R

R

► R Packages for Social Network Analysis

R Packages for Social Network Analysis

Ian McCulloh and Alexander Perrone
Applied Physics Laboratory, Johns Hopkins University, Laurel, MD, USA

Synonyms

Ergm; igraph; Network; Network science; R; R-project; RSiena; SNA; SocialMediaLab; Statnet

Glossary

CLI	Command Line Interface is a software interface that requires users to enter typed commands to perform computational operations
CRAN	The Comprehensive R Archive Network is a network of FTP sites around the world that stores identical code and documentation for R. The CRAN is maintained by the R Foundation
GUI	Graphical User Interface is a software interface designed to simplify and standardize the use of computer programs
INSNA	The International Network of Social Network Analysts is the academic professional society for social network analysis, founded in the 1970s
Package	A set of computer code, functions, and commands that have been bundled, tested, reviewed, and are available for use in the R programming environment
R	An open source computer language for statistical computing
R-Project	An activity supported by the nonprofit R Foundation to promote open source statistical computing
Social Media	An online means of communication, where large groups of people communicate, share information, and develop personal and professional contacts
SOCNET	A list-serv where social network scholars share knowledge and information pertaining to social network analysis research and application
Sunbelt Social Networks Conference	The annual international conference of INSNA

R. Alhajj, J. Rokne (eds.), *Encyclopedia of Social Network Analysis and Mining*,
https://doi.org/10.1007/978-1-4939-7131-2

Definition

R is an open source computer language and environment for statistical computing. Many users contribute to the R environment by publishing packages that perform various computational functions. There has been a recent growth in the use of R for social network analysis and with it the introduction of numerous packages. In some cases, these packages provide redundant capabilities and may be incompatible with each other. In other cases, packages provide unique analytics, but may not be widely known. This entry provides a review of many of the common R packages used in social network analysis. Recommendations for users and future R development are provided.

Introduction

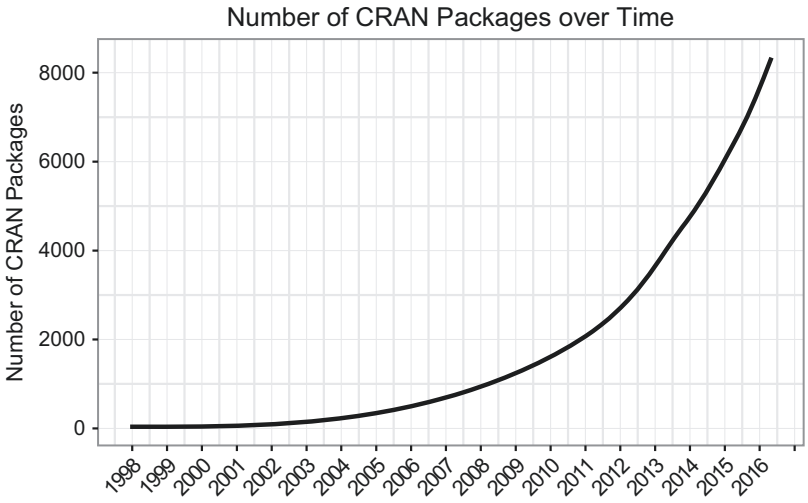
Social network analysis (SNA) is an academic methodology to investigate the structure of social interactions among people (Wasserman and Faust 1994; McCulloh et al. 2013). SNA relies on a wide range of interdisciplinary theories from throughout the social and natural sciences. Social networks consist of actors, represented by nodes (or vertices), and relationships between actors, represented by links (or edges). There are many types of relationships that may be represented, such as friendship, respect, or co-affiliation. There are infinite potential relationships that can be modeled or investigated. SNA is often extended more broadly than networks of people. These extended networks are referred to as meta-networks (Carley and Krackhardt 1998; Carley 1999) and can include nodes such as organizations, resources, skills, tasks, events, roles, beliefs, and more.

A key aspect of SNA that differentiates it from other analytic methods is its focus on structure. In most scientific analyses, data are assumed to be independent and identically distributed (i.i.d.). In cases where this does not hold, the data are often transformed so that the i.i.d. assumption is met. This assumption allows the application of powerful statistical methods. In most applications of SNA, the i.i.d. assumption is assumed to be false. The behavior and attributes of one actor are not independent of the other actors within their network and they are not identically distributed. SNA is interested in how these structures form and how they impact the behavior of the nodes and groups as a whole. Interest in network structure is not exclusive to scientists studying social groups. There are many applications of networks, and the methods and measures developed in SNA have found broader application to a range of scientific disciplines.

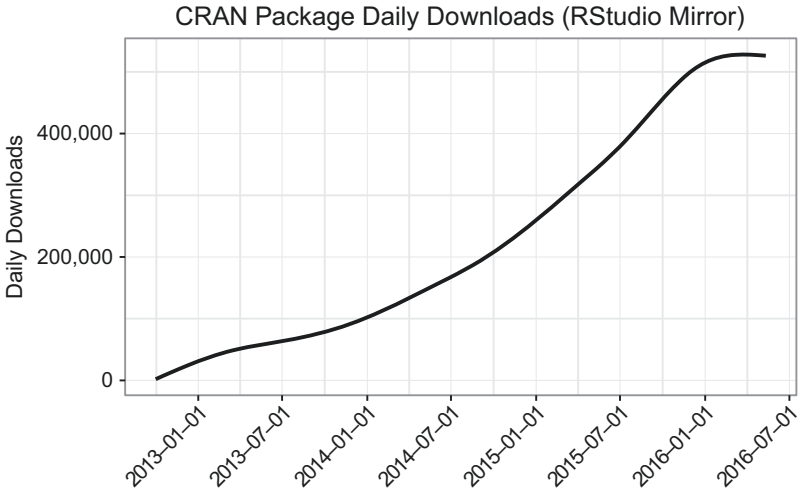
The broad scope of SNA has led to a wide range of software. R is an open-source language and environment for statistical computing and graphics (Ihaka and Gentleman 1996), which now includes additional applications such as SNA. Communities of contributors write and maintain packages for use in R. Still others write and share scripts to perform certain analyses. It is common that a package or script intended for one application will be repurposed for another. There are also multiple packages for R that accomplish some of the same functions. For example, the packages 'igraph' and 'sna' both support SNA and offer commands to calculate some of the same common structural measures such as centrality (Freeman 1977, 1978; Freeman et al. 1979) or triad census (Wasserman 1977).

Historical Background

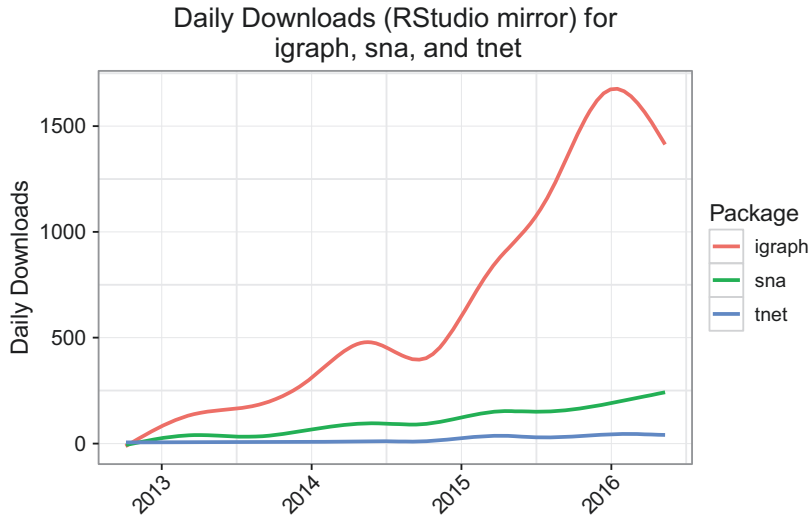
There has been tremendous growth in both the use of R and the number of available packages. As of the writing of this entry, there have been over 8000 R packages published to the Comprehensive R Archive Network (CRAN), the main repository of packages, and documentation for R. Figure 1 displays the number of CRAN packages over time. This shows exponential growth in the contributions to R. Figure 2 displays an estimate of the number of user downloads of R packages over time. This figure shows a similar explosion in



R Packages for Social Network Analysis, Fig. 1 The number of R packages available over time. The CRAN page for each package was used to provide a dataset of packages. The main page for each package often (but not always) provides a link to "old sources" which contains an archive of previous releases. The earliest release date on the archive was used as the initial release date for the package. If no archive existed, the published date on the main page was used as the initial release date, since there was only one version of the package. The initial release dates were then sorted by date from earliest to latest, and a cumulative sum was computed to count the number of packages over time, which is displayed in the plot



R Packages for Social Network Analysis, Fig. 2 R Package Daily Downloads. Obtained using the 'cran.stats' package in R, which provides functions for scraping and analyzing download logs from RStudio's CRAN mirror. The download logs were obtained from 2012–10-01 (earliest available date) to 2016–05-11. The plot shows the daily downloads smoothed using locally estimated scatterplot smoothing (LOESS) (Cleveland 1979) with span set to 0.75 to remove daily variability. These data are used as a proxy for R package downloads. These data are therefore sensitive to the popularity of RStudio as an environment for using R, with the proxy becoming better the more RStudio has become popular among R users



R Packages for Social Network Analysis, Fig. 3 Estimated daily downloads of SNA packages in R over time. The download logs from the RStudio mirror were subset to the packages igraph, sna, and tnet, and their downloads are shown in the plot. The daily estimates were smoothed using LOESS with span set to 0.75

activity related to R with over half a million package downloads per day.

R has also become increasingly popular for SNA. Figure 3 shows the growth in three common R packages for use with social networks: 'igraph', 'sna', and 'tnet'. While SNA use in R is not as dramatic as overall R usage, the trend is generally increasing. Daily download estimates of SNA packages in R exceed 1500.

Key Points

Given the growth of SNA use in R and the explosion in the number of packages, a review of available R packages is necessary to assist users in finding packages with needed functionality. Due to the volume of packages, however, a package by package comparison was deemed infeasible. The remainder of this entry is therefore organized as follows. Advantages and disadvantages of the use of R compared to commercial SNA software are briefly discussed. Definitions that are relevant to the R community are provided. Given the number of R packages in use for SNA and the broader R community, a methodology is introduced that is used to identify and compare key packages for SNA. Data obtained using this methodology are also described. Descriptions and comparisons of key packages are discussed. This includes key packages for SNA as well as packages intended to add niche functionality to other SNA packages. Recommendations are provided for those using R for SNA. Finally, future directions for the use and implementation of SNA in R are proposed.

Advantages and Disadvantages of R

There are advantages and disadvantages for the use of R as an SNA software tool. This is also true of particular packages in R. R is free and open source. This makes the use of R available to anyone, even with no budget. While this free tool does not have direct customer support, there is a growing user community that will provide help and advice. In addition, there are numerous training opportunities such as in-person or online bootcamps, academic conference workshops, online courses, and learning platforms. There are also numerous online Coursera courses using R, including two R specializations by JHU and Duke (Peng et al. 2016; Çetinkaya-Rundel et al. 2016). Many of the organizations offering these training opportunities are also available to offer tailored workshops for organizations wishing to adopt R.

A disadvantage of R is that commands are entered via code at the command line, rather than a point-and-click graphical user interface

(GUI). While there are some GUI options available, such as R Commander, users typically need to have some comfort with programming to take full advantage of R. Recently, the Shiny web application framework has become a popular way to create user interfaces using only R. At this point, R users typically need to develop their own user interfaces, although there have been some R packages utilizing Shiny which provide user interfaces for other R packages, such as statnetWeb (for statnet) and ShinyStan (for Stan, JAGS, and related languages). It is likely that these applications will become more common, making the use of R more accessible to a wider range of users. Nevertheless, like the R package landscape, it will likely remain a decentralized array of user interfaces built for particular purposes, rather than a single, general GUI application.

Since R content is generated by a user community, it is possible that the algorithms developed for applications are not tested to the level of rigor expected of commercial software (Wasserman and Capra 2007). In addition, customers of commercial software have the expectation that the code base will be maintained, updated, and quality-controlled and that in some cases support services will be provided. The tradeoff, however, is that novel analytics often appear in R before they are available in commercial software, if they ever appear in commercial software at all. Users also maintain a great deal of flexibility by being able to modify or develop their own analytics in R. This flexibility is valued by users who are often faced with novel questions that cannot be addressed with traditional software packages.

The R Development Environment

R is a computer language and environment for statistical computation and graphics. It is available as free software distributed under the GNU General Public License, version 2. The goal of this license is to ensure that developers maintain the freedoms to run R for any purpose, access and study its source code, redistribute copies, and redistribute any modified versions of R under the terms of the license. In addition, many R packages are distributed with either a free software license or an open-source license.

There are four primary methods that R developers share their code. The Comprehensive R Archive Network (CRAN) is a collection of web servers that offer the R software environment, extensions or packages contributed by the R community, and documentation for R. The master R site is located at https://CRAN.R-project.org. This is the official repository maintained by the R Foundation and a team of volunteers. Developers create packages to extend R functionality and submit them through the CRAN website for consideration to be included on the CRAN repository. Policies for CRAN package development and submission can be found at https://cran.r-project.org/web/packages/policies.html. The policies ensure that the package is free of legal restrictions, conforms to certain standards, and does not create major conflicts or problems with existing packages, such as two packages with the same name. It also requires that a single individual maintain the package and that they can easily be contacted by the CRAN team. The CRAN is the primary location to find stable R code for application.

Two other locations for R code are R-Forge and GitHub. These platforms provide version control and aid in collaboration among software developers. It allows two or more developers to modify code at the same time. Others in the community can review and make comments on the code that may offer helpful suggestions or efficiency. There are thousands of packages available on R-Forge and GitHub. A developer may or may not choose to submit their finished code to CRAN, as the CRAN standards are more restrictive than R-Forge or GitHub. In addition, using a package from one of these sites is just as easy as using one from CRAN. For example, a GitHub package can be loaded into the R environment with the following commands:

```
install.packages("devtools")
devtools::install_github
("username/packagename")
```

The disadvantage of R-Forge and GitHub is that the packages may not be maintained and may not meet the compatibility and quality standards required by CRAN. For this reason, we restrict our exploration of SNA packages in R to the CRAN site only.

The fourth method that R developers share code is through scripts. Scripts can be shared as a text file or as files written in R and saved with a .R extension. Scripts may not be under version control and may not be part of a package available on CRAN, R-Forge, or GitHub. Most R users, however, will maintain personal scripts for using R that they might share with students or colleagues. Scripts do not need to be loaded like a package (i.e., using the library command) since they are a collection of commands that can be run directly at the command line.

The R package is the basic unit of shareable R code. An R package includes functions and documentation describing their use and often includes sample data. An R package may also require dependencies. A dependency is another R package that is used by the primary package. A dependency has two levels of strength. An import dependency is a package that must be loaded in order for the primary package to work. A suggest dependency is a package that can be used by the primary package to extend functionality. Import dependencies are automatically loaded with the primary package, whereas suggest dependencies are not.

R does not allow cyclic dependencies. In other words, if package A is required by package B and B is required by package C, then C cannot be required by package A. The dependency relation in R therefore allows the construction of a hierarchical network with no cycles.

This section is by no means comprehensive and such an undertaking is well beyond the scope of this entry. This section is meant to provide a brief orientation to the R development environment and packages. CRAN and dependency relations will be used to identify packages that are available and useful for conducting SNA in R.

Method and Data Collection

Given the large number of packages available in R, it was important to develop a methodology for which SNA relevant packages could be identified. The methodology used an inductive and a segmental approach (Valente 2012). The inductive approach began with seeking feedback from the SNA community about common R packages and applications. The segmental approach involved scraping all R packages and using a community detection approach to identify categories of related packages that might find application with SNA. This method resulted in 135 R packages that find application in SNA and nine categories of packages.

The inductive approach began with identifying a broad base of SNA professionals. The authors identified the International Network of Social Network Analysts (INSNA). This organization is the academic professional society for SNA, founded in 1977. Each year, INSNA hosts an annual conference known as the International Sunbelt Social Networks Conference. In 2016, this event was attended by over 800 people. The conference offers a number of workshops at the beginning of the conference. INSNA also maintains a list-serv called "SOCNET" where the SNA community can post questions, share resources, and provide assistance to others in the community.

Once a community of experts was defined, we identified the use of R in the Sunbelt conference workshops. Of the 33 workshops, 16 of the workshops used R. The nine specific packages that were taught include: 'statnet', 'ergm', 'tergm', 'igraph', 'netdiffuseR', 'RSiena', 'BlauNet', 'egonetR', 'SocialMediaLab'. This step was used to identify popular SNA packages in current use by the SNA community.

Step 2 required a search of the CRAN for the following key words: "network," "matrix," and "graph." This returned 53 additional packages including: 'amen', 'assortnet', 'bipartite', 'blkergm', 'blockmodeling', 'data.table', 'devtools', 'd3Network', 'DCG', 'dils', 'dnc',

'Dominance', 'geomnet', 'GGally', 'ggnetwork', 'influenceR', 'intergraph', 'latentnet', 'keyplayer', 'linkcomm', 'Matrix', 'mcPAFit', 'multiplex', 'ndtv', 'NetCluster', 'netcoh', 'NetComp', 'nets', 'NetSim', 'NetSwan', 'nettools', 'NSUM', 'network', 'networkDynamic', 'networksis', 'networkD3', 'queueing-package', 'sand', 'slam', 'sna', 'SNscan', 'SocialNetworks', 'sparkR', 'SPARQL', 'SparseM', 'spnet', 'statnetWeb', 'statools', 'timeordered', 'tnam', 'tnet', 'tsna', and 'visNetwork'. Upon inspection, it was clear that the primary application for several of these packages was not SNA. We therefore removed the following packages from consideration: 'data.table', 'devtools', 'GGally'. Several additional packages used the term "network" for nonsocial network application, for example, "neural networks" or "Bayesian networks." These packages were excluded from additional analysis.

The third step involved searching the CRAN for well-known social media platforms. This search returned 12 additional packages. They consisted of 'graphTweets', 'instar', 'RedditExtractoR', 'Rfacebook', 'RKlout', 'Rlinkedin', 'RNewsflow', 'SocialMediaMineR', 'tumblR', 'twitteR', 'WikipediaR', and 'WikiSocio'. While the authors recognize the difference between SNA and social media analysis (SMA), the SMA packages remained in the data set under investigation.

For the fourth step, the consolidated list of R packages was posted to the SOCNET list-serv with the following request: "If you are an R user, I'd appreciate it if you could look at the list [of R packages] and let me know if I am missing any packages." List-serv members were also asked to provide "any thoughts on things [related to R and SNA] that might [be] useful in a review entry on this topic." The list-serv remained active on this topic for approximately 1 month and one of the authors continued to receive personal emails regarding the use of R for SNA for approximately two months. This resulted in 10 additional packages to include: 'cssTools', 'astsa', 'forecast', 'hergm', 'PCIT', 'RSienaTest', 'mixer', 'dplyr', 'ggplot', and 'rvest'.

The fifth and final step in the induction approach was to scrape the descriptions of all R packages and conduct a key word search for "igraph," "graph," "network," "sna," and "social network analysis." This resulted in identifying 386 packages, some of which were already identified. Many of the 386 packages were not applicable to social network analysis, but may have used a network analysis approach for some other application. The authors made a qualitative judgment on which of the 386 packages should be included. The total list of SNA-related packages consists of 135 and is described below.

The segmental approach intended to identify categories of R packages. Categories for R packages were posted on the INSNA SOCNET list-serv and also hosted on a Google Sheets page, where people in the SNA community were able to nominate categories for SNA packages. The nominations consisted of (1) basic SNA, (2) statistical analysis of networks, (3) network visualization, (4) social media analysis, (5) community detection and subgroup analysis, and (6) a miscellaneous category. Upon further examination of the consolidated list of packages, the authors added three additional categories consisting of (7) simulation, (8) data management, and (9) example data packages. One of the people on the SOCNET list-serv suggested doing a consensus analysis, however, due to the cross-discussion, people's responses were not provided independently from the input of others. Therefore, a key assumption underlying consensus analysis was missing. Ideally, future similar efforts should attempt to conduct a proper consensus analysis with a pile sort to determine categories. For this project, 13 people provided input into the category selection similar to a focus group, but conducted asynchronously, online.

The final step in the segmental approach involved creating a network visualization, where nodes represented R packages and links represented dependency relations as reported on the CRAN package page as of May 5, 2016. Nodes are colored based on the category in which they were grouped. The authors provide

qualitative discussion and insights based on the network visualization.

The analysis revealed two primary basic SNA packages used in R: 'igraph' and 'sna'. These packages perform some redundant analytic calculations, they use different data structures, and they create conflicts with each other. Therefore, the authors conducted several performance comparisons between the two packages. Performance criteria included system time and consistency across calculation. Memory as a performance criterion was compared for the two different data structures used by 'igraph' and 'sna'. In order to evaluate the performance, four SNA tasks were selected that could be performed in both 'igraph' and 'sna'. Those tasks included two centrality measures, degree and betweenness, as well as two graph level measures, diameter, and triad census. These were selected to vary the computational complexity of the calculation as well as the focus of measure. In addition, the size and topology of the networks were varied. While the performance of the basic SNA package cannot speak to the efficiency and functionality of the other packages that depend upon it, it may provide some additional insight for selecting which basic SNA package to build upon for new development.

Categorization of R Packages

A full explanation and comparison of the 135 packages used for SNA in R is well beyond the scope of this entry. This section will identify packages used for common SNA functions and provide additional detail for the most popular and widely used packages. The nine categories of packages are: basic SNA, statistical analysis of networks, network visualization, social media analysis, community detection/subgroup analysis, niche SNA, simulation, data management, and data.

Basic SNA

There are two main packages for basic SNA in R: 'igraph' and 'sna'. The package 'igraph' is by far the most popular by download statistics. This may be due, in part, to its application in network analysis that is not necessarily social. Another possible explanation is that 'igraph' is not exclusive to R. It is also available for Python and C/C++. The package 'sna', on the other hand, is primarily used by the package 'statnet' and associated dependencies for the statistical analysis of networks solely within R.

The two packages use different data structures and are not fully compatible. The 'igraph' package uses its own class in R. The command "as.igraph()" will coerce a matrix object into an 'igraph' object. The 'igraph' package also has an "as_adj_list()" command to coerce a data structure into an adjacency list format, which is often more computationally efficient. The 'sna' package, however, uses the 'network' class and relies on the 'network' package.

There is a package 'intergraph' that provides functions to coerce network data between a 'network' object and an 'igraph' object. This package does not make commands interoperable between the 'sna' and 'igraph' however. It will be necessary to load both 'sna' and 'igraph' in addition to 'intergraph' in order to coerce network data from one format to the other. The "library()" command is used to load a package into R. Once the network data has been converted into the desired format, the "detach()" command should be used to remove the unneeded package from the environment. Alternatively, one could also reference functions directly if there is a conflict without using the detach command. For example, the command "igraph::betweenness(N)" will calculate the betweenness centrality of graph N using the 'igraph' package, while the command 'sna::betweenness(N)' will calculate betweenness centrality of graph N using the 'sna' package. Referencing functions without detaching them is necessary if one still needs both packages loaded for some reason. Failure to resolve conflicting packages can result in excessive bugs and errant analysis.

Table 1 contrasts the difference in basic capability and associated command for common SNA routines. It can be seen in the table that each package contains measures that the other does

R Packages for Social Network Analysis, Table 1 Comparison of basic SNA package functions

Measure	igraph	sna
Bonacich Centrality	alpha_centrality	–
Assortativity coefficient	assortativity	–
Authority Centrality	authority_score	–
Centralization	centralize	centralization
Centralization, betweenness	centr_betw	centralization
Centralization, closeness	centr_clo	centralization
Centralization, degree	centr_degree	centralization
Centralization, eigenvector	centr_eigen	centralization
Cliques	cliques	clique.census
Centrality, betweenness	betweenness estimate_betweenness	betweenness
Centrality, closeness	closeness	closeness
Centrality, degree	degree	degree
Centrality, eigenvector	eigen_centrality	evcent
Diameter	diameter	~geodist
Dyad Census	dyad_census	dyad.census
Find Ego Network	ego	ego.extract
Hub Centrality	hub_score	–
Page Rank	page_rank	–
Bonacich Power Centrality	power_centrality	bonpow
Radius	radius	–
Reciprocity	reciprocity	grecip mutuality
Transitivity	transitivity	gtrans
Triad Census	triad_census	triad.census
Blockmodel	–	blockmodel
Connectedness	–	conectedness
Efficiency	–	efficiency
Density	–	gden
Hierarchy	–	hierarchy
Information Centrality	–	infocent
Least Upper Bound	–	lubness
Triad Classification	–	triad.classify

not. Both packages calculate basic centrality and common graph level measures.

The two basic SNA packages 'igraph' and 'sna' were compared for performance differences using simulation. Directed networks were simulated using 'igraph's built-in simulation functions. Half of the networks were generated as Erdos and Renyi (1959) random networks using the command "erdos.renyi.game()." The other half of the networks were generated as scale-free networks using a preferential attachment mechanism (Barabási and Albert 1999) using the command "barabasi.game()." Half of the networks were small networks, consisting of 30 nodes. The other half of the networks were mid-size networks consisting of 150 nodes. The density of the network was fixed at 0.1. The "m" parameter on the scale-free network was used to calibrate the scale-free network generation to achieve a comparable network density. For the 30 node network, $m = 3$, and for the 150 node network, $m = 2$. A total of 1000 simulations were run for each of the four conditions. For each simulation run, the elapsed time was recorded for degree centrality,

R Packages for Social Network Analysis, Table 2 Comparison t-Statistics for 'igraph' versus 'sna' basic SNA packages

		2-Sample t-tests					Pairwise t-test				
Nodes	Topology	Deg	Btw	Dia	Triad	Size	Deg	Btw	Dia	Triad	Size
30	Random	−61.5	−24.7	105.3	162.7	118.3	−58.8	−26.7	103.5	175.5	123.2
30	Scale-free	−68.7	−49.6	115.1	139.5	22410	−63.8	−50.9	111.2	142.4	22,410
150	Random	180.6	414.6	390.8	539.7	560.6	183.6	423.6	397.4	566.2	583.7
150	Scale-free	34.2	59.0	437.1	302.7	58272	33.9	68.9	447.1	301.3	58,272

betweenness centrality, diameter, and triad census on the generated 'igraph' network object. The 'intergraph' package was then used to coerce the simulated network from an 'igraph' object into a 'network' object for use with the 'sna' package. The rationale was to compare a computationally fast algorithm, such as degree centrality with a computationally slow algorithm, such as betweenness centrality. The diameter and triad census measures provided graph level measures. The object size in memory of the generated network, stored as an 'igraph' object and as a 'network' object were also recorded.

For each measure, a two-sample t-test and a pairwise distance t-test were performed to evaluate differences. In all cases, elapsed time for 'igraph' was subtracted from elapsed time for 'sna'," so a positive value of the t statistic indicates that 'igraph' is faster, while a negative t statistic indicates that 'sna' is faster. For memory the same convention was applied, meaning that a positive value for the t statistic indicates 'network' objects are more memory intensive, while a negative value indicates 'igraph' objects are more memory intensive. Table 2 provides the t-statistics for the different simulation conditions. All tests are significant at the $p < 0.0001$ level.

Neither package is uniformly superior to the other. The negative t-statistics in Table 2 demonstrate that the 'sna' package is computationally faster for calculating centrality measures on small networks around 30 nodes. In all other cases, however, 'igraph' outperforms 'sna' in terms of elapsed time and the object size in memory. In the range of network size and topology where 'sna' outperforms 'igraph', however, the networks are small enough that the user is unlikely to notice the difference in performance. While the difference is statistically significant, the effect size of two to four ten thousandth of a second difference is undetectable to the user.

Another basic SNA package is 'tnet'. This package is designed to analyze weighted networks, two-mode networks, and longitudinal networks. It provides a method of analysis to prevent information loss when these networks are simplified for dichotomous, single-mode, and static networks. This package depends upon igraph and uses the igraph class as its data structure.

For the remaining sections on categories of packages, each table of packages will reference the basic SNA package that is listed as a dependency for each package in the category. In the event that a package does not have a basic SNA package listed as a dependency, then "n/a" will be recorded in the Table.

10.2. Statistical Analysis of Networks

There are 35 identified packages for the statistical analysis of social networks in R as shown in Table 3. Most use the 'network' class and rely on the 'sna' basic SNA package. A number of packages simply work with a 'Matrix' class object. In the authors' opinion, 'sna' provides access to a wider range of useful statistical analysis functions for use with network data.

Network Visualization

There are several approaches to network visualization. There are basic visualization tools that provide the code necessary to plot static networks. Enhanced visualizations are often made possible by JavaScript. JavaScript can be accessed through a package such as 'networkD3' or 'visNetwork'

R Packages for Social Network Analysis, Table 3 Statistical analysis packages

Package	SNA Pkg	Application
amen	sna	Analysis of dyadic relations using additive and multiplicative effects (AME)
asnipe	igraph, sna	Animal Social Network Inference and Permutations
Bergm	sna	Tools for Bayesian exponential random graph models
blkergm	sna	Fitting block ERGM given the block structure on social networks
bootnet	n/a	Bootstrap Methods for Various Network Estimation Routines
btergm	sna	Temporal ergm by Bootstrapped Pseudolikelihood
degreenet	igraph, sna	Models for Skewed Count Distributions Relevant to Networks
ergm	sna	Fit, Simulate and Diagnose Exponential-Family Models for Networks
ergm.count	sna	Fit, Simulate and Diagnose ergm with Count Edges
ergm.ego	sna	Fit, Simulate and Diagnose ergm to Egocentrically Sampled Network Data
ergm.graphlets	sna	ERG Modeling Based on Graphlet Properties
ergm.rank	sna	Fit, Simulate and Diagnose ergm for Rank-Order Relational Data
ergm.userterms	sna	User-specified terms for the statnet suite of packages
GERGM	igraph	Tools for generalized exponential random graph models
hergm	sna	Hierarchical exponential random graph models with local dependence
HLSM	n/a	Hierarchical Latent Space Network Model (HLSM)
JRF	n/a	Joint Random Forest (JRF) for the Simultaneous Estimation of Multiple Related Networks
latentnet	sna	Latent position and cluster models for networks
lvm4net	igraph, sna	Latent Variable Models for Networks
netcoh	n/a	Statistical models incorporating network cohesion as an effect
netdiffuseR	igraph	Tools for estimating models of the diffusion of innovations
nets	igraph	Network estimation for time series
PCIT	n/a	Identifies meaningful correlations to define edges in a weighted network
Rambo	sna	The Random Subgraph Model
rem	sna	Relational event models
RSiena	n/a	Fits stochastic actor oriented models to longitudinal network data
statnet	sna	An umbrella package for many network analysis packages
statnet.common	sna	Nonstatistical utilities used by statnet and related packages
statnetWeb	sna	A web-based interface for using the statnet package
staTools	n/a	Tools for modeling the statistical distribution of a vector of observations
tergm	sna	Temporal ergm for dynamic networks over time
timeordered	igraph	Time ordered and time aggregated network analysis
tnam	sna	Temporal network autocorrelation models
tsna	sna	Temporal SNA tools for continuous and discrete time networks over time
xergm.common	sna	Extensions of exponential random graph models

which provide different libraries for visualization. Other packages, such as 'plotly', translate network layouts into an interactive, web-based version. In the authors' subjective opinions, the 'igraph' basic SNA package provides access to better visualization layouts, routines, and downstream packages than the 'sna' basic SNA package. There are 17 visualization packages identified as shown in Table 4.

Social Media Analysis

There were 13 packages for analyzing social media identified in R and shown in Table 5. With the exception of 'SocialMediaLab', all tools focused on simply extracting data from an application program interface and storing the data as a list, data.frame, data.table, or some other general class. The data can then be coerced into either a 'network' or 'igraph' object for further analysis.

R Packages for Social Network Analysis, Table 4 Network visualization packages

Package	SNA Pkg	Application
bipartite	igraph,sna	Visualize bipartite networks
d3Network	n/a	Tools for D3 JavaScript – *Active development is done in network D3*
diagram	n/a	Functions for visualizing simple graphs, plotting flow diagrams
edgebundleR	igraph	Circle Plot with Bundled Edges
geomnet	sna	Single geometry approach to network visualization with ggplot2
ggnetwork	sna	Geometries to plot networks with ggplot2
ggplot2	n/a	Implementation of the grammar of graphics
HiveR	sna	2D and 3D Hive Plots for R
igraphinshiny	igraph	Use shiny to Demo igraph
ndtv	sna	Tools for animated visualizations of networks
networkD3	igraph	D3 JavaScript network graphs from R
networkDynamic	sna	Support for dynamic, temporal networks
plotly	n/a	Translate ggplot2 into an interactive web-based version
SPARQL	n/a	Select or update queries
spnet	igraph	Plotting social networks on maps
threejs	igraph	Interactive 3D Scatter Plots, Networks, and Globes
visNetwork	igraph	Uses vis.js library for network visualizations

R Packages for Social Network Analysis, Table 5 Social media analysis packages

Package	SNA Pkg	Application
graphTweets	igraph	Visualise Twitter Interactions
instar	n/a	Tools for Instagram data
RedditExtractoR	n/a	Tools for Reddit
Rfacebook	n/a	Tools for Facebook data
RKlout	n/a	Fetch Klout Scores for Twitter Users
Rlinkedin	n/a	Tools for LinkedIn data
RNewsflow	igraph	Tools for Analyzing Content Homogeneity and News Diffusion
SocialMediaLab	igraph	Tools for collecting and constructing networks from social media data
SocialMediaMineR	n/a	Investigates popularity and reach of URL(s) on social media
tumblR	n/a	Access to Tumblr v2 API
twitteR	n/a	Tools for Twitter data
WikipediaR	n/a	Wikipedia web API. Login not required
WikiSocio	igraph	store wiki archives to R object – data-frame, lists, vector, and variables

Community Detection/Subgroup Analysis

There are several clustering and community detection packages available in R. Some packages are intended for use with 'igraph', others with 'sna', and still others use the 'Matrix' object. Based on the investigation described in this entry, community detection and subgroup analysis remains limited. There is more functionality to visualize clusters and compare the quality of different clustering solutions than there exists to perform clustering and community detection. A total of 18 packages were identified for community detection and subgroup analysis, which are shown in Table 6.

Niche SNA

The following table lists 30 R packages that support important SNA capability, but do not seem to

R Packages for Social Network Analysis, Table 6 Community detection and subgroup analysis packages

Package	SNA Pkg	Application
blockmodeling	sna	Generalized blockmodeling for valued networks
blockmodels	n/a	Latent and stochastic blockmodel estimation
cvxbiclustr	igraph	Convex Biclustering Algorithm
cvxclustr	igraph	Splitting methods for convex clustering
DCG	n/a	Random walks to find community structure in networks
dnc	igraph	Community detection for dynamic networks (repeated measures)
fcd	n/a	Fused Community Detection
linkcomm	igraph	Reveals nested and overlapping structure in networks
MCL	n/a	Markov Cluster Algorithm
mixer	n/a	Random graph clustering
modMax	igraph	Community Structure Detection via Modularity Maximization
NetCluster	sna	Evaluate clustering solution
rnetcarto	igraph	Fast Network Modularity and Roles by Simulated Annealing
sbmSDP	n/a	Semidefinite Programming for Fitting Block Models of Equal Block Sizes
SNscan	igraph	Test network clustering patterns using scan statistics
SOMbrero	igraph	SOM Bound to Realize Euclidean and Relational Outputs
VBLPCM	sna	Variational Bayes Latent Position Cluster Model for Networks
wfg	igraph	Weighted Fast Greedy Algorithm

fit within the previous categories. Most of these packages are intended for use with 'igraph'. These packages are shown in Table 7.

Simulation

A great deal of research in SNA requires the simulation of networks. For example, comparing the performance of 'igraph' and 'sna' required simulation of networks. The following table lists R packages that can be used to simulate network data under different conditions and topologies. There were seven packages identified as shown in Table 8.

Data Management

Some R packages are used just to manipulate network data or coerce one data object into another. While it might have been helpful to include general purpose data management packages such as 'data.table', or 'dplyr', we restrict our list of packages to those that are often listed as dependencies for network packages. There are ten packages in the list, which are shown in Table 9.

Four packages included in this section have broader application than SNA. They are 'Matrix', 'slam', 'spam', and 'SparseM'. These are different packages used for storing and manipulating sparse matrices. When the density of a matrix is approximately greater than 0.5, an adjacency matrix is most efficient in terms of memory. When the density is smaller than 0.5 the matrix becomes sparse and an edge list, or adjacency list becomes more efficient.

It is important to note that the SNA basic packages will use their own data object, either 'igraph' or 'network'. Therefore, once the data is encoded in the package specific data object, the sparse matrix packages will not be useful unless data is converted back into a matrix format for operation. The sparse matrix packages are used for gross data manipulation and not for SNA-specific calculations such as centrality values. Performance comparisons of the matrix packages are outside the scope of this entry.

Datasets

The final category of packages contains packages that primarily provide sample datasets. These packages are often used to support training and education of the R community. They are shown in Table 10.

R Packages for Social Network Analysis, Table 7 Niche or uncategorized network analysis packages

Package	SNA Pkg	Application
agop	igraph	Aggregation operators for scientometrics and bibliometrics
assortnet	n/a	Calculate assortativity
backShift	igraph	An algorithm to estimate the connectivity matrix of a directed (possibly cyclic) graph with hidden variables.
bingat	sna	Binary Graph Analysis Tools
Blaunet	sna	Calculate and Analyze Blau Status for Measuring Social Distance
CITAN	igraph	CITation ANalysis Toolpack
crimelinkage	igraph	Statistical Methods for Crime Series Linkage
cssTools	sna	Estimate networks from random sample of cognitive social structures
dils	igraph	Data informed link strength: combines multiple networks into one
disparityfilter	igraph	Disparity Filter Algorithm for Weighted Networks
Dominance	igraph	Calculate dominance indices
ebdbNet	igraph	Infer the adjacency matrix using an empirical Bayes estimation
egonet	sna	Analysis of ego-centric network data
influenceR	igraph	Node centrality focused at identifying opinion leaders
interactiveIGraph	igraph	Extends igraph to interactively work with igraph objects
keyplayer	igraph,sna	Calculates group centrality scores
mcPAFit	n/a	Estimate preferential attachment from a single networks using MCMC
NetSwan	igraph	Analysis of network robustness, resilience, and vulnerability
nettools	igraph	Network comparison
networkreporting	n/a	Functions for producing estimates from data collected using network reporting techniques like network scale-up, indirect sampling, network reporting, and sibling history
optrees	igraph	Finds optimal trees in weighted graphs
PAFit	n/a	Nonparametric Estimation of Preferential Attachment and Node Fitness in Temporal Complex Networks
qgraph	igraph, sna	Used to visualize data as networks as well as provides an interface for visualizing weighted graphical models
QuACN	igraph	Topological measures for structural analysis of complex networks
SDDE	igraph	Compares network X to an augmented network Y by counting the number of Shortcuts, Detours, Dead Ends (SDDE), equal paths, and disconnected nodes.
SocialNetworks	n/a	Generates social networks based on node distance
spatgraphs	n/a	Graph component calculations for spatial locations
spatialnbda	n/a	Network-based diffusion analysis (NBDA) allows inference on the asocial and social transmission of information
SSrat	sna	Sociometric status determination with rating scales
VertexSimilarity	igraph	Vertex Similarity matrix of an undirected graph

Network Analysis of R SNA Packages

A simple network analysis of R packages shows two clusters surrounding the two basic SNA packages. Figure 4 shows a network visualization of the above R packages. Nodes represent packages and links represent dependency relations between them. Isolate nodes, representing packages with no dependencies to other listed packages, are omitted from the visualization. The node color indicates the category of R package. It is important to note that 'igraph' is both an SNA package and a network data object, while 'sna' is an SNA package that uses the 'network' network data object. Therefore, 'sna' and 'network' would need to be merged into a single node for a more equitable comparison. It is clear to see the two

R Packages for Social Network Analysis, Table 8 Network simulation packages

Package	SNA Pkg	Application
CCMnet	sna	Simulate Congruence Class Model for Networks
CIDnetworks	igraph	Generative Models for Complex Networks with Conditionally Independent Dyadic Structure
NetComp	n/a	Network generation and comparison
NetSim	igraph	Social network simulation tool
networksis	sna	Tools to simulate bipartite networks/graphs with the degrees of the nodes fixed and specified
snowboot	igraph	Network simulation nonparametric methods of analysis center around snowball and bootstrap sampling
SpaDES	igraph	Implement a variety of simulation models, with a focus on spatially explicit agent-based models

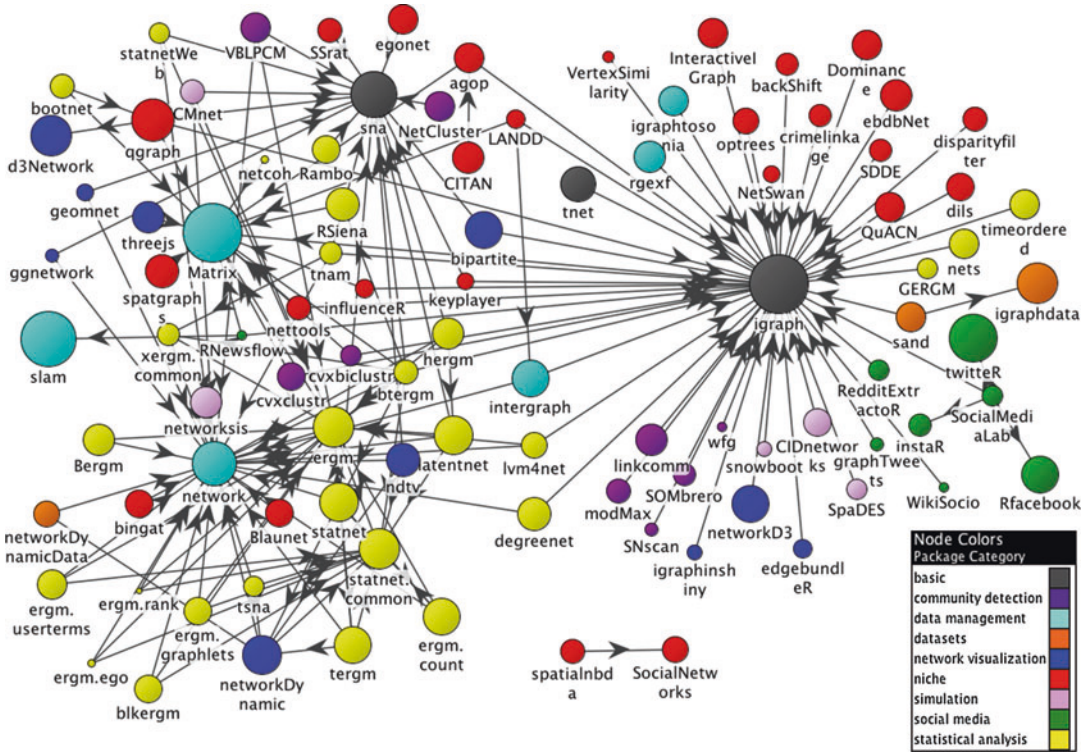
R Packages for Social Network Analysis, Table 9 Data management packages

Package	SNA Pkg	Application
igraphtosonia	igraph	Exports igraph objects to the SoNIA file format
intergraph	igraph,sna	Converts data objects between igraph and network classes
Matrix	n/a	Methods and operations for dense and sparse matrices
multiplex	n/a	Relational algebra tools for the analysis of multiple social networks
network	sna	Tools to create and modify network objects
rgexf	igraph	Create, read and write GEXF (Graph Exchange XML Format) files used in Gephi and other programs
slam	n/a	Data structures and algorithms for sparse arrays and matrices
SNFtool	n/a	Takes multiple views of a network and fuses them together to construct an overall status matrix
spam	n/a	Set of functions for sparse matrix algebra
SparseM	n/a	Basic linear algebra functionality for sparse matrices

R Packages for Social Network Analysis, Table 10 Packages with sample datasets

Package	SNA Pkg	Application
historydata	n/a	Sample datasets intended for historians learning R to include network analysis with R
igraphdata	igraph	Datasets for use with igraph
networkDynamicData	sna	A collection of dynamic network datasets from various sources and authors represented as networkDynamic formatted objects
sand	igraph	Datasets for use with igraph for statistics application

clusters present in Fig. 4. The packages that serve as bridges between the two clusters are ‘qgraph’, ‘keyplayer’, ‘degreenet’, ‘lvm4net’, ‘bipartite’, and ‘intergraph’. It should be noted that the purpose of ‘intergraph’ is to convert data objects between ‘igraph’ and ‘network’. The majority of the packages in the sna-network cluster are statistical analysis packages with several visualization packages that are specifically intended for use with ‘statnet’ and ‘sna’. By contrast, the ‘igraph’ cluster contains more community detection, subgroup analysis packages as well as most of the



R Packages for Social Network Analysis, Fig. 4 SNA packages in R. A directed edge between package A and package B means package A depends on or imports packages B. Nodes size represents log of downloads according to the RStudio CRAN mirror, and node color represents the category of the package

niche SNA packages. All of the social media analysis packages reside in the 'igraph' cluster.

It is important to recognize that just because a particular package might depend upon other packages in the 'sna'-'network' or 'igraph' clusters does not mean it cannot be used with both. In some cases, packages perform operations on matrix objects that are neither 'igraph' nor 'network' objects. In other cases 'intergraph' can be used to convert between data objects. The purpose of the network analysis of packages is to identify the intended purpose and availability of packages in R. There are well over 100 packages of value for SNA application. Those packages that tend to rely on 'sna' and 'network' are typically used for the statistical analysis of networks. Those packages that tend to rely on 'igraph' include community detection and subgroup analysis, better network visualization features, or contain niche functions. While social media analysis packages tend to rely on 'igraph', they are more concerned with importing data from a particular social media application program interface (API) than with any particular analytic package. Data from these packages are primarily used with 'igraph'. However, they can be readily converted to a network data object for use with 'sna' and the related statistical analysis packages.

Key Applications

R is a language for statistical computing and can be used for most social network analysis computation and visualization. R provides access to the latest advances in the statistical analysis of

networks, such as exponential random graph models (ergm) and stochastic actor oriented models (also known as SIENA models). R can also take advantage of advanced visualization packages. The availability of such a diverse and complete set of packages and functions make R an attractive environment for conducting social network analysis.

Future Directions

R is a powerful tool for social network analysis. While its command-line interface has a steeper learning curve, it maintains a great deal of flexibility for the competent analyst. Perhaps the biggest challenge for an analyst is to know where to look for existing code. R hosts certain sites called "CRAN Task Views." These sites provide a general description of a broad class of analytic problems and identify all of the R packages that support that class of analytic problems. As of the writing of this entry, there are 33 CRAN Task View pages, but none devoted to social network analysis, network analysis, or network science. A CRAN Task View, possibly organized based on this entry, would be useful to the community of R users investigating SNA. In the interim, this entry provides an overview of the types of packages available for use in R.

Many R packages provide redundant capability, leaving users a difficult choice on deciding which set of packages they should use. In terms of basic performance, 'igraph' tends to outperform 'sna' for most of the common SNA functions. The 'igraph' package has greater compatibility with a range of other packages, especially for working with social media data or community detection packages. The chief strength of the 'sna' package is compatibility with the majority of packages for statistical analysis of networks. Given the availability of the 'intergraph' package, it may be advantageous for a user to primarily use 'igraph' and convert the 'igraph' objects to 'network' objects for statistical analysis, but otherwise rely on 'igraph' for basic network analysis, visualization, and community detection.

Another issue that analysts may have with using R as an SNA tool is the availability of user support and help files. A quick search of the Internet, using search terms such as "network," "R," and "tutorial," will reveal a number of tutorials posted by the user community. An analyst that is new to R, however, is unlikely to notice the differences between 'igraph' and 'sna'. They may also have difficulty becoming aware of less popular packages. Perhaps the INSNA website will host a web page that provides social network analysts a link to an SNA CRAN Task View or a similar page that identifies SNA related R packages and tutorials, possibly patterned after this entry. This service would allow R developers to communicate the tools that they offer the community. It would allow social network analysts a more direct way to stay abreast of new packages and advanced analytics that are available to them.

Those analysts that use R must be careful to understand the dependencies and conflicts that may occur when loading R packages. As of the writing of this entry, igraph and statnet/sna are not compatible. Users should consider detaching unneeded packages to reduce the risk of conflicts.

Institutions of higher education should consider R methods courses to empower students to use and develop R. It can be frustrating for a student to learn powerful analytic methods, only to find that the tools for accessing them are cost prohibitive after their course ends. With R, analytic tools are free. Students that become proficient in R can maintain their analytic capability for use in other courses, independent research, and in support of clients.

Cross-References

- ▶ Actor-Based Models for Longitudinal Networks
- ▶ Exponential Random Graph Models
- ▶ Flickr and Twitter Data Analysis
- ▶ Network Analysis
- ▶ Siena: Statistical Modeling of Longitudinal Network Data

Acknowledgments This work was supported, in part, by the U.S. Army Research Laboratory. The authors would like to thank the many social network professionals who contributed comments and advice via the SOCNET listserv, personal email, and during the 2016 International Sunbelt Social Networks Conference.

References

Barabási AL, Albert R (1999) Emergence of scaling in random networks. Science 286(5439):509–512
Carley KM (1999) On the evolution of social and organizational networks. Res Sociol Organ 16(0):1–32
Carley K, Krackhardt D (1998) A PCANS model of structure in organizations. In symposium on command and control research and technology (pp. 113–119)
Çetinkaya-Rundel M, Banks D, Rundel C, Clyde M (2016) Statistics with R Specialization. Retrieved from https://www.coursera.org/specializations/statistics
Cleveland WS (1979) Robust locally weighted regression and smoothing scatterplots. J Am Stat Assoc 74(368):829–836
Erdös P, Rényi A (1959) On random graphs, I. Publ Math Debr 6:290–297
Freeman LC (1977) A set of measures of centrality based on betweenness. Sociometry 40:35–41
Freeman LC (1978) Centrality in social networks conceptual clarification. Soc Networks 1(3):215–239
Freeman LC, Roeder D, Mulholland RR (1979) Centrality in social networks: II. Experimental results. Soc Networks 2(2):119–141
Ihaka R, Gentleman R (1996) R: a language for data analysis and graphics. J Comput Graph Stat 5(3):299–314
McCulloh I, Armstrong H, Johnson A (2013) Social network analysis with applications. Wiley, Hoboken
Peng R, Caffo B, Leek J (2016). Data science specialization. Retrieved from https://www.coursera.org/specializations/jhu-data-science
Valente TW (2012) Network interventions. Science 337(6090):49–53
Wasserman SS (1977) Random directed graph distributions and the triad census in social networks†. J Math Sociol 5(1):61–86
Wasserman S, Faust K (1994) Social network analysis: methods and applications, vol 8. Cambridge university press, Cambridge
Wasserman T, Capra E (2007) Evaluating software engineering processes in commercial and community open source projects

R&D (Research and Development) Collaborations

► Innovator Networks

R&D Networks

Jan Kratzer
Technical University Berlin, Berlin, Germany

Glossary

Design Structure Matrix (DSM)	Symmetric matrix that indicates the links/interfaces between decomposed product components
Hierarchical Decomposition	Methods to decompose products in to components and subcomponents following product hierarchies
Systematic Variation	Method that refers to the search for and combination of solutions to design subproblems
Satisficing	Method that refers to the evaluation and selection of alternative solutions and the understanding that searches should not be focused on finding the optimal solution
Discursiveness	Method that refers to a step-by-step, yet iterative, approach to the product development process
Lead User	Person who are ahead of trends and develop and/or modify for their own benefit new products and processes

Definition

Perhaps the first attempts to characterize industrial organizations as networks were contained in the records of the Hawthorne Experiments. Shortly later, the analytic tools to scientifically engage in networks were presented: the sciogram introduced in 1934 and the sociomatrix introduced in 1946. The decades after, with increasing competitions, globalization, and customer individualization, the pressure

on organizational research and development efforts has dramatically increased. This process brought research and development networks (R&D networks) into the picture of academic research.

These early studies also exemplify the multilevel character of such R&D networks. The smallest elements in R&D are humans, so interaction networks among them mold the lowest level. Humans are grouped into teams in aggregation departments and functional divisions, so there are a number of levels within organizations. Further, organizations are embedded in environments with partners, competitors, and customers within an economic, political, and societal system. Hence, one dimension in defining R&D networks is the inherent existence of different levels (Gabbay and Leenders 1999). Another dimension is the nature of nodes and arcs. Nodes may be humans, but also teams and departments. However, nodes may also be product components (Sosa et al. 2004) in R&D networks. In this case, the linking element, the arcs, would be interfaces between product components, whereas interaction between humans, teams, and departments most often refers to some kind of communication. In addition, like all networks, R&D networks can be open or closed, are denoted by strong or weak ties, and have structural features such as centrality. Generally speaking, R&D networks are like other networks and can be defined as other networks, as a number of ties or arcs between a number of nodes, whereby arcs and nodes are embedded in a multilevel hierarchy and can be of different kind, strength, and structural consequence. Research on different levels of analysis has shown that social network ties have an impact on performance: On an individual level, Burt (1992) shows that managers with a high quantity of disconnected, nonredundant social network ties achieve faster promotions to managerial positions. On an organizational level, Tsai and Ghosal (1998) found that social interaction, as a manifestation of the structural dimension, is significantly related to the extent of interunit resource exchange. On an interorganizational level, Gabbay and Leenders (1999) illustrate how a key position in a cohesive clique of an interorganizational network provides a corporate actor with a rent-seeking capacity enabling a business organization to extend its profitability or to accrue valuable resources necessary for corporate success. R&D networks are distinct in being embedded in research and development efforts, and with this the focus is partly on specific nodes such as product components, arcs such as problem-solving communication, and levels of analyses such as R&D teams.

Examples

Following there are two examples of typical R&D network research. The first example is adapted from Journal of Product Innovation Management (Leenders et al. 2007), and the second from Research Policy (Kratzer et al. 2008). The structure of formal and informal networks of teams in R&D projects define the opportunities potentially available to create new knowledge. As many scholars have argued, networks of organizational linkages are critical to a host of organizational processes and outcomes (e.g., Reagans and Zuckerman 2003).

Can organizations exert control and provide structure for R&D activities while at the same time encouraging and managing creative performance? This question was addressed in the publication "Systematic Design Methods and the Creative Performance of New Product Teams: Do They Contradict or Complement Each Other?" (Leenders et al. 2007). Most R&D projects are executed with the R&D team as the organizational nucleus. As a result, managing creativity in R&D thus implies managing the creativity of R&D teams. Besides having to manage creative performance, companies are generally also concerned with improving the efficiency and effectiveness of the R&D process. Modern R&D projects therefore have the need for an approach that can be planned, optimized, and verified. As a consequence, systematic design methods have become widely used in R&D. In this article a conceptual model is developed of the effect of modern design methodology on the creative performance of R&D teams. It is then proposed that four principles underlie modern design methodology: hierarchical decomposition, systematic variation, satisficing, and discursiveness.

These principles affect R&D communication by, respectively, influencing the establishment of subgroups, the frequency of communication, the level of agreement or disagreement in the team, and the level of centralization of communication. These patterns of communication are then related to team-level creative performance. The main conclusion of the entry, is that the design principles work together and need to be considered as an integrated whole: the creative performance of R&D teams can only effectively be managed by using and aligning all four of them.

In another publication "Revealing dynamics and consequences of fit and misfit between formal and informal networks in multi-institutional product development collaborations" (Kratzer et al. 2008), the interplay between communication networks and product component or design networks is highlighted. The size and complexity of most multi-team R&D project structures characterize the importance of addressing and defining the interfaces between product sub-components; it is also important to determine if the teams actually interact according to their formally ascribed interfaces, an inevitable requirement for the project to function. Unfortunately, informal communication networks often compete with such aspects of organizations as formal structure (Cross et al. 2002). One of the most consistent findings in the social science literature is that who you know often has a great deal to do with what you come to know (e.g., Szulanski 1996). In multi-team R&D projects, therefore, it would be naive to expect a perfect alignment between design interfaces – the "Design Structure Matrix" – and the informal communication network as Sosa et al. (2004) have shown. The study revealed three important findings: (1) formally ascribed design interfaces and informal communication networks correlate only marginally. The main reason is that informal communication is much more dense than ascribed; (2) although the formally ascribed design interfaces change, the structure of informal communication remains largely stable throughout time; and (3) the most intriguing finding is that this communicational misfit is associated with higher effectiveness, but it negatively impacts the institutional unit's efficiency.

Other Directions and Future Directions

These two examples show that R&D networks do not solely focus on human interaction, but also take the structure of products and processes as systematic methods into account. There are other examples of research on R&D networks reaching beyond the organizational boundaries. Another stream of research is focused on the diffusion of innovation (Rogers 1974), on the identification of certain roles as lead users important to propel R&D efforts (Kratzer and Lettl 2009) and R&D alliances (Hagedoorn 2002) among others. The investigations of R&D networks in the future may study networks increasingly by addressing more thoroughly the multilevel character, may focus more on longitudinal research designs, may apply more sophisticated statistical analytics to capture the dynamics of networks, and finally may close the gap between qualitative and quantitative research designs.

Cross-References

- ▶ Innovator Networks
- ▶ Interorganizational Networks
- ▶ Intraorganizational Networks
- ▶ Networks of Practice
- ▶ Top Management Team Networks

References

Burt R (1992) Structural holes. Harvard University Press, Cambridge

Cross R, Borgatti SP, Parker A (2002) Making invisible work visible: using social network analysis to support strategic collaboration. Calif Manage Rev 44:25–46

Gabbay SM, Leenders RT (1999) The structure of advantage and disadvantage. In: Leenders RT, Gabbay SM (eds) Corporate social capital and liability. Kluwer, Boston, pp 1–14

Hagedoorn J (2002) Inter-firm R&D partnerships: an overview of major trends and patterns since 1960. Res Policy 3:477–492

Kratzer J, Lettl C (2009) Distinctive roles of lead users and opinion leaders in the social networks of schoolchildren. J Consum Res 36:646–659

Kratzer J, Gemuenden HG, Lettl C (2008) Revealing dynamics and consequences of fit and misfit between formal and informal networks in multi-institutional product development collaborations. Res Policy 37:1356–1370

Leenders RTAJ, Van Egelen JML, Kratzer J (2007) Systematic design methods and the creative performance of new product teams: do they contradict or complement each other? J Prod Innov Manag 24:166–179
Reagans R, Zuckerman E (2003) Networks, diversity, and productivity: the social capital of corporate R&D units. Organ Sci 12:502–517
Rogers DL (1974) Sociometric analysis of interorganizational relations: application of theory and measurement. Rural Sociol 39:487–503
Sosa ME, Eppinger SD, Rowles CM (2004) The misalignment of product architecture and organizational structure in complex product development. Manag Sci 50:1674–1689
Szulanski G (1996) Exploring internal stickiness: impediments to the transfer of best practices within the firm? Strateg Manag J 17:27–43
Tsai W, Ghosal S (1998) Social capital and value creation: the role of intrafirm networks. Acad Manag J 41:464–476

Random Networks

► Sources of Network Data

Random Processes

► Probabilistic Analysis

Random Structures

► Probabilistic Analysis

Random Walks

► Legislative Prediction with Political and Social Network Analysis

Ranking

► Misinformation in Social Networks: Analyzing Twitter During Crisis Events

Ranking Methods for Networks

Yizhou Sun[1] and Jiawei Han[2]
[1]College of Computer and Information Science, Northeastern University, Boston, MA, USA
[2]Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, IL, USA

Synonyms

Identify influential nodes; Importance ranking; Link-based ranking; Relevance ranking

Glossary

Global ranking	Objects are assigned ranks globally
Learning to rank	Ranking is learned according to examples via supervised or semi-supervised methods
Heterogeneous information network	Networks that contain more than one type of objects and/or one type of relationships
Homogeneous information network	Networks that contain one type of objects and one type of relationships
Ranking	Sort objects according to some order
Query-dependent ranking	Objects are assigned with different ranks according to different queries
Proximity ranking	Objects are ranked according to proximity or similarity to other objects

Definition

Ranking objects in a network may refer to sorting the objects according to importance, popularity, influence, authority, relevance, similarity, and proximity, by utilizing link information in the network.

Introduction

In this entry, we introduce the ranking methods developed for networks. Different from other ranking methods defined in text or database systems, links or the structure information of the network are significantly explored. For most of the ranking methods in networks, ranking scores are defined in a way that can be propagated in the network. Therefore, the rank score of an object is determined by other objects in the network, usually with stronger influence from closer objects and weaker influence from more remote ones.

Methods for ranking in networks can be categorized according to several aspects, such as global ranking versus query-dependent ranking, based on whether the ranking result is dependent on a query; ranking in homogeneous information networks versus ranking in heterogeneous information networks, based on the type of the underlying networks; importance-based ranking versus proximity-based ranking, based on whether the semantic meaning of the ranking is importance related or similarity/proximity related; and unsupervised versus supervised or semi-supervised, based on whether training is needed.

Historical Background

The earliest ranking problem for objects in a network was proposed by sociologists, who introduced various kinds of centrality to define the importance of a node (or actor) in a social network. With the advent of the World Wide Web and the rising necessity of web search, ranking methods for web page networks are flourishing, including the well-known ranking methods PageRank (Brin and Page 1998) and Hyperlink-Induced Topic Search (HITS) (Kleinberg 1999). Later, in order to better support entity search instead of web page ranking, object ranking algorithms are proposed, which usually consider more complex structural information of the network, such as heterogeneous information networks. Moreover, in order to better personalize search quality, ranking methods that can integrate user guidance are proposed. Learning to rank techniques are used in such tasks, and not only the link information but the attributes associated with nodes and edges are commonly used.

Methods and Algorithms

In this section, we introduce the most representative ranking methods for networks.

Centrality and Prestige

In network science, various definitions and measures are proposed to evaluate the prominence or importance of a node in the network. According to Wasserman and Faust (1994), centrality and prestige are two concepts to quantify prominence of a node within a network, where centrality focuses on evaluating the involvement of a node no matter whether the prominence is due to the receiving or the transmission of the ties, whereas prestige focuses on evaluating a node according to the ties that the node is receiving.

Given a network $G = (V, E)$, where V and E denote the vertex set and the edge set, several frequently used centrality measures are listed in the following:

- Degree centrality. Degree centrality (Nieminen 1974) of a node u is defined as the degree of nodes in the network: $C_D(u) = \Sigma_v A_{u,v}$, where A is the adjacency matrix of G. Normalized degree $\left(C'_D(u) = C_D(u)/(N-1)\right)$ can also be used to measure the relative importance of a node, where N is the total number of nodes in the network and $N-1$ is the maximum degree that a node can have.
- Closeness centrality. Closeness centrality (Sabidussi 1966) assigns a high score to a node if it is close to many other nodes in the network and is calculated by the inverse of the sum of geodesic distance (shortest distance) between the node and other nodes:

$$C_C(u) = \frac{1}{\sum_v d(u,v)}$$

where $d(u, v)$ is the geodesic distance between u and v. A normalized closeness centrality score (Beauchamp 1965) is defined as

$$C'_C(u) = \frac{(N-1)}{\sum_v d(u,v)}$$

where $N-1$ is the possible minimum sum of distances between a node and the remaining N–1 nodes.

- Betweenness centrality. Betweenness centrality evaluates how many times the node falls on the shortest or geodesic paths between a pair of nodes:

$$C_B(u) = \sum_{v<w} \frac{g_{vw}(u)}{gvw}$$

where g_{vw} is the number of shortest paths between v and w and $g_{vw}(u)$ is the number of shortest paths between v and w containing u. A normalized betweenness centrality score is given in Freeman (1977):

$$C'_B(u) = \frac{2C_B(u)}{N^2 - 3N + 2}$$

where $(N^2-3\ N+2)/2$ can be proved to be the maximum value of $C_B(u)$, when u is a center point in a star network.

The readers may refer to Freeman (1978) and Wasserman and Faust (1994) for detailed introduction of these centrality measures.

In Wasserman and Faust (1994), several prestige measures are proposed for directed networks.

- Degree prestige. Degree prestige is defined as the in-degree of each node, as a node is prestigious if it receives many nominations:

$$P_D(u) = d_{in}(u) = \sum_v A_{v,u}$$

The normalized version of degree prestige is

$$P'_D(u) = \frac{P_D(u)}{N-1}$$

where N is the total number of nodes in the network and thus $N-1$ is the maximum in-degree that a node can have.

- Eigenvector-based prestige. In order to capture the intuition that a node is prestigious if it is linked by a lot of prestigious nodes, eigenvector-based prestige is proposed in an iterative form:

$$P(u) = \frac{1}{\lambda}\sum_v A_{v,u}P(v)$$

It turns out that $p = (P(1),\ldots, P(N))'$ is the primary eigenvector of the transpose of adjacency matrix A^T. p is also called eigenvector centrality.

- Katz prestige. In Katz (1953), attenuation factor α is considered for influence with longer length transmissions, and the Katz score is calculated as a weighted combination of influence with different lengths:

$$P_{Katz}(u) = \sum_{k=1} \alpha^k \sum_v \left(A^k\right)_{vu}$$

which can be written into the matrix form:

$$\boldsymbol{P_{Katz}} = \left((I - \alpha A)^{-1} - I\right)'\mathbf{1}$$

where $\mathbf{P_{Katz}} = (P_{Katz}(1), \ldots, P_{Katz}(N))'$, I is the identity matrix, and 1 is an all-one vector with length N. Katz score is also called Katz centrality.

Global Ranking

Along with the flourish of web applications, many link-based ranking algorithms are proposed. We first introduce the ranking algorithms that assign global ranking scores to objects in the network.

PageRank. In information network analysis, the most well-known ranking algorithm is PageRank (Brin and Page 1998), which has been successfully applied to the web search problem. PageRank is a link analysis algorithm that assigns a numerical weight to each object in the information network, with the purpose of "measuring" its relative importance within the object set.

More specifically, for a directed web page network G with adjacency matrix A, the PageRank rank score of a web page u is iteratively determined by the scores of its incoming neighbors:

$$PR(u) = \frac{1-\alpha}{N} + \alpha \sum_{\upsilon} A_{\upsilon u} PR(\upsilon)/d_{out}(\upsilon)$$

where $\alpha \in (0, 1)$ is a damping factor and is set as 0.85 in the original PageRank paper, N is the total number of nodes in the network, and $d_{out}(\upsilon) = \sum_w A_{\upsilon w}$ is the degree of outgoing links of υ. The iterative formula can also be written in the following matrix form:

$$\mathbf{PR} = \frac{1-\alpha}{N}\mathbf{1} + \alpha M^T \mathbf{PR}$$

where M is the row-normalized matrix of A, i.e., $M_{u\upsilon} = A_{u\upsilon}/\sum_{\upsilon'} A_{u\upsilon'}$, and 1 is an all-one vector with length N.

The iterative formula can be proved to converge to the following stable point:

$$\mathbf{PR} = \left(I - \alpha M^T\right)^{-1} \frac{1-\alpha}{N}\mathbf{1}$$

where I is the identity matrix.

PageRank score can be viewed as a stationary distribution of a random walk on the network, where a random surfer either randomly selects an out-linked web page υ of the current page u with probability $\alpha/d_{out}(u)$ or randomly selects a web page from the whole web page set with probability $(1-\alpha)/N$.

Query-Dependent Ranking

Different from global ranking, query-dependent ranking produces different ranking results for different queries.

HITS. Hyperlink-Induced Topic Search (HITS) (Kleinberg 1999) ranks objects based on two scores: authority and hub. Authority estimates the value of the content of the object, whereas hub measures the value of its links to other objects.

HITS is designed to be applied on a query-dependent subnetwork, where the most relevant (e.g., by keyword matching) web pages to the query are first extracted. Then, the authority and hub scores are calculated according to the following two rules:

1. An object has a high authority score if it is pointed by many nodes with high hub scores.
2. An object has a high hub score if it has pointed to many nodes with high authority scores.

Mathematically, the two rules can be represented as two formulas:

$$Auth(u) = \sum_{\upsilon} A_{\upsilon u} Hub(\upsilon)$$
$$Hub(u) = \sum_{\upsilon} A_{\upsilon u} Auth(\upsilon)$$

where A is the adjacency matrix of the subnetwork. The two formulas are calculated iteratively, where normalization is needed after each iteration such that the score summation for each type equals to 1.

By reforming the two formulas into matrix form, we can find the authority score vector is the primary eigenvector of $A^T A$ matrix, and the hub score vector is the primary eigenvector of AA^T matrix.

Note that the authority and hub scores can only be calculated at query time, as the subnetwork needs first to be extracted according to the query. Therefore, efficiency is a major issue of the HITS algorithm.

Topic-Sensitive PageRank. In order to obtain both the offline computation benefit as PageRank and the query-dependent ranking benefit as HITS, topic-sensitive PageRank is proposed in Haveliwala (2002).

The topic-sensitive PageRank is comprised of two steps. In step 1, a biased PageRank score vector is computed for each predefined topic offline, and in step 2, the probabilities that a query belongs to each topic are determined online, and the final query-dependent ranking is a weighted combination of the rankings for each topic.

More specifically, in step 1, let T_j be the web page set for topic c_j, and let p_j be the initial ranking score vector for topic c_j, where $p_j(u) = 1/|T_j|$ if web page $u \in T_j$ and $P_j(u) = 0$;

otherwise, the biased PageRank score for topic c_j is calculated as

$$\mathbf{PR}_j = (1-\alpha)M^T \times \mathbf{PR}_j + \alpha \mathbf{p}_j$$

where M is the row-normalized matrix of adjacency matrix A, as defined in PageRank section, and α is the parameter indicating the weight for the initial ranking vector. Note that in PageRank, the initial ranking score is $1/N$ for all the web pages in the network.

In step 2, for a given query q, the probability that it belongs to each topic c $_j$ is calculated according to the term distribution in each topic:

$$P(c_j|q) \propto P(c_j)P(q|c_j)$$

where $P(c_j)$ is the prior distribution of topic c_j and $P(q|c_j)$ is the probability that query q can be generated in topic c_j according to term distribution in c_j. Then, the query q-dependent importance score for web page u can be calculated as

$$s_{qu} = \sum_j P(c_j|q) PR_j(u)$$

where PR_j (u) is the biased PageRank score for web page u for topic c_j.

Personalized PageRank. In Jeh and Widom (2003), personalized PageRank is proposed and how to scale the computation is introduced. Personalized PageRank aims at calculating biased PageRank score to a personalized query vector q, which is called preference vector:

$$\mathbf{PPR}_q = (1-\alpha)M^T \times \mathbf{PPR}_q + \alpha \mathbf{q}$$

where M is the row-normalized matrix for the network and $\alpha \in (0,1)$ is the parameter indicating the probability a random walk will teleport to the query vector. PPR $_q$ is called the personalized PageRank vector (PPV) for preference vector q.

Different from topic-sensitive PageRank, where the query vectors are fixed for predefined topics, query vectors in personalized PageRank are arbitrary. Therefore, how to compute personalized PageRank efficiently online becomes critical, and the readers may refer to Jeh and Widom (2003) for more discussions.

A similar idea, TrustRank, that is used for ranking web pages according to their trustability is proposed in Gyöngyi et al. (2004), where the query vector is determined by a set of carefully selected trustable websites.

Ranking in Heterogeneous Information Networks

Traditional ranking problem is considered in homogeneous information networks, where the networks contain only one type of objects and the objects are connected via one type of relationships. Recently, ranking algorithms for heterogeneous information networks are proposed, where the networks contain multiple types of objects and/or multiple types of relationships.

ObjectRank. ObjectRank is proposed in Balmin et al. (2004), which aims at ranking the objects according to a keyword-based query in a database. A database is represented using a labeled data graph, $D(V_D, E_D)$, where nodes represented objects from different types and links represented relationships from different types. A schema graph, $G(V_G, E_G)$, is used to describe the structure of the data graph. Each node also contains several attribute-value pairs, which determine a set of keywords each node is associated with.

An authority transfer schema graph, $G^A(V_G, E_G^A)$, is then defined according to the schema graph, where authority transfer rates are given to the edges in the schema graph, that is, a certain link type in the data graph. The rate is specified by domain experts or obtained by trial and error. Afterward, an authority transfer data graph, $D^A(V_D, E_D^A)$, can be derived, where the authority transfer rate between two objects u and υ is defined by

$$M(u,\upsilon) = \begin{cases} \dfrac{w(T)}{d_{out}(u,T)} & \text{if } \mathrm{d}_{out}(u,T) > 0 \\ 0 & \text{if } \mathrm{d}_{out}(u,T) = 0 \end{cases}$$

where T is the type of edge $e = (u,\upsilon)$, $w(T)$ is the authority transfer rate on the type of edges T, and

$d_{out}(u, T)$ is the total number of out edges from u and of type T. After defining the authority transfer data graph and obtaining the new transition matrix M defined on objects, the online query processing is similar to personalized PageRank. For a keyword query k, the system will prepare the query vector q according to the set of objects containing the keyword. If an object u contains the keyword, then $q(u) = 1/N_k$, where N_k is the total number of objects containing the keyword k; otherwise, $q(u) = 0$. Then, the ObjectRank vector for objects given the keyword k is defined as

$$\mathbf{OR}_q = (1-\alpha)M^T \times \mathbf{OR}_q + \alpha\mathbf{q}$$

where α is the parameter indicating the probability a random walk will teleport to the query vector.

PopRank. In Nie et al. (2005), PopRank is proposed to rank web objects by using both web links and object relationship links. The PopRank score vector R_X for objects from type X is defined as a combination of their web popularity R_{EX} and impacts from objects from other types:

$$\mathbf{R}_X = \epsilon\mathbf{R}_{EX} + (1-\epsilon)\sum_Y \gamma_{YX} M_{YX}^T \mathbf{R}_Y$$

where ϵ is the weighting parameter of the two components, γ_{YX} is the popularity propagation factor (PPF) of the relationship link from an object of type Y to an object of type X, and $\sum_Y \gamma_{YX} = 1$, M_{YX} is the row-normalized adjacency matrix between type Y and type X, and R_Y is the PopRank score vector for type Y.

In this entry, a simulated annealing-based algorithm for learning popularity propagation factor γ_{YX} is also proposed, according to some partial ranking lists given by users. Note that PopRank assigns a global score for every object.

Authority Ranking for Heterogeneous Bibliographic Network. In reality, ranking function is not only related to the link property of an information network but also dependent on the hidden ranking rules used by people in some specific domain. Ranking functions should be combined with link information and user rules in that domain. Authority ranking for heterogeneous bibliographic network is proposed in Sun et al. (2009a, b), which gives an object higher rank score if it has more authority.

Without using citation information, as citation information could be unavailable or incomplete (such as in the DBLP data, where there is no citation information imported from CiteSeer, ACM Digital Library, or Google Scholar), two simple empirical rules similar to HITS are proposed to rank authors and venues:

- Rule 1: Highly ranked authors publish many papers in highly ranked venues.
- Rule 2: Highly ranked venues attract many papers from highly ranked authors.

Let X and Y denote the venue type and author type, respectively, and W_{YY} and W_{YX} denote the adjacency matrices for coauthor relationships and author-venue relationships in a bibliographic network; according to Rule 1, each author's score is determined by the number of papers and their publication forums:

$$\mathbf{r}_Y(j) = \sum_{i=1}^{m} W_{YX}(j,i)\mathbf{r}_X(i) \tag{1}$$

At the end each step $r_Y(j)$ is normalized by $\mathbf{r}_Y(j) \leftarrow \frac{\mathbf{r}_Y(j)}{\sum_{j'=1}^{n}\mathbf{r}_Y(j')}$. According to Rule 2, the score of each venue is determined by the quantity and quality of papers in the venue, which is measured by their authors' rank scores:

$$\mathbf{r}_X(i) = \sum_{j=1}^{n} W_{XY}(i,j)\mathbf{r}_Y(j) \tag{2}$$

The score vector is then normalized by $\mathbf{r}_X(i) \leftarrow \frac{\mathbf{r}_X(i)}{\sum_{i'=1}^{m}\mathbf{r}_X(i')}$.

The two formulas will converge to the primary eigenvector of W_{XY} W_{YX} and W_{YX} W_{XY}, respectively.

When considering the coauthor information, the scoring function can be further refined by a third rule:

- Rule 3: The rank of an author is enhanced if he or she coauthors with many highly ranked authors

Adding this new rule, we can calculate rank scores for authors by revising Eq. 1 as

$$\mathbf{r}_Y(i) = \alpha \sum_{j=1}^{m} W_{YX}(i,j)\mathbf{r}_X(j) + (1-\alpha)\sum_{j=1}^{m} W_{YY}(i,j)\mathbf{r}_Y(j) \quad (3)$$

where parameter $\alpha \in [0,1]$ determines how much weight to put on each factor, which can be assigned based on one's belief or learned by some training dataset.

Similarly, we can prove that r_Y should be the primary eigenvector of $\alpha W_{YX}\ W_{XY} + (1-\alpha)\ W_{YY}$ and r_X should be the primary eigenvector of $\alpha\ W_{XY}\ (I - (1-\alpha)\ W_{YY})^{-1}\ W_{YX}$. Since the iterative process is a power method to calculate primary eigenvectors, the rank score will finally converge.

The idea is extended to ranking medical treatments based on medical literature, and an algorithm called MedRank is proposed in Chen et al. (2013).

Proximity Ranking

Different from previous ranking methods that either rank objects according to their global importance or find the important objects that are relevant to a query, ranking objects according to their similarity or proximity to a given object is also important. Note that proximity ranking does not necessarily return highly visible objects in the network.

SimRank. SimRank is proposed in Jeh and Widom (2002) to calculate pairwise similarity between objects in a network based on the link information. The intuition of the similarity model is based on the idea that "two objects are similar if they are related to similar objects." In other words, the similarity between objects can be propagated from pair to pair via links.

For a directed graph $G = (V, E)$, the similarity between two nodes a and b is defined to be 1, if $a = b$, that is, $s(a, b) = 1$ when $a = b$. Otherwise, it is calculated iteratively via the following formula:

$$s(a,b) = \frac{C}{|I(a)||I(b)|}\sum_{i=1}^{|I(a)|}\sum_{j=1}^{|I(b)|} s\big(I_i(a), I_j(b)\big)$$

where C is the damping factor and is set as 0.8 in the paper, $I(a)$ represents the in-neighbors of node a, $|I(a)|$ is the total number of in-neighbors of a, and $I_i(a)$ represents the ith in-neighbor of a.

SimRank can also be applied to bipartite networks, where similarity between one type enhances the quality of the other type alternatively.

It can be shown that SimRank computation on a network G is equivalent to the pairwise random surfer model on a network of G^2. The rank score of a node in G^2 represents the similarity score of a pair of nodes in the original network G. The convergence of the SimRank computation can be guaranteed.

The time complexity of computing SimRank is high, as the similarity score between a pair of objects is dependent on the similarityZ between every other pair of objects. Different algorithms are proposed to fast computing SimRank, such as Li et al. (2010a, b).

PathSim. PathSim (Sun et al. 2011) is designed to evaluate peer similarity between objects in a heterogeneous information network. Different from previous query-based ranking and similarity measure, PathSim is proposed for (1) evaluating similarity between objects in a heterogeneous information network and (2) evaluating similarity in terms of peers between objects.

In heterogeneous information networks, objects can be connected via different types of connections, and similarity with different semantics can be defined using different types of connections. Meta-path, the meta-level connection between objects, is then proposed to systematically capture how objects are connected in a heterogeneous network.

In many scenarios, finding similar objects in networks is to find similar peers, such as finding similar authors based on their fields and reputation, finding similar actors based on their movie

styles and productivity, and finding similar products based on their functions and popularity. A meta-path-based similarity measure, called PathSim that captures the subtlety of peer similarity, is proposed. The intuition behind it is that two similar peer objects should not only be strongly connected but also share comparable visibility. Given a symmetric meta-path P, PathSim between two objects x and y of the same type is

where $p_{x \rightsquigarrow y}$ is a path instance between x and y, $p_{x \rightsquigarrow x}$ is that between x and x, and $p_{y \rightsquigarrow y}$ is that between y and y.

Meta-path-based similarity is a general framework, on which other measures can be defined to evaluate similarity or proximity between objects. For example, Shi et al. (2012) propose a proximity measure between different types of objects:

$$s(x,y) = \frac{2 \times \left|\left\{P_{x \rightsquigarrow y} : P_{x \rightsquigarrow y} \in P\right\}\right|}{\left|\left\{P_{x \rightsquigarrow x} : P_{x \rightsquigarrow x} \in P\right\}\right| + \left|\left\{P_{y \rightsquigarrow y} : P_{y \rightsquigarrow y} \in P\right\}\right|}$$

Learning to Rank

Most of the previously discussed ranking methods are unsupervised. However, in many cases, ranking should be different for different datasets and/or for different purposes. Thus, learning is important to select the best parameters for a parameterized ranking method. For example, the previously mentioned PopRank (Nie et al. 2005) can automatically learn the best popularity propagation probabilities between object types. Besides PopRank, there are several other recently proposed supervised or semi-supervised ranking methods, as introduced below.

Adaptive PageRank. In Tsoi et al. (2003), the authors propose to help administrators alter PageRank scores according to their preference by modifying PageRank equations and introducing constraints.

The administrator of a system may want to intervene the PageRank score, such as modify the page scores to some target scores or establish a predefined ordering on the pages. These constraints can be represented as some linear constraints. At the same time, the administrator wants to find a scoring function that is most similar to the original PageRank scoring function. The problem can then be transformed to a quadratic programming problem with an inequality constraint set. And the parameters can be automatically learned to derive an administrator preferred ranking function.

Learn to Rank Networked Entities (NetRank). In Agarwal et al. (2006), the authors propose to parameterize the conductance values between objects and rank networked entities based on Markov walks with these parameterized conductance value. The goal is to learn those parameters according to a given preference order among objects.

The conductance value between two objects u and υ is defined as the network flow between u and υ:

$$p_{u\upsilon} = Pr(u \rightarrow \upsilon) = p_u p(\upsilon|u)$$

where p_u is the probability that a random surfer stays at node u and $p(\upsilon|u)$ is the transition probability from u to υ.

The conductance value is considered to be parameterized in two ways. First, it can be parameterized according to the hidden communities that the two nodes belong to. Intuitively, edges within the same community have a higher conductance and edges that bridge different communities have a lower conductance. Second, the conductance value can be parameterized according to the edge type that (u υ) belongs to. Intuitively, different types of edges may have different conductance.

Semi-supervised PageRank. A semi-supervised learning framework, called semi-supervised PageRank, is proposed in Gao et al. (2011), which aims at ranking nodes on a very large graph. In the algorithm, the objective function is defined based upon Markov random walk on the graph. The transition probability and the reset probability of the Markov model are

defined as parametric models based on the features on both nodes and edges.

For the objective function, the goal is to find a ranking that is as close to the parametric Markov process stationary probability as possible. At the same time, the constraints indicate the guidance from the users and require that the ranking is as consistent with the user supervision as possible.

It turns out that adaptive PageRank and NetRank are both special cases of the proposed approach.

Similarity Search by Meta-Path Selection. A query-dependent semi-supervised ranking method in heterogeneous information network is proposed in Yu et al. (2012), which aims to find entities with high similarity to a given query entity.

Due to the diverse semantic meanings in a heterogeneous information network that contains multi-typed entities and relationships, similarity measurement can be ambiguous without context. A meta-path-based ranking model ensemble is proposed to represent semantic meanings for similarity queries. Users can provide several sample similar objects while issuing the query, and the algorithm will automatically select the best ranking model according to such hints and dispatch the query to the selected ranking model online.

Key Applications

Ranking methods are important for many applications. For example, ranking is critical for search engine systems, either web search or entity search. It can also be used in entity ranking for applications in a particular domain, such as in a bibliographic database or a medical information system. Proximity ranking turns out to be very useful in recommender systems. Identifying the most influential actors in social networks can help viral marketing. Ranking can also be used for spam detection and trustworthy analysis.

Cross-References

- ▶ Centrality Measures
- ▶ Data Mining
- ▶ Eigenvalues: Singular Value Decomposition
- ▶ Social Influence Analysis
- ▶ Social Web Search

References

Agarwal A, Chakrabarti S, Aggarwal S (2006) Learning to rank networked entities. In: Proceedings of the 12th ACM SIGKDD international conference on knowledge discovery and data mining, KDD'06, pp 14–23. http://doi.acm.org/10.1145/1150402.1150409

Balmin A, Hristidis V, Papakonstantinou Y (2004) Objectrank: authority-based keyword search in databases. In: Proceedings of the thirtieth international conference on very large data bases – volume 30, VLDB Endowment, VLDB'04, pp 564–575

Beauchamp MA (1965) An improved index of centrality. Behav Sci 10:161–163

Brin S, Page L (1998) The anatomy of a large-scale hypertextual web search engine. Comput Netw 30(1–7):107–117

Chen L, Li X, Han J (2013) Medrank: discovering influential medical treatments from literature by information network analysis. In: Proceeding of the 2013 Australasian database conference, ADC'13, Adelaid

Freeman LC (1977) A set of measures of centrality based on betweenness. Sociometry 40:35–41

Freeman LC (1978) Centrality in social networks conceptual clarification. Soc Netw 1(3):215–239. https://doi.org/10.1016/0378-8733(78)90021-7MathSciNet

Gao B, Liu TY, Wei W, Wang T, Li H (2011) Semi-supervised ranking on very large graphs with rich metadata. In: Proceedings of the 17th ACM SIGKDD international conference on knowledge discovery and data mining, KDD'11, pp 96–104. http://doi.acm.org/10.1145/2020408.2020430

Gyöngyi Z, Garcia-Molina H, Pedersen J (2004) Combating web spam with trustrank. In: Proceedings of the thirtieth international conference on very large data bases – volume 30, VLDB endowment, VLDB'04, pp 576–587. http://dl.acm.org/citation.cfm?id=1316689.1316740

Haveliwala TH (2002) Topic-sensitive pagerank. In: Proceedings of the 11th international conference on world wide web, WWW'02, pp 517–526

Jeh G, Widom J (2002) Simrank: a measure of structural-context similarity. In: Proceedings of the eighth ACM SIGKDD international conference on knowledge discovery and data mining, KDD'02, pp 538–543. http://doi.acm.org/10.1145/775047.775126

Jeh G, Widom J (2003) Scaling personalized web search. In: Proceedings of the 12th international conference on world wide web, WWW'03, New York, pp 271–279. https://doi.org/10.1145/775152.775191

Katz L (1953) A new status index derived from sociometric analysis. Psychometrika 18(1):39–43, MATH

Kleinberg JM (1999) Authoritative sources in a hyperlinked environment. J ACM 46(5):604–632, MATH MathSciNet

Li C, Han J, He G, Jin X, Sun Y, Yu Y, Wu T (2010) Fast computation of simrank for static and dynamic information networks. In: Proceedings of the 13th international conference on extending database technology, EDBT'10, pp 465–476. http://doi.acm.org/10.1145/1739041.1739098

Li P, Liu H, Xu J, Jun Y, Du HX (2010) Fast single-pair simrank computation. In: In Proceedings of the SIAM international conference on data mining, SDM'10

Nie Z, Zhang Y, Wen JR, Ma WY (2005) Object-level ranking: bringing order to web objects. In: Proceedings of the 14th international conference on world wide web, WWW'05, pp 567–574. https://doi.org/10.1145/1060745.1060828

Nieminen J (1974) On the centrality in a graph. Scand J Psychol 15(1):332–336

Sabidussi G (1966) The centrality index of a graph. Psychometrika 31:581–603, MATHMathSciNet

Shi C, Kong X, Yu PS, Xie S, Wu B (2012) Relevance search in heterogeneous networks. In: Proceedings of the 15th international conference on extending database technology, EDBT'12, pp 180–191. http://doi.acm.org/10.1145/2247596.2247618

Sun Y, Han J, Zhao P, Yin Z, Cheng H, Wu T (2009) Rankclus: integrating clustering with ranking for heterogeneous information network analysis. In: Proceedings of the 12th international conference on extending database technology (EDBT'09), pp 565–576

Sun Y, Yu Y, Han J (2009) Ranking-based clustering of heterogeneous information networks with star network schema. In: Proceedings of the 15th ACM SIGKDD international conference on knowledge discovery and data mining, KDD'09, pp 797–806

Sun Y, Han J, Yan X, Yu PS, Wu T (2011) Pathsim: meta path-based top-k similarity search in heterogeneous information networks. In: Proceeding of 2011 international conference on very large data bases, VLDB'11

Tsoi AC, Morini G, Scarselli F, Hagenbuchner M, Maggini M (2003) Adaptive ranking of web pages. In: Proceedings of the 12th international conference on world wide web, WWW'03, pp 356–365. http://doi.acm.org/10.1145/775152.775203

Wasserman S, Faust K (1994) Social network analysis: methods and applications. Cambridge University Press, Cambridge, UK

Yu X, Sun Y, Norick B, Mao T, Han J (2012) User guided entity similarity search using meta-path selection in heterogeneous information networks. In: Proceedings of the 21st ACM international conference on information and knowledge management, CIKM'12, pp 2025–2029. https://doi.org/10.1145/2396761.2398565

RDF

Thomas Gottron and Steffen Staab
Institute for Web Science and Technologies, Universität Koblenz-Landau, Koblenz, Germany

Glossary

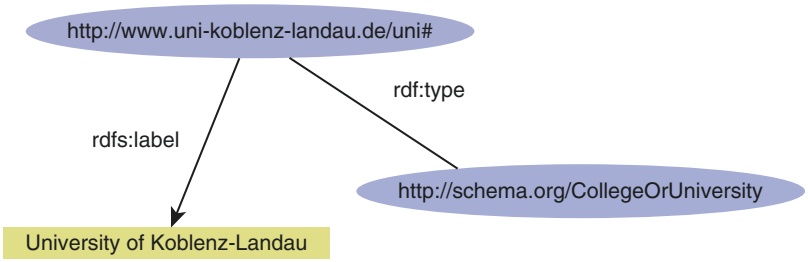
RDF	Resource Description Framework
RDFS	RDF Schema
URI	Uniform Resource Identifier

Definition

The Resource Description Framework (RDF) provides a model for representing data. Its background is set historically in a web environment where it is used for representing information in a device- and platform-independent way. The data model of RDF corresponds to a directed, labelled graph. Technically RDF consists of several W3C recommendations which define its concepts and abstract syntax (Klyne and Carroll). Work on the RDF 1.1 specifications has commenced and reached a draft status at W3C (Cyganiak and Wood).

The core idea of RDF is to represent "things" by URIs. Information is provided by statements about the things and statements are expressed as triples. These triples consist of a subject, a predicate, and an object and express that the subject is in a certain relation (identified by the predicate) with the object. The relations between things can be interpreted and represented in a graph format, where subjects and objects are graph nodes and the predicate is a labelled edge between the nodes.

The choice of using URIs to represent things is intentionally broad in its definition. In RDF a URI can stand for a web resource (e.g., an HTML web document), for a real-world entity (e.g., a person), for abstract constructs (e.g., a user account in an online community), a class concept (e.g., a class type for documents), or to denote the properties connecting entities (e.g., the relation "creator" linking an author to a document he or she



RDF, Fig. 1 An example RDF data graph

wrote). The advantage of using URIs in the triple statements for subject, predicate, and object is that URIs are typically assigned to an authority. For instance, http://west.uni-koblenz.de/staff/Staab# is under the authority of the Institute WeST at Universität Koblenz-Landau; hence, this identifier is not easily to be confused with an identifier of another Steffen Staab, which might be given by https://www.facebook.com/steffen.staab.9#. In this way the URIs provide globally unique identifiers. The object of a triple can also be a literal value (e.g., to denote the name of a person by a String value). Subject and object can furthermore be implemented by so-called blank nodes, which represent nodes in the graph which are neither identifiable by a URI nor are they a literal.

RDF Schema (RDFS) (Brickley and Guha 2004) is an extension to RDF for defining specific vocabularies for RDF applications, i.e., for defining the predicates and class types to be used in a specific application context. To this end, RDFS implements and provides some of the concepts used in RDF itself. For instance, RDFS provides a concrete property for assigning a type class to a URI and defines how to specify a URI to be a property which can be used as predicate in triples. With RDFS it is also possible to provide some information about the vocabulary terms which allow for simple forms of inferencing. For instance, it is possible to define the domain and range of properties or to define a subsumption relation between class concepts.

Example

Assume a scenario where we want to provide information about the university of Koblenz-Landau. We can use the URI http://www.uni-koblenz-landau.de/uni# to represent the university as an institution. We also want to state that the thing identified by this URI is an entity of type College or University as defined by the schema.org vocabulary. Furthermore, we want to state that it can be presented to human users by the String "University of Koblenz-Landau." While the human readable label is given as a literal value, the type is represented as a URI as well. The graph representation of these two statements is shown in Fig. 1. Using the subsumption relations in the schema.org vocabulary, we can infer, for instance, that the described entity is also an educational organization, as http://schema.org/CollegeOrUniversity is modelled to be a subclass of the type http://schema.org/EducationalOrganization.

RDF itself is defined only via an abstract syntax. There are several ways to serialize an RDF data graph into a machine-readable format. The most common serializations are RDF/XML (Beckett 2004), N3 (Berners-Lee and Connolly 2011), N-Triples (Beckett 2013), and Turtle (Beckett 2013). While all serializations are intended for the exchange of RDF-encoded data between applications, some serializations are deemed more human readable (e.g., N3) while other are easier to integrate due to well-established tool chains (e.g., RDF/XML).

Cross-References

- ▶ Linked Open Data
- ▶ SPARQL

References

Beckett D (2004) RDF/XML syntax specification (Revised). http://www.w3.org/TR/2004/REC-rdf-syntax-grammar-20040210/. Accessed 19 Aug 2013
Beckett D (2013) N-Triples. http://www.w3.org/TR/2013/NOTE-n-triples-20130409/. Accessed 19 Aug 2013
Beckett D, Berners-Lee T (2013) Eric Prud'hommeaux and Gavin Carothers. Turtle. http://www.w3.org/TR/2013/CR-turtle-20130219/. Accessed 19 Aug 2013
Berners-Lee T, Connolly D (2011) Notation3 (N3): a readable RDF syntax. http://www.w3.org/TeamSubmission/2011/SUBM-n3-20110328/. Accessed 19 Aug 2013
Brickley D, Guha RV (2004) RDF vocabulary description language 1.0: RDF Schema. http://www.w3.org/TR/2004/REC-rdf-schema-20040210/. Accessed 19 Aug 2013
Cyganiak R, Wood D. RDF 1.1 concepts and abstract syntax (W3C last call working draft 23 July 2013). http://www.w3.org/TR/2013/WD-rdf11-concepts-201307-23/. Accessed 19 Aug 2013
Klyne G, Carroll JJ. Resource description framework (RDF): concepts and abstract syntax. http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/. Accessed 19 Aug 2013

Reality Mining

► Assessing Individual and Group Behavior from Mobility Data: Technological Advances and Emerging Applications

Real-Time Detection of Topics in Twitter Streams

Rania Ibrahim, Ahmed Elbagoury, Khaled Ammar, Mohamed S. Kamel and Fakhri Karray
University of Waterloo, Waterloo, ON, Canada

Synonyms

Apache Giraph; Exemplar-based topic detection; MapReduce

Glossary

Exemplar	Tweet
HDFS	Hadoop distributed file system
LDA	Latent dirichlet allocation
LSA	Latent semantic analysis
NMF	Non-negative matrix factorization
SVD	Singular value decomposition

Definition

Topic detection from Twitter is defined as the task of discovering the underlying key topics that occur in a set of tweets. Additionally, scalable topic detection techniques are topic detection techniques that scale well with extracting topics from huge number of tweets.

Introduction

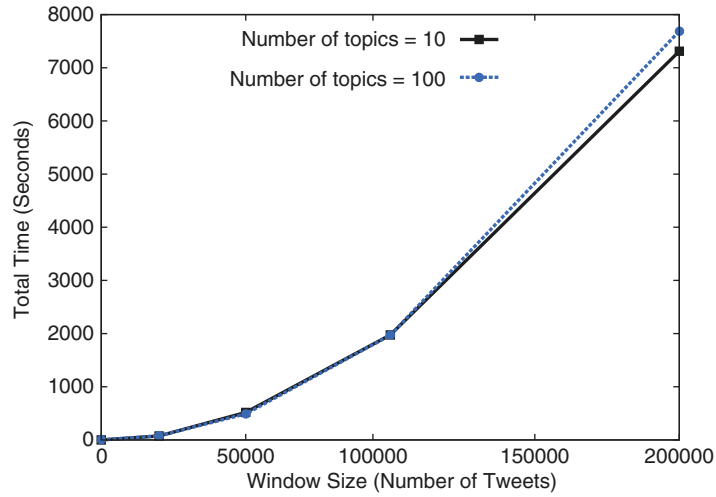
Recently Twitter has become one of the most popular social networks, where users can express themselves by *tweeting* their thoughts in a post of 140 characters at most. The increasing number of users – that reached more than 288 million users in 2014 – who are producing more than 500 million tweets daily (http://www.statisticbrain.com/twitter-statistics/), motivates a lot of celebrities and organizations to post their updates on Twitter. This large number of users and organizations who are producing large amount of data motivates researchers from different areas like machine learning, data mining, and database to develop scalable techniques that can analyze these data and infer useful information. One of the most important tasks is *real-time topic detection*, which is the task of discovering the underlying topics that are discussed by a set of tweets in a real-time manner. This can be useful in discovering natural disasters as early as possible, market analysis, political events like elections or revolutions, and understanding users' sentiments toward some products. Many techniques have been developed for detecting topics in Twitter, these techniques can be categorized into: clustering techniques, which follow the assumption that tweets in the same cluster talk about the same topic; matrix factorization techniques that find the latent basis for the tweets and consider them as the representative topics; and sketch-based techniques, which keeps a sketch for every set of

tweets and use them in topic detection tasks. These techniques represent each topic by a set of terms which could be uncorrelated and may result in topics that cannot be easily understood. On the other hand using exemplar-based approaches has been shown to be effective as in (Elbagoury et al. 2016), this motivated us to develop an exemplar-based topic detection approach (Elbagoury et al. 2015), where each topic is represented by one tweet instead of using keywords from different tweets that can be unrelated to each other. Table 1 shows an example of topics detected by the exemplar-based topic detection and Kmeans. As shown in the table, the topics detected by the exemplar-based approach contain more correlated terms that are more interpretable and can be easily understood. In addition, representative tweets are identified based on their similarities with other tweets, instead of just relying on the tweet's terms alone. Working with the similarity matrix allows using different semantic similarity approaches that can be useful for short text as in (Yan et al. 2012).

Although it was shown that the exemplar-based topic detection approach is superior to other approaches (like Kmeans and non-negative matrix factorization), its scalability needs to be enhanced to handle the large number of daily generated tweets. One factor that affects both the quality of the detected topics and the response time is the size of the time window. On one hand, small time windows will have small number of tweets and hence a lower response time, however, there will not be enough time for topics to be shaped; on the other hand, using large time windows will be enough for topics to be formed but the number of tweets to be processed will be large and the system will have a large delay. It was shown in (Mishne et al. 2013) that 10 min is a suitable time window size for topics to be formed and in the same time to be able to detect topics as soon as they occur. Figure 1 shows the effect of changing window size (the number of tweets) on the response time of the exemplar-based approach (the algorithm is implemented in Java and executed on one machine with 16GB RAM and i7 3.6GHz CPU), we can see that setting the time window to be 10 min, which corresponds to 2,000,000 English tweets using firehose (http://www.internetlivestats.com/twitter-statistics/#trend) (20, 000 English tweets using streaming API) will result in a delay of more than 30 min for both firehose and streaming API which is unacceptable for real-time applications. So, the objective of this work is to enhance the scalability of the exemplar-based approach in order to handle large-size windows efficiently while preserving a high quality of the detected topics. This can be achieved by scaling out the computations to be executed on multiple machines instead of one machine. One way to scale out the computations is using MapReduce. However, MapReduce is not suitable for real-time applications as it needs to perform multiple read and write operations from and to Hadoop Distributed File System (HDFS). To alleviate this problem, we propose a distributed version of the exemplar-based topic detection approach using Apache Giraph, which is an open-source graph processing system. Experimental results on four different datasets show that efficient Giraph-based implementation of the algorithm achieves a speedup up to a factor of 19X while preserving good quality of the discovered topics. There are two versions of time windows which are nonoverlapping windows and sliding windows. In sliding windows, the window

Real-Time Detection of Topics in Twitter Streams, Table 1 Sample topics detected by exemplar-based and Kmeans approaches on US election and super Tuesday datasets (Aiello et al. 2013)

Approach	US Elections	Super Tuesday
Exemplar	RT @AP: AP RACE CALL: Obama wins Vermont; Romney wins Kentucky. #Election 2012	BREAKING NEWT: Gingrich wins Georgia republican primary (AP)
Kmeans	Gonna popping yeahitsope twitter live wins win obama 2012 obama night election romney rt im election 2012	Rt romney http gingrich georgia supertuesday newt wins virginia ohio santorum mitt paul primary win



Real-Time Detection of Topics in Twitter Streams, Fig. 1 Centralized exemplar-based running time

shifts with a step smaller than the window size. As nonoverlapping windows cannot detect topics that span multiple windows, we use a sliding windows model and extend Giraph to handle it efficiently by keeping the same Giraph instance running and modifying the graph structure to delete the old tweets and incorporate the new tweets in each time slot.

Notations: The following notations are used throughout the paper. Scalars are shown in small letters, sets are shown in script letters, vectors are denoted by small bold italic letters, and matrices are denoted by capital letters. In addition:

- For a matrix $A \in \mathbb{R}^{n \times m}$: A_i denotes the i-th row of A, and A_{ij} denotes the (i, j)-th entry of A.
- For a vector $\mathbf{x} \in \mathbb{R}^n$: $\mathbf{x_i}$ denotes the i-th element of $\mathbf{x}$.

Key Points

The contributions of the paper can be summarized as follows:

- Developing and optimizing a solution to scale out the exemplar-based topic detection approach using Apache Giraph.
- Extending the Giraph solution to handle sliding windows efficiently.
- Empirically evaluating the proposed distributed Giraph approach and comparing it against the centralized implementation and related work techniques (Kmeans, LDA, NMF, and LSA) using four datasets.
- Deploy the system to detect topics from real-time Twitter streams and show its scalability.

Historical Background

We start this section by reviewing some of the related work in real-time topic detection and topic detection that led to the development of exemplar-based topic detection.

Real-Time Topic Related Work

Real-Time Topic Detection Using Clustering

Several proposed solutions handle scalable topic detection in Twitter using clustering where the assumption is that the tweets that fall in one cluster talk about the same topic (Ibrahim et al. 2016). The work reported in (Zhang et al. 2013) belongs to this category, where it developed a system that performs distributed online clustering. The system clusters the new arriving tweets either by creating new clusters or by assigning them to existing clusters. Other techniques model the topic detection task in Twitter as a clustering problem over a dynamic graph as in (Agarwal et al. 2012; Lee et al. 2014; Yuan et al. 2013). In Agarwal et al. (2012) the system is modeled as a

graph where each node represents a keyword. Two keywords are connected if they co-occur together in at least k tweets. The goal of the system is to find dense clusters in this dynamic graph where each cluster represents the keywords of a topic. The objective of the approach in (Lee et al. 2014) is to build a system that detects events in a dynamic streaming data and keeps track of the events' changes. The system constructs a network where each node represents a user's tweet, and two tweets are connected with an edge weighted with their time-based Jaccard similarity if this similarity value is above a certain threshold. The approach uses a DBSCAN like clustering technique to cluster the graph where the events are identified as clusters of tweets. In (Yuan et al. 2013) a clustering technique for streaming graphs that can track cluster's evolution is proposed. The system also maintains at each time instance t, two hash tables that contain clustering information for each tweet. Tables in two different time slots are compared to find out which clusters have grown, shrunk, created, or removed.

Sketch-Based Real-Time Topic Detection

Other scalable topic detection techniques for Twitter are based on maintaining a sketch of the streaming data and utilizing this sketch to detect topics in Twitter like (Budak et al. 2013; Li et al. 2012, 2013; Xie et al. 2013). Xie et al. (2013) detects bursty topics in a real-time manner. The approach starts by maintaining a sketch of the data, which contains the total number of tweets, the number of occurrence of words, and the number of co-occurrence of words pairs. The paper adapts exponential moving average to measure these quantities and uses the change in these quantities to capture the topics' burst. Each tweet is modeled as a mixture of multiple latent topics, where each topic has a fixed distribution over words. Using the maintained values, the paper infers these distributions. Also the paper proposes a dimensionality reduction technique by hashing distinct values into buckets and drawing the topic distribution over buckets instead of words. In addition, (Budak et al. 2013) adds the geographic constraint, as it wants to detect trending topics related to geographic locations. It starts by describing the characteristics of geographical trending topics (geotrends) it wants to detect, which are: (1) geo-trend's location has to be a popular location in the current sliding window, (2) probability of topic x occurring in location y needs to be above a threshold θ, and (3) probability of location y containing topic x needs to be above a threshold σ. To do this efficiently, the paper constructs a Location-StreamSummary-Table which keeps track of popular locations' counts, where each location cell has $StreamSummary_{y_i}$ list which contains topics that occur in location y_i with a probability above θ. Also, to keep track of the probability mentioned in point (3), the system keeps another table Topic-StreamSummary-Table, which keeps track of each topic's count, and the locations associated with the topic that has probabilities above σ. The paper also drives memory and time requirements, where the memory requirement is shown to be sublinear and the time requirement is shown to have an amortized running time. Moreover, (Li et al. 2013) monitors a certain topic by finding the keywords related to it. The used sketch in this paper is the topic's keywords. The paper assumes that topics' classifiers exist, and we need to construct a sampling tweets technique and then run the topic's classifier on the sampled tweets. The sampling process is composed of two steps either using the streaming API, or sampling keywords from tweets and using the filtering API (Twitter API that retrieves tweets containing the provided keywords) with the sampled keywords. After that, the paper extracts the significant keywords of the tweets that belong to this topic and uses these keywords to monitor the topic. The previous steps are done in an iterative manner to detect new keywords that are related to the topic. Finally, (Li et al. 2012) also uses keywords as a sketch of the topic. The paper starts by crawling tweets that have crime keywords. It identifies the crime keywords iteratively by first selecting a small set of keywords manually and then automatically observing the coming tweets to find more related terms. After that, the tweets are passed to a crime classifier, which decides if the tweet is about crime or not using its features. If the tweet contains hashtag, url, time, or location, then

the approach extracts the events related features from it like time and location. The approach also performs user location prediction from the tweet's location, the tweet's text, or the user's profile location. Finally, the paper ranks the tweets using content and user features.

Real-Time Streaming Systems

Several techniques for topic detection depend on computing the similarity matrix between all pairs of tweets. Therefore, several techniques are proposed in the literature to compute similarity matrix in a scalable manner like (Pantel et al. 2009). Pantel et al. (2009) calculates similarity matrix between all pairs of points using map only Hadoop job. The approach first constructs an inverted index of the features and if two data points share a nonzero entry, it then computes the similarity of these data points. In detail, each mapper computes the pairwise similarity between all the points and subset of them. However, this approach assumes that all the points can fit in a single machine.

Other real-time streaming systems are developed in Twitter for different tasks like related query suggestion (Mishne et al. 2013) or revenue, user services, search, and content discovery (Toshniwal et al. 2014). (Mishne et al. 2013) proposes a real-time query suggestion system that can make suggestions related to breaking events in less than 10 min. The first solution is to use Hadoop. However, this solution is shown to have a bad response time due to log bottleneck. The second solution is to use in-memory structures. Tweet streams and query streams first get processed by statistic collector and their statistics is stored in in-memory stores. Then all of these stores are used for the ranking algorithm, which writes its result to HDFS. Finally, a cache loads data from HDFS and replies to client queries. In addition, there is a background process that runs over older data to keep track of slower queries. Moreover, (Toshniwal et al. 2014) explains the usage of Storm system in Twitter. The paper starts by explaining Storm, which is in-memory system that consists of a stream of tuples going through a topology. Edges represent data flow and nodes have two types, which are spouts and bolts. Spouts are tuple sources, while bolts process their input tuples, generate tuples, and pass them to the next bolt. Storm has a nimbus, zookeeper, supervisor, and workers to manage the work between nodes and to ensure fault tolerance. In Twitter, Storm is used for revenue, user services, search, content discovery, and developers can visualize Storm's operations.

Topic Detection Related Work

Recently, a lot of scalable topic detection approaches have been developed. This section summarizes some of this work, which can be categorized into four groups which are: clustering, matrix factorization, probabilistic topic modeling, and sketch-based techniques. We also shed the light on some real-time streaming systems.

Topic Detection Using Clustering

Kmeans is one of the most popular clustering techniques which aims to partition n samples into k clusters iteratively, by initially selecting k random points as centroids. Then, assigning each data point to the nearest centroid and recomputing each cluster centroid as the average of the assigned points. The last two steps are repeated until the centroids or the data points' clusters are not changed. For the purpose of detecting topics: Each document is represented by the term frequency-inverse document frequency (tfidf) scheme and the number of topics to be discovered is used as the number of clusters k. One of the limitations of Kmeans is that it doesn't detect arbitrary shape or arbitrary density clusters.

Topic Detection Using Matrix Factorization

Several techniques were proposed in the literature for matrix factorization. The problem of matrix factorization is defined in the context of topic detection as factorizing a data matrix X, where rows of X represent documents and columns of X represent terms. Each entry (i, j) of X corresponds to how related document i to term j. Matrix factorization techniques like (Landauer et al. 1998) factorizes the matrix X into the multiplication of three matrices $U\Sigma V^T$. This can be interpreted as

matrix U representing the relations between the documents and the topics, Σ representing the relations between the topics and V^T representing the relations between the topics and the terms. Therefore, documents can be assigned to topics based on matrix U and topics can be visualized according to their terms relations using matrix V^T. However, LSI has two problems, which are: (1) U and V^T can have negative values and thus interpreting them is difficult; and (2) the detected topics are latent and don't correspond to real documents or have clear meaning.

To overcome the aforementioned problem, another category of matrix factorization techniques were proposed like nonnegative matrix factorization (NMF), where it factorizes the matrix X into the multiplication of two matrices W and H. Additionally, NMF enforces a restriction on the values of W and H to ensure that the values are nonnegative and thus H and W can be easily interpreted. Still, NMF didn't solve the problem of using latent topics.

Probabilistic Topic Models

Probabilistic topic models are generative models that consider each document as weighted mixture of a set of topics, where each of these topics has a hidden distribution over terms (Blei et al. 2003). Each document has a hidden distribution over the set of topics, and each word in the document is generated from one of the topics, based on the document over topics and topic over words hidden distributions. LDA proposed in (Blei et al. 2003) is one of the well-known probabilistic topic modeling approaches that infers the hidden distributions by observing the terms in each document. However, it was reported in (Mehrotra et al. 2013; Newman et al. 2011) that LDA has a problem with the data sparsity in short text.

Exemplar-Based Topic Detection

This section discusses the exemplar-based topic detection approach presented in (Elbagoury et al. 2015).

The basic idea behind this algorithm is to represent each of the detected topics using one tweet which is the most descriptive one for this topic. This tweet (i.e., the exemplar) is much easier to be interpreted by the user as it contains related terms and it represents a topic that is of direct importance to the user.

Problem Formulation

Given a set of tweets T of size n, our goal is to detect the under lying topics in this set and represent each of them using only one tweet (exemplar). The selection criterion should be able to choose a tweet for each topic such that each tweet is descriptive for one topic and discriminates this topic from the other topics.

Exemplar Selection Criterion

The criterion used in this algorithm is based on the following observation. A tweet which is similar to a set of tweets and dissimilar to the rest of the tweets is a good topic representative. This can be formulated by defining a similarity matrix $S_{n\times n}$ where S_{ij} is the similarity between tweet t_i and tweet t_j. The distribution of similarities between each tweet t_i and the rest of the tweets can be classified into three cases:

- Tweet t_i is similar to many tweets. Therefore, its similarity distribution will have low sample variance.
- Tweet t_i is very similar to a set of tweets and less similar to the others. Therefore, its similarity distribution will have high sample variance.
- Tweet t_i is not similar to most of the other tweets. So, its similarity distribution will have low sample variance.

The tweets that fall in the second case are good candidates for representing topics, as each tweet is very similar to a set of tweets and therefore it can capture their underlying topic. On the other hand, each of these tweets is different from the rest of the tweets which means it can distinguish between its topic and the rest of the topics. This suggests using the variance of the similarity distribution of each tweet as a criterion for selecting topics representatives, where the sample variance of the similarities for each tweet t_i is computed as follows:

$$\mathrm{var}(S_{:i}) = \frac{1}{n-1}\sum_{j=1}^{n}\left(S_{ij} - \mu_i\right)^2,$$

where μ_i is the mean of the similarities of tweet t_i: $\mu_i = \frac{1}{n}\sum_{j=1}^{n} S_{ij}$.

Exemplar Selection Algorithm

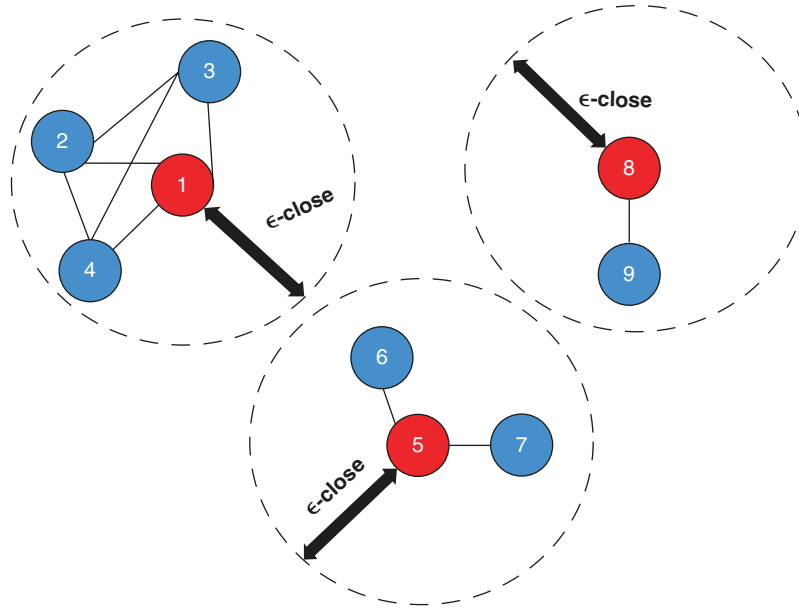
Choosing exemplars for detected topics can be done in an iterative manner by choosing the tweet with the highest variance in each iteration as an exemplar for a topic. However, this approach does not guarantee that the selected tweets are talking about different topics. So after choosing each exemplar, we have to remove its effect to ensure that no more tweets about the same topic will be selected as exemplars of another topics. One way to remove the effect of an exemplar t_i is to disqualify the tweets that are ϵ close to it from being exemplars, and consider the tweet that has the highest variance of similarity and is not ϵ close to t_i as the exemplar of the next topic. Figure 2 shows an example where each node represents a tweet and its ϵ close tweets are within the dotted circle. Tweets are sorted descendingly based on the variance of their similarities with the rest of the tweets and each tweet is labeled in accordance by this order. In this example, after choosing tweet 1 as the first topic exemplar, tweets 2, 3, and 4 are not chosen as exemplars of new topics as they are very close to tweet 1 and do not represent new topics. Tweet 5 which is the tweet with the highest variance of similarity and not ϵ close to tweet 1 is chosen as an exemplar of a new topic. Similarly, tweets 6 and 7 are not chosen as exemplars while tweet 8 is chosen.

Algorithm 1 Exemplar-Based Topic Detection

Data: T set of tweets, k number of topics and ϵ similarity threshold

Result: ε set of k tweets each representing a topic

```
1  S ← similarity(T, T)
2  v ← zeros(n)    // Vector of size n
3  i ← 1
4  while i ≤ n do
5    v_i ← var(S_:i), i ← i + 1
6.  v ← sort(v, "descending")
7.  topic ← 1, i ← 1
8  while (topic < k) do
9.   ε.add(v_i), i ← i + 1
10   while similarity(v_i, ε(topic)) ≥ ϵ do
11     i ← i + 1
12   topic ← topic + 1
```



Real-Time Detection of Topics in Twitter Streams,
Fig. 2 Exemplar-based illustration example

The set of exemplars ϵ is constructed iteratively using the following objective function, at each iteration i:

$$\max_{t_i \in \mathcal{T}} \text{var}(S_{:i}), \text{s.t.} S_{ij} \leq \epsilon \forall t_i, t_j \in \varepsilon \quad \text{and} \quad i \neq j$$

This objective function can be solved by iterating through the tweets in a descending order of the variance and considering the first tweet that is not ϵ close to t_i as the exemplar of the next topic. Algorithm 1 shows the pseudocode of the approach.

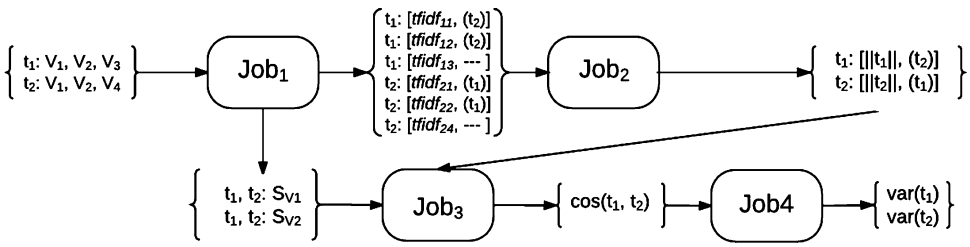
Scaling Exemplar-Based Topic Detection

This section shows the details of scaling out the exemplar-based topic detection approach. We start by discussing different options and justify our choice of Giraph, then we explain our Giraph implementation in detail. The cosine similarity is used in both models, as it is reported to be more suitable for text.

MapReduce is a programming framework which eases the development of parallel applications that process large volume of data. Usually, these applications run on a cluster of commodity machines, where MapReduce handles data partitioning, task scheduling, and fault tolerance. The exemplar-based approach can be modeled as a MapReduce workflow that consists of four jobs. First job_1 reads the tweets text from HDFS as an input and it outputs two files. The first one contains key-value pairs, where each key represents tweet id and the value contains the tf-idf weight of each term in this tweet and the ids of the tweets that share this term with this tweet. The second file contains key-value pairs too, where each key consists of the ids of two tweets that share at least one term, and the value is the similarity between these two tweets based on one shared term only (as both tweets have this term only). Then job_2 reads the first output of job_1 and outputs the norm of each tweet and the list of tweets that share at least one term with this tweet. After that, job_3 reads the second output of job_1 and the output of job_2 and outputs the cosine similarity between all pairs of tweets that share at least one term. Job_4 reads the output of job_3 as its input and computes the variance of the similarities of each tweet as its output. Finally a java component that runs on one machine, reads the outputs of job_3 and job_4, and selects the exemplar tweets. Figure 3 shows the workflow of MapReduce jobs that compute the variance of the similarities for two tweets t_1, that contains terms v_1, v_2, and v_3, and tweet t_2 that contains terms v_1, v_2, and v_4. The figure also shows the inputs and the outputs of each job.

This modeling is similar to the modeling in (Elsayed et al. 2008). However, the approach in (Elsayed et al. 2008) computes the similarity using term frequency weighting scheme only without the document frequency which was shown to be inefficient for down-weighting the less important terms that occur in many documents (Salton and Buckley 1988). In addition, we don't only provide a model that calculates the similarity but that also calculates the variance of the similarities.

Although MapReduce approach is simple, it has some limitations. First, there are four jobs in the workflow, where each job reads its input from the HDFS then writes its output to the HDFS so that it can be read by subsequent jobs. Reading



Real-Time Detection of Topics in Twitter Streams, Fig. 3 MapReduce workflow

and writing from and to HDFS are costly operations, which result in slowing down the computations. Second, MapReduce jobs suffer from the stragglers problem, where the whole job has to wait for one straggler machine to finish that is shown in (Mishne et al. 2013) to slow down the computations. Due to these limitations, it is obvious that MapReduce approach will not be able to detect topics from a large number of tweets in a real time. That is why we decided to use a graph-based system that keeps everything in memory.

There are many graph processing systems like Spark GraphX (Gonzalez et al. 2014), Cassovary (Gupta et al. 2013), and GraphLab (Low et al. 2014). Spark GraphX is a Spark API for graph processing. However, it is still an alpha version and it doesn't support graph mutations, which are needed to support sliding windows efficiently as will be shown. Another graph processing system is Cassovary, which is a graph processing framework for processing large graphs that is implemented in Scala. It is developed by Twitter and contains common graph representations and graph traversal algorithms. However, it doesn't work in a distributed manner. Another remarkable graph processing system is GraphLab, which is a distributed graph processing system that is implemented in C++. GraphLab uses asynchronous model, where there is no global synchronization barrier and vertices can access the latest values of the other vertices and edges. However, the open-source GraphLab version doesn't support graph mutations, which is needed for the sliding windows part. Therefore, our choice of Giraph is based on two factors. First, it supports graph mutations which are needed by the exemplar-based approach as will be shown. Second, it is an open-source project that is supported by an active community.

Giraph Approach

This section shows the details of using Giraph system to scale out the exemplar-based topic detection approach. We first show how to model the problem as a general graph problem and we show how to solve it using Giraph.

The exemplar-based topic detection approach is built based on calculating the variance of the similarities of each tweet t_i with the rest of the tweets, which is calculated as:

$$\mathrm{var}(S_{:i}) = \frac{1}{n-1}\sum_{j=1}^{n}\left(S_{ij}-\mu_i\right)^2 \qquad (1)$$

Let $\mathcal{N}_i$ be the set of tweets that have nonzero similarities with tweet t_i and $\overline{\mathcal{N}}_i$ be the set of tweets that have zero similarities with tweet t_i, then:

$$\mathrm{var}\,(S_{:i}) = \frac{1}{n-1}\left(\sum_{j\in\mathcal{N}_i}\left(S_{ij}-\mu_i\right)^2 + \sum_{j\in\overline{\mathcal{N}}_i}\left(S_{ij}-\mu_i\right)^2\right)$$

As $S_{ij} = 0,\ \forall j \in \overline{N}_i$ and $|\mathcal{N}_i| + \left|\overline{\mathcal{N}}_i\right| = n$. Therefore

$$\mathrm{var}(S_{:i}) = \frac{1}{n-1}\left(\sum_{j\in\mathcal{N}_i}\left(S_{ij}-\mu_i\right)^2 + (n-|\mathcal{N}_i|)\mu_i^2\right) \qquad (2)$$

where, $\mu_i = \frac{1}{n}\sum_{j\in\mathcal{N}_i}\left(S_{ij}\right)$. Based on Eq. 2, to calculate the variance of similarities of each tweet t_i, it is only needed to calculate the similarities with the tweets in the set $\mathcal{N}_i$. To calculate the set $\mathcal{N}_i$, let $\mathcal{W}_i$ be the set of terms that occur in tweet t_i. As S is the cosine similarity measure, we note that:

$$S_{ij} \neq 0 \leftrightarrow \mathcal{W}_i \cap \mathcal{W}_j \neq \varnothing$$

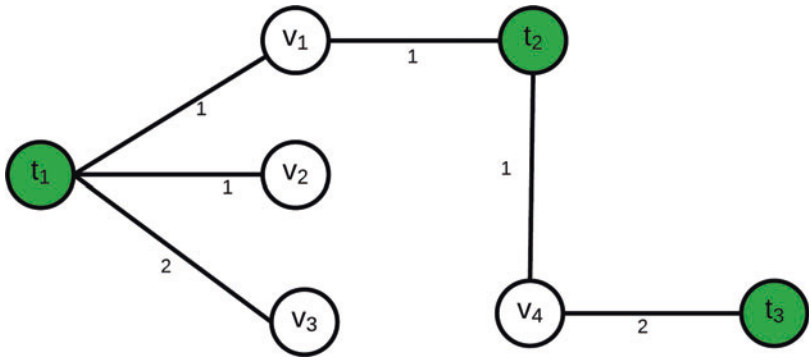
And by definition: $j \in \mathcal{N}_i \leftrightarrow S_{ij} \neq 0$.

Then: $j \in \mathcal{N}_i \leftrightarrow \mathcal{W}_i \cap \mathcal{W}_j \neq \varnothing$. Based on the previous definition, for each tweet t_i, the set $\mathcal{N}_i$ is the set of tweets that share at least one term with t_i, and can be computed as follows:

$$\mathcal{N}_i = \cup_{vj} \in w_i \delta\left(v_j\right) \qquad (3)$$

where $\delta(v_j)$ is the set of all tweets that contain the term v_j.

Based on Eqs. 2 and 3, computing the variance of the similarities of each tweet can be mapped to



Real-Time Detection of Topics in Twitter Streams, Fig. 4 Graph modeling

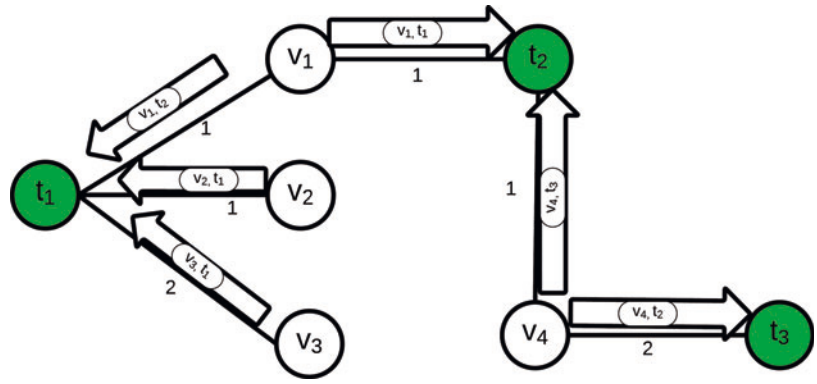
a graph that contains two types of vertices. The first type represents tweets and the second represents terms, and there is a weighted edge between a tweet vertex t_i and a term vertex v_j, $\forall v_j \in \mathcal{W}_i$, where the weight is initially set to the frequency of occurrence of the term v_j in the tweet t_i. Figure 4 shows the graph representation of three tweets, t_1 that contains the terms v_1, v_2, and v_3, tweet t_2 that contains the terms v_1 and v_4, and tweet t_3 that contains the term v_4, we will use this example through the rest of the paper to illustrate the steps of the computations. Based on this representation, each term vertex v_j can compute the set $\delta(v_j)$ locally by using its edges, then it can compute its inverse document frequency idf_{v_j} and update the weights on the edges between it and all the tweets in the set $\delta(v_j)$ with the tf-idf values. After that, each term vertex v_j broadcasts the set $\delta(v_j)$ to each tweet vertex that belongs to the set $\delta(v_j)$, so that each tweet vertex can know the rest of the tweets that share the term v_j with it. As each tweet t_i knows $\delta(v_j)$, $\forall v_j \in \mathcal{W}_i$, it can compute the set $\mathcal{N}_i$ and calculate its similarity values with tweets t_j, $\forall_j \in \mathcal{N}_i$. Finally, after computing the similarities, each tweet t_i can calculate the variance of these similarities var($S_{:i}$). Next subsections show the details of implementing this model using Giraph and further optimizations. In the rest of the paper, we will refer to the tweet vertex and the term vertex as tweet and term respectively when it is not ambiguous to do so.

Giraph Modeling

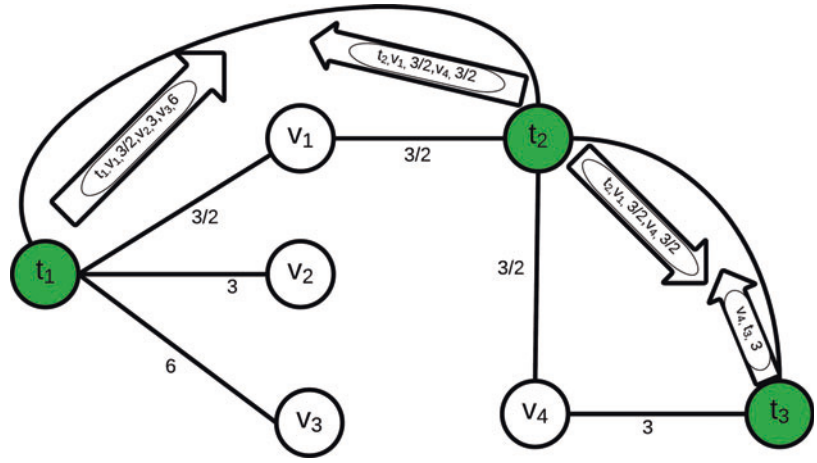
To implement the aforementioned graph modeling in Giraph, we first need to have two types of vertices while Giraph has only one Vertex class. So, to distinguish between tweet vertices and term vertices, positive ids are assigned to tweets, while terms are assigned nonpositive ids. The computation of the similarities and the variance of tweets are performed in three supersteps.

- Superstep 0:
  - Term vertices compute the set $\delta(v_j)$ which contains all the tweets vertices that are connected to the vertex v_j, then it broadcasts this list to each member in this list (all the tweets vertices it is connected to). This list is customized before being sent to each tweet t_i by removing the vertex t_i from it in order to reduce the size of the message. Figure 5 illustrates superstep 0 on the example shown in Fig. 4, where term v_1 computes the list $\delta(v_1) = \{t_1, t_2\}$, then it broadcasts it to each member.
- Superstep 1:
  - Based on messages sent in superstep 0, each tweet vertex t_i calculates inverse document frequency $\text{idf}_{v_j} = \frac{n}{|\delta(v_j)|}$, where n is the number of tweets. Then it updates the edge's weight between it and each term vertex v_j to be the tf-idf$_{ij}$ = tf$_{ij}$ × idf$_{vj}$. Each tweet vertex t_i can determine the set of its neighboring tweets $\mathcal{N}_i$ and sends its tf-idf feature vector ($\boldsymbol{x_i}$) to all elements in the set $\mathcal{N}_i$. As shown in Fig. 6, tweet t_1 sends a message to tweet t_2 containing term id and tf-idf weight of each term that appears in t_1.

R



Real-Time Detection of Topics in Twitter Streams, Fig. 5 Giraph superstep 0



Real-Time Detection of Topics in Twitter Streams, Fig. 6 Giraph superstep 1

- Superstep 2:
  - Each tweet node t_i receives the tf-idf vector $\boldsymbol{x_j}$, $\forall j \in \mathcal{N}_i$, then it calculates the cosine similarity S_{ij}, $\forall j \in \mathcal{N}_i$. Finally it calculates the variance of its similarities S_{ij}, $\forall j \in \mathcal{N}_i$. Finally it calculates the variance of its similarities using Eq. 2 and writes it to HDFS. In addition, for each tweet t_i the ids of all tweets that have similarity values with t_i above certain E are written in the HDFS as conflict tweets to this tweet. Conflict tweets are tweets that are similar to this tweet, which means they cannot both be detected as two different topics.

Finally, a centralized component sorts the tweets based on the variance of their similarities and selects the top tweets with the highest variances that don't conflict with each other as the topics.

Giraph Optimizations

Running this implementation on one time slot containing 20,000 tweets results in 282 s which is a large delay; therefore, one optimization is applied in superstep 1. In order to calculate the cosine similarity between a pair of tweets, each tweet needs to know from the other tweet only its tf-idf weights of the terms shared between them and its norm. Therefore, each pair of tweets doesn't need to share the tf-idf weights of non-shared terms. After applying this optimization, the running time of Giraph approach on 20,000 tweets is reduced to 163 s.

Another optimization is to let each term vertex v_j send its tf-idf weights with all the tweets in the set $\delta(v_j)$ along with the set itself in superstep 0 instead of sending the set alone. By doing this there is no need to send the tf-idf weights in superstep 1 which saves 4 bytes integer

of term id during the communication between pairs of tweets. Applying this optimization, the running time on 20,000 tweets is reduced from 163 s to 84 s.

The final optimization is to complete all the computation within Giraph without the need to write an intermediate output to the HDFS, read this output, and process it in a centralized component. We use Giraph aggregator for this purpose, where Giraph has a two-level hierarchy of aggregators. The first level is on each worker where the values are partially aggregated on each machine then the partially aggregated value is forwarded to the second level on the master machine to obtain the final aggregated value. The aggregator approach is described as follows:

- When a worker aggregator receives a new tweet's variance message, it scans its sorted list of candidate tweets, if the new tweet conflicts with a tweet with higher variance, then it will neglect the new tweet and keep the list unchanged. However, if it doesn't, the aggregator will add the new tweet to the list in the right position that maintains the list sorted then it will remove the tweets that have a lower variance and conflict with the new tweet, if there are any. If the size of the list exceeds m, it will only keep the top m tweets.
- When the master aggregator receives worker aggregators' lists, it will combine and sort them. Finally it will choose the top variance tweets that don't conflict with each other as topics.

The worker aggregator runs in $O\left(\frac{n}{w}m\right)$ and the master aggregator runs in $O(wm\log(wm))$ where w is the number of workers. Therefore, the total aggregator runs in $O\left(\frac{n}{w}m + wm\log(wm)\right)$. The parameter m should be chosen carefully, because a small value of m could decrease the topic quality, as each worker aggregator makes a local decision by reporting its m tweets as candidate exemplars. However, its m tweets could conflict with other worker aggregators' tweets and therefore redundant topics will be detected as different topics, which harm the quality of the detected topics. On the other side, large values of m increases the message size that worker aggregators send to the master aggregator. Therefore, the parameter m has a great impact on the results and needs to be tuned carefully.

Real-Time Detection of Topics in Twitter Streams, Table 2 Running time, storage, and communication bounds of Girpah algorithm

	Running time	Communication cost	Memory usage
Superstep 0	$O(d_v)$	$O\left(d_v^2\right)$	$O\left(d_v^2\right)$
Superstep 1	$O(d_{tt})$	$O(d_{tt})$	$O(d_{tt})$
Superstep 2	$O(d_{tt})$	$O(1)$	$O(d_{tt})$

Giving that d_v is the average degree of each term vertex, d_{tt} is the average degree of each tweet vertex with the other tweets, and d_{tv} is the average degree of each tweet vertex with term vertices. Due to the tweet length limitation (140 characters), d_{tv} is a small constant number. Table 2 shows the running time, storage, and communication bounds on Giraph algorithm per vertex, where the memory usage is calculated as the messages size generated by the vertex plus the memory used by the vertex compute method. As shown in the table, the running time is linear. However, the communication cost and memory usage are quadratic.

Memory Optimized Giraph Approach

The memory complexity of the previous approach is quadratic in the average term degree $O\left(d_v^2\right)$ which does not scale for a large number of tweets. Thus, in this section, another Giraph approach is proposed to reduce the memory complexity to be linear so that the approach can handle large volume of tweets. The quadratic complexity of memory is due to the step in which each term vertex shares the set of its neighboring tweets with the neighboring tweets. This step can be avoided by reformulating Eq. 1 to be:

$$\mathrm{var}(S_{:i}) = \frac{1}{n-1}\left(\sum_{t_j \in \mathcal{N}_i} S_{ij}^2 - 2\mu_i \sum_{t_j \in \mathcal{N}_i} S_{ij} + n\mu_i^2\right) \tag{4}$$

where $\mu_i = \frac{1}{n}\sum_{v_j \in \mathcal{N}_i} S_{ij}$. As the set $\mathcal{N}_i$ can be expressed as $\cup_{v_k \in w_i}\delta(v_k)$. Then, the term $\sum_{t_j \in \mathcal{N}_i} S_{ij}$ in Eq. 4 can be computed as the sum of

$\sum_{t_j \in \delta(v_k)} S_{ij}, \forall v_k \in \mathcal{W}_i$ and with considering that if two tweets t_i and t_j share more than one term, then only the term vertex with highest id, v_k, calculates the $\sum_{t_j \in \delta(v_k)} S_{ij}$. The terms $\sum_{t_j \in \mathcal{N}_i} S_{ij}^2$ and μ_i can be computed in a similar manner. Thus, to compute Eq. 4, some term vertices can compute parts of $\sum_{t_j \in \delta(v_k)} S_{ij}$ and then the tweet vertex adds these parts together without the need to send messages of quadratic order. To achieve this every term vertex v_k needs the feature vector of the tweets $t_i \in \delta(v_k)$. This information is obtained as follows:

- Each term vertex broadcasts its degree to its neighboring tweets.
- Using the term vertex degree, the tweet vertex calculates each term tf-idf weight, constructs its tf-idf vector, and sends it to all its neighboring term vertices.
- Each term vertex v_k calculates the similarity between each pair of tweets tf-idf vectors, and then it sends for each tweet the values $\sum_{j \in \delta(k)} S_{ij}$ and $\sum_{j \in \delta(k)} S_{ij}^2$. Also, term vertex sends to each tweet its top conflicting tweets (the ones with ϵ close similarity to it). If two tweets share more than one term, then only the term vertex with highest id calculates their pairwise similarity.
- Each tweet vertex adds the sum of similarities and the sum of square of similarities it receives from its term vertices and using these two values, it can compute its variance of similarities using Eq. 4. In addition, each tweet vertex unions the conflict lists it receives from its term vertices. Finally, the tweet sends its variance and conflict list to the worker aggregator and the rest of the approach.
- Each worker aggregator accumulates the tweets it receives and then sends them to the master aggregator. Finally, the master aggregator performs the same logic explained in the previous section.

Table 3 shows the bounds on running time, communicate cost, and memory usage of the memory optimized Giraph approach. As shown in the table, the communication cost and memory usage are linear, however the running time is quadratic. As Giraph is paralleling the running time of the vertices across the distributed machines, the overall running time is reduced to $O\left(\frac{n}{w} d_v^2\right)$.

Real-Time Detection of Topics in Twitter Streams, Table 3 Running time, storage, and communication bounds of memory optimized Girpah algorithm

	Running time	Communication cost	Memory usage
Superstep 0	$O(d_v)$	$O(1)$	$O(d_v)$
Superstep 1	$O(1)$	$O(1)$	$O(1)$
Superstep 2	$O\left(d_v^2\right)$	$O(Cd_v)$	$O(d_v)$
Superstep 3	$O(1)$	$O(1)$	$O(Cd_{tt})$

Sliding Windows

Time windows that shift with a step smaller than the window size are called sliding windows. By using sliding windows, topics that span multiple windows and are not frequent in any of them can still be detected. In the aforementioned Giraph model, the sliding windows are handled by running different instance of Giraph for each time slot. However, one drawback of handling sliding windows in such a way is the overhead of initializing multiple Giraph instances and also the tweets that exist in two consecutive time slots are loaded twice. This section describes how to extend the Giraph model to handle the sliding windows without the need to run different instances for each time slot.

The basic idea to handle sliding windows is to keep the Giraph instance running and only load the part of the graph that changes between time slots. Given a time window of size l and a shifting of sliding window of size s. At each time slot, s tweets are removed and s tweets are inserted. Instead of removing vertices from Giraph and adding new ones, the identities of s vertices that are supposed to be removed will be changed so that they can represent the new tweets instead of the old ones. Changing the identity of a tweet vertex means updating its connections with the term vertices such that it becomes connected only to the terms that are contained in the new tweet. This can be done by removing the

connections between the tweet vertex and the term vertices and adding connections to the terms that are in the new tweet. Applying this model will save the cost of initializing Giraph multiple times and the cost of deleting and inserting vertices.

In order to maintain load balancing between worker machines, the file containing the new s tweets is divided into w parts, where w is the number of worker machines, and then each worker arbitrary selects one vertex from the vertices it has to read the worker corresponding part of the input file, then this vertex will update the topology of the graph.

Experimental Results

Experimental Setup

Experiments are conducted using Giraph 1.0.0, Hadoop 1.0.4, and GraphLab v2.2. Four datasets are used which are:

- **US Election Dataset:** which contains 578,837 tweets. It is divided into 26 time slots, where each time slot has around 20,000 tweets.
- **Super Tuesday Dataset:** which contains 353,650 tweets. It is divided into eight time slots, where some time slots contain around 20,000 tweets and others contain around 60,000 tweets. However, there is one-time slot that contains around 80,000 tweets, we have removed this time slot from our evaluation as the native implementation's memory footprint for it is 25GB.
- **120 M Tweets Dataset:** which contains around 120 million tweets covering more than 500 events and collected by (McMinn et al. 2013). As we were able to retrieve only portion of the tweets, we were unable to use the ground truth and therefore only the running time is reported.
- **2 M Streaming API Tweets:** which contains two million English tweets collected using the streaming API.

Four types of experiments are conducted, which are:

- **Validating Topic Quality Experiments:** The purpose of this experiment is measuring the quality of the topics detected by the centralized native implementation and Giraph distributed implementations. This experiment is performed on a cluster of three machines, each contains 16GB RAM and i7 3.6GHz CPU using US Election and Super Tuesday datasets. Sliding window is not used in these experiments as these datasets have nonoverlapping time slots. In addition, the effect of changing m of the aggregator is measured on the quality of the topics. The quality of the detected topics is measured using:
  - **Topic recall:** Percentage of topics successfully retrieved.
  - **Term precision:** Number of correct keywords in the detected topics over the total number of keywords.
    - **Term recall:** Number of correct keywords over the total number of keywords in the ground truth topics.
    - **Running time:** Total time needed to detect the topics.

Topic precision is not used as not all the topics covered by Twitter appear in news sources as in (Aiello et al. 2013).

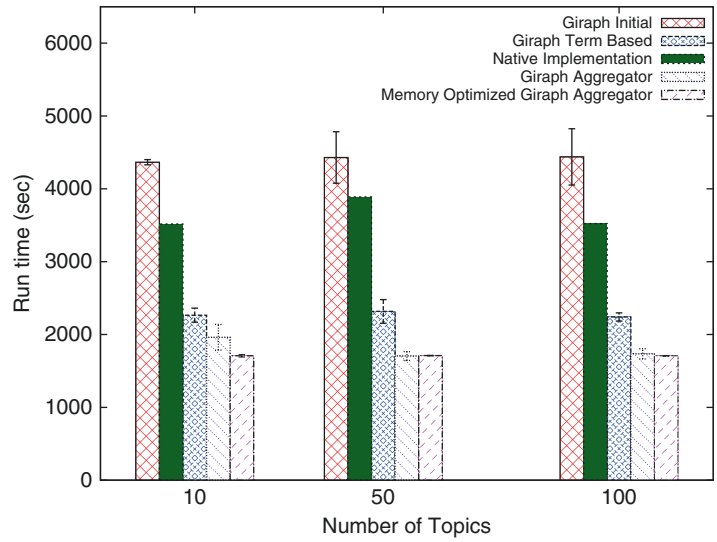
- **Related work comparison experiments:** The purpose of this experiment is to compare the running time and the topic quality of proposed approach to other topic detection approaches like SVD, LDA, LSA, and Kmeans. This experiment is performed on the same settings as in validating topic quality experiment.
- **Sliding window experiment:** The purpose of this experiment is to compare the running time of sliding window Giraph implementation to the Giraph implementation, while increasing the number of tweets. This experiment is performed on the same settings as in validating topic quality experiment.
- **Scalability experiment:** To measure the scalability of the proposed approach, it is evaluated on clusters of 8,16, and 32 machines. Each machine contains 30.5GB RAM and 4 cores.

US Election and Super Tuesday Results

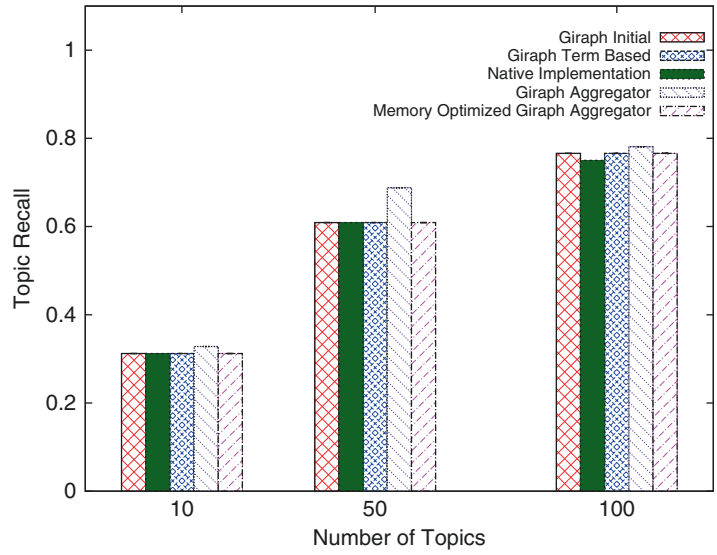
Four variations of the distributed implementation are compared to the centralized native Java implementation, which are:

- **Giraph initial:** Giraph implementation with no optimization
- **Giraph term-based:** Giraph implementation with the optimizations that let each term vertex v_j send its tf-idf weight in superstep 0 along with the set $\delta(v_j)$.
- **Giraph aggregator:** Giraph term-based implementation which uses an aggregator to compute the topics instead of the centralized component, where $m = 5000$.
- **Memory-optimized Giraph aggregator:** Memory optimized Giraph algorithm, C is chosen to equal 1000 empirically as it achieves the best topic quality.

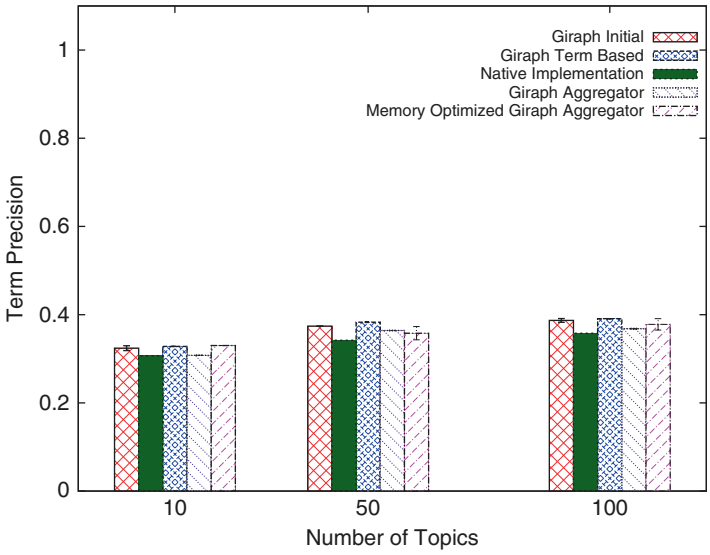
All runs are repeated three times and 95% confidence interval is reported in the figures. Figures 7, 8, 9, and 10 show the results of running



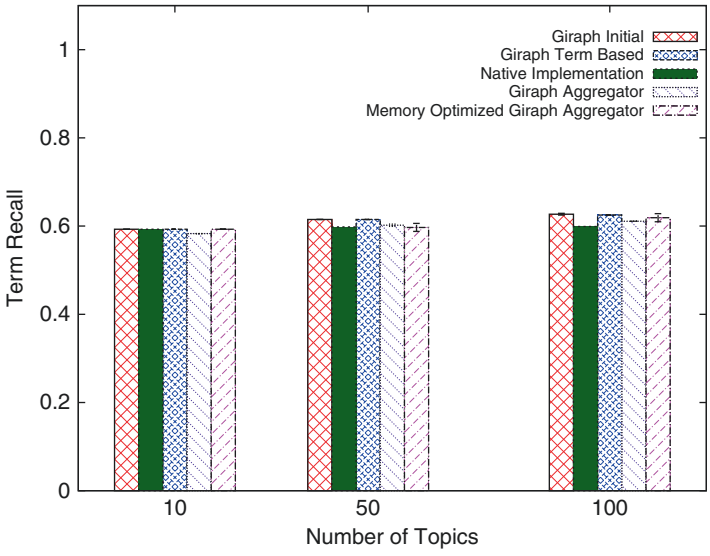
Real-Time Detection of Topics in Twitter Streams, Fig. 7 US election running time comparison



Real-Time Detection of Topics in Twitter Streams, Fig. 8 US election topic recall comparison



Real-Time Detection of Topics in Twitter Streams, Fig. 9 US election term precision comparison

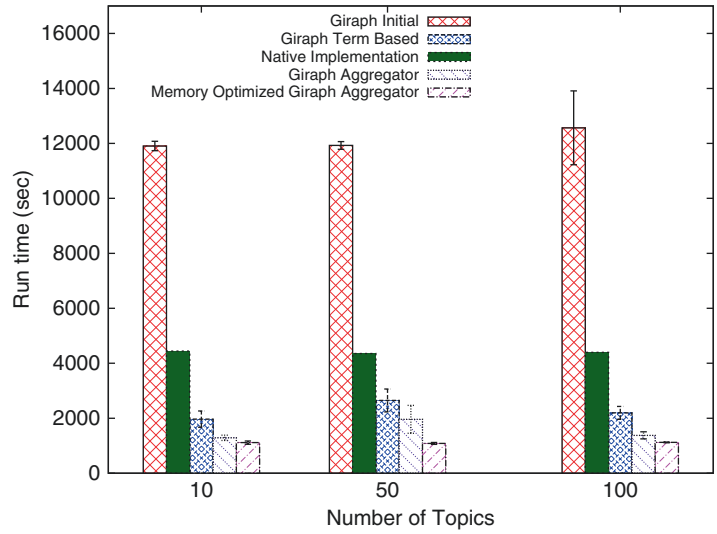


Real-Time Detection of Topics in Twitter Streams, Fig. 10 US election term recall comparison

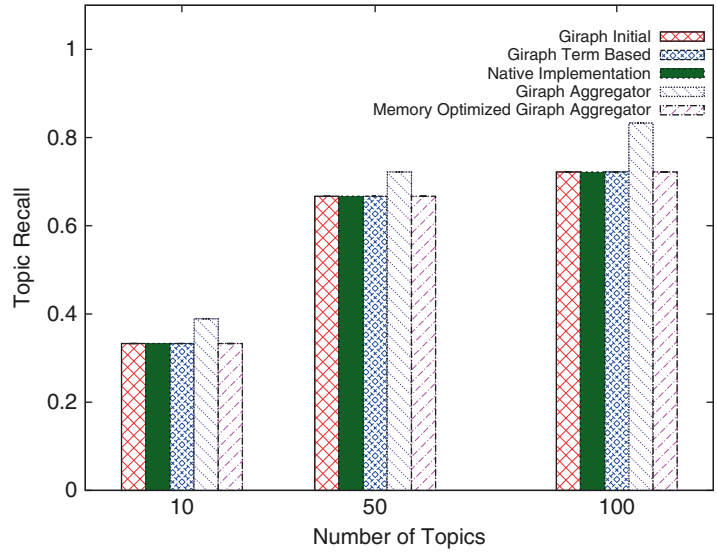
time, topic recall, term precision, and term recall respectively on US election dataset. Results show that our proposed Giraph approach is able to speed up the native implementation by a factor of ∼2X, while maintaining or increasing the topics quality. Also, results show that changing the number of detected topics has a small effect on the running time as the dominating time in the exemplar-based approach is the time to calculate the similarity matrix between the tweets. Moreover, Figs. 11, 12, 13, and 14 show the results of running time, topic recall, term precision, and term recall respectively on Super Tuesday dataset. Results also show that Giraph approach is able to speed up the native implementation by a factor of ∼3X with good topic quality.

Table 4 shows the average CPU idle time, memory usage and the number of the transmitted bytes for the Native implementation and different variations of Giraph implementation. As shown in

R

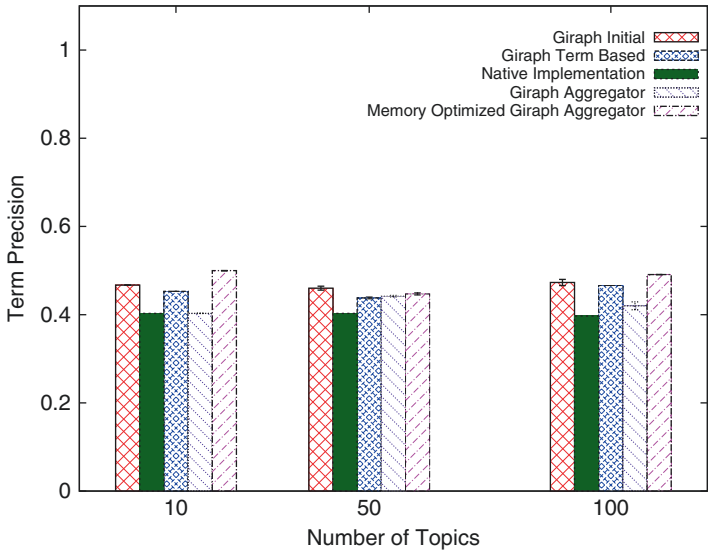


Real-Time Detection of Topics in Twitter Streams, Fig. 11 Super Tuesday running time comparison

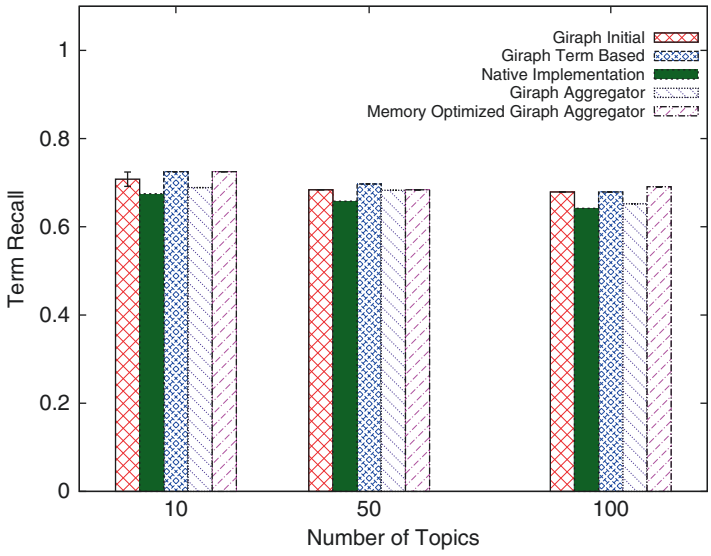


Real-Time Detection of Topics in Twitter Streams, Fig. 12 Super Tuesday topic recall comparison

the table, memory-optimized Giraph Aggregator has the lowest memory footprint in both US election and Super Tuesday datasets. It also achieves the lowest CPU utilization in both US election and Super Tuesday. However, the number of transmitted is the largest due to the aggregators usage and that the worker aggregators send all tweets to the master aggregator. For both Giraph Initial and Giraph Term-based, the variance of each tweet is stored by Giraph in the HDFS and then a centralized Java component collects these variances from the HDFS to select the representative exemplars. Thus, the number of transmitted bytes in this case is the sum of the bytes transmitted within Giraph and the bytes sent between all the machines and the machine that runs the centralized component. In the case of US Election dataset, the number of bytes transmitted in the Giraph term-based approach is dominated by the bytes transmitted by Giraph, that is why using the aggregator in the Giraph aggregator did not reduce the number of transmitted bytes. On the other hand, for the Super



Real-Time Detection of Topics in Twitter Streams, Fig. 13 Super Tuesday term precision comparison



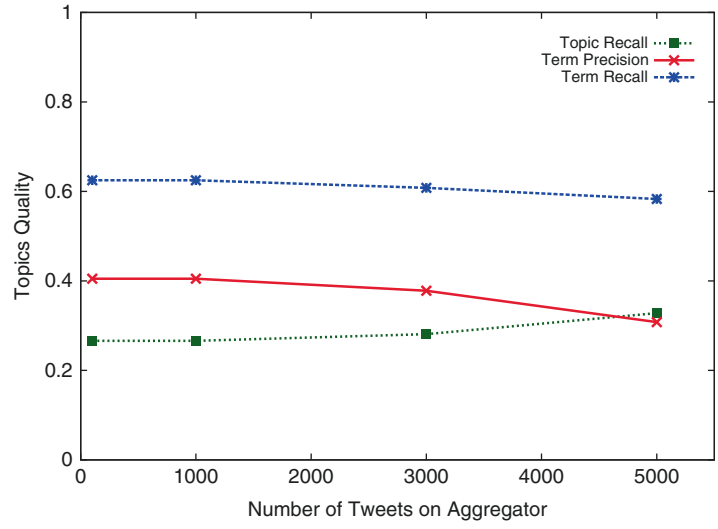
Real-Time Detection of Topics in Twitter Streams, Fig. 14 Super Tuesday term recall comparison

Tuesday dataset, the bytes sent between machines by the centralized Java component are dominating; as a result the cost introduced due to using the aggregator in the Giraph Aggregator approach is out-weighted by eliminating the significant part of the network overhead (moving the data between machines to the centralized Java component). This appeared in Super Tuesday dataset only as Super Tuesday time slots have more tweets than US election; therefore, HDFS cost increases while worker aggregators sent data is fixed ($m = 5000$).

For measuring the effect of changing the parameter m on the aggregators, we varied the parameter m from 100 to 5000 and measured the change in the running time and the quality of the detected topics on the US election dataset. As shown in Fig. 16, increasing m increases the running time of Giraph as the aggregator runs in $O\left(\frac{n}{w}m + wm\log(wm)\right)$. In addition, Fig. 15 shows that increasing m increases the topic recall and as the detected topics increased, the topic's quality (term precision and term recall) decreased.

Real-Time Detection of Topics in Twitter Streams, Table 4 CPU, memory, and network analysis of native implementation and Giraph distributed implementations (Number of topics = 100)

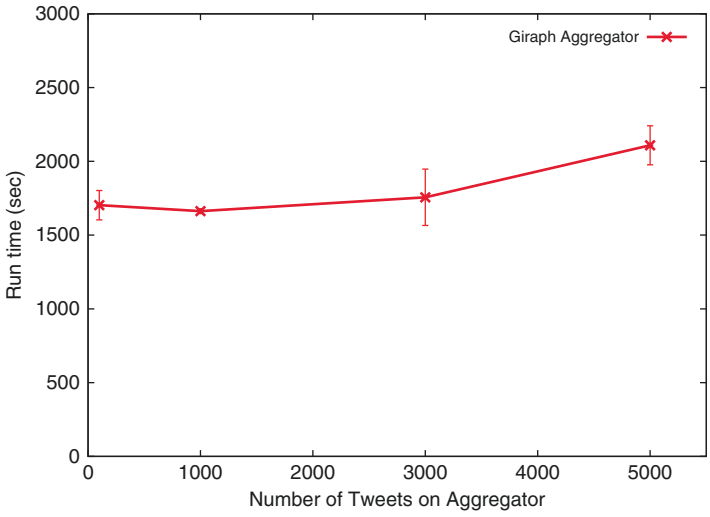
Dataset	Approach	Avg CPU idle time (%)	Avg memory size (GB)	Number of transmitted bytes
US election	**Native implementation**	86.91	9.35	–
	Giraph initial	86.01	10.07	3.69×10^{12}
	Giraph term based	84.75	7.77	2.13×10^{12}
	Giraph aggregator	84.56	7.62	1.18×10^{13}
	Memory-optimized Giraph aggregator	93.12	5.30	2.19×10^{13}
Super Tuesday	**Native implementation**	86.71	7.74	–
	Giraph initial	86.54	9.84	1.23×10^{13}
	Giraph term based	83.46	9.61	1.25×10^{13}
	Giraph aggregator	81.46	9.03	5.13×10^{12}
	Memory-optimized Giraph aggregator	92.57	4.63	2.19×10^{13}



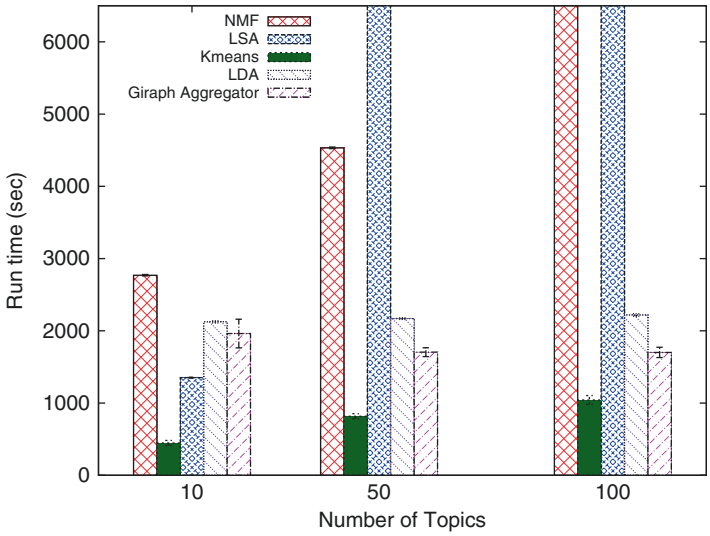
Real-Time Detection of Topics in Twitter Streams, Fig. 15 Aggregator parameter *m* effect on topic detection quality (number of detected topics = 10)

Related Work Comparison

We compared our approach to the Graphlab implementations as Giraph has no existing implementation of the related work. It is known in the literature that Graphlab implementations are faster than Giraph, therefore showing that our Giraph approach is faster than Graphlab-related work implementations, means that if our approach was implemented in Graphlab it will be even faster. Our approach is compared against LSA, NMF using alternating least squares, Kmeans and LDA. As Graphlab implementation of LDA takes the running time as a parameter, we chose to set the running time of LDA to be equal to the maximum time needed by our approach to finish one time slot which is 75 and 275 s for US election and Super Tuesday datasets respectively. Each run is repeated three times and 95% confidence interval is reported. For US election dataset we achieved the second lower running time after Kmeans, while algorithms like NMF



Real-Time Detection of Topics in Twitter Streams, Fig. 16 Aggregator Parameter *m* effect on running time (number of detected topics = 10)

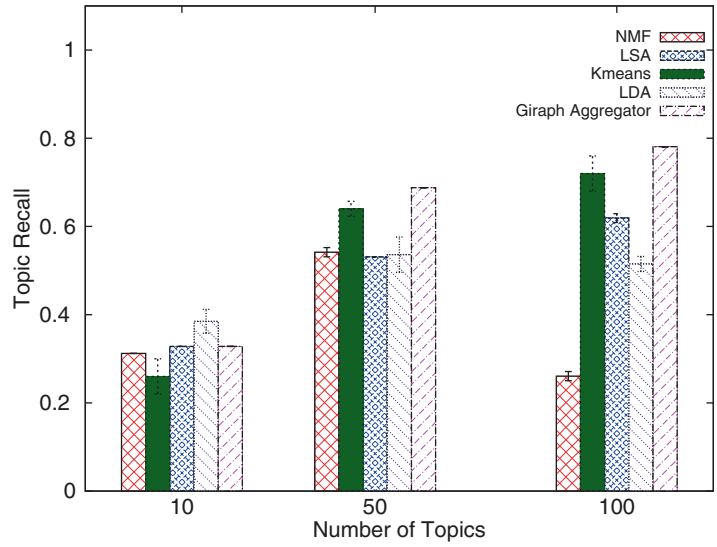


Real-Time Detection of Topics in Twitter Streams, Fig. 17 US election running time related work comparison

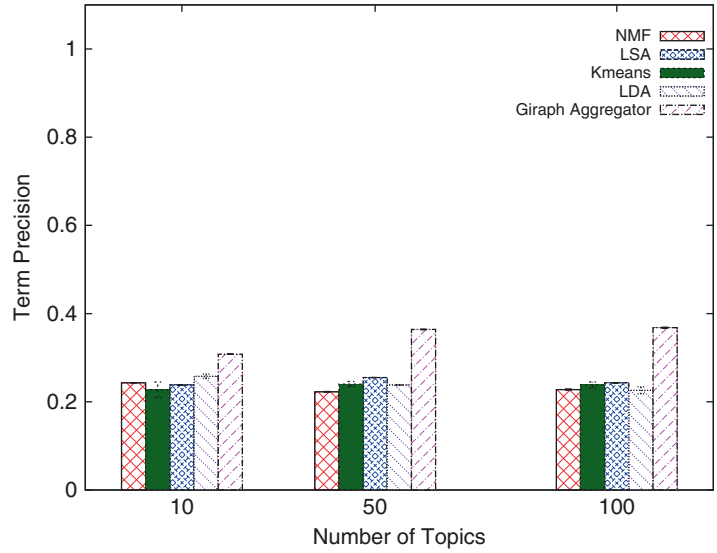
and LSA don't scale with increasing the number of topics as shown in Fig. 17. Note that LSA and NMF took more than 6500 s. However, we didn't show the complete bar in Fig. 17, so that figure can be clear. Our approach obtains the highest topic recall except for 10 topics, where LDA was higher with around 6% while our approach was higher than LDA with around 15% and 27% for 50 and 100 topics respectively as shown in Fig. 18. For term precision, our approach achieves the highest value for all number of topics as in Fig. 19, while maintaining good term recall as shown in Fig. 20.

The running time on the Super Tuesday dataset follows the same pattern, where Kmeans achieves the best running time, while LSA does not scale as shown in Fig. 21. Our algorithm achieves the best topic recall and term precision while maintaining a good term recall as shown in Figs. 22, 23 and 24. It is worth noting that LSA achieves the highest term recall as it detects a few number of correct topics which is reflected in its low topic recall.

R



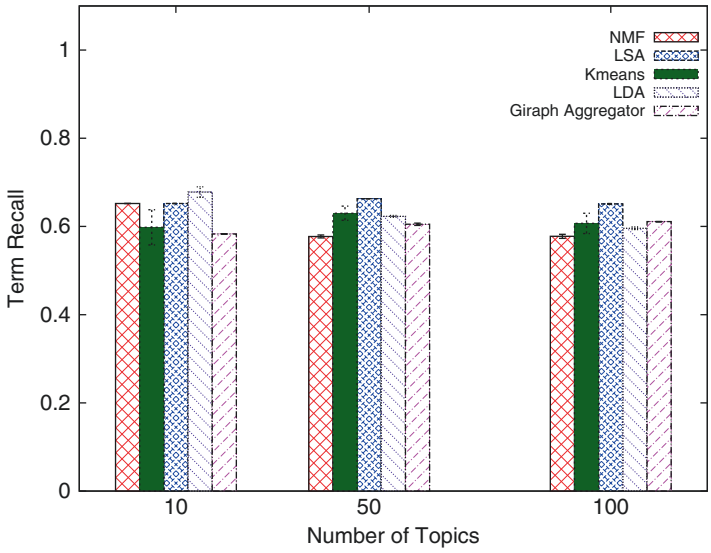
Real-Time Detection of Topics in Twitter Streams, Fig. 18 US election topic recall related work comparison



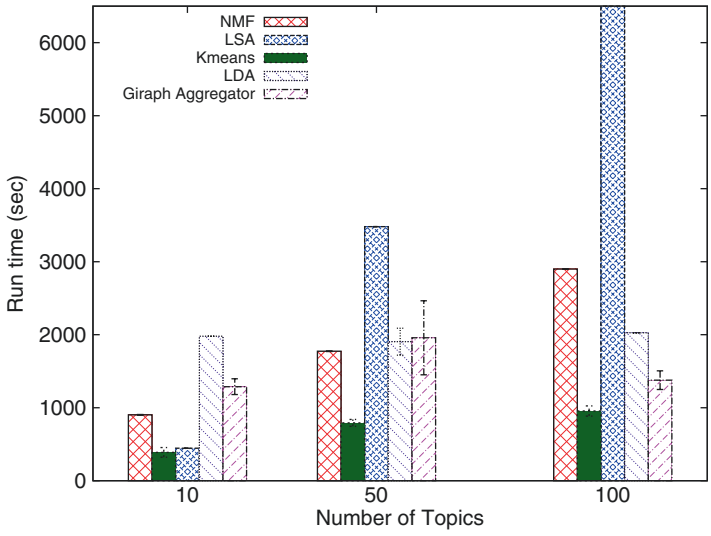
Real-Time Detection of Topics in Twitter Streams, Fig. 19 US election term precision related work comparison

Giraph Sliding Window

In this section, Giraph sliding windows technique is compared against the Giraph aggregator. Both systems run using three workers on 10 time slots, which are a subset of 120 M tweets dataset. Table 5 shows the running time of both systems by varying the window size l and fixing the shifting size s to 10,000, the number of topics to be detected to 10 and the number of tweets to keep at each aggregator m to 5000. Results show that Giraph Sliding Windows technique is faster than Giraph Aggregator and achieves a speedup of factor 19X over the native implementation. However this gap gets reduced as the window size increases, as the computation time dominates the total running time instead of Giraph setup and vertices' loading time.



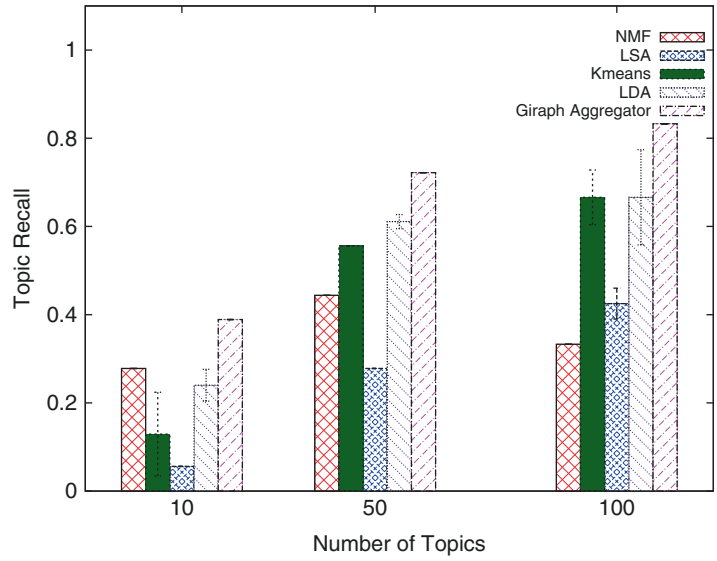
Real-Time Detection of Topics in Twitter Streams, Fig. 20 US election term recall related work comparison



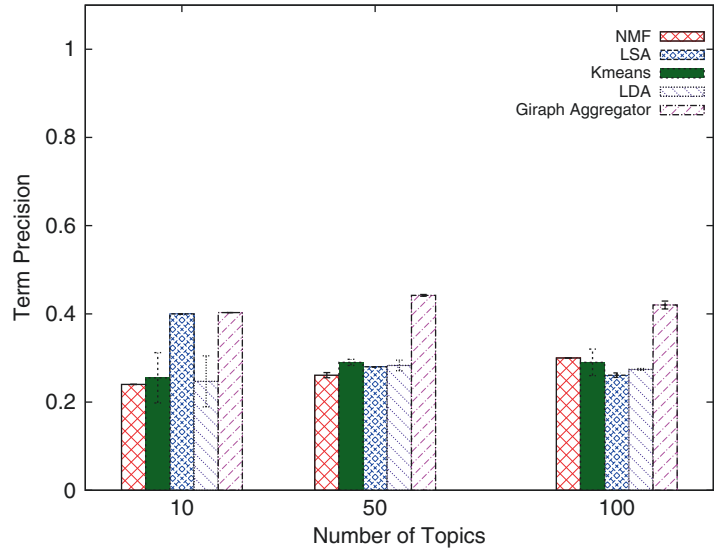
Real-Time Detection of Topics in Twitter Streams, Fig. 21 Super Tuesday running time-related work comparison

Scalability Results

To demonstrate the efficiency of our proposed system in a real scenario, we have collected two million English tweets using the streaming API, which is the amount of collected English tweets in 10 min according to Twitter statistics. In this experiment, we have varied the number of machines from 8 to 16 and 32 machines and measured the running time. Each run is replicated three times and the 95% confidence interval is reported. As shown in Fig. 25 using only 16 machines, the topic detection task is finished in around 7 min, which is less than the 10 min window and therefore using a small cluster of 16 machines is enough for our approach to run in a real settings and to detect topics from real Twitter streams.



Real-Time Detection of Topics in Twitter Streams, Fig. 22 Super Tuesday topic recall related work comparison



Real-Time Detection of Topics in Twitter Streams, Fig. 23 Super Tuesday term precision related work comparison

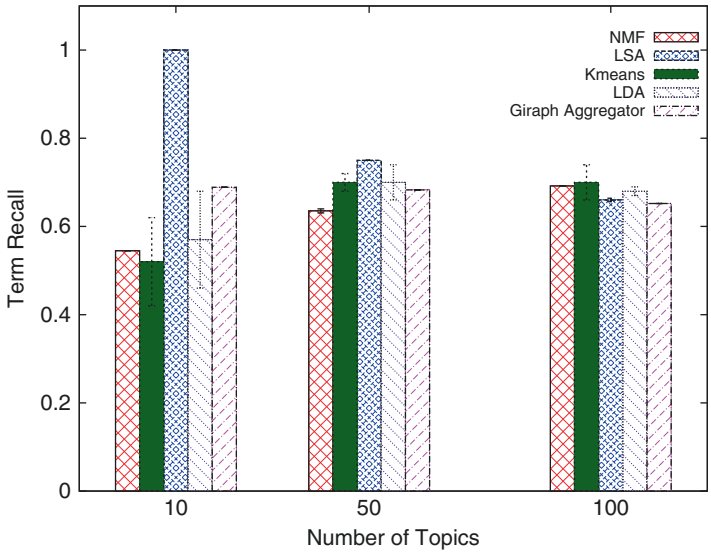
Key Applications

The proposed system can be used in Twitter or other microblog sites to detect the topics from their data streams in a scalable manner.

Future Directions

"Think like a Graph" Model

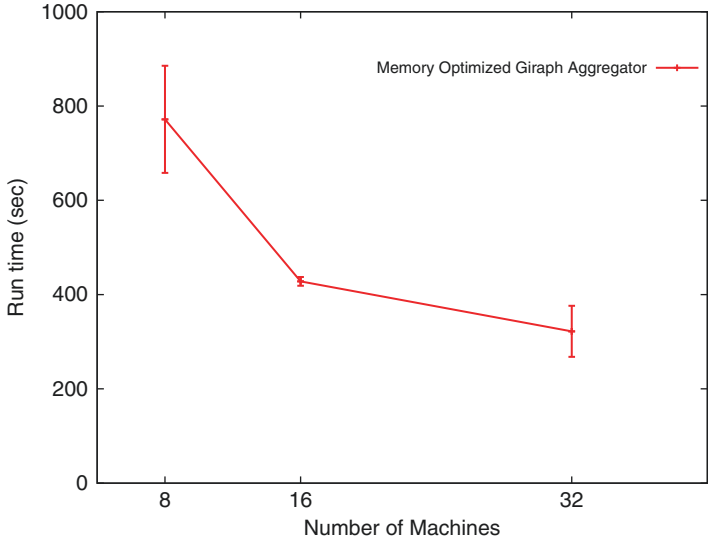
A new "Think like a graph" model is proposed in (Tian et al. 2013). The basic idea behind the new model is that the communication between the vertices within the same partition can be performed without exchanging messages, as all the information about these vertices are located on the same partition. The new model uses the information that each partition has, so that the information can flow freely inside one partition. In this paradigm the computations are expressed in terms of what each partition has to do as opposed to what each vertex has to do in the vertex-centric paradigm. Employing this model will reduce the running time. However, to reduce



Real-Time Detection of Topics in Twitter Streams, Fig. 24 Super Tuesday term recall related work comparison

Real-Time Detection of Topics in Twitter Streams, Table 5 Running time comparison (in seconds) for Giraph sliding windows using 120 M tweets dataset

n			
Approaches	20000	50000	100000
Native implementation	75.0	518.0	1975.0
Giraph aggregator	33.1	43.5	107.7
Giraph sliding window	24.0	31.9	102.6



Real-Time Detection of Topics in Twitter Streams, Fig. 25 The effect of increasing the number of machines on the running time

the running time the system assumes that the vertices are partitioned such that there are no crossing edges between partitions, while this process is very costly. Thus, an efficient partitioning heuristic is needed to make use of the "Think like a graph" model.

Partitioning Strategies

Using a partitioning strategy that reduces the number of edges between different partitions will reduce the communication overhead. Although graph partition problems fall under the category of NP-hard problem, the following heuristic partitioning strategy can be used to place the tweet vertex and the term vertices that are connected to it on one partition. However, this may introduce imbalanced partitions. To alleviate the imbalanced partitions problem, one of the dynamic load balancing techniques as in (Gonzalez et al. 2012) or graph Voronoi diagram based partitioner used by Blogel (Yan et al. 2014) can be used.

Parameters Tuning

As the parameters k, ϵ, and the window size are important parameters that have great effect on the running time and the detected topic quality of the exemplar-based topic detection approach. It is essential to select these parameters carefully. Therefore, several approaches are needed to be developed to automatically tune the values of these three parameters and to adjust them dynamically in different time slots as the tweets keep coming.

Cross-References

- ▶ Automatic Document Topic Identification Using Social Knowledge Network
- ▶ Spatiotemporal Topic Detection from Social Media
- ▶ Topic Modeling in Online Social Media, User Features and Social Networks for

Acknowledgments This publication was made possible by a grant from the Qatar National Research Fund through National Priority Research Program (NPRP) No. 06-1220-1-233. Its contents are solely the responsibility of the authors.

References

Agarwal MK, Ramamritham K, Bhide M (2012) Real time discovery of dense clusters in highly dynamic graphs: identifying real world events in highly dynamic environments. Proc VLDB Endowment 5(10):980–991, Turkey

Aiello LM, Petkos G, Martin C, Corney D, Papadopoulos S, Skraba R, Goker A, Kompatsiaris I, Jaimes A (2013) Sensing trending topics in twitter. IEEE Trans Multimedia 15(6):1268–1282

Blei DM, Ng AY, Jordan MI (2003) Latent dirichlet allocation. J Mach Learn Res 3:993–1022

Budak C, Georgiou T, Agrawal D, El Abbadi A (2013) Geoscope: online detection of geo-correlated information trends in social networks. Proc VLDB Endowment 7(4):229, Italy

Elbagoury A, Ibrahim R, Farahat A, Kamel M, Karray F (2015) Exemplar-based topic detection in twitter streams. In: Ninth international AAAI conference on weblogs and social media, Oxford, UK

Elbagoury A, Ibrahim R, Kamel MS, Karray F (2016) Ebek: exemplar-based kernel preserving embedding. In: Proceedings of the 25th international conference on artificial intelligence. AAAI Press, New York, USA

Elsayed T, Lin J, Oard DW (2008) Pairwise document similarity in large collections with mapreduce. In: Proceedings of the 46th annual meeting of the Association for Computational Linguistics on human language technologies: short papers. Association for Computational Linguistics, pp 265–268, Ohio, USA

Gonzalez JE, Low Y, Gu H, Bickson D, Guestrin C (2012) Powergraph: distributed graph-parallel computation on natural graphs. OSDI 12:2, Hollywood, CA, USA

Gonzalez JE, Xin RS, Dave A, Crankshaw D, Franklin MJ, Stoica I (2014) Graphx: graph processing in a distributed dataflow framework. In: Proceedings of the 11th USENIX symposium on operating systems design and implementation (OSDI), broomfield, CO, USA

Gupta P, Goel A, Lin J, Sharma A, Wang D, Zadeh R (2013) Wtf: the who to follow service at twitter. In: Proceedings of the 22nd international conference on world wide web, pp 505–514. International World Wide Web Conferences Steering Committee, Rio de Janeiro, Brazil

Ibrahim R, Elbagoury A, Kamel MS, Karray F (2016) Lvc: local variance-based clustering. In: Neural networks (IJCNN), 2016 international joint conference on. IEEE, Vancouver, Canada

Landauer TK, Foltz PW, Laham D (1998) An introduction to latent semantic analysis. Discourse Processes 25(2–3):259–284

Lee P, Lakshmanan LV, Milios EE (2014) Incremental cluster evolution tracking from highly dynamic network data. In: Data engineering (ICDE), 2014 I.E. 30th international conference on. IEEE, pp 3–14, Chicago, IL, USA

Li R, Lei KH, Khadiwala R, Chang K-C (2012) Tedas: a twitter-based event detection and analysis system. In:

Data engineering (icde), 2012 ieee 28th international conference on. IEEE, pp 1273–1276, Arlington, VA, USA
Li R, Wang S, Chang KC-C (2013) Towards social data platform: automatic topic-focused monitor for twitter stream. Proc VLDB Endowment 6(14):1966–1977, Italy
Low Y, Gonzalez JE, Kyrola A, Bickson D, Guestrin CE, Hellerstein J (2014) Graphlab: a new framework for parallel machine learning. In: arXiv preprint arXiv:1408.2041
McMinn AJ, Moshfeghi Y, Jose JM (2013) Building a large-scale corpus for evaluating event detection on twitter. In: Proceedings of the 22nd ACM international conference on conference on information & knowledge management. ACM, pp 409–418, San Francisco, CA, USA
Mehrotra R, Sanner S, Buntine W, Xie L (2013) Improving lda topic models for microblogs via tweet pooling and automatic labeling. In: Proceedings of the 36th international ACM SIGIR conference on research and development in information retrieval. ACM, pp 889–892, Dublin, Ireland
Mishne G, Dalton J, Li Z, Sharma A, Lin J (2013) Fast data in the era of big data: Twitter's real-time related query suggestion architecture. In: Proceedings of the 2013 ACM SIGMOD international conference on management of data. ACM, pp 1147–1158, New York, USA
Newman D, Bonilla EV, Buntine W (2011) Improving topic coherence with regularized topic models. In: Advances in neural information processing systems, pp 496–504, Granada Spain.
Pantel P, Crestan E, Borkovsky A, Popescu A-M, Vyas V (2009) Web-scale distributional similarity and entity set expansion. In: Proceedings of the 2009 conference on empirical methods in natural language processing: volume 2-volume 2. Association for Computational Linguistics, pp 938–947, Singapore
Salton G, Buckley C (1988) Term-weighting approaches in automatic text retrieval. Inf Process Manag 24(5): 513–523
Tian Y, Balmin A, Corsten SA, Tatikonda S, McPherson J (2013) From "think like a vertex" to "think like a graph". Proc VLDB Endowment 7(3):193–204, Italy
Toshniwal A, Taneja S, Shukla A, Ramasamy K, Patel JM, Kulkarni S, Jackson J, Gade K, Fu M, Donham J et al (2014) Storm@ twitter. In: Proceedings of the 2014 ACM SIGMOD international conference on management of data. ACM, pp 147–156, Utah, USA
Xie W, Zhu F, Jiang J, Lim E-P, Wang K (2013) Topicsketch: real-time bursty topic detection from twitter. In: Data mining (ICDM), 2013 I.E. 13th international conference on. IEEE, pp 837–846, Dallas, Texas, USA
Yan X, Guo J, Liu S, Cheng X-q, Wang Y (2012) Clustering short text using ncut-weighted non-negative matrix factorization. In: Proceedings of the 21st ACM international conference on information and knowledge management. ACM, pp 2259–2262, Maui, HI, USA
Yan D, Cheng J, Lu Y, Ng W (2014) Blogel: a block-centric framework for distributed computation on real-world graphs. Proc VLDB Endowment 7(14):1981, Hangzhou, China
Yuan M, Wu K-L, Jacques-Silva G, Lu Y (2013) Efficient processing of streaming graphs for evolution-aware clustering. In: Proceedings of the 22nd ACM international conference on conference on information & knowledge management. ACM, pp 319–328, San Francisco, CA, USA
Zhang Z, Shu H, Chong Z, Lu H, Yang Y (2013) C-cube: elastic continuous clustering in the cloud. In: Data engineering (ICDE), 2013 I.E. 29th international conference on. IEEE, pp 577–588, Brisbane, Australia

Real-Time Recombination

▶ Spatiotemporal Recommendation in Geo-Social Networks

Real-Time Social Media Analysis

▶ Twitris: A System for Collective Social Intelligence

Reciprocal Averaging

▶ Correspondence Analysis

Reciprocity

▶ Exchange Networks

Recommendation System

▶ Recommender Systems Using Social Network Analysis: Challenges and Future Trends

Recommendation Systems

▶ Recommender Systems, Basics of
▶ Recommender Systems Evaluation
▶ Recommender Systems Based on Social Networks
▶ Recommender Systems: Models and Techniques

Recommender Engine

▶ Recommender Systems Using Social Network Analysis: Challenges and Future Trends

Recommender Platform

▶ Recommender Systems Using Social Network Analysis: Challenges and Future Trends

Recommender Systems

▶ Community Detection and Recommender Systems
▶ Social Web Search

Recommender Systems Based on Linked Open Data

Tommaso Di Noia and Paolo Tomeo
SisInf Lab, Polytechnic University of Bari, Bari, Italy

Synonyms

DBpedia; Hybrid recommender systems; Information filtering; Semantic Web; Web of data

Glossary

RSs	Recommender Systems
LOD	Linked Open Data
Semantic Web	Web of linked data
RDF	Resource Description Framework
SPARQL	Query language for the Semantic Web
SPrank	Semantic Path-based ranking algorithm for recommendation

Definition

The Linked Open Data initiative (Bizer et al. 2009) has allowed the publication of a vast amount of data in the Semantic Web. Concurrently, growing massive amount of information on the web has led us in the Information Overload era, where the enormous amount of information and choices undermines the user experience. Recommender Systems help users to find what is relevant for them in a vast range of possibilities. Recommender Systems can benefit from the use of knowledge encoded in the Linked Open Data to provide better recommendations.

Introduction

In the last years, we assisted to the shift of the Web from a distributed collection of hyper-linked documents to a huge distributed repository of linked data. Boosted by the technologies behind the creation of the Semantic Web and by the `Linked Open Data (LOD)` initiative, a vast amount of RDF data have been produced and published online in freely distributed datasets which are in turn connected with each other to form the so called LOD cloud. The information they encode spans on different and diverse knowledge domains and is a mine for data-intensive applications. Moreover, a massive amount of information has been growing rapidly on the Web with the recent diffusion of social networks and pervasive mobile devices.

This has led us into an era of Information Overload: the information amount exceeds the users' capability of processing and using it (Speier et al. 1999). Huge and fast growing number of possibilities overwhelms users, leading them to make poor decisions and feel anxiety and unsatisfaction (Schwartz 2005). Recommender Systems (RSs) (Ricci et al. 2015) are a family of information filtering tools which have proven to be valuable means in assisting users to find, in a personalized manner, what is relevant for them in such overflowing complex information spaces. They provide users with personalized access to large collections of resources.

Key Points

In this chapter, we introduce all the notions and elements needed to build and evaluate the effectiveness of a RS which leverages the data accessible in the LOD cloud. In the next section, we briefly recall some of the relevant scientific literature in the field. Section "Using Linked Open Data to Solve the Recommendation Problem" is devoted to the description of the recommendation problem while in section "LOD-Enabled Recommender Systems," we discuss on how to exploit the knowledge encoded in the `Linked Open Data` cloud to design a semantics-aware recommender system. Some relevant graph-based algorithms developed to deal with `LOD` datasets are presented section "Graph-Based Recommender Systems Using LOD." Future directions in the field of recommender systems feed by `Linked Open Data` close the chapter.

In the following, we enumerate the main key points we present here:

- Brief introduction to Recommender Systems.
- How to exploit Linked Data to build a semantics-aware recommendation engine.
- Presentation of three algorithms leveraging the graph-based nature of Linked Data.
- Evaluation of the algorithms and discussion of the experimental results.

Historical Background

Many researches in the past have proposed different ways of using domain ontologies and taxonomies to improve the quality of conventional RSs. In Maidel et al. (2008), the authors presented a content-based filtering approach wherein user and item profiles are described in terms of concepts belonging to a domain taxonomy. The authors propose different possible matches between user and item such as exact or partial and different match scores depending also on the hierarchical distance between concepts. The authors of Middleton et al. (2004) describe an approach for ontological user profiling and an application of such approach for building a research paper recommendation system where both research papers and user profiles are described in terms of topics organized in taxonomy. In Mobasher et al. (2004), the authors present a semantically enhanced collaborative filtering approach where structured semantic knowledge about items is used in conjunction with user-item ratings to create a combined similarity measure for item comparisons. Taxonomic information is used in Ziegler et al. (2004) to represent the user's interest in categories of products. Consequently, user similarity is determined by common interests in categories and not by common interests in items. In Anand et al. (2007), the authors present an approach that infers user preferences from rating data using an item ontology. The system collaboratively generates recommendations using the ontology and infers preferences during similarity computation. Another hybrid ontological recommendation system is proposed in Cantador et al. (2008) where user preferences and item features are described by semantic concepts to obtain users' clusters corresponding to implicit *Communities of Interest*. Most of the presented works used ontologies to compute better user-user or item-item similarities in memory-based collaborative filtering approaches. However, little work has been done in exploiting ontologies for computing model-based recommendations. A detailed description of recommendation techniques based on ontological filtering is given in de Gemmis et al. (2015)

and Jannach et al. (2010). In the last few years with the availability of Linked Open Data, a new class of recommender systems has emerged which can be named as LOD-based recommender systems. One of the first approaches that exploits Linked Open Data for building recommender systems is presented in Heitmann and Hayes (2010). Here the authors, for the first time propose a recommender system fed by Linked Open Data. In Fernández-Tob´ ıas et al. (2011), the authors present a knowledge-based framework leveraging DBpedia for computing cross-domain recommendations. In Lommatzsch et al. (2011), the authors propose a graph-based recommendation approach utilizing model and memory-based link prediction methods. In Marie et al. (2013), LOD datasets are used for personalized exploratory search using a spreading activation method. *dbrec* (Passant 2010) is a music content-based recommender system leveraging the DBpedia dataset. They define the *Linked Data Semantic Distance* in order to find semantic distances between resources and then compute recommendations. A full SPARQL-based recommendation engine named RecSPARQL has been presented in 2015 as a standalone project, its data flowed into the Wikidata presented in Ayala et al. (2014). In Khrouf and Troncy (2013), the authors present an event recommendation system based on linked data and user diversity. A semantic-aware extension of the SVD++ model, named SemanticSVD++, is presented in Rowe (2014a). In Rowe (2014b), the authors improve their previous work for dealing with cold-start items by introducing a vertex kernel for getting knowledge about the unrated semantic categories starting from those categories which are known. Another interesting direction about the usage of LOD for content-based RSs is explored in Musto et al. (2014) where the authors present Contextual eVSM, a content-based context-aware recommendation framework that adopts a semantic representation based on distributional models and entity linking techniques. In Dojchinovski and Vitvar (2014), the authors propose the usage of recommendation techniques for providing personalized access to Linked Data. The proposed recommendation method is a user-user collaborative filtering recommender wherein the similarity between the users takes into account the commonalities and informativeness of the resources instead of treating resources as plain identifiers. In Nguyen et al. (2015a), authors presented an evaluation of two graph-based similarity metrics – SimRank and Personalized PageRank – applied upon resources in RDF graph to feed a content based recommender system, compared against other two similarity metrics – VSM and a simplified variation of it presented in Di Noia et al. (2012). The results in the music domain showed that SimRank and Personalized PageRank may lead to accurate and novel recommendations but penalizing the aggregate diversity. Most of aforementioned works use DBpedia as external knowledge graph. A comparative analysis between DBpedia and Freebase was carried out in Nguyen et al. (2015b) demonstrating that Freebase may lead to better accuracy and aggregate diversity respect to DBpedia. One of the well-known problem recently raised in recommender systems is to recommend relevant items to cold-start users – those, namely, who have provided little or no information about their interests. The experimental results showed that hybrid recommendation methods that jointly exploit user ratings and item metadata extracted from Linked Data can provide accurate recommendations even for users with very few likes, hence addressing the cold-start user problem (Tomeo et al. 2016). In particular, those results demonstrated that graph-based methods effectively exploit content information and are the best choice in domains where content information is more important that collaborative information, such as book and movie domain. More recently, in Di Noia et al. (2017), DBpedia has been used as the reference dataset to propose a user modeling able to diversify recommendation results in a personalized way.

Using Linked Open Data to Solve the Recommendation Problem

Recommender Systems

Recommender Systems are software tools and techniques providing suggestions for items that a user may like or wish to utilize (Ricci et al. 2015).

The main task of a recommendation engine is typically to estimate the relevance of unknown items for a target user and recommend the Top-N items by considering for each user the best *N* items with highest utility. Typically, the utility of an item is represented by a numerical rating, which indicates how much a particular user liked such item. However, the utility is not defined for each pair of users and items and usually it is available only for a small subset of them. Such subset represents the input of a generic recommender system, whose objective is to estimate the utility of all the remaining pairs. A formal definition of the recommendation problem has been given in Adomavicius and Tuzhilin (2005) and defined as follows.

Definition 1 *Let U represent the set of users and I the set of items in the system. Potentially, both sets can be very large. Let f: $U \times I \rightarrow R$, where R is a totally ordered set, be a utility function measuring the relevance of item $i \in I$ for user $u \in U$. Then, the recommendation problem consists in finding for each user u such item $i^{max,u} \in I$ maximizing the utility function f. More formally, this corresponds to the following:*

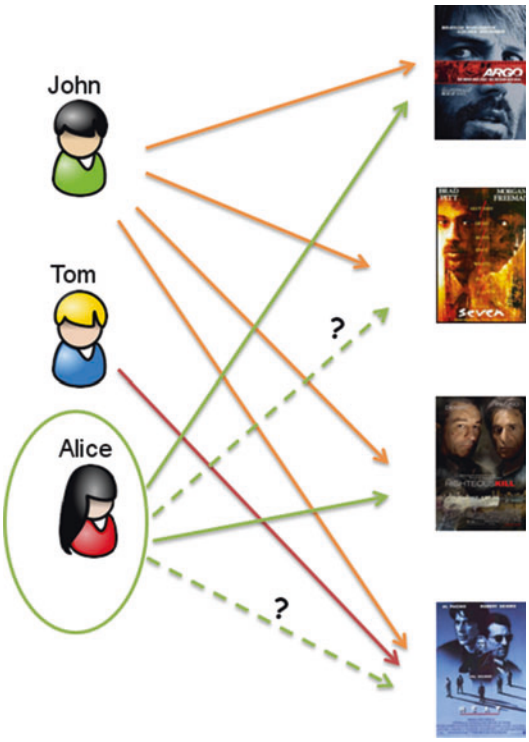
$$\forall u \in U, i^{max,u} = arg\, max_{i \in I} f(u, i)$$

Depending on the way, the utility function is estimated and the availability of additional data about the characteristics of items for example, there are different types of recommendation techniques. The main two are: collaborative filtering and content-based. A complete list of techniques is given in Burke (2007) and in Ricci et al. (2015).

Collaborative Filtering Recommendation

Collaborative Filtering is the process of filtering or evaluating items using the opinions of other people (Schafer et al. 2007).

The only input data that CF-RSs need is the user-item ratings matrix. Figure 1 shows a simple example of collaborative filtering case. If we consider `Alice` as target user, as said before, recommendations are generated considering the ratings given by other users with similar tastes.



Recommender Systems Based on Linked Open Data, Fig. 1 Illustration of a CF-based recommender system

According to Breese et al. (1998), there are two main types of collaborative filtering methods: memory-based and model-based. Memory-based CF uses a particular type of Machine Learning methods: nearest neighborhood (k-NN) algorithm. In particular, it does not require any preliminary model building phase, since predictions are made by aggregating the ratings of the closest neighbors. Conversely, model-based techniques first learn a predictive model which is eventually used to make predictions. Memory-based approaches can be classified either in user-based (Jin et al. 2004) or item-based (Sarwar et al. 2001).

Content-Based Recommendation

Content-based RSs recommend an item to a user based upon a description of the item and a profile of the user's interests (Pazzani and Billsus 2007). Briefly, the basic process performed by a content-based recommender consists is matching up the attributes of a user profile with the attributes of a

content object (item) (de Gemmis et al. 2015). Differently from collaborative filtering, such recommendation approach relies on the availability of content features describing the items. A high level architecture of a content-based RS is presented in de Gemmis et al. (2015). There are two main content-based recommendation approaches: *heuristic-based* or *model-based*. Approaches using heuristic functions have their roots in Information Retrieval and Information Filtering. Items are recommended based on a comparison between their content and a user profile. The idea is to represent both items and users using typical IR techniques (Balabanović and Shoham 1997), e.g., vectors of terms, and compute a match between their representations. The user profile consists in a vector of terms built from the analysis of the items liked by the user. Model-based approaches (Pazzani and Billsus 1997) use Machine Learning techniques to learn a model of the user's preferences by analyzing the content characteristics of items the user rated. Content-based methods can have several limitations. For a complete and detailed description of content-based recommendation techniques refer to de Gemmis et al. (2015) and Pazzani and Billsus (2007). The main one is the `content overspecialization` which consists in the incapability of the system to recommend relevant items which are different to the ones the user already knows.

Evaluation Metrics

Establishing the best algorithm for a given recommendation purpose requires a comparative analysis based on an appropriate experimental evaluation (Gunawardana and Shani 2015). Originally, evaluations were performed mainly focused on recommendation accuracy, which measures the ability of a system to provide correct (accurate) recommendations. Examples of accuracy metrics are `Precision`, `Recall`, and `Normalized Discounted Cumulative Gain`, usually computed considering incremental list sizes for Top-N recommendation task, with N varying typically from 1 to 50 or 100. Recently, it has been strongly demonstrated that attention needs to be paid also to other quality dimensions, such as diversity and novelty (Castells et al. 2015). A comprehensive review of evaluation metrics, techniques, and types of experiments is provided in (Gunawardana and Shani 2015).

LOD-Enabled Recommender Systems

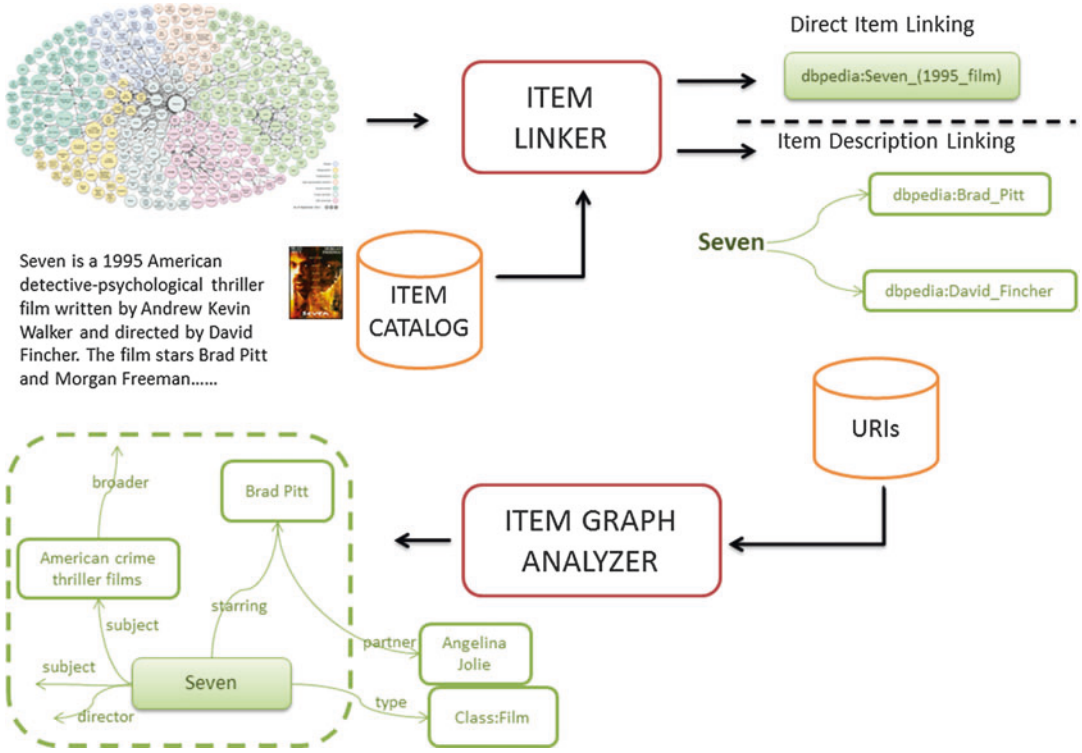
Nowadays `Linked Open Data` datasets represent a huge repository of different kinds of knowledge spanning from sedimentary-one such as encyclopedic, linguistic, common-sense, and so on, to real-time one such as data streams, events, etc. The main advantages of using LOD for content-based and hybrid recommender systems can be summarized as:

- Availability of a great amount of multidomain and ontological knowledge freely available for feeding the system.
- Semantic Web standards and technologies to retrieve the required data and hence no need for content analysis tasks for obtaining a structured representation of the items content.
- The ontological and relational nature of the data allows the system to analyze item descriptions at a semantic level.

If we consider encyclopedic datasets such as `DBpedia` (Lehmann et al. 2015) or `Freebase` (Bollacker et al. 2008), we have access to a huge amount of factual knowledge referring to a variety of topics. As pointed out by Semeraro et al. (2009), factual and common sense knowledge bases can provide the system with the "cultural" background knowledge needed to compute an accurate content analysis. For example, we can obtain information about actors starring in the movie *Pulp Fiction* from `DBpedia` via a simple `SPARQL` query:

```
PREFIX dbr: <http://dbpedia.org/
resource/>
PREFIX dbo: <http://dbpedia.org/
ontology/>
SELECT ?actor WHERE {
     dbr:Pulp_Fiction dbo:starring ?
actor .
}
```

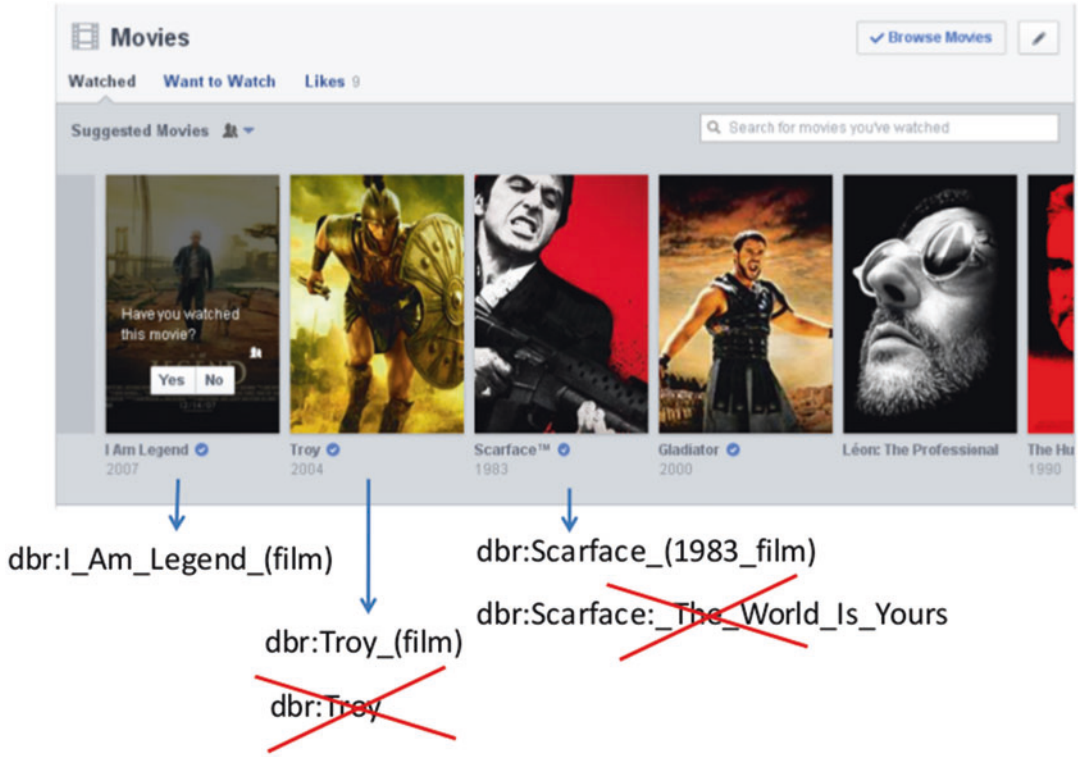
Given the `URI` corresponding to an item, we have then an entry point in the knowledge sub-



Recommender Systems Based on Linked Open Data, Fig. 2 High-level architecture for feeding RSs with LOD

graph corresponding to the specific entity (Pulp Fiction in this case). One of the advantages of using LOD is the availability of well structured graph-based item descriptions. In fact, each entity is usually classified by a class belonging to an ontological taxonomy and is in turn connected to other entities by means of semantic relations/properties. For example, the resource dbr:Bruce Willis in DBpedia is an instance of the class dbo:Person which is sub-class of dbo:Agent. Analogously, the property dbo:starring which connects dbr:Pulp Fiction to dbr:Bruce Willis has domain dbo:Work and range dbo:Actor which is subclass of dbo:Person. Thanks to the semantic relations among entities, the system can perform a deeper semantic analysis of the item content which results harder in a keyword-based approach. In order to effectively incorporate Linked Open Data in recommendation applications, there are several aspects to consider. Ultimately, the goal is to provide the system with background knowledge about the domain of interest in the form of a knowledge graph. In Fig. 2, we show a high level architecture of a component in charge of retrieving portions of the LOD graph regarding the items in the system which are used to form the knowledge graph. Such component consists of two main modules: the Item Linker and the Item Graph Analyzer.

Item Linker The Item Linker addresses the task of linking the items in the system with the corresponding resources in the LOD knowledge bases. The aim of such component is bridging the gap between the items in the catalog and LOD. We have hypothesized two main ways for performing the linking task: Direct Item Linking and Item Description Linking. These modules take as input any dataset in the Linked Open Data cloud and the list of items in the catalog with associated side information, if available, and return either the mapping between items



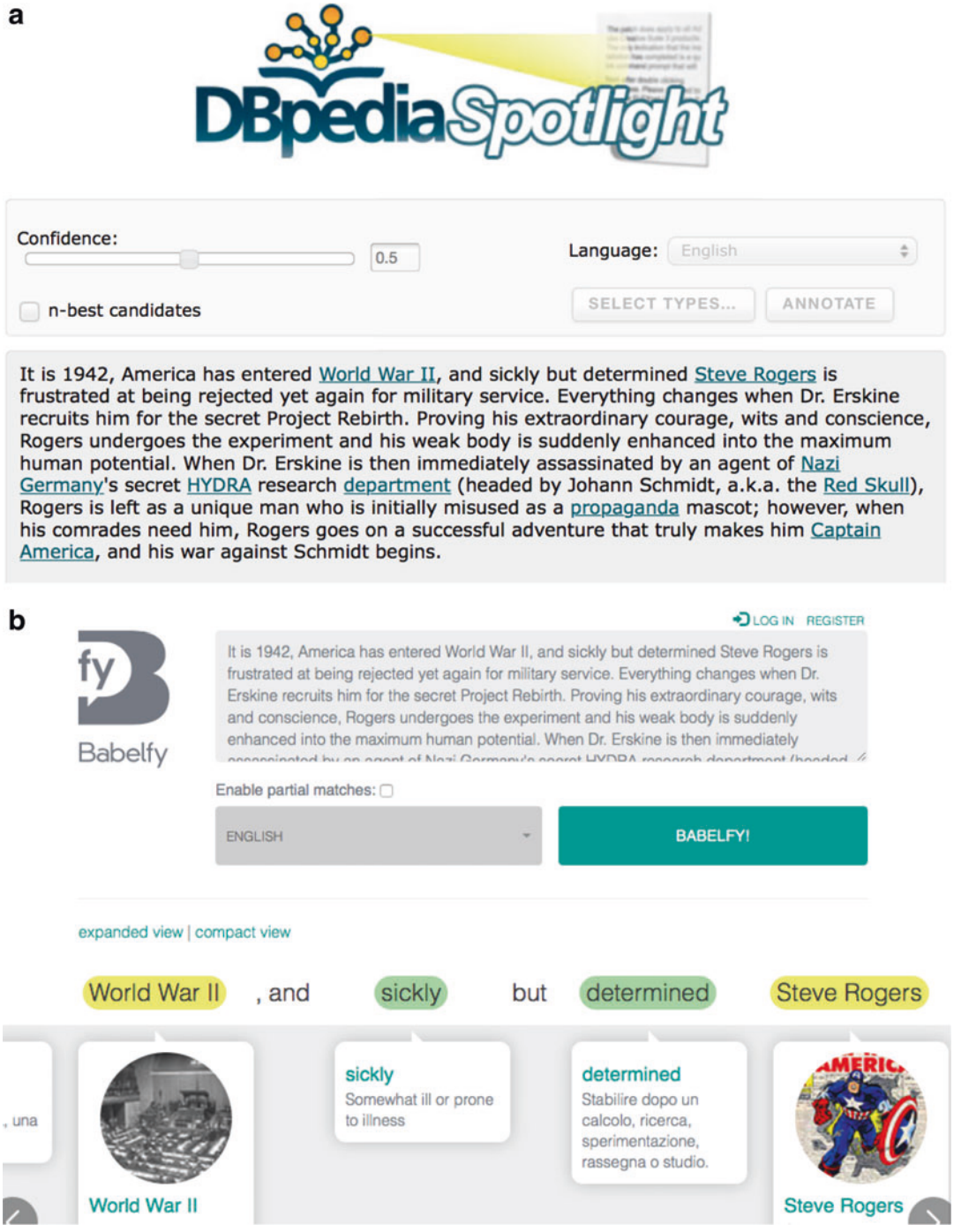
Recommender Systems Based on Linked Open Data, Fig. 3 Example of direct linking of movies

and URIs or the set of URIs found in each item description, depending on the selected module.

Direct Item Linking This approach is the more straightforward way for accessing `LOD` datasets. However, it requires that items have to be `Linked Open Data` resources, otherwise it cannot be used. As shown in Fig. 3, using movie title and year information it is possible to find the relative DBpedia resource. However, it is important to solve possible cases of ambiguity. The simplest solution is to exploit the class of the ontology that the item belongs to. For instance, in the movie domain we may select the resources with class `dbo:Film`.

Item Description Linking This approach bases on the exploitation of side information about the items such as textual descriptions or attributes. Such information can be used as input for entity linking tools in order to have access to `LOD` resources and link them to the item. Specifically, Entity Linking is the task of linking the entity mentioned in the text with the corresponding real world entity in the existing knowledge base (Shen et al. 2012). Many Entity Linking tools have been proposed in the literature and made available on the Web. Some of them are: Babelfy (Moro et al. 2014), Dexter (Ceccarelli et al. 2013), DBpedia Spotlight (Mendes et al. 2011), TAGME (Ferragina and Scaiella 2010), and NERD (Rizzo and Troncy 2012). In Fig. 4, we show two examples of item description linking using Babelfy (http://babelfy.org) and DBpediaSpotlight (http://dbpedia-spotlight.github.io), where the initial text is a blog post related to the movie *Captain America: The First Avenger*.

Item Graph Analyzer This module is responsible of the extraction from the knowledge base of a descriptive and informative subgraph for each



Recommender Systems Based on Linked Open Data, Fig. 4 Example of item description linking using DBpediaSpotlight and Babelfy for the movie *Captain America: The First Avenger*

item that is a set of RDF triples somehow related to the item resource. Eventually, all the extracted portions of the original graph can be merged to obtain a specific knowledge graph representative of the domain of interest covered by the recommender. It takes as input the list of items URI returned by the Item Linker and returns a set of RDF triples for each item. Potentially, each item resource may be connected to a big portion of the LOD graph. However, not all entities and relations may be informative and descriptive of the item content. Several strategies to select a relevant subset of RDF triples for each item may be considered and adopted (Musto et al. 2016; Ragone et al. 2017). One strategy can be to manually define a set of properties or sequences of properties by using some domain knowledge. SPARQL queries are a powerful tool to prefilter subgraphs relevant for the recommendation task.

Evaluating LOD-Based RSs

There are many datasets available for the evaluation of recommender systems. However, such datasets are not appropriate for evaluating LOD-based recommendation algorithms because they do not contains links to URIs. In order to evaluate LOD-based RSs, three datasets belonging to different domains (movies, music, and book) have been processed to compute a mapping between items (movies, artists, books) and their corresponding DBpedia URIs. The mappings for the datasets are available at https://github.com/sisinflab/LODrecsys-datasets.

Movielens This dataset is based on the MovieLens 1 M dataset (http://www.grouplens.org/node/73) released by the GroupLens research group. The original dataset contains 1,000,209 1–5 stars ratings given by 6040 users to 3883 movies. We found a valid mapping for 3300 out of the all movies.

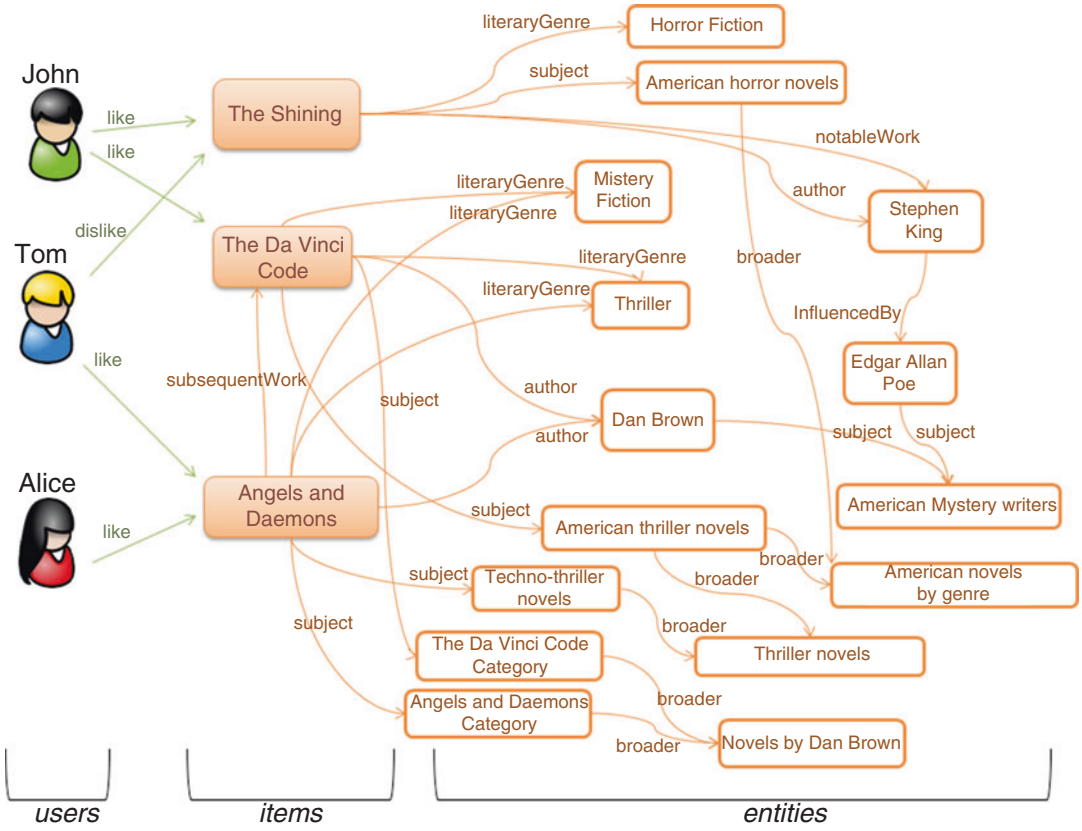
LibraryThing Derived from the LibraryThing (http://www.librarything.com) dataset (http://www.macle.nl/tud/LT/). This dataset is related to the book domain and contains 7112 users, 37,231 books and 626,000 ratings ranging from 1 to 10. In this case, we found a match for 11,694 books.

LastFM Differently from the previous ones, this third dataset is based on implicit feedback consisting of user-artist listening data. Its data come from recent initiatives on information heterogeneity and fusion in recommender systems (http://ir.ii.uam.es/hetrec2011/datasets.html) (Cantador et al. 2011) and has been built on top of the Last.fm music system (http://www.lastfm.com). The original dataset contains 1892 users, 17,632 artists and 92,834 relations between a user and a listened artist together with their corresponding listening counts. For this dataset, we found a match for 11,180 out of all artists.

Key Applications

Graph-Based Recommender Systems Using LOD

Recently, graph-based methods have been proposed and analyzed for implementing hybrid recommender systems, namely able to leverage collaborative and content-based information agglomerated in a uniform graph-based representation. **SPrank** (Semantic Path-based ranking), presented in Ostuni et al. (2013) computes top-N item recommendations in a learning to rank setting and combines ontological knowledge belonging to the Web of Data with collaborative user preferences in a unified graph-based data model. SPrank explores *paths* in a *semantic* graph to find items that are related to the ones the user is interested in. Specifically, it uses *path-based* features to represent the connectivity between users and items exploiting the multirelational structure of the graph and eventually, on top of such features it applies a *learning to rank* algorithm for getting a ranking function able to recommend the most relevant items to the user. Let us refer to the user-item ratings matrix with $\widehat{R}$, where each entry $\widehat{r}_{ui}$ represents the rating of user u on item i. We can use $\widehat{R}$ for building a bipartite graph where users and items are the nodes and users' feedback



Recommender Systems Based on Linked Open Data, Fig. 5 Example of the graph data model G

are the links. We then define a labeling function ψ_R: $\mathbb{R} \rightarrow R$ for mapping the ratings to symbolic user-item relations expressing the item relevance for the user. In this way, we are able to extend the original knowledge graph with collaborative information thus representing all the information in an integrated and uniform way (see Fig. 5).

The multirelational graph is defined as $G = \{t \mid t \in V \times \Sigma \times V\}$, where V denotes the set of vertices and Σ indicates the set of edge labels or relations. Three relevant subsets of V are defined: U, I, and E representing *users*, *items*, and *entities*, respectively. As $V = U \cup E$ and $I \subseteq E$, we consider items as a particular type of entities. Similarly, Σ contains two categories of relationships. In fact, we have $\Sigma = R \cup P$ where $R = \{r \mid \exists (u, r, i) \in G \wedge u \in U \wedge i \in I\}$ and $P = \{p \mid \exists (e_j, p, e_k) \in G \wedge e_j, e_k \in E\}$. In other words, a relation $r \in R$ links a user $u \in U$ to an item $i \in I$ while a relation $p \in P$ connects either an item i to another entity $e \in E$ in the graph or an entity $e_j \in E \setminus I$ to another entity $e_k \in E$. We will use u, i, e, and v to represent a generic node belonging to U, I, E, and V, respectively. Analogously, we will denote with σ, p, and r generic relations in Σ, P and R. For each $(v_j, \sigma, v_k) \in G$, we assume there always exists (v_k, σ^{-1}, v_j) where σ^{-1} represents the inverse relation of σ. When no confusion arises, we write $\sigma = (v_j, v_k)$ as an alternative representation of (v_j, σ, v_k).

Figure 5 shows an example of G where U = {Alice, John, Tom} and I = {AngelsAndDaemons, TheDaVinciCode, TheShining}. In this example, the user ratings are binary R = {like, dislike}. All the other relations belong to P = {literaryGenre, subject, author, notableWork, influencedBy, subsequentWork, broader}. Differently from traditional

approaches, SPrank directly formulate the problem of computing *top-N* recommendations in the standard learning to rank setting adopted in Web search (Liu 2009) by replacing queries with users and document with items and using user's ratings as relevance scores. For each user-item pair $(u, i) \in U \times I$, SPrank encodes the matching between user interests and item content in the feature vector $\mathbf{x_{ui}} \in \mathbb{R}^D$, where D is the dimension of the feature space. Each component in $\mathbf{x_{ui}}$ represents the relevance of i for u with respect to a specific feature. For each user $u \in U$, $I_u = \left\{ i \in I | \widehat{r}_{ui} \in \widehat{R} \right\}$ is the set of items rated by u. The ratings $\widehat{r}_{ui}$ associated to the items in I_u can be used to induce a ranking among them.

Path-Based Features Given the multirelational graph G, we define `path` the sequence of relations of the form $(r_1 \ldots \sigma_l \ldots \sigma_L)$ such that $r_1 = (u, v_1)$, $\sigma_l = (v_{l-1}, v_l)$, and $\sigma_L = (v_{L-1}, i)$ with $u \in U$, $i \in I$. We refer to the actual sequence of nodes and relations $u \rightarrow^{r_1} v_1 \ldots \rightarrow^{\sigma_1} \ldots v_{L-1} \rightarrow^{\sigma_L} i$ which generates the particular path as *path instance*. We also define the *length of a path* as the number of relations contained within such path. We only consider paths having length greater than 1 and less or equal than a given L. Considering a user-item pair (u, i), we denote with $\#path_{u,i}(j)$ the number of path instances between u and i corresponding to the specific $Path(j)$ entry in the index. In other words, it represents how many paths of type $Path(j)$ connect u and i. Then, we define the j-th component in the feature vector $\mathbf{x_{ui}}$ as:

$$x_{\mathrm{ui}}(j) = \frac{\#path_{u,i}(j) - \min_{i \in I}\left(path_{u,i}(j)\right)}{\max_{i \in I}\left(path_{u,i}(j)\right) - \min_{i \in I}\left(path_{u,i}(j)\right)} \tag{1}$$

Equation 1 represents the importance of the path $Path(j)$ between u and i in the graph involving all nodes reachable in L hops starting from u. Given the user u and considering the specific path $Path(j)$, we observe how this path is distributed among all items for that user. Once we have computed all path count values $\#path_{ui}(j)$, we can compute $\min_{i \in I}(\#path_{ui}(j))$ and $\max_{i \in I}(\#path_{ui}(j))$ values for each user-path combination. For example, for user `Alice` referring to $Path(10)$, we have $\max_{i \in I}(\#path_{\texttt{Alice},i}(10)) = 2$ and $\min_{i \in I}(\#path_{\texttt{Alice},i}(10)) = 0$. Indeed we have $\#path_{\texttt{Alice},\texttt{TheDaVinciCode}}(j) = 2$ with $j \in \{8, 10\}$ while we have $\#path_{\texttt{Alice},\texttt{AngelsAndDaemons}}(j) = 0$ for $j \in \{1 \ldots 20\}$. Finally we can compute the feature vectors using Formula 1.

Experimental results presented in (Ostuni et al. 2013) demonstrated that SPrank improve the quality recommendation respect to several state-of-the-art methods for implicit feedback in two datasets: MovieLens and Last.fm. A further and more comprehensive experimental analysis of SPrank and several collaborative-filtering and hybrid algorithms show that SPrank would be the best choice on dataset with fewer ratings and more content information, while other hybrid algorithms, properly adapted to exploit metadata, such as BPR-SSLIM and Factorization Machines can better exploit collaborative information (Di Noia et al. 2016).

Two **graph-based kernel methods** have been proposed in (Ostuni et al. 2016): the one based on entities and the other on paths, called *entity-based* and *path-based item neighborhood mapping*, respectively.

Entity-based item neighborhood mapping. In this mapping, each feature refers to an entity in E and the corresponding score represents the weight associated to that entity in G_i^h. The resulting feature vector $\phi_E\left(G_i^h\right)$ is:

$$\phi_E\left(G_i^h\right) = \left(w_{i,e_1}, w_{i,e_2}, \ldots w_{i,e_m}, \ldots w_{i,e_t}\right)$$

where the weight associated to the generic entity e_m is computed as follows:

$$w_{i,e_m} = \sum_{l=1}^{h} \alpha_l \cdot c_{l,e_m}$$

with

$$\alpha_l = \frac{1}{1 + \log(l)}$$

and

$$c_{l,e_m} = \left| \left\{ (e_n, p, e_m) \middle| e_n \in \widehat{E}_i^{l-1} \vee e_m \in \widehat{E}_i^{l} \right\} \cup \left\{ (e_m, p, e_n) \middle| e_m \in \widehat{E}_i^{l} \vee e_n \in \widehat{E}_i^{l-1} \right\} \right|$$

where $\widehat{E}_i^l = E_i^l \backslash E_i^{l-1}$ is the set of entities *exactly l* hops far from *i*.

In particular, c_{l,e_m} corresponds to the number of triples connecting e_m to entities in the previous hop $(l-1)$, whether e_m appears either as subject or object of the triple. In other words, c_{l,e_m} can be seen as the *occurrence* of the entity e_m in the item neighborhood at distance *l*. The more the entity e_m is connected to neighboring entities of *i*, the more it is descriptive of *i*. α_l represents a decay factor depending on the distance *l* from the item *i*, whose aim is to incrementally penalize farther entities from the item. It allows us to take into account the *locality* of those entities in the graph neighborhood. The discount factor can also be parametrized defining a specific weight for each hop.

With reference to example discussed before, the $c_{l,em}$ values are computed as follows: $c_{1,e_1} = 1$, $c_{1,e_2} = 1$, $c_{1,e_4} = 1$, $c_{2,e_5} = 2$, $c_{2,e_6} = 1$, $c_{2,e_7} = 2$, $c_{2,e_8} = 1$, $c_{3,e_9} = 2$, $c_{3,e_{10}} = 2$, $c_{3,e_{11}} = 1$. All the others are zero.

Path-based item neighborhood mapping. Differently from the previous mapping, in the `path-based item neighborhood mapping` a feature is represented as a sequence of nodes in *G*. Given two entities e_1 and e_n, we consider the sequence of nodes $e_1 \cdot e_2 \cdot \ldots \cdot e_{n-1} \cdot e_n$ met while traversing the graph to go from e_1 to e_n and we refer to such sequence as *path*. Therefore, each feature refers to several variants of paths rooted in the item node. We first collect all the paths rooted in *i* which can be indicated as sequence of entities $i \cdot e_1 \cdot e_2 \cdot \ldots \cdot e_{n-1} \cdot e_n$. Then, from those paths, other features are defined considering every sub-paths. Specifically sub-paths are composed by only those entities progressively farther from the item. Considering the path given above we build the following features: $e_1 \cdot e_2 \cdot \ldots \cdot e_{n-1} \cdot e_n$, $e_2 \cdot \ldots \cdot e_{n-1} \cdot e_n, \ldots, e_{n-1} \cdot e_n, e_n$. This choice allows to explicitly represent substructures shared between items with no overlapping in their immediate neighborhoods but somehow connected at farther distance. Items connected to the same entities have same common structures because both closer and farther entities are shared. Items connected to different entities which are however linked directly or at a farther distance to same entities share less or none sub-paths depending on how much far the common entities are, if any.

More formally, let P_i be the set of paths rooted in *i* and P_i^* be the list of all possible sub-paths extracted from them. We use $p_m(i)$ and $p_m^*(i)$ to refer to the $m - th$ elements in P_i and $P_m^*(i)$, respectively. Then, the feature mapping for item *i* is:

$$\phi_P(G_i^h) = \left(w_{i,p_1^*}, w_{i,p_2^*}, \ldots w_{i,p_m^*}, \ldots, w_{i,p_t^*} \right)$$

where each w_{i,p_m^*} is computed as:

$$w_{i,p_m^*} = \frac{\#p_m^*(i)}{|\, p_m \,| - |\, p_m^* \,|}$$

where $|p_m|$ indicates the length of path p_m and $\# p_m^*(i)$ the occurrence of $p_m^*(i)$ in P_i^*. The denominator is a discounting factor which takes into account the difference between the original path p_m and its subpath p_m^*. The shorter the subpath, the more the discount, because it contains entities farther from the item.

Experimental Evaluation

The algorithms we introduced in the previous section have been extensively evaluated in order to determinate their ability to provide valuable recommendations for the users. SPrank has been compared against several baselines: a popularity-based baseline (MostPop), a matrix factorization-based method optimized with Bayesian Personalized Ranking optimization criterion (BPR-MF), a weighted matrix factorization method (WRMF), a Sparse Linear method for learning a sparse aggregation coefficient matrix (SLIM), SLIM with side information and BPR optimization criterion (BPR-SSLIM), hybrid matrix factorization method able to work with sparse datasets (BPRLinear), a

Recommender Systems Based on Linked Open Data, Table 1 Comparative results on Last.fm. The differences between SPRank and the comparative approaches are statistically significant according to the Student's paired t-test with $p < 0.0001$ (except with respect to BPRMF in terms of prec@10 with $p > 0.1$)

Alg	nDCG@10	prec@10	rec@10
SPRank	**0.1918**	**0.0841**	0.2043
SLIM	0.1395	0.0723	0.1548
BPRMF	0.1496	0.0785	0.1523
MostPop	0.0684	0.0537	0.0756
BPRLin	0.1058	0.0521	0.1235
RankMF	0.0689	0.0487	0.0876
MCMC-FM	0.0317	0.0102	0.0574
KNN-FM	0.0248	0.0081	0.0445
Ranking-FM	0.0460	0.0160	0.0902
HeteRec	0.0669	0.0207	0.1199
PathRank	0.1442	0.0387	0.1878
BPR-SSLIM	0.1737	0.0467	**0.2605**

Recommender Systems Based on Linked Open Data, Table 2 Comparative results on Movielens. The differences between SPRank and the comparative approaches are statistically significant according to the Student's paired t-test with $p < 0.0001$ (for Ranking-FM the differences are still significant but with $p < 0.01$)

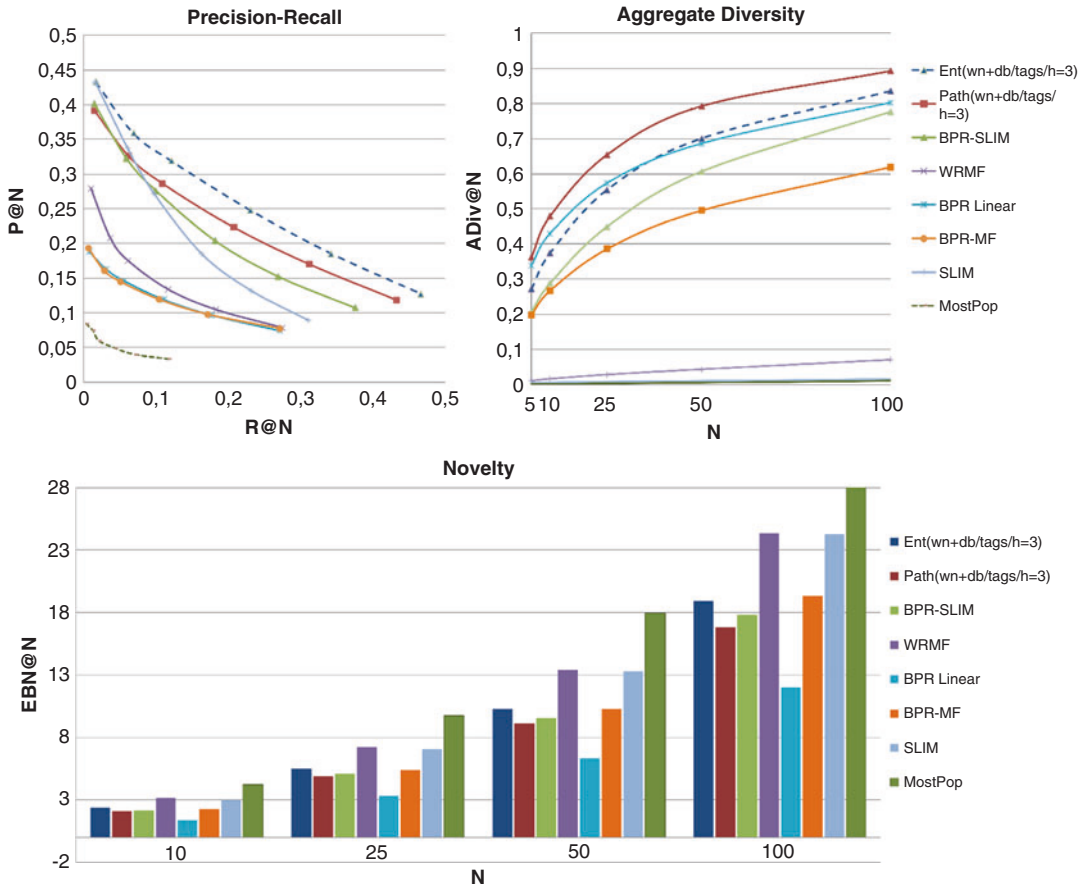
Alg	nDCG@10	prec@10	rec@10
SPRank	0.4056	0.1613	0.0785
MF	0.3618	0.0945	**0.0422**
SLIM	0.4597	0.2631	0.1469
BPR-MF	0.4359	0.2572	0.1354
MostPop	0.3915	0.1663	0.0753
BPRLin	0.3752	0.1437	0.0731
RankMF	0.3695	0.1428	0.0739
MCMC-FM	0.3576	0.0902	0.0414
KNN-FM	0.3521	0.0870	0.0388
Ranking-FM	0.4179	0.1850	0.0846
HeteRec	0.3404	0.0884	0.0478
PathRank	0.3477	0.0881	0.0488
BPR-SSLIM	**0.4674**	**0.2636**	0.1439

Soft Margin Ranking Matrix Factorization (RankingMF), three different versions of Factorization Machines based method (MCMC-FM, KNN-FM, Ranking-FM), and two graph-based methods (HeteRec, PathRank). As shown in Table 1, SPrank shows its effectiveness on dataset with fewer ratings and more content information, as Last.fm in this case that is the most sparse and enriched dataset. As MovieLens is the less sparse and enriched dataset in this work, results in Table 2 show that SPrank is not as effective as the other hybrid learning to rank approaches where the collaborative component plays an important role. This is confirmed by the results on LibraryThing (Table 3) that has intermediate statistical values and where SPrank strongly overcomes almost all the other algorithms. Summing up, SPrank is an interesting alternative for those datasets where the rating matrix is sparse but the content-based part may benefit from rich LOD datasets. A more comprehensive discussion about these results is shown in (Di Noia et al. 2016).

The two graph-based kernel methods showed in section "Graph-Based Recommender Systems Using LOD" have been compared against some of the aforementioned baselines on Last.fm. Figure 6

Recommender Systems Based on Linked Open Data, Table 3 Comparative results on LibraryThing. The differences between SPRank and the comparative approaches are statistically significant according to the Student's paired t-test with $p < 0.0001$ (except with respect to BPR-SSLIM in terms of nDCG@10 with p-value >0.1)

Alg	nDCG@10	prec@10	rec@10
SPRank	**0.3068**	**0.1019**	0.2232
MF	0.1423	0.017	0.0209
SLIM	0.2317	0.0543	0.0988
BPRMF	0.1858	0.0449	0.0751
MostPop	0.1598	0.0397	0.0452
BPRLin	0.275	0.0794	0.1723
RankMF	0.1714	0.0369	0.0459
MCMC-FM	0.1421	0.0362	0.1299
KNN-FM	0.1723	0.0249	0.0655
Ranking-FM	0.1958	0.0431	0.0798
HeteRec	0.2894	0.0930	0.2125
PathRank	0.2568	0.0521	0.1116
BPR-SSLIM	0.3051	0.0962	**0.2328**



Recommender Systems Based on Linked Open Data, Fig. 6 Precision-recall, novelty, and aggregate diversity plots in Last.fm dataset

shows the experimental results. The precision-recall show that the entity-based kernel is the most accurate method on both the dataset, and the path-based method is almost always the second best one. In terms of aggregate diversity (ADiv@N in the plots), the path-based method obtains the best results, while the other overcomes all the other methods except for BPRLinear. Finally, the results in terms of novelty demonstrate that such kernel methods have trends similar to the other methods, or even better. Therefore, the two graph-based kernel methods can recommend less popular items with higher accuracy than other collaborative filtering and hybrid algorithms in music domain, providing a better level of personalization and better exploring the long tail in both artists and songs recommendation scenario. A more comprehensive evaluation of the two kernel methods is available in (Ostuni et al. 2016).

Future Directions

Recommender systems can be considered as a killer application for the exploitation of the huge knowledge encoded in LOD datasets freely available on the Web. Although, many solutions have been proposed and implemented to deliver this new generation of semantics-aware recommendation engines, some important issues remain still open to a deeper investigation. Among them, we cite the most important ones: feature selection, distributed computing, cross-domain recommendation, computing explanation, and the role of formal reasoning in the recommendation process.

Feature selection: The richness of Linked Open Data datasets may result in a pitfall for data-intensive tasks (as computing recommendations) as they potentially introduce noise in the data. Selecting the right features in LOD-enabled recommender systems results not just in getting the minimal meaningful subset of properties which are directly connected to an item but, given the graph-based nature of the underlying data, the minimal meaningful set of semantic paths, of arbitrary length, which result representative of the item itself.

Distributed computing: Over the last years, solutions to horizontally distribute graph-based data manipulation have been proposed also boosted by the increasing production of data coming from social networks. All the methods, algorithms, and frameworks work quite well with multirelational graphs where the number of possible relations are just a few compared to that of Linked Open Data. New approaches need to be proposed and developed to easily deal with all the semantics encoded in LOD datasets.

Cross-domain recommendation: The highly interconnected nature of datasets such as DBpedia or Wikidata represents an opportunity to develop cross-domain recommender systems. That is, systems able to recommend items in a knowledge domain which is not the same of the user profile. As an example, we may be able to recommend books given the user profile collects information on movies.

Computing explanation: Sometimes, receiving recommendations may result frustrating as we do not know the reason why the system suggested such items to us. Computing explanation for recommendations has been identified as a *must have* feature for the new generation of recommender systems. In this direction, all the knowledge available as Linked Open Data may surely play a key role.

Formal reasoning: LOD datasets are not just a mere collection of data represented in a graph-based way. They usually refer to a rich ontology which in turns can be represented by means of expressive logical languages as Description Logics. The adoption of such languages enables the application of formal logical reasoning over the underlying data. As of today, due to its high computational complexity, such reasoning has not been exploited to its full potential, but it can surely add new value to real knowledge-enabled recommender systems.

Cross-References

- ▶ Linked Open Data
- ▶ Recommender Systems
- ▶ Recommender Systems Evaluation
- ▶ Recommender Systems, Semantic-Based
- ▶ Recommender Systems: Models and Techniques

References

Adomavicius G, Tuzhilin A (2005) Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. IEEE Trans Knowl Data Eng 17(6):734–749

Anand SS, Kearney P, Shapcott M (2007) Generating semantically enriched user profiles for web personalization. ACM Trans Internet Technol 7(4):22

Ayala VAA, Przyjaciel-Zablocki M, Hornung T, Schätzle A, Lausen G (2014) Extending sparql for recommendations. In: Proceedings of semantic web information management on semantic web information management, SWIM'14, ACM, New York, 2014, pp 1:1–1:8

Balabanović M, Shoham Y (1997) Fab: content-based, collaborative recommendation. Commun ACM 40(3):66–72

Bizer C, Heath T, Berners-Lee T (2009) Linked data – the story so far. Int J Semantic Web Inf Syst 5(3):1–22

Bollacker K, Evans C, Paritosh P, Sturge T, Taylor J (2008) Freebase: a collaboratively created graph database for structuring human knowledge. In: Proceedings of the 2008 ACM SIGMOD international conference on management of data, SIGMOD '08, ACM, New York, 2008, pp 1247–1250

Breese JS, Heckerman D, Kadie C (1998) Empirical analysis of predictive algorithms for collaborative filtering. In: Proceedings of the fourteenth conference on uncertainty in artificial intelligence, UAI'98. Morgan Kaufmann, San Francisco, pp 43–52

Burke RD (2007) Hybrid web recommender systems. In: The adaptive web, methods and strategies of web personalization. Springer, Berlin/Heidelberg, pp 377–408

Cantador I, Bellog´ın A, Castells P (2008) A multilayer ontology-based hybrid recommendation model. AI Commun Special Issue on Rec Sys 21(2–3):203–210

Cantador I, Brusilovsky P, Kuflik T (2011) Second workshop on information heterogeneity and fusion in recommender systems (hetrec 2011). In: Proceedings of the 5th ACM conference on recommender systems, RecSys 2011, ACM, New York, 2011

Castells P, Hurley NJ, Vargas S (2015) Novelty and diversity in recommender systems. Springer, Boston, pp 881–918

Ceccarelli D, Lucchese C, Orlando S, Perego R, Trani S (2013) Dexter: an open source framework for entity linking. In: Proceedings of the sixth international workshop on exploiting semantic annotations in information retrieval, ESAIR '13, ACM, New York, 2013, pp 17–20

Di Noia T, Mirizzi R, Ostuni VC, Romito D, Zanker M (2012) Linked open data to support content-based recommender systems. In: Proceedings of the 8th international conference on semantic systems, I-SEMANTICS '12, ACM, New York, 2012, pp 1–8

Di Noia T, Ostuni VC, Tomeo P, Di Sciascio E (2016) Sprank: semantic path-based ranking for top-n recommendations using linked open data. ACM Trans Intell Syst Technol (TIST) 8(1):9:1–9:34

Di Noia T, Rosati J, Tomeo P, Di Sciascio E (2017) Adaptive multi-attribute diversity for recommender systems. Inform Sci 382–383:234–253

Dojchinovski M, Vitvar T (2014) Personalised access to linked data. EKAW 8876:121–136

Fernández-Tob´ ıas I, Cantador I, Kaminskas M, Ricci F (2011) A generic semanticbased framework for cross-domain recommendation. In: Proceedings of the 2nd international workshop on information heterogeneity and fusion in recommender systems, HetRec '11, ACM, New York, 2011, pp 25–32

Ferragina P, Scaiella, U (2010) Tagme: on-the-fly annotation of short text fragments (by wikipedia entities). In: Proceedings of the 19th ACM international conference on information and knowledge management, CIKM '10, ACM, New York, 2010, pp 1625–1628

de Gemmis M, Lops P, Musto C, Narducci F, Semeraro G (2015) Semantics-aware content-based recommender systems. Springer, Boston, pp 119–159

Gunawardana A, Shani G (2015) Evaluating recommender systems. Springer, Boston, pp 265–308

Heitmann B, Hayes C (2010) Using linked data to build open, collaborative recommender systems. In: AAAI spring symposium: linked data meets artificial intelligence. AAAI Press, Stanford

Jannach D, Zanker M, Felfernig A, Friedrich G (2010) Recommender systems and the next-generation web. In: Recommender systems. Cambridge University press, Cambridge Books Online, New York, pp 253–288

Jin R, Chai JY, Si L (2004) An automatic weighting scheme for collaborative filtering. In: Proceedings of the 27th annual international ACM SIGIR conference on research and development in information retrieval, SIGIR '04, ACM, New York, 2004, pp 337–344

Khrouf H, Troncy R (2013) Hybrid event recommendation using linked data and user diversity. In: Proceedings of the 7th ACM conference on recommender systems, RecSys '13, ACM, New York, 2013, pp 185–192

Lehmann J, Isele R, Jakob M, Jentzsch A, Kontokostas D, Mendes PN, Hellmann S, Morsey M, van Kleef P, Auer S, Bizer C (2015) DBpedia – a large-scale, multilingual knowledge base extracted from wikipedia. Semant Web J 6(2):167–195

Liu T-Y (2009) Learning to rank for information retrieval. Found Trends Inf Retr 3(3):225–331

Lommatzsch A, Plumbaum T, Albayrak S (2011) A linked dataverse knows better: boosting recommendation quality using semantic knowledge. In: Proceedings of the 5th International Conference on advances in semantic processing, IARIA, Wilmington, 2011, pp 97–103

Maidel V, Shoval P, Shapira B, Taieb-Maimon M (2008). Evaluation of an ontologycontent based filtering method for a personalized newspaper. In: Proceedings of the 2008 ACM conference on recommender systems, RecSys 2008, Lausanne, 23–25 Oct 2008. ACM, New York, pp 91–98

Marie N, Corby O, Gandon F, Ribière M (2013) Composite interests' exploration thanks to on-the-fly linked data

spreading activation. In: Proceedings of the 24th ACM conference on hypertext and social media, HT '13. ACM, New York, pp 31–40

Mendes PN, Jakob M, A. Garc´ıa-Silva, Bizer, C (2011) Dbpedia spotlight: shedding light on the web of documents. In: Proceedings of the 7th international conference on semantic systems, I-semantics '11, ACM, New York, 2011, pp 1–8

Middleton SE, Shadbolt NR, De Roure DC (2004) Ontological user profiling in recommender systems. ACM Trans Inf Syst 22:54–88

Mobasher, B, Jin X, Zhou Y (2004) Semantically enhanced collaborative filtering on the web. In: Berendt B, Hotho A, Mladenic D, Someren M, Spiliopoulou M, Stumme G (eds) Web mining: from web to semantic web, vol 3209 of lecture notes in computer science. Springer, Berlin/Heidelberg, pp 57–76

Moro A, Raganato A, Navigli R (2014) Entity linking meets word sense disambiguation : a unified approach. Trans Assoc Comput Linguist 2:231–244

Musto C, Semeraro G, Lops P, de Gemmis M (2014) Combining distributional semantics and entity linking for context-aware content-based recommendation. In: User modeling, adaptation, and personalization – 22nd international conference, UMAP, Aalborg, 7–11 July 2014, pp 381–392

Musto C, Lops P, Basile P, de Gemmis M, Semeraro G (2016) Semantics-aware graph-based recommender systems exploiting linked open data. In: Proceedings of the 2016 conference on user modeling adaptation and personalization, UMAP '16, ACM, New York, 2016, pp 229–237

Nguyen P, Tomeo P, Di Noia T, Di Sciascio E (2015a) An evaluation of simrank and personalized pagerank to build a recommender system for the web of data. In: Proceedings of the 24th international conference on World Wide Web, WWW '15 companion, ACM, New York, 2015, pp 1477–1482

Nguyen PT, Tomeo P, Di Noia T, Di Sciascio, E (2015b) Content-based recommendations via DBpedia and Freebase: a case study in the music domain. Springer International Publishing, Cham, pp 605–621

Ostuni, VC, Di Noia, T, Di Sciascio, E, Mirizzi R (2013) Top-n recommendations from implicit feedback leveraging linked open data. In: Proceedings of the 7th ACM conference on recommender systems, RecSys '13, ACM, New York, 2013, pp 85–92

Ostuni VC, Oramas S, Di Noia T, Serra X, Di Sciascio E (2016) Sound and music recommendation with knowledge graphs. ACM Trans Intell Syst Technol, 8(2):21:1–21:21

Passant A (2010) Measuring semantic distance on linking data and using it for resources recommendations. In: AAAI spring symposium: linked data meets artificial intelligence. AAAI, 2010

Pazzani M, Billsus D (1997) Learning and revising user profiles: the identification ofinteresting web sites. Mach Learn 27(3):313–331

Pazzani MJ, Billsus D (2007) Content-based recommendation systems. In: The adaptive web. Springer, Berlin/Heidelberg, pp 325–341

Ragone A, Tomeo P, Magarelli C, Di Noia T, Palmonari M, Maurino A, Di Sciascio E (2017) Schema-summarization in linked-data-based feature selection for recommender systems. In: 32nd ACM SIGAPP symposium on applied computing. ACM, New York

Ricci F, Rokach L, Shapira B, eds (2015) Recommender systems handbook. Springer, New York

Rizzo G, Troncy R (2012) Nerd: a framework for unifying named entity recognition and disambiguation extraction tools. In: Proceedings of the demonstrations at the 13th conference of the European chapter of the Association for Computational Linguistics, EACL '12, Association for Computational Linguistics, Stroudsburg, 2012, pp 73–76

Rowe M (2014a) Semanticsvd++: incorporating semantic taste evolution for predicting ratings. In: 2014 IEEE/WIC/ACM international conferences on web intelligence, Wi 2014. IEEE Computer Society, Washington, DC

Rowe M (2014b) Transferring semantic categories with vertex kernels: recommendations with semanticsvd++. In: The semantic web – ISWC 2014 – 13th international semantic web conference, 2014. Springer, New York

Sarwar B, Karypis G, Konstan J, Riedl J (2001) Item-based collaborative filtering recommendation algorithms. In: Proceedings of the 10th international conference on World Wide Web, WWW '01, 2001, pp 285–295

Schafer JB, Frankowski D, Herlocker J, Sen S (2007) Collaborative filtering recommender systems. In: The adaptive web. Springer, Berlin/Heidelberg, pp 291–324

Schwartz B (2005). The paradox of choice: why more is less. Harper Perennial, New York City

Semeraro G, Lops P, Basile P, de Gemmis M (2009). Knowledge infusion into content-based recommender systems. In: Proceedings of the third ACM conference on recommender systems, RecSys '09, ACM, New York, 2009, pp 301–304

Shen W, Wang J, Luo P, Wang M (2012) Linden: linking named entities with knowledge base via semantic knowledge. In: Proceedings of the 21st international conference on World Wide Web, WWW '12, ACM, New York, 2012, pp 449–458

Speier C, Valacich JS, Vessey I (1999) The influence of task interruption on individual decision making: an information overload perspective. Decis Sci 30 (2):337–360

Tomeo P, Fernández-Tob´ ıas I, Di Noia T, Cantador I (2016) Exploiting linked open data in cold-start recommendations with positive-only feedback. In: Proceedings of the 4th Spanish conference on information retrieval, CERI '16, ACM, New York, 2016, pp 11:1–11:8

Ziegler C-N, Lausen G, Schmidt-Thieme L (2004) Taxonomy-driven computation of product recommendations. In: Proceedings of the thirteenth ACM international conference on information and knowledge management, CIKM '04, ACM, New York, 2004, pp 406–415

Recommender Systems Based on Social Networks

Fatemeh Vahedian and Robin Burke
College of Computing and Digital Media, DePaul University, Chicago, IL, USA

Synonyms

Collaborative recommendation; Content-based recommendation; Link prediction; Recommendation systems

Glossary

Recommender system	System that can automatically produce personalized lists of items or products to help a user find the most relevant items or information
Link prediction	The task of predicting missing or yet-to-be formed edges in a network
Trust-aware recommender systems	Recommender systems that, in addition to user rating or user preference data, consider the user trust relations in order to generate recommendation

Definition

Recommender systems (or recommendation systems) provide personalized suggestions to users in domains where the volume and variety of products and information make other forms of search and navigation ineffective and time-consuming. Recommender systems have been widely deployed throughout the information ecosystem, particularly in electronic commerce. Social networks can be used as an input to a recommendation process (as in trust-aware recommendation), as the domain in which recommendations are sought (as in link prediction in online social networks), or as a general data model supporting recommendation algorithms (as in network-oriented formulations of the recommendation problem).

Introduction

Recommender systems are personalized information access systems and are a common feature of electronic commerce and information access systems. For example, an e-commerce retailer might present returning users with a sidebar consisting of products "recommended for you" or might enhance the presentation of a product by listing products that other users bought as well under the heading "Users Also Bought." These types of interactions are quite familiar to contemporary users of the World Wide Web. Recommender systems research encompasses a wide variety of approaches and methodologies surveyed in several recent texts (Aggarwal 2016; Jannach et al. 2010; Ricci et al. 2010).

Key Points

Social networks may relate to recommender systems in three key ways:

1. A social network may provide addition sources of data to a recommender system. When that data is about items and their properties, we would consider this a semantic-based recommender system: see "Recommender System, Semantic-based." The data may represent relationships between users, such as explicit trust links, which can be used to find reliable peers. A social network may also enhance what is known about items. For example, a communication network in an organization may be useful in determining similarity between experts for the purpose of expertise recommendation.
2. Alternatively, a social network may also serve as the domain in which recommendation is

generated. For example, social networking sites often recommend to users individuals whom they may wish to connect: In LinkedIn, this is the "You may also know" feature.

3. Finally, there are approaches to recommendation in which the data about users and items is represented as a network, and recommendations are derived by considering paths through the network. This can take a variety of forms, including approaches derived from link prediction research in network science and those that build on more complex representations such as bipartite or complex heterogeneous networks.

Historical Background

The recommendation problem may have different formulations depending on the role that recommendation plays in a given system or interface. We will concentrate in this chapter on the "Recommended Items" problem (Herlocker et al. 2004), where the task of the recommender system is to present the user with a ranked list of items that she is likely to prefer.

Without loss of generality, we can formulate the ranked list problem as one of deriving a user-specific scoring function for items, where the high-scoring items are those predicted to be of most interest to the target user. With such a function in hand, the generation of a ranked list is trivial.

More formally, let U be the set of all users and let I be the set of all possible items that can be recommended. ψ is a utility function measuring the usefulness of item i to user u, $\psi : U*I \rightarrow R$ where R is a real value.

Conceptually then, the recommendation function R takes a user and a list length k and returns a ranked list of the k items of highest preference for that user:

$$R(u,k) = sort(top_k(argMax_{i \in I}\psi(u,i))) \quad (1)$$

Actual recommender systems may realize this function through a wide variety of implementations, but the key points are that the recommendations are personalized to each user and can be produced on demand based on what is known about the user at that time.

Recommendation Types

Personalization is achieved by having a representation of the user's needs or interests. This representation is known at the *user profile*. Recommendation techniques differ based on the types of user profiles that they maintain and how those profiles are applied to the recommendation task.

There are three well-studied and widespread approaches to achieving personalization recommendation: collaborative, content based, and knowledge based. Collaborative recommendation (also known as collaborative filtering) assumes a database of profiles that consist of user's preferences for items. For example, a movie recommender might have profiles in which users have rated movies that they have seen. The estimation of $\psi(u,i)$ for some movie i that u has not rated is performed by extrapolating from the ratings of other users who have seen i. The other user's profiles are essential to the technique, which is why this is a "collaborative" technique.

Content-based recommendation can take a variety of forms, but often a user profile consists, as before, of a collection of items and associated ratings for each user. However, in addition, a database of meta-data features for each item is also required. For example, in a restaurant recommender, these might be the type of cuisine served, the price range, and other data. The content-based recommender acquires its function ψ for each user by noting what features appear to be liked and disliked by the user, essentially learning a classifier for each user.

The third type of recommendation is knowledge-based recommendation in which the system, in addition to having data about items and profile information from users, also has knowledge about how items suit users' needs. The determination of the usefulness of an item can therefore involve an inferential process.

Each basic type of recommendation has its strengths and weaknesses (Burke and Ramezani 2011), and for this reason, a variety of hybrid recommender systems have been created that meld these techniques and types of data in different ways.

Collaborative Recommendation

Collaborative recommendations are based on the ratings or behavior of other users in the system. The main assumption of this recommendation approach is that other user's opinions can be selected and aggregated in such a way as to provide a reasonable prediction of the target user's preference. Intuitively, these algorithms assume that similar users to you most likely have similar opinions on items as you do; therefore their opinions can be recommendations for you.

Conceptually, therefore, collaborative recommendation is a two-step process. The first step is to identify a set of peer users with tastes similar to the target user. The second step is to use the profiles of these peers to estimate the preferences of the target user.

A critical design decision in implementing collaborative recommendation is the choice of similarity function for peers. Several different similarity functions have been proposed and evaluated in the literature. A typical choice is the cosine similarity measure which for a user profile p_v and profile of active user p_u is defined as

$$Sim(u,v) = \frac{p_u \cdot p_v}{|| p_u || \quad || p_v ||} \tag{2}$$

Once similarities between users are known, the next task is to collect the k nearest neighbors of the target user and produce personalized score (Resnick et al. 1994). A standard technique is to make a list of K nearest neighbor for each active user u based on correlation measure and then calculate the user rating prediction for each unseen items. The personalized score for each pair of target user and item (u,i) is calculated as

$$\psi(u,i) = \overline{p}_u + \frac{\sum_{v \in N_u} Sim(u,v)(p(v,i) - \overline{p}_v)}{\sum_{v \in N_u} Sim(u,v)} \tag{3}$$

where the $Sim(u,v)$ denotes the similarity value between users u and v. Systems that implement a direct comparison between users (or items) to produce such predictions are known as memory-based collaborative recommendation systems (Adomavicius and Tuzhilin 2005).

An alternative to these memory-based approaches are model-based ones in which a statistical or machine learning model is constructed from the user profiles and the model is used for prediction. This is an area of active research, and many avenues have been explored, including Bayesian classifiers (Park et al. 2007), neural networks (Roh et al. 2003), fuzzy systems (Yager 2003), latent features (Zhong and Li 2010), and matrix factorization (Koren et al. 2009; Luo et al. 2012; Ma et al. 2008).

Content-Based Recommendation

On the key advantages of collaborative recommendation is that no descriptive features for items need to be known, only user profiles related to those items. Content-based recommender systems rely on such descriptive features to build association between user preferences and item characteristics (Balabanović and Shoham 1997; Pazzani and Billsus 1997, 2007). Often, a vector space model is used to represent the features of items, and preferences over these features are learned from user profile data. Essentially the task is similar to a machine learning classification task with a different classifier for each user. Such models borrow from research in information retrieval (Baeza-Yates and Ribeiro-Neto 1999) and information filtering (Belkin and Croft 1992). A variety of models can be used including heuristic (Baeza-Yates and Ribeiro-Neto 1999; Garcia and Amatriain 2010; Salton 1989), nearest neighbors (Billsus et al. 2000; Yang 1999), classification algorithms (Cohen 1995; Joachims 1998; Kim et al. 2001), probabilistic methods (de Gemmis et al. 2008; Mooney and Roy 2000; Pazzani and Billsus 1997), and Rocchio's algorithm (Balabanović and Shoham 1997; Lang 1995).

Hybrid Recommendation

Hybrid recommender systems combine the techniques described above and can take a wide variety of forms. See Burke (2002) for a survey of the ways in which hybrids may be created. A simple

technique is a linear weighted hybrid, in which a system is created comprising multiple recommendation components, each of which returns a real-valued score for a combination of user and item. The scores from all the components are combined in a weighted sum (Burke 2002). More formally,

$$s(u,i) = \sum_j \alpha_j s_j(u,i) \quad (4)$$

where $s(u,i)$ is the score computed for a user–item combination, $s_j(u,i)$ is the score computed by the jth component, and α_j is the weight associated with the jth component.

Recommender Systems and Social Networks

Social network data may enhance or replace any of the forms of data on which recommendation calculations depend. The group of techniques that we classify under the heading of *link prediction* are those that use network-oriented measures to generate predicted (and, thus, recommended) links. *Trust-based models* are those that use social network information and, often, explicit trust statements by users, to supplement or replace the calculation of neighbor similarity for collaborative recommendation. Finally, heterogeneous network and multi-relational techniques provide a means of constructing recommendation hybrids involving a wide variety of data based on network relations.

Link Prediction

Link prediction is well-studied problem in network science. In its simplest form, link prediction starts with a graph and predicts additional edges that could or should be added to it. This has applications in inferring future states of the graph and in completing networks where the edges are only partially observed. Usually, link prediction algorithms are evaluated by removing some number of links from a known network and testing to see how well an algorithm can reconstruct them. See Al Hasan and Zaki (2011) for a survey of recent work in this area.

This evaluation technique makes clear that the emphasis in link prediction is somewhat different than that in recommendation. In particular, personalization is not a goal: the goal is to find predicted links for the network overall and not a list of links relevant to a given user. However, link prediction techniques can be applied in situations where a set of links is the desired recommendation output. For example, recommending social contacts to users joining an online social network is a well-established recommendation application to which link prediction can be applied (Schall 2015).

Link prediction models can be categorized in two general classes: unsupervised and supervised. Unsupervised models are those in which a score is calculated for each pair of nodes based on topological properties of the graph. For example, one such score is the number of common neighbors that two nodes share. Other scores include *Adamic–Adar* (Adamic and Adar 2003), *Katz*, and *Leicht–Holme–Newman Index* (Lü and Zhou 2011). These models use predefined scores that are invariant to the specific structure of the input graph and do not involve learning. See the Link Prediction section of this volume for additional details.

Supervised models try to predict a link by learning a classifier based on features of the source and target nodes and/or the structure of the network. The choice of these terms depends on the model type, which can be feature-based models (Wang et al. 2007), graph regularization models (Zhu and Ghahramani 2002), latent class models (Airoldi et al. 2008), and latent feature models (Koren et al. 2009).

Feature-Based Models

The link prediction problem can be modeled as a supervised classification (Al Hasan and Zaki 2011) task where each data point corresponds to a pair of nodes in the social network graph. The classification model can be learned through a training set *Tr*, and the predictions of future links can be evaluated from the test set *T*. In order to formulate this model, assume u and v are two nodes in the graph $G(V, E)$ and $y^{<u,v>}$ denotes the label of the data point $<u, v>$.

$$y^{\langle u,v\rangle} = \begin{cases} +1, & if \ \langle u,v\rangle \in E \\ -1, & if \ \langle u,v\rangle \notin E \end{cases} \quad (5)$$

With this labeling of the training set *Tr*, the link prediction problem can be seen as a typical classification task to predict the unknown labels of a pair of nodes <*u*, *v*> in test set *T*. Any of the popular supervised classification methods, such as naive Bayes, neural networks, support vector machines (SVM), and k nearest neighbors, can be used. The main problem in feature-based link prediction is to choose a set of features for the classification task.

One important class of features are those based on graph topology (Kashima and Abe 2006; Liben-Nowell and Kleinberg 2007). Link prediction models using such features compute the similarity based on the node neighborhoods or based on the combination of paths between a pair of nodes. Graph topological approaches can be divided in two categories:

1. Node neighborhood based, which measure the similarity between nodes based on neighbors such as *common neighbors* (Newman 2001), *Jaccard coefficient*, and *Adamic–Adar (*Adamic and Adar 2003)
2. Path-based methods, which measure node similarities based on paths connecting nodes such as *shortest path distance* (Al Hasan et al. 2006), Katz (1953), *hitting time*, and *Rooted PageRank* (Chung and Zhao 2010)

Another important class of features are node and edge attributes. Several researchers (Al Hasan et al. 2006; Doppa et al. 2009) have shown that methods that include node or edge attributes as proximity features can enhance the performance of link prediction tasks. Several approaches have been studied to incorporate these features in a link prediction task such as *vertex feature aggregation* (Al Hasan et al. 2006; Barabâsi et al. 2002), *kernel feature conjunction* (Basilico and Hofmann 2004; Oyama and Manning 2004), *extended graph formulation* (Al Hasan et al. 2006), and *generic SimRank* (Jeh and Widom 2002).

Bayesian Probabilistic Models

Bayesian models take a probabilistic approach to link prediction, computing a posterior probability that represents the chance of co-occurrence of the each node pair. A variety of formulations have been attempted. The work of Wang et al. (2007) uses Markov random fields. This model predicts the edge between a pair of nodes x and y by introducing the concept of central neighborhood set, which consists of other nodes that appear in the local neighborhood of x or y. Calculation is performed by calculating the joint probability of the interested node pair <*x*, *y*> and their local neighbors.

Another approach is to represent probabilities in terms of network evolution. For example, the evolution model in Kashima and Abe (2006) considers only the topological (structural) properties of the network. For a graph with a set of node V and $G(V, \phi)$, $\phi : V \times V \rightarrow [0, 1]$ label function, $\phi(x,y)$ denotes the probability that an edge exists between two nodes in graph G. The label function ϕ is defined at time t where ϕ_t denotes the edge label function at time t, which changes over time. In addition to that, this model is a Markov model in a way that the ϕ_{t+1} only depends on ϕ_t.

The evolution of the graph over time is modeled such that a label of one edge is copied from node x to node y randomly with probability of p_{xy}. This model decides on x and y and then selects an edge label uniformly from one of x edge labels to copy as y edge label. The evolution model closely resembles the transitive closure property of social networks, such that a friend of a friend becomes a friend (Al Hasan and Zaki 2011). The model assumes that the observed network is in a stationary state, which means that $\phi_\infty(x, y) = \phi_{t+1}(x, y) = \phi_t(x, y)$.

The predictions can be obtained by optimizing the log likelihood of equation below and obtaining the requisite probabilities for each non-observed edge:

$$\phi_\infty(x,y) = \frac{\sum_{k \notin (x,y)} \left(p_{ky}\phi_\infty(k,x) + \left(p_{kx}\phi_\infty(k,x)\right)\right.}{\sum_{k \notin (x,y)} p_{ky} + p_{kx}} \quad (6)$$

Another probabilistic alternative is the *hierarchical probabilistic model* (Clauset et al. 2008), based on the assumption of hierarchical structure in the network, namely, that the network is divided into groups that further subdivide into subgroups over multiple scales. The learning task is to use the observed network data to fit the most likely hierarchical structure through statistical inference. This model makes advantage of the combination of the maximum likelihood approach and Monte Carlo sampling (Al Hasan and Zaki 2011).

Linear Algebraic Methods

Link prediction can be derived from algebraic operations on the network's adjacency matrix. Such methods generalize graph kernels and dimensionality reduction methods (Kunegis and Lommatzsch 2009). This model learns a spectral transformation function F, which maps from the training data A to the test data B with minimal error. This results in the following optimization problem:

$$min_F \|F(A) - B\| \tag{7}$$

where the $\|\cdot\|$ denotes the *Frobenius norm*. Using the eigendecomposition of training matrix $A = U\Delta U^T$ and the fact that the *Frobenius norm* is invariant under multiplication by an orthogonal matrix, Eq. 7 can be written as below:

$$\left\|UF(\Delta)U^T - B\right\| = \left\|F(\Delta) - U^T B U\right\| \tag{8}$$

As a result, the final optimization function on the matrix can be reduced to a one-dimensional least square curve fitting problem:

$$min_f \sum_i \left(F(\Delta_{ii}) - U_i^T B U_i\right)^2 \tag{9}$$

Trust-Based Recommendation

Link prediction assumes that the recommendation/prediction is focused on the network itself and no external data is required. In trust-based recommendation, the assumption is that ordinary collaborative recommendation is to be performed with the social network providing clues about the relations between the users. Incorporating this social information has been shown to improve recommendation accuracy in a variety of settings (Miller et al. 2003). In such models, trust is defined as user's belief toward others in providing reliable and accurate rating. Trust intensity between users is defined using different trust metrics. Some approaches proposed explicit trust evaluations by users (Golbeck 2006), while others inferred trust from implicit indicators (Fu-guo and Sheng-hua 2007; O'Donovan and Smyth 2005).

Three steps are required for trust-based recommendation. First, trust measurement generates a $|u| \times |u|$ matrix, where $|u|$ is the number of users. Each entry of this matrix represents the trust intensity between two different users. This trust can be implicit or explicit, positive or negative. Generally, this matrix will be quite sparse as observed. As a result, trust propagation is usually necessary. By trust propagation methods, the system calculates the trust intensity between any two users based on nonempty cells of the trust matrix.

The second step is neighbor selection, a typical component of collaborative recommendation. However, in trust-based recommendation, the observed or calculated trust value intensity between users is used to choose reliable users as neighbors (Massa and Bhattacharjee 2004). Ziegler and Lausen (2004) studied the correlation between trust and user similarity in online communities and showed that users are significantly more similar to their trusted peers than to the whole population. Despite this research and the intuitive appeal of this idea, recommendation research has shown that peer users with high trust intensity may not always be satisfactory recommendation partners due to differences in tastes (Guo et al. 2014; Ray and Mahanti 2010). A combination of traditional peer similarity and trust intensity must be used in neighbor selection. The third step is rating prediction. Standard collaborative prediction techniques can be used here, factoring in the trust value as part of the similarity calculation.

Several trust metrics have been proposed. In his work on robust collaborative recommendation, Levin proposed *Advogato* (Levien 2009) to help recommenders distinguish between genuine

users and attackers. *Appleseed*, proposed in Ziegler (2005), models trust as energy and propagates it using spreading activation approach. This method generates the top n nearest trust neighbors for every user. The main novelty of this metric lies in the propagation step in which a source node u is activated through an injection of energy.

TidalTrust (Golbeck 2006) is a trust propagation technique using shortest paths. The maximum of the minimum edge weights on a path is calculated, and multiplication is used to calculate the weight of a path from its edge weights. The path weight is then used to propagate trust.

O'Donovan and Smyth (2005) took a different approach to calculating implicit trust. Their metric calculates trust as a function of recommendation performance – how often a neighbor contributed to a correct prediction. A new metric is proposed to deduce global implicit trust incorporating two types of trust (item level and profile level). Item-level trust, the perceived trustworthiness of the user for each item, is measured based on the ratio of correct predictions of the user to all of his/her predictions about the item. Profile-level trust is calculated as the ratio of correct predictions to all predictions, independent of items.

A related approach from DuBois et al. (2009) proposed clustering users based on the trust between them using correlation clustering to improve recommendation accuracy. This model modifies the collaborative filtering algorithm to use trust for generating recommendations independently of the clusters. The algorithm is modified to give stronger weight to ratings from nodes in the same cluster as the user for whom the rating was being generated.

Random Walk and Related Methods

Trust-based recommendation uses a specific type of network data (trust links) to supplement collaborative recommendation. Other recommendation techniques discussed here make use of the network structure more directly in recommendation generation.

A random walk is a finite and time-reversible Markov chain (Lovsz 1993): see the Random Walks entry in this volume. For a graph $G = (V, E)$, a random walk starts at a node v_0, and at the step, t will be at a node v_t. The walker moves to a neighbor of v_t with probability $1/degree(v_t)$. Let $M = (p_{ij})\ ,\ i\ ,\ j\ \in\ V$ be the transition probability matrix of this Markov chain. So

$$P^t = \begin{bmatrix} \dfrac{1}{\text{degree}(i)} & ,\text{if}(ij \in E) \\ 0 & ,\text{otherwise} \end{bmatrix} \tag{10}$$

If we let A be the adjacency matrix of G and let D be the diagonal matrix, with entries as $(D)_{ii} = 1/d(i)$, then $M = DA$ and the distribution of the point t can be expressed by the simple equation $p_t = (M^T)^t p_0$ where p_0 denotes the probability distribution of the start node.

Several researchers have demonstrated the effectiveness of random walks for recommendation in networks. The random walk method can be applied directly to generate recommendations (Gori et al. 2007) by projecting a bipartite graph built from a training data into a correlation graph that consists of item nodes (the node type to be recommended) and then applies a personalized ranking algorithm on this graph. A query-centered method proposed by Cheng et al. (2007) first builds a k-partite graph based on item features and then performs multi-way clustering to induce a subgraph. Recommendations are generated by random walks on the subgraph.

Singh et al. (2007) proposed an absorbing random walk method on the user–item bipartite graph, enhanced with user–user (social) links. An absorbing random walk is one in which certain states – in this case, items – have no outbound transitions. The probability distribution of an absorbing random walk is considered as personalized ranking for each user. In Singh's model, the walker starts from the target user, either transitioning to a friend with probability α or to an item with probability $1-\alpha$. If the walker transitions to an item, the walker cannot leave because, by definition, items are absorbing states. The probability that the walker is still in a user state after t steps is α^t and that the walk is absorbed with probability $(1-\alpha^t)$ after t steps.

The absorbing distributions $P_u(G)$ can be calculated through a transition matrix of P^∞, which is computationally expensive. A more tractable

naive approach computes a t step walk for which the absorbing distributions can be obtained by reading off the upper right hand block of the transition matrix below:

$$P^t = \begin{bmatrix} \alpha^t U^t & ,\left(1-\alpha\sum_{k=0}^{t-1}\alpha^k U^k G\right) \\ 0 & ,I \end{bmatrix} \quad (11)$$

Yildirim and Krishnamoorthy (2008) proposed an item-oriented algorithm that first develops an item graph where the nodes are items and the edges between nodes represent similarities of items. In the recommendation model, users make a finite-length random walk, so ranking becomes dependent on the initial distribution of ratings to items by the user.

The random walk with restart has been studied (Konstas et al. 2009) to measure the effect of social networking and social tagging in collaborative recommendation. In this work, random walk with restart allows the system to predict directly the preference of users for particular items by incorporating user tagging behavior and social networks. Specifically, the graph is created by representing users, item, and tags as nodes in the graph. Relationships between those nodes are encoded using edges between the corresponding nodes. Konstas et al. propose a music recommendation model using this technique (Konstas et al. 2009), which takes into account both the social annotation and friendships obtained in the social graph.

PageRank is a well-known random walk model (Page et al. 1999) used in search engines to rank items globally. This method computes the stationary distribution of a random walk. A PageRank model starts from a node v_0 and then walks randomly in a graph. At each step, the walker jumps to a random node with the probability α and walks along the outgoing edges with the probability $1-\alpha$. In the limit, the stationary probability of each node tends to be a constant independent of the starting node and is called PageRank score (Jin et al. 2012). The PageRank score $\pi_u(v)$ of node v satisfies the following equation with starting node u:

$$\pi_u(v) = \alpha\delta(u) + (1-\alpha)\sum_{w(w,u)\in E}\pi_u(w) \quad (12)$$

In a recommender framework, a personalized version of PageRank can be used to generate a personalized ranking score (Kim and El Saddik 2011). Personalized PageRank differs from the standard formulation in that the nonlocal transitions in the model always return to a particular node or set of nodes dependent on the user instead of randomly selecting any node in the network. For example, Jaschke et al. (2007) proposed a tag recommendation model in folksonomies using an adapted PageRank. This model converts the folksonomy $F = (U, T, R, Y)$ into an undirected tripartite graph $G = (V, E)$ in such a way that node set V consists of the disjoint union of the sets of tags, users, and resources. The edge set E is generated based on all co-occurrences of *tags–users*, *users–resources*, and *tags–resources*. Let $\vec{w}$ be a weight vector for each node, A be the row-stochastic version of the adjacency matrix of the graph G, $\vec{p}$ be the preference vector, and $d \in [0,1]$ be the influence factor of $\vec{p}$. The ranking of nodes in the graph G can be obtained as entries of fixed point of vector $\vec{w}$:

$$\vec{w} = dA \quad \vec{w} + (1-d)\ \vec{p} \quad (13)$$

Hotho et al. (2006) also proposed and evaluated a variation of adapted PageRank on a social tagging system to recommend bookmarks associated with collaborative annotation. A weight-spreading ranking scheme was used on an undirected, weighted, tripartite graph, in addition to a PageRank method that takes into account the edge weights. Personalized PageRank has also been used for trust-based recommendation (Andersen et al. 2008).

A temporally extended PageRank algorithm was proposed by Xiang et al. (2010) as temporal personalized random walk (*TPRW*) to allow recommendation generation to take the time of rating into consideration. In this model, a user-specific personalization vector d is maintained, which contains both the user's own node and also a session node that represents the current state of the user's

interaction. The nonlocal transitions in PageRank therefore incorporate both the user identity and session state. The issue of time was also addressed by Bahmani et al. (2010), which proposed incremental computation of PageRank in dynamically evolving networks as well as efficient updating of personalized PageRank.

Various other biased versions of PageRank have been developed for recommendation. Topic-sensitive PageRank adds a topic vector to control personalized search (Haveliwala 2002). The *ItemRank* algorithm (Gori and Pucci 2007) is a biased version of PageRank proposed to rank products according to expected user preferences. In this algorithm, a weighted correlation graph G_C of the items is constructed first with an edge (a_i, a_j) if and only if $C(i, j) > 0$. The bias is created from the user's ratings of each item. More formally,

$$ItemRank_{ui} = \alpha CItemRank_{ui} + (1 - \alpha)d_{ui} \quad (14)$$

Social Networks as Heterogeneous Networks

The social tagging systems discussed above in the context of PageRank are examples of multipartite networks with links between different types of nodes: tags, users, and items. Like a multipartite network, a heterogeneous information network (HIN) is a network with multiple types of nodes (Shi et al. 2015a) (e.g., user, groups, pages, and photos) and multiple types of edges (e.g., a "friendship" relation between users, "follow" relation between user and pages, and a "like" relation between a user and a post). As in a multipartite network, edge types are defined by the types of nodes that they connect. However, unlike multipartite networks, there are no restrictions on the connectivity of the network. Such networks are a natural way to express the multiplicity of connections between types of information in social media applications: consider LinkedIn's users, employers, interest groups, educational institutions, job postings, posts, comments, etc. as just one example. However, the complexity of heterogeneous networks poses two challenges for recommender systems: (1) the problem of integrating a wide variety of data effectively into a recommendation framework and (2) the problem of responding to many potential recommendation tasks, because of the wide variety of items present.

The main intuition in work with heterogeneous information networks is that nodes with meaningful semantics can be found by following certain typed paths on a heterogeneous network (Yu et al. 2014). In other words, in a *HIN*, two nodes can be connected via different types of paths. Due to the multiplicity of node and edge types in heterogeneous networks, these paths may contain different node types and link types in different orders, and they can have various lengths. For example, the relation between LinkedIn users u and v might be friend of a friend, friend of coworker, or friend of a fellow alumnus, to name just a few of the possible two-step paths.

A network schema is a high-level view of a heterogeneous network showing the node types and edge types. For example, in a bibliographic network, the *authored-by* edge type connects nodes of type *author* to nodes of type *publication.* A meta-path is a sequence of edges through the network schema – in other words a sequence of edge types. Traversing a meta-path on a heterogeneous network means following all edges of a given type from a node to all possible successors.

Research has shown that recommender systems can benefit from incorporating the diversity of relations and data dimensions found in heterogeneous networks. The challenge is to identify productive relations and meta-paths in the multiplicity of possible connections. A variety of approaches have been developed, achieving this integration by different means.

The Weighted Hybrid of Low-Dimensional Recommenders (*WHyLDR*) model uses a linear combination of recommendation components generated through meta-paths (Burke et al. 2014; Vahedian 2014). Components are built based on two-dimensional matrices commonly used in collaborative recommendation, both item based and user based. Each recommendation task carries with it a "target relation" that represents the predictions being sought. A user-based matrix is one in which the rows are users and the columns are

some other class of node around which a user profile can be organized. An item-based matrix is similar, except that the rows are items. Weights for the hybrid can be calculated through a non-linear optimization technique, but research has also shown it is possible to estimate the weights (and control meta-path expansion) using an information gain measure (Burke and Vahedian 2013; Vahedian and Burke 2014).

The HeteRec model also combines user feedback with different types of entity relationships in a collaborative manner. This algorithm uses diffusion of user preferences along different meta-paths to represent user interests and obtains a latent factor representation of users and items through matrix factorization. These latent features are combined in a weighted way to define a global recommendation model. In addition, this model generates different entity recommendation models for different user clusters to distinguish user interests using *Bayesian ranking* optimization (Yu et al. 2013, 2014).

The random walk techniques discussed above have also been extended for heterogeneous networks. A hybrid random walk-based model was proposed to rank nodes in a heterogeneous graph for a specific application domain. This graph-based recommender system extends personalized PageRank to exploit an effective node ranking measure which can capture various semantics of different nodes and edges in heterogeneous systems. This model produces different node ranking lists by using various sets of meta-paths (Lee et al. 2013).

Shi et al. (2015b) proposed the *SemRec* model, which uses weighted meta-paths to consider attribute values on links in information networks, calculating a path-based similarity between users. The rating prediction for user u on an item i under a specific meta-path is calculated as below:

$$\widehat{R}_{u,i} = \sum_{i=1}^{N} r \times \frac{Q_{u,i,r}}{\sum_{k=1}^{N} Q_{u,i,k}} \tag{15}$$

where the $Q_{u,t,k}$ denotes the rating intensity for the target user for item i. Following this rating prediction model, several prediction scores can be made for a pair $<u,i>$. This model also construct the final hybrid recommendation model as a weighted score to integrate generated meta-paths.

A random walk-based recommendation over heterogeneous social networks (Zhang et al. 2008) was proposed which formalized recommendation as a ranking problem, estimating the importance of each object in the heterogeneous network. This model introduced a pair-wise learning algorithm to learn the weight for each type of relationship. One key advantage of this model is that it can be used to recommend different types of objects simultaneously. The relevance score of each object is calculated regarding two different scenarios. First, when a person searches for one type of object, the relevance score is calculated based on the query. Second, when the person is browsing an object, a relevance score can be calculated based on the object profile (which can be extracted from the key terms) to recommend other objects. The random walk used to estimate the global importance is an adapted PageRank model that considers the different types of objects in heterogeneous network.

HeteRS (Pham et al. 2015) model was proposed as a general-purpose heterogeneous graph-based recommendation system capable of addressing multiple recommendation tasks. The system builds a heterogeneous graph to represent the interactions among users, events, groups, tags, and venues in event-based social networks. This work offers three different recommendation tasks: *group to user* recommendation, *tag to group* recommendation, and *event to user* recommendation. The recommendation problem is formulated as a node proximity calculation problem to be solved as a multivariate Markov chain.

Multi-relational Factorization

Another approach to recommender systems in heterogeneous information networks is multi-relational factorization, which extracts latent user features from multiple relations and uses them together to predict whether each of the items is interesting to a given user. *Collective matrix factorization* (Singh and Gordon 2008) is an example of this type of approach. This model factorizes

each relation matrix with a generalized-linear link function, but whenever an entity type is involved in more than one relationship, the factors of different models are tied together. Krohn-Grimberghe et al. (2012) formalized multi-relational factorization for item recommendation in social networks as a form of link prediction and extended the Bayesian personalized ranking (*BPR*) (Rendle et al. 2009) to provide ranking results in this context.

DMF (Drumond et al. 2014) was proposed to predict multiple target relations in multi-relational domains. In this model, different latent feature models are defined for each relation. Parameters are learned from the factorization process in such a way that they are optimized for the best performance on each relation individually.

DMF model was extended to heterogeneous networks by integrating additional information obtained through meta-paths (Vahedian et al. 2015, 2016). In this model one latent feature vector model is associated with each relation *r*. Different feature matrices are associated with each relation for different target relations. The DMF loss function decomposes over the target relation, and each component can be optimized independently of each other.

Key Applications

Social network-based recommender systems appear in many applications of different domains. Some examples include:

- Item or tag recommendation in social annotation (tagging) systems (Burke et al. 2014; Gemmell et al. 2010; Kim and El Saddik 2011)
- Friend recommendation (Moricz et al. 2010; Silva et al. 2010), "who to follow" recommendation in twitter (Gupta et al. 2013), and top-k tweet recommendation (Kim and Shim 2011)
- Semantic-based citation recommendation and venue recommendation for computer science bibliography (DBLP) (Vahedian et al. 2016.
- Business and movie recommendation in heterogeneous information networks (Vahedian and Burke 2014; Yu et al. 2014.

Future Directions

There has been significant research on the incorporation of social network data of different types into recommender systems as outlined above. However, this area is extremely active and likely to see further advances. As sites incorporate recommendation into all aspects of the user interface, multi-target recommendation systems that can be personalized beyond a single-target relation would seem to be extremely advantageous. Recommendation in enterprise social networks has been recently explored to improve employees' social interaction and motivations. There has been little work in recommendation explanation and interactivity where social network data is involved. Other future directions include group recommendation when all members of the group are part of a social network and recommendation in mobile social networks.

Cross-References

- ▶ Link Prediction
- ▶ Random Walks
- ▶ Recommender Systems
- ▶ Recommender Systems, Semantic-Based

Acknowledgments This work was supported in part by the National Science Foundation under Grant No. IIS-1423368 (Multidimensional Recommendation in Complex Heterogeneous Networks).

References

Adamic LA, Adar E (2003) Friends and neighbors on the web. Soc Netw 25(3):211–230

Adomavicius G, Tuzhilin A (2005) Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. Knowl Data Eng, IEEE Trans 17(6):734–749

Aggarwal CC (2016) Recommender systems – the textbook. Springer, Heidelberg, Germany

Airoldi EM, Blei DM, Fienberg SE, Xing EP (2008) Mixed membership stochastic blockmodels. J Mach Learn Res 9(Sep):1981–2014

Al Hasan M, Zaki MJ (2011) A survey of link prediction in social networks. In: Social network data analytics, Springer, pp 243–275

Al Hasan M, Chaoji V, Salem S, Zaki M (2006) Link prediction using supervised learning. In: SDM06: workshop on link analysis, counter-terrorism and security

Andersen R, Borgs C, Chayes J, Feige U, Flaxman A, Kalai A, Mirrokni V, Tennenholtz M (2008) Trust-based recommendation systems: an axiomatic approach. In: Proceedings of the 17th international conference on World Wide Web. ACM, pp 199–208

Baeza-Yates RA, Ribeiro-Neto B (1999) Modern information retrieval. Addison-Wesley Longman, Boston

Bahmani B, Chowdhury A, Goel A (2010) Fast incremental and personalized pagerank. Proc VLDB Endowment 4(3):173–184

Balabanović M, Shoham Y (1997) Fab: content-based, collaborative recommendation. Commun ACM 40(3):66–72

Barabâsi AL, Jeong H, Néda Z, Ravasz E, Schubert A, Vicsek T (2002) Evolution of the social network of scientific collaborations. Phys A: Stat Mech Appl 311(3):590–614

Basilico J, Hofmann T (2004) Unifying collaborative and content-based filtering. In: Proceedings of the twenty-first international conference on machine learning. ACM, pp 9–16

Belkin NJ, Croft WB (1992) Information filtering and information retrieval: two sides of the same coin? Commun ACM 35(12):29–38

Billsus D, Pazzani MJ, Chen J (2000) A learning agent for wireless news access. In: Proceedings of the 5th international conference on intelligent user interfaces, IUI '00. ACM, New York, pp 33–36

Burke R (2002) Hybrid recommender systems: survey and experiments. User Model User-Adap Inter 12(4):331–370

Burke R, Ramezani M (2011) Matching recommendation technologies and domains. In: Recommender systems handbook, Springer, pp 367–386

Burke R, Vahedian F (2013) Social web recommendation using metapaths. In: Proceedings of the Fifth ACM RecSys workshop on recommender systems and the social web. ACM. http://ceur-ws.org/Vol-1066/

Burke RD, Vahedian F, Mobasher B (2014) Hybrid recommendation in heterogeneous networks. UMAP 2014:49–60

Cheng H, Tan PN, Sticklen J, Punch WF (2007) Recommendation via query centered random walk on k-partite graph. In: Seventh IEEE international conference on data mining (ICDM 2007). IEEE, pp 457–462

Chung F, Zhao W (2010) Pagerank and random walks on graphs. In: Fete of combinatorics and computer science. Springer, pp 43–62

Clauset A, Moore C, Newman ME (2008) Hierarchical structure and the prediction of missing links in networks. Nature 453(7191):98–101

Cohen WW (1995) Fast effective rule induction. In: In Proceedings of the twelfth international conference on machine learning. Morgan Kaufmann, San Mateo, CA, pp 115–123

Doppa JR, Yu J, Tadepalli P, Getoor L (2009) Chance-constrained programs for link prediction. In: NIPS workshop on analyzing networks and learning with graphs

Drumond LR, Diaz-Aviles E, Schmidt-Thieme L, Nejdl W (2014) Optimizing multi-relational factorization models for multiple target relations. In: Proceedings of the 23rd ACM international conference on conference on information and knowledge management. ACM, pp 191–200

DuBois T, Golbeck J, Kleint J, Srinivasan A (2009) Improving recommendation accuracy by clustering social networks with trust. Recommender Syst Soc Web 532:1–8

Fu-guo Z, Sheng-hua X (2007) Topic-level trust in recommender systems. In: 2007 international conference on management science and engineering. IEEE, pp 156–161

Garcia R, Amatriain X (2010) Weighted content based methods for recommending connections in online social networks. In: Workshop on recommender systems and the social web, Citeseer, pp 68–71

Gemmell J, Schimoler T, Mobasher B, Burke R (2010) Hybrid tag recommendation for social annotation systems. In: Proceedings of the 19th ACM international conference on information and knowledge management. ACM, pp 829–838

de Gemmis M, Lops P, Semeraro G, Basile P (2008) Integrating tags in a semantic content-based recommender. In: Proceedings of the 2008 ACM conference on recommender systems, RecSys '08. ACM, New York, pp 163–170

Golbeck J (2006) Generating predictive movie recommendations from trust in social networks. In: International conference on trust management. Springer, pp 93–104

Gori M, Pucci A (2007) Itemrank: a random-walk based scoring algorithm for recommender engines. In: Proceedings of the 20th international joint conference on artificial intelligence, IJCAI'07. Morgan Kaufmann, San Francisco, pp 2766–2771

Gori M, Pucci A, Roma V, Siena I (2007) Itemrank: a random-walk based scoring algorithm for recommender engines. IJCAI 7:2766–2771

Guo G, Zhang J, Thalmann D (2014) Merging trust in collaborative filtering to alleviate data sparsity and cold start. Knowl-Based Syst 57:57–68

Gupta P, Goel A, Lin J, Sharma A, Wang D, Zadeh R (2013) Wtf: the who to follow service at twitter. In: Proceedings of the 22nd international conference on World Wide Web. ACM, pp 505–514

Haveliwala TH (2002) Topic-sensitive pagerank. In: Proceedings of the 11th international conference on World Wide Web, WWW '02. ACM, New York, pp 517–526

Herlocker JL, Konstan JA, Terveen LG, Riedl JT (2004) Evaluating collaborative filtering recommender systems. ACM Trans Inf Syst (TOIS) 22(1):5–53

Hotho A, Jäschke R, Schmitz C, Stumme G (2006) Information retrieval in folksonomies: search and ranking.

In: European semantic web conference, Springer, pp 411–426
Jannach D, Zanker M, Felfernig A, Friedrich G (2010) Recommender systems: an introduction, 1st edn. Cambridge University Press, New York
Jäschke R, Marinho L, Hotho A, Schmidt-Thieme L, Stumme G (2007) Tag recommendations in folksonomies. In: European conference on principles of data mining and knowledge discovery. Springer, pp 506–514
Jeh G, Widom J (2002) Simrank: a measure of structural-context similarity. In: Proceedings of the eighth ACM SIGKDD international conference on knowledge discovery and data mining. ACM, pp 538–543
Jin Z, Shi D, Wu Q, Yan H, Fan H (2012) Lbsnrank: personalized pagerank on location-based social networks. In: Proceedings of the 2012 ACM conference on ubiquitous computing. ACM, pp 980–987
Joachims T (1998) Text categorization with support vector machines: learning with many relevant features. In: Proceedings of the 10th European conference on machine learning, ECML '98. Springer, London, pp 137–142
Kashima H, Abe N (2006) A parameterized probabilistic model of network evolution for supervised link prediction. In: Proceedings of the sixth international conference on data mining, ICDM '06. IEEE Computer Society, Washington, DC, pp 340–349
Katz L (1953) A new status index derived from sociometric analysis. Psychometrika 18(1):39–43
Kim HN, El Saddik A (2011) Personalized pagerank vectors for tag recommendations: inside folkrank. In: Proceedings of the fifth ACM conference on recommender systems. ACM, pp 45–52
Kim Y, Shim K (2011) Twitobi: a recommendation system for twitter using probabilistic modeling. In: 2011 I.E. 11th international conference on data mining. IEEE, pp 340–349
Kim JW, Lee BH, Shaw MJ, Chang HL, Nelson M (2001) Application of decision-tree induction techniques to personalized advertisements on internet storefronts. Int J Electron Commer 5(3):45–62
Konstas I, Stathopoulos V, Jose JM (2009) On social networks and collaborative recommendation. In: Proceedings of the 32Nd international ACM SIGIR conference on research and development in information retrieval, SIGIR '09. ACM, New York, pp 195–202
Koren Y, Bell R, Volinsky C et al (2009) Matrix factorization techniques for recommender systems. Computer 42(8):30–37
Krohn-Grimberghe A, Drumond L, Freudenthaler C, Schmidt-Thieme L (2012) Multi-relational matrix factorization using Bayesian personalized ranking for social network data. In: Proceedings of the fifth ACM international conference on web search and data mining. ACM, pp 173–182
Kunegis J, Lommatzsch A (2009) Learning spectral graph transformations for link prediction. In: Proceedings of the 26th annual international conference on machine learning. ACM, pp 561–568
Lang K (1995) Newsweeder: learning to filter netnews. In: Proceedings of the 12th international machine learning conference (ML95)
Lee S, Park S, Kahng M, Lee SG (2013) Pathrank: ranking nodes on a heterogeneous graph for flexible hybrid recommender systems. Expert Syst Appl 40 (2):684–697
Levien R (2009) Attack-resistant trust metrics. In: Computing with social trust. Springer, pp 121–132
Liben-Nowell D, Kleinberg J (2007) The link-prediction problem for social networks. J Am Soc Inf Sci Technol 58(7):1019–1031. https://doi.org/10.1002/asi.v58:7
Lovsz L (1993) Random walks on graphs: a survey
Lü L, Zhou T (2011) Link prediction in complex networks: a survey. Phys A: Stat Mech Appl 390(6):1150–1170
Luo X, Xia Y, Zhu Q (2012) Incremental collaborative filtering recommender based on regularized matrix factorization. Knowl-Based Syst 27:271–280
Ma H, Yang H, Lyu MR, King I (2008) Sorec: social recommendation using probabilistic matrix factorization. In: Proceedings of the 17th ACM conference on information and knowledge management, CIKM '08. ACM, New York, pp 931–940
Massa P, Bhattacharjee B (2004) Using trust in recommender systems: an experimental analysis. In: International conference on trust management. Springer, pp 221–235
Miller BN, Albert I, Lam SK, Konstan JA, Riedl J (2003) Movielens unplugged: experiences with an occasionally connected recommender system. In: Proceedings of the 8th international conference on intelligent user interfaces. ACM, pp 263–266
Mooney RJ, Roy L (2000) Content-based book recommending using learning for text categorization. In: Proceedings of the fifth ACM conference on digital libraries, DL '00. ACM, New York, pp 195–204
Moricz M, Dosbayev Y, Berlyant M (2010) Pymk: friend recommendation at myspace. In: Proceedings of the 2010 ACM SIGMOD international conference on management of data. ACM, pp 999–1002
Newman M (2001) Clustering and preferential attachment in growing networks. Phys Rev E 64(2):025,102
O'Donovan J, Smyth B (2005) Trust no one: evaluating trust-based filtering for recommenders. IJCAI, Citeseer 5:1663–1665
Oyama S, Manning CD (2004) Using feature conjunctions across examples for learning pairwise classifiers. In: European conference on machine learning. Springer, pp 322–333
Page L, Brin S, Motwani R, Winograd T (1999) The pagerank citation ranking: bringing order to the web
Park MH, Hong JH, Cho SB (2007) Location-based recommendation system using Bayesian users preference model in mobile devices. In: International conference on ubiquitous intelligence and computing. Springer, pp 1130–1139

Pazzani M, Billsus D (1997) Learning and revising user profiles: the identification of interesting web sites. Mach Learn 27(3):313–331

Pazzani MJ, Billsus D (2007) Content-based recommendation systems. In: Brusilovsky P, Kobsa A, Nejdl W (eds) The adaptive web. Springer, Berlin, pp 325–341. http://dl.acm.org/citation.cfm?id=1768197.1768209

Pham TAN, Li X, Cong G, Zhang Z (2015) A general graph-based model for recommendation in event-based social networks. In: 2015 I.E. 31st international conference on data engineering. IEEE, pp 567–578

Ray S, Mahanti A (2010) Improving prediction accuracy in trust-aware recommender systems. In: System sciences (HICSS), 2010 43rd Hawaii international conference on. IEEE, pp 1–9

Rendle S, Freudenthaler C, Gantner Z, Schmidt-Thieme L (2009) Bpr: Bayesian personalized ranking from implicit feedback. In: Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence, UAI '09. AUAI Press, Arlington, pp 452–461. http://dl.acm.org/citation.cfm?id=1795114.1795167

Resnick P, Iacovou N, Suchak M, Bergstrom P, Riedl J (1994) Grouplens: an open architecture for collaborative filtering of netnews. In: Proceedings of the 1994 ACM conference on computer supported cooperative work, CSCW '94. ACM, New York, pp 175–186

Ricci F, Rokach L, Shapira B, Kantor PB (2010) Recommender systems handbook, 1st edn. Springer, New York

Roh TH, Oh KJ, Han I (2003) The collaborative filtering recommendation based on som cluster-indexing cbr. Expert Syst Appl 25(3):413–423

Salton G (1989) Automatic text processing: the transformation, analysis, and retrieval of information by computer. Addison-Wesley Longman, Boston

Schall D (2015) Social network-based recommender systems. Springer

Shi C, Li Y, Zhang J, Sun Y, Yu PS (2015a) A survey of heterogeneous information network analysis. arXiv preprint arXiv:151104854

Shi C, Zhang Z, Luo P, Yu PS, Yue Y, Wu B (2015b) Semantic path based personalized recommendation on weighted heterogeneous information networks. In: Proceedings of the 24th ACM international on conference on information and knowledge management. ACM, pp 453–462

Silva NB, Tsang R, Cavalcanti GD, Tsang J (2010) A graph-based friend recommendation system using genetic algorithm. In: IEEE congress on evolutionary computation. IEEE, pp 1–7

Singh AP, Gordon GJ (2008) Relational learning via collective matrix factorization. In: Proceedings of the 14th ACM SIGKDD international conference on knowledge discovery and data mining. ACM, pp 650–658

Singh AP, Gunawardana A, Meek C, Surendran AC (2007) Recommendations using absorbing random walks. North East Student Colloquium on Artificial Intelligence

Vahedian F (2014) Weighted hybrid recommendation for heterogeneous networks. In: RecSys '14, pp 429–432

Vahedian F, Burke RD (2014) Predicting component utilities for linear-weighted hybrid recommendation. In: RSWeb 2014. http://ceurws.org/Vol-1271/Paper7.pdf

Vahedian F, Burke RD, Mobasher B (2015) Network-based extension of multi-relational factorization models. In: Poster proceedings of the 9th ACM conference on recommender systems, RecSys 2015, Vienna, 16 Sept 2015

Vahedian F, Burke RD, Mobasher B (2016) Meta-path selection for extended multi-relational matrix factorization. In: Proceedings of the twenty-ninth international Florida Artificial Intelligence Research Society conference, FLAIRS 2016, Key Largo, 16–18 May 2016, pp 566–571

Wang C, Satuluri V, Parthasarathy S (2007) Local probabilistic models for link prediction. In: seventh IEEE international conference on data mining (ICDM 2007). IEEE, pp 322–331

Xiang L, Yuan Q, Zhao S, Chen L, Zhang X, Yang Q, Sun J (2010) Temporal recommendation on graphs via long- and short-term preference fusion. In: Proceedings of the 16th ACM SIGKDD international conference on knowledge discovery and data mining, KDD '10. ACM, New York, pp 723–732. http://doi.acm.org/10.1145/1835804.1835896

Yager RR (2003) Fuzzy logic methods in recommender systems. Fuzzy Sets Syst 136(2):133–149

Yang Y (1999) An evaluation of statistical approaches to text categorization. Inf Retr 1(1–2):69–90. https://doi.org/10.1023/A:1009982220290

Yildirim H, Krishnamoorthy MS (2008) A random walk method for alleviating the sparsity problem in collaborative filtering. In: Proceedings of the 2008 ACM conference on recommender systems. ACM, pp 131–138

Yu X, Ren X, Sun Y, Sturt B, Khandelwal U, Gu Q, Norick B, Han J (2013) Recommendation in heterogeneous information networks with implicit user feedback. In: Proceedings of the 7th ACM conference on recommender systems. ACM, pp 347–350

Yu X, Ren X, Sun Y, Gu Q, Sturt B, Khandelwal U, Norick B, Han J (2014) Personalized entity recommendation: a heterogeneous information network approach. In: Proceedings of the 7th ACM international conference on web search and data mining. ACM, pp 283–292

Zhang J, Tang J, Liang B, Yang Z, Wang S, Zuo J, (2008) Recommendation over a heterogeneous social network. In: Web-age information management, 2008. WAIM'08. The ninth international conference on. IEEE, pp 309–316

Zhong J, Li X (2010) Unified collaborative filtering model based on combination of latent features. Expert Syst Appl 37(8):5666–5672

Zhu X, Ghahramani Z (2002) Learning from labeled and unlabeled data with label propagation. Tech. Rep., CMU-CALD-02, Carnegie Mellon University

Ziegler CN (2005) Towards decentralized recommender systems. PhD thesis, Albert-Ludwigs-Universitt Freiburg, Freiburg i.Br.
Ziegler CN, Lausen G (2004) Analyzing correlation between trust and user similarity in online communities. In: International conference on trust management. Springer, pp 251–265

Recommender Systems Evaluation

Alejandro Bellogín[1] and Alan Said[2]
[1]Universidad Autónoma de Madrid, Madrid, Spain
[2]School of Informatics, University of Skövde, Skövde, Sweden

Synonyms

Evaluation; Methods; Metrics; Recommendation systems; Reproducibility

Glossary

AUC	Area under the curve
CF	Collaborative filtering
CTR	Click-through rate
DCG	Discounted cumulative gain
ILD	Intra-list diversity
IR	Information retrieval
MAE	Mean absolute error
MAP	Mean average precision
ML	Machine learning
RMSE	Root-mean-squared error
ROC	Receiver operating characteristic
RS	Recommender system

Definition

The evaluation of RSs has been, and still is, the object of active research in the field. Since the advent of the first RS, recommendation performance has been usually equated to the accuracy of rating prediction, that is, estimated ratings are compared against actual ratings, and differences between them are computed by means of the MAE and RMSE metrics. In terms of the effective utility of recommendations for users, there is, however, an increasing realization that the quality (precision) of a ranking of recommended items can be more important than the accuracy in predicting specific rating values. As a result, precision-oriented metrics are being increasingly considered in the field, and a large amount of recent work has focused on evaluating top-N ranked recommendation lists with the above type of metrics.

Besides that, other dimensions apart from accuracy – such as coverage, diversity, novelty, and serendipity – have been recently taken into account and analyzed when considered what makes a good recommendation (Said et al. 2014b; Cremonesi et al. 2011; McNee et al. 2006; Bellogín and de Vries 2013; Bollen et al. 2010).

So, what makes a good evaluation? The realization that high prediction accuracy might not translate to a higher perceived performance from the users has brought a plethora of novel metrics and methods, focusing on other aspects of recommendation (Said et al. 2013a; Castells et al. 2015; Vargas and Castells 2014). Recent trends in evaluation methodologies point toward there being a shift from traditional methods solely based on statistical analyses of static data, i.e., raising precision performance of algorithms on offline data (Ekstrand et al. 2011b) – offline data in this case being recorded user interactions such as movie ratings or product purchases.

Evaluation is the key to identifying how well an algorithm or a system works. Deploying a new algorithm in a new system will have an effect on the overall performance of the system – in terms of accuracy and other types of metrics. Both prior deploying the algorithm, and after the deployment, it is important to evaluate the system performance.

It is in the evaluation of an RS one needs to decide on what should be sought-for, e.g., depending on whether the evaluation is to be performed from the users' perspective (accuracy, serendipity, novelty), the vendor's perspective (catalog, profit, churn), or even from the technical perspective of the system running the RS (CPU load, training time, adaptability). Given the

context of the system, there might be other perspectives as well; in summary, what is important is to define the key performance indicator (KPI) that one wants to measure.

Let us imagine an online marketplace where customers buy various goods, an improved recommendation algorithm could result in, e.g., increased numbers of sold goods, more expensive goods sold, more goods from a specific section of the catalog sold, customers returning to the marketplace more often, etc. When evaluating a system like this, one needs to decide on what is to be evaluated – what the sought-for quality is – and how it is going to be measured.

Introduction

Recommender systems have been a popular research topic within personalized systems and information retrieval since the mid-1990s. Throughout this time, various models of recommendation have been developed, e.g., approaches using *collaborative filtering* or *matrix factorization* for purposes such as retrieval of ranked lists of items for consumption, or for the rating prediction task (which was made very popular through the Netflix Prize; Bennett et al. (2007)). Today, the use of recommender systems has spread to a very wide area of topics, including personalized healthcare (Elsweiler et al. 2015; Luo et al. 2016), online news portals (Said et al. 2014a), food (Elahi et al. 2014, 2015), social networks (Guy 2015), exercise (Berkovsky et al. 2012), jobs (Abel 2015), investment (Zhao et al. 2015), transportation (Bistaffa et al. 2015), shopping (Jannach et al. 2015), etc. The list can be made even longer. The very many domains that recommender systems are applied to give rise to a plethora of recommendation approaches, algorithms, and settings that are tailored to the specific context and domain of the recommendation.

Given the various situations recommendations can be applied to, it follows that evaluation of these systems needs to be tailored to the specific setting, domain, userbase, context, etc. Moreover, RSs have several properties that affect the user experience (Gunawardana and Shani 2015): accuracy, robustness, scalability, coverage, confidence, novelty, diversity, etc. These properties are associated to one or many evaluation metrics, whose values are, in the end, what will be compared for different recommendation algorithms in comparative studies of RS literature, where different evaluation methodologies are applied depending on the experimental settings (Bellogín 2012; Said 2013).

This chapter attempts to give an overview of some of the more commonly used evaluation methods and metrics used for various types of recommendation. The overview should be seen as an introduction to evaluation and not a definitive guide to RS evaluation.

Key Points

To accurately measure the efficiency of a recommender system, a proper evaluation that is tailored to the sought-for qualities of the recommender system needs to be created, aiming to adapt metrics, strategies/protocols, user tasks, etc. to the scenario at hand. Given the many situations in which recommender systems can be applied, there are many variations for each evaluation component (metric, protocol, etc.). In this entry, we describe the processes, methods, strategies, and metrics used for the purpose of evaluating recommender systems. We also present the main challenges, applications, and future directions in the area, evidencing that evaluation of recommender systems is a popular and active research problem and, still, an open problem in the field.

Historical Background

In 1994, Resnick and colleagues published what is considered the first modern research paper on RSs (Resnick et al. 1994). In the two decades that have passed since, RS has grown exponentially as research topic. In the early stages of RS research and development, evaluation focused on predicting future ratings that users will give to items which have not been rated thus far. Generally, this meant: the closer the predicted rating to the actual rating provided by the user, the better the recommendation.

Over the years, RSs have changed, and along them, so have the evaluation methods used to evaluate them. Historically, evaluation methods and metrics (see the next section for a thorough definition of all these) were adapted from IR, ML, and classification systems (precision and recall, nDCG, ROC, AUC, etc.). RMSE became the focus of RS evaluation after having been used as the default evaluation measure in the Netflix Prize. After the prize concluded, many new methods and metrics were developed, specifically tailored for RSs used within Web 2.0, the interactive web, by integrating and adapting to how users interact with this new generation of RSs.

Let us return to the Netflix Prize to exemplify this. When the service started, customers would order a set of DVDs from the website, wait for a few days until the package arrived, watch the movies, rate them, and return the DVDs. The newly added rating would then be used to generate new recommendations for the customers, for the next round of DVD rentals. The turn-around between recommendations, consumption, and rating was at least a few days. This changed when, instead of ordering DVDs, customers began to stream content; the turn-around period went from a few days to minutes. This is the context of recommenders today. The feedback loop to the RS is instantaneous; the evaluation methods and metrics have adapted to this. Instead of focusing on predicting the rating given for a movie, a song, or a product, the evaluation focuses on how much of a movie has been watched, how many times a song has been skipped in a playlist, users' dwell-times while reading online newspaper articles, and so on.

Once these *first-generation metrics* (that measure prediction error) became obsolete, metrics from IR were adapted to take into account the ranking presented to the user, instead of the actual rating predicted for the user-item pair (acknowledging the fact that an algorithm that learns better than another the 1's and 2's of a particular user has no practical utility, since those items will never be recommended). Nowadays, these metrics are adapted to many different contexts including very small interfaces (short rankings) and online recommendation. A *third generation* of metrics appears when user data and their associated recommendations move from batch mode (as in standard offline experimentation) to online or streaming mode. In this situation, usually only clicks (CTR) can be measured and additional statistics should be computed if significance of the result has to be estimated (because of a high variance in the population). Finally, side data associated with the interaction between the user and the system has also been considered in the last years. Dwell time, returning visitors, buying events, etc. can be computed in specific domains assuming we can identify the user on a sequence of visits to the system. These metrics are, in principle, closer to measuring the real value of the recommendation – as we shall discuss in following sections – and therefore are being incorporated in the evaluation of real RSs such as newspapers and e-commerce businesses.

In parallel to the evolution of evaluation metrics and tasks, more datasets have been made publicly available; some of them include new dimensions (social, temporal, clicks, multidimensional ratings, personality, etc.) and domains (microblogging, movies, music, e-commerce, recipes, and more). This has caused the evaluation ecosystem to adapt itself: as an example, context-based RSs need their own evaluation metrics that are aware of other dimensions (time, emotion, etc.) (Campos et al. 2014; Tkalcic et al. 2016). A further proof of this evolution are the different challenges and competitions – KDD Cup, RecSys Challenge, and some of the competitions hosted on Kaggle – related to RS where either ad-hoc metrics or others based on rankings have been used. While none of these challenges has had the huge impact the Netflix Prize had, it is true that more and more practitioners are being introduced to the field through these venues, and hence, the experimental methodologies and datasets used there might be considered standard practice.

Evaluating Recommender Systems

The evaluation of RSs has been a major object of study in the field since its earliest days, and is still a topic of ongoing research, where open questions remain (Herlocker et al. 2004; Gunawardana and

Shani 2015). It is acknowledged that the evaluation of RSs should take into account the goal of the system itself. For example, (Herlocker et al. 2004) identify two main user tasks: *annotation in context* and *find good items.* In these tasks, the users only care about errors in the item rank order provided by the system, not the predicted rating value itself. Based on this consideration, researchers have started to use precision-based metrics to evaluate recommendations, although most works also still report error-based metrics for comparison with state-of-the-art approaches. Moreover, other authors (Herlocker et al. 2004; Gunawardana and Shani 2015) encourage considering alternative performance criteria, like the novelty of the suggested items and the item coverage of a recommendation method. We describe the above types of evaluation metrics in the subsequent sections.

Not all the evaluation metrics can be computed under any experimental settings. Different evaluation protocols exist and they impose constraints on the type of data that can be measured and analyzed. Two main evaluation protocols are usually considered: offline and online. These protocols present a clear tradeoff between effort (time, users, etc.) and usefulness/trustworthiness of the results, which will be discussed in later sections.

Finally, we present some of the most important challenges that appear when evaluating RSs. As we stated previously, several open questions remain in RS evaluation, and some of the most important ones are related to how replicable the conducted experiments in RSs could be; besides, a deeper understanding of the biases present in evaluation and how to combine the different dimensions and metrics is still needed in the field – these aspects will be presented and discussed in the last part of this section.

Evaluation Metrics

In the classical formulation of the recommendation problem, user preferences for items are represented as numeric ratings, and the goal of a recommendation algorithm consists of predicting unknown ratings based on known ratings and, in some cases, additional information about users, items, and the context. In this scenario, the accuracy of recommendations has been commonly evaluated by measuring the error between predicted and known ratings, using error metrics. Although dominant in the literature, some authors have argued this evaluation methodology is detrimental to the field since the recommendations obtained in this way are not the most useful for users (McNee et al. 2006). Acknowledging this, recent work has evaluated top-N ranked recommendation lists with precision-based metrics (Cremonesi et al. 2010; McLaughlin and Herlocker 2004; Bellogín et al. 2011; Said and Bellogín 2014), drawing from well-studied evaluation methodologies in the information retrieval field. We present and discuss the most prominent of these metrics later on.

In a more modern formulation of the recommendation problem, the ratings are no longer important. Instead, the consumption of recommended items by users is key, i.e., whether a recommended movie will be seen, a music track listened to, a product purchased, etc. In this context, it is important to ask oneself what should the recommender system bring the user? If a recommender system recommends an item that the user already knows of, what is the value of the system? Will this recommendation result in the consumption of the item? The assumption is that a recommender system, in this context, should bring the user something she might not yet be familiar with, i.e., something novel, unexpected, or serendipitous (Vargas et al. 2014; Vargas and Castells 2013). Still, the novel, unexpected, or serendipitous recommendations need to fulfill the requirement of the items being of actual interest to the user. These, so-called, nonaccuracy metrics focus on the variety, popularity, novelty, and similar aspects of the items, or lists of items, that are recommended (Ziegler et al. 2005; Ziegler and Lausen 2009). In this section, we present some of the most common nonaccuracy metrics and the motivations behind them.

An often forgotten dimension of evaluation, at least in academic research, is *business-related* metrics. Within this domain, there are metrics which are purely based on economy and market growth, e.g., customer churn and profit margin on customer purchases; additionally, there are metrics which are related to the development, and maintenance of the recommender system itself,

such as CPU cost per recommendation, storage cost, cost of retraining the model, etc. A brief overview of this metric type ends this section.

Error Metrics

A classic assumption in the RS literature is that a system that provides more accurate predictions will be preferred by the user (Gunawardana and Shani 2015). Although this has been further studied and refuted by several authors (McNee et al. 2006; Cremonesi et al. 2011; Bollen et al. 2010), the issue is still worth being analyzed. Traditionally, the most popular metrics to measure the accuracy of an RS have been the **mean absolute error** (MAE), and the **root-mean-squared error** (RMSE):

$$\mathrm{MAE} = \frac{1}{|\mathrm{Te}|} \sum_{(u,i) \in \mathbf{Te}} \left| \tilde{r}(u,i) - r(u,i) \right| \quad (1)$$

$$\mathrm{RMSE} = \sqrt{\frac{1}{|\mathrm{Te}|} \sum_{(u,i) \in \mathbf{Te}} (\tilde{r}(u,i) - r(u,i))^2} \quad (2)$$

where $\tilde{r}$ and r denote the predicted and real rating, respectively, and Te corresponds to the test set. The RMSE metric is usually preferred to MAE because it penalizes larger errors.

Different variations of these metrics have been proposed in the literature. Some authors **normalize MAE** and **RMSE** with respect to the maximum range of the ratings (Goldberg et al. 2001; Gunawardana and Shani 2015) or with respect to the expected value if ratings are distributed uniformly (Marlin 2003; Rennie and Srebro 2005). Alternatively, **per-user** and **per-item** average errors have also been proposed in order to avoid biases from the error on a few very frequent users or items (Massa and Avesani 2007; Gunawardana and Shani 2015).

A critical limitation of these metrics is that they do not make any distinction between the errors made on the top items predicted by a system, and the errors made for the rest of the items. Furthermore, they can only be applied when the recommender predicts a score in the allowed range of rating values. Because of that, implicit and some content-based and probabilistic recommenders cannot be evaluated in this way, since $\tilde{r}(u,i)$ would represent a probability or, in general, a preference score, not a rating. Hence, these methods can only be evaluated by measuring the performance of the generated ranking using precision-based metrics.

Ranking Metrics

Ranking or precision-based metrics measure the accuracy of a list of recommendations, usually taking into account the natural browsing order (Gunawardana and Shani 2015). In contrast to the previous metrics, any algorithm that sorts the items in a particular way could be evaluated – that is, we are not restricted to those predicting an explicit rating. These metrics can be classified into three groups: metrics that only use one ranking, metrics that compare two rankings (typically, one of them is a reference or ideal ranking), and metrics from the ML field.

Examples of metrics based on one ranking are precision, recall, normalized discounted cumulative gain, mean average precision, and mean reciprocal rank (see Table 1). Each of these metrics captures the quality of a ranking from a slightly different angle. More specifically, **precision** accounts for the fraction of recommended items that are relevant, whereas **recall** is the fraction of the relevant items that has been recommended. Both metrics are inversely related, since an improvement in recall typically produces a decrease in precision. They are typically computed up to a ranking position or cutoff k, being denoted as *P@k* and *R@k* (Baeza-Yates and Ribeiro-Neto 2011). Note that recall has also been referred to as **hit-rate** in (Deshpande and Karypis 2004). Hit-rate has also been defined as the percentage of users with at least one correct recommendation (Bellogín et al. 2013), corresponding to the **success** metric (or first relevant score), as defined by TREC (Tomlinson 2012).

The **mean average precision** (MAP) metric provides a single summary of the user's ranking by averaging the precision figures obtained after each new relevant item is obtained (Baeza-Yates and Ribeiro-Neto 2011). **Normalized discounted cumulative gain** (nDCG) uses graded relevance that is accumulated starting at the top of the ranking and may be reduced, or discounted, at lower

Recommender Systems Evaluation, Table 1 Formulation of ranking metrics[a]

$\text{P@k} = \frac{1}{\lvert \mathrm{U} \rvert} \sum_{u \in \mathrm{U}} \frac{\lvert \text{Rel}_u @k \rvert}{k}$
$\text{R@k} = \frac{1}{\lvert \mathrm{U} \rvert} \sum_{u \in \mathrm{U}} \frac{\lvert \text{Rel}_u @k \rvert}{\lvert \text{Rel}_u \rvert}$
$\text{MAP} = \frac{1}{\mathrm{U}} \sum_{u \in \mathrm{U}} \frac{1}{\lvert \text{Rel}_u \rvert} \sum_{i \in \text{Rel}_u} P@\text{rank}(u,i)$
$\text{nDCG} = \frac{1}{\mathrm{U}} \sum_{u \in \mathrm{U}} \frac{1}{\text{IDCG}_u^{pu}} \sum_{p=1}^{p_u} f_{\text{dis}}\big(\text{rel}\big(u,i_p\big),p\big), \quad \text{usually } f_{\text{dis}}(x,y) = \frac{2^x-1}{\log(1+y)}$
$\text{HL} = 100\big(\sum_{u \in \mathrm{U}} \text{HL}_u^{\max}\big)^{-1} \sum_{u \in \mathrm{U}} \text{HL}_u, \quad \text{HL}_u = \sum_{p=1}^{p_u} \frac{\max\big(\tilde{r}\big(u,i_p\big) - d, 0\big)}{2^{(p-1)}/(\alpha-1)}$
$\text{MRR} = \sum_{u \in \mathrm{U}} \frac{1}{s_r(u)}$
$\text{NDPM} = \frac{1}{\mathrm{U}} \sum_{u \in \mathrm{U}} \frac{2C_u^{\text{con}} + C_u^{\text{tie}}}{2C_u}$

[a]Notation: Rel_u represents the set of relevant items for user u, Rel_u @k is the number of relevant recommended items up to position k, rank(u, i) outputs the ranking position of item i in the user's u list; IDCG_u^k denotes the score obtained by an ideal or perfect ranking for user u up to position k; p_u denotes the maximum number of items evaluated by each user (sometimes assumed to be a cutoff k, the same for all the users); d is the default rating (or neutral vote), and α is the half-life utility that represents the rank of the item on the list such that there is a 50% chance that the user will view that item. Breese et al. (1998) use a value of 5 in their experiments, and note that they did not obtain different results with a half-life of 10. Moreover $s_r(u)$ is a function that returns the position of the first relevant item obtained for user u. Finally, C_u is the number of pairs of items for which the reference ranking asserts an ordering, i.e., the items are not tied. Besides, C_u^{con} denotes the number of discordant item pairs between the method's ranking and the reference ranking, and C_u^{tie} represents the number of pairs where the reference ranking does not tie, but where the method's ranking does.

ranks (Järvelin and Kekäläinen 2002). Using a different discount function, the **rank score** or **half-life utility** metric (Breese et al. 1998; Herlocker et al. 2004) can be obtained from the nDCG formulation (see Table 1).

Mean reciprocal rank (MRR) favors rankings whose first correct result occurs near the top ranking results (Baeza-Yates and Ribeiro-Neto 2011). This metric is similar to the **average rank of correct recommendation** (ARC) proposed in Burke (2004) and to the **average reciprocal hit-rank** (ARHR) defined in Deshpande and Karypis (2004).

Additionally, specific metrics have been defined in the context of RS evaluation that take as inputs two rankings (ideal vs. estimated) instead of just one. A first example is the normalized distance-based performance measure (NDPM), used in Balabanovic and Shoham (1997), and proposed in Yao (1995). This metric compares two different weakly ordered rankings, considering the number of concordant and discordant pairs. This metric is comparable across datasets since it is normalized with respect to the worst possible scenario. Furthermore, it provides a perfect score of 0 to systems that correctly predict every preference relation asserted by the reference, and a worst score of 1 to methods that contradict every reference preference relation. Besides, a penalization of 0.5 is applied when a reference preference relation is not predicted, whereas predicting unknown preferences (i.e., they are not ordered in the reference ranking) receives no penalization.

Rank correlation metrics such as **Spearman's** ρ and **Kendall's** τ have also been proposed to directly compare the system ranking to a preference order given by the user. These correlation coefficients provide scores in the range of -1 to 1, where 1 denotes a perfect correlation between the two rankings, and -1 represents an inverse correlation. These two metrics, along with NDPM, suffer from the interchange weakness (Herlocker et al. 2004), that is, interchanges at the top of the ranking have the same weight that interchanges at the bottom of the ranking.

Finally, metrics from the ML literature have also been used, although they are not as popular as those described above. For instance, metrics based on the **receiving operating characteristic** (ROC) curve and the **area under the curve** (AUC) provide a theoretically grounded alternative to precision and recall (Herlocker et al. 2004). The ROC model attempts to measure the extent to which an information filtering system can successfully distinguish

between signal (relevant items) and noise. Starting from the origin of coordinates at (0,0), the ROC curve is built by considering, at each rank position, whether the item is relevant or not for the user; in the first case, the curve goes one step up, and in the second, one step right. A random recommender is expected to produce a straight line from the origin to the upper right corner; on the other hand, the more leftward the curve leans, the better is the performance of the system. These facts are related to the area under the ROC curve, a summary metric that is expected to be higher when the recommender performs better, where the expected value of a random recommender is 0.5, corresponding to a diagonal curve in the unit square.

Nonaccuracy Metrics

Nonaccuracy metrics attempt to measure the quality of the recommendation not in terms of how well the system is able to mimic the users' history. Research and development of these metrics has grown rapidly in recent years, the reason for this is the realization that the data in a user's prior preferences might not necessarily be enough to generate future models of the user's preference. Additionally, as will be discussed later in this entry, traditional *offline* evaluation methods often suffer from various biases, e.g., popularity, recency, etc.

These metrics attempt to measure a holistic quality of the recommendations, i.e., how well will the recommendation be received by the end user, how will the users' experience of the system be affected by the recommendation, and also how the system performs in terms of, e.g., the catalog of items available. Due to the nature of some of these metrics, it is difficult to create a ground truth dataset to use with them. Thus, recommendation algorithms which are specifically tailored toward nonaccuracy metrics will often perform badly in terms of accuracy-based metrics (Said et al. 2013a); and symmetrically, if an algorithm is tailored toward accuracy metrics, it will often perform badly in terms of nonaccuracy metrics.

Due to the nature of nonaccuracy metrics, there are often various definitions of them, each tailored toward the context they are used in. In this chapter, we follow the same definitions as in the *Recommender Systems Handbook* (Ricci et al. 2015).

Perhaps the most well-known nonaccuracy metric, serendipity, attempts to model what is often referenced to as a *pleasant and unexpected surprise*, similar to finding a banknote on the pavement. Serendipity is a compound metric, or concept, of amongst others novelty and diversity. Novelty expresses how new a recommended item is (for a user). The underlying motivation for novelty being an interesting aspect of recommendation is that items which are old, or rather not new, can already have been seen by the user. If this is the case, recommending items which are already known by the user might not be of much value, since the recommendation does not actually present the user with something she could not find herself. Novelty can be directly measured in online experiments by asking users whether they are familiar with the recommended item (Celma and Herrera 2008). However, it is also interesting to measure novelty in an offline experiment, so as not to restrict its evaluation to costly and hardly reproducible online experiments. While novelty often can have a negative effect on accuracy, diversity can often be increased without necessarily sacrificing accuracy. Diversity expresses, as implied, the variety of the recommended items. There are multiple ways of measuring diversity in a set of recommended items; commonly, this is done by measuring the *intralist diversity* (Smyth and McClave 2001), which is defined as

$$\mathrm{ILD} = \frac{1}{|R|(|R|-1)} \sum_{i \in R} \sum_{j \in R} (1 - s(i,j)) \quad (3)$$

where $s(i, j)$ is a similarity measure reporting on the similarity of items i and j given some predefined set of item features. When using ILD as a measure, the goal is to generate a list of recommended items that contains items that are both accurate and diverse.

Metrics based on information theoretic properties of the items being recommended have also been proposed by several authors. In Bellogín et al. (2010), the entropy function is used to

capture the novelty of a recommendation list; in Zhou et al. (2010), the authors use the self-information of the user's top recommended items; and in Filippone and Sanguinetti (2010), the Kullback–Leibler divergence is used.

Other metrics such as privacy, adaptivity, and confidence have been explored to a lesser extent, but their importance and application to recommender systems have been discussed, making clear their relation with the user's experience and satisfaction, which is the ultimate goal of a "good" recommender system (Herlocker et al. 2004; McNee et al. 2006; Gunawardana and Shani 2015).

Business Metrics

As stated earlier, business metrics belong to an often neglected dimension in research literature. However, it should be safe to assume that business metrics are among the most important indicators of how well a recommender performs. This contradiction is in part due to the difficulty in measuring the business impact of a recommendation algorithm prior to deployment within a service, which could in turn explain the lesser focus on this from the research community.

Where error metrics, ranking metrics, and non-accuracy metrics attempt to measure an aspect of data interaction such as a user clicking on a recommendation, a user rating an item, or similar, business metrics measure the quality of the recommendation as seen from the vendor's perspective. It should not be forgotten that most systems that utilize recommendation do so with the purpose of increasing revenue, and naturally keeping their users satisfied with the offered service. One common business metric quite simply measures the **coverage** of the list of recommended item, i.e., how well does the list correspond to what is currently in stock or else-ways available. Coverage does, however, not only apply to the catalog of available items, it also applies to the users, i.e., an algorithm which can recommend very accurate items, but only for a small portion of users might not be as "good" as a slightly less accurate algorithm with a higher user coverage. In Gunawardana and Shani (2015), two metrics are proposed for measuring item coverage: one based on the Gini index and another based on Shannon's entropy. In Ge et al. (2010), the authors propose simple ratio quantities to measure such metrics, and to discriminate between the percentage of the items for which the system is able to generate a recommendation (*prediction coverage*), and the percentage of the available items that are effectively ever recommended (*catalog coverage*).

A metric that cannot be measured without a running system with a large number of users is **churn rate**. Churn rate is not specifically linked to RSs; however, for a service which focuses on recommendation, it should be an important indicator of the service quality. The churn rate is a measure of the users that are leaving the service for another. One variation of churn rate, revenue churn, measures the changes in revenue.

Additional business-related measures of recommendation quality state the balance between the complexity of an algorithm in terms of development cost, CPU cost, maintenance, versus the benefits it brings to the vendor (Amatriain and Basilico 2012).

Evaluation Methods

The metrics presented before can be applied under different experimental conditions. On the one hand, offline evaluation allows to compare a wide range of candidate algorithms at a low cost; it is easy to conduct and does not require any interaction with real users. On the other hand, user studies and online experiments are more trustworthy – since the system is used by real users and interacted with in real time – but care must be taken to consider biases in the experimental design (Gunawardana and Shani 2015).

Offline Evaluation

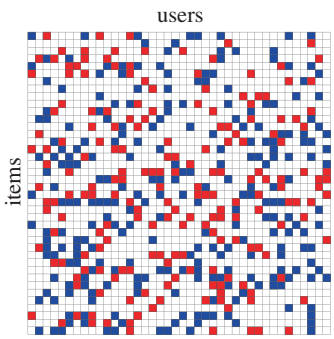
An important decision in the experimental configuration of a recommender evaluation is the dataset partition strategy. How the datasets are partitioned into training and test sets may have a considerable impact on the final performance results, and may cause some recommenders to obtain better or worse results depending on how this partition is configured.

First, we have to choose whether or not to take time into account (Gunawardana and Shani 2015). Time-based approaches naturally require the availability of user interaction data timestamps. A simple approach is to select a time point in the available interaction data timeline to separate training data (all interaction records prior to that point) and test data (dated after the split time point). The split point can be set so as to, for instance, have a desired training/test ratio in the experiment. The ratio can be global, with a single common split point for all users, or user-specific, to ensure the same ratio per user. Time-based approaches have the advantage of more realistically matching working application scenarios, where "future" user likes (which would translate to positive response to recommendations by the system) are to be predicted based on past evidence. As an example, the well-known Netflix Prize provided a dataset where the test set for each user consisted on her most recent ratings (Bennett et al. 2007).

If we ignore time, there are at least the following three strategies to select the items to hide from each user: (a) sample a fixed number (different) for each user; (b) sample a fixed (but the same for all) number for each user, also known as *given n* or *all but n* protocols; (c) sample a percentage of all the interactions using cross-validation. The most usual protocol is the last one (Goldberg et al. 2001), although several authors have also used the *all but n* protocol (Breese et al. 1998). Figure 1 shows an example of a random dataset partition.

Nonetheless, independently from the dataset partition, it is recognized that the goals for which an evaluation is performed may be different in each situation, and thus, a different setting (and partition protocol) should be developed (Herlocker et al. 2004; Gunawardana and Shani 2015). If that is not the case, the results obtained in a particular setting would be biased and difficult to use in further experiments; for instance, in an online experimentation.

Regarding the actual evaluation process, there is a relation between the evaluation protocol and the evaluation metrics that can be computed. Error metrics require explicit ground truth values for every user-item pair that needs to be evaluated – that is, only items in the test set of each user will be considered. Ranked recommendations (using the metrics described before), on the other hand, require for a target user u to select two sets of items, namely relevant and not relevant items. The following candidate generation strategies, where L_u denotes the set of target items the recommender ranks (candidate items), have been proposed (we follow the notation presented in Said and Bellogín (2014)):



Recommender Systems Evaluation, Fig. 1 How the dataset would be split into training (*blue*) and test (*red*) sets. *White* cells denote unknown values in the user-item matrix

UserTest: This strategy takes the same target item sets as standard error-based evaluation: for each user u, the list L_u consists of items rated by u in the test set. The smallest set of target items for each user is selected, including no unrated items. A relevance threshold is used to indicate which of the items in the user's test are considered relevant. Threshold variations can be static for all users (Jambor and Wang 2010), or per-user (Basu et al. 1998).

TrainItems: Every rated item in the system is selected – except those rated by the target user. This strategy is useful when simulating a real system where no test is available, i.e., no need to look into the test set to generate the rankings (Bellogín et al. 2011). The relevant items for each user consist of those included in her test set; the use of a threshold to consider only highly rated items is optional although recommended.

RelPlusN: For each user, a set of highly relevant items is selected from the test set. Then, a set of nonrelevant items is created by randomly selecting N additional items. In Cremonesi et al. (2010), N is set to 1000 stating that the nonrelevant items are selected from items in the test set not rated by u. Finally, for each highly relevant item i, the recommender produces a ranking of the union between this item and the nonrelevant items.

As it was observed in Bellogín et al. (2011) and Said and Bellogín (2014), each of these candidate generation strategies, may produce a different ranking of recommendation performance – TrainItems and RelPlusN produce consistent results (although with different absolute values), whereas UserTest obtains results closer to those from error-based metrics. Hence, we should pay attention to the strategy used when ranking metrics are computed, since the amount of relevant items considered can drastically change the output of the experiment. In contrast to IR, in RS we have to define training and test sets, whereas in IR, we would have the whole dataset available, first, for the indexing task, and then, for the retrieval and evaluation tasks. In RS, we need to separate the data into training and test; the more the training available, the better the algorithm will learn the users' preferences. However, the smaller the test set, the smaller the confidence on the obtained results. This sparsity in the ground truth dimension may produce biases in the evaluation results, as observed in Bellogín (2012).

Online Evaluation

Online evaluation, as opposed to traditional offline evaluation, is performed through direct involvement of a system's users in order to establish a *qualitative* assessment of the system's quality *as perceived by the end users.*

To illustrate one of the key differences between offline evaluation and online evaluation, consider this top-N recommendation scenario: We have a user-item interaction matrix, as shown in Table 2. The table shows a matrix of 5 users and 6 items and their interactions, e.g., 1 represents an interaction (rating, purchase, etc.), 0 the lack of such.

Recommender Systems Evaluation, Table 2 A user-item matrix divided into a training set (above the dashed line) and a test set (below the *dashed line*)

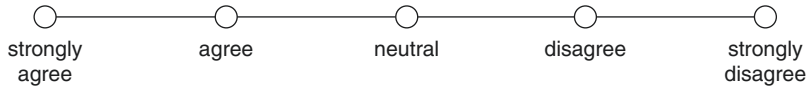
	u_1	u_2	u_3	u_4	u_5
i_1	1	1	0	0	1
i_2	1	0	1	1	1
i_3	0	0	0	1	0
i_4	1	0	1	0	1
i_5	0	0	1	1	0
i_6	0	0	0	1	0

The training/test split is illustrated by the dashed line. In this case, an offline evaluation will only recognize item i_5 as a true positive recommendation for user u_3 and items i_5 and i_6 for user u_4. Users u_1, u_2, and u_5 will not have any true positive recommendations since they have not interacted with any of the items. The evaluation does not consider that the items might actually be liked by the user, if recommended in a real-world situation. Similarly, the fact that u_3 has interacted with i_5 does not need to imply that the item is a good recommendation.

In order to overcome this deficiency, online evaluation attempts to capture the quality of the recommendation as perceived by the users by analyzing their interaction patterns with the system together with explicitly asking questions.

Online evaluation commonly involves a user study. Users can be made aware or encouraged to participate, or participate unknowingly. In real-life systems, the concept of *A/B testing* is readily used to estimate different algorithms' qualities (Kohavi et al. 2009). A/B testing involves assigning a subset of a system's users to the algorithm under evaluation. In studies of real-life systems, users are usually not made aware of their participation in tests (Kohavi et al. 2009). The interactions of the users are then analyzed and compared to a baseline.

More elaborate user studies, including questionnaires and other explicitly collected information, serve as an alternative to A/B testing. This type of studies commonly involve asking the users questions throughout, or after, their interaction with the system. In studies like this, the participants are naturally aware of their participation



Recommender Systems Evaluation, Fig. 2 A Likert scale where users are asked to agree or disagree with a specific question

in the study. In order to be able to analyze the results quantitatively, the users are asked to agree or disagree with a question in the form of a statement. The scale of (dis)agreement is represented as a *Likert* scale; a basic example of a Likert scale is shown in Fig. 2.

Online user-centric evaluation is a means to measure the level of *subjectively perceived quality* by the users of a system. There is a large body of work conducted on how to perform and evaluate user studies and surveys based on statistical significance tests.

There is no default, quality-related, set of questions to ask when performing a recommender systems user study; instead, questions are based on the type of quality that is sought-for, whether relating to the concepts mentioned above or to rather technical qualities, e.g., time of recommendation, number of items recommended, etc. This type of user studies needs to be meticulously planned and executed. If poorly executed, there is a risk of changing the users' opinions, e.g., through suggestive questions, or excessive work load or time involved in answering the questions. Work load and time-related issues can be mitigated by creating an incentive for the users to fulfill the survey, e.g., raffling off vouchers, prizes, etc. If no incentive is given, the time involved in answering the survey creates a decaying effect on the fraction of users who complete the study. When the users are given an incentive, there is a risk that some users will answer the questions quickly (at random) in order to be eligible for the award (Swearingen and Sinha 2001). In order to mitigate these effects, the number of questions and work load should be kept relatively low.

Challenges

Evaluating recommender systems is not an easy task. In all fairness, it is not *a* task, it is a set of several interconnected and standalone tasks that, when viewed together, should result in one (or several) measure(s) which then state the quality of the recommender. So far, we have presented measures, metrics, methods, and a general history of recommender system evaluation. In this section, we focus on the challenges, the pitfalls, and other related concepts that make evaluation difficult.

Technical Challenges

Earlier in this chapter, evaluation measures and evaluation methods have been presented. In order to accurately evaluate an RS, a combination of metrics and methods are used. The challenge becomes, for instance, what metrics to use when evaluating, or how many metrics to use. This poses the problem of how to combine the various metrics into something that can be optimized. Even though there is a significant body of work on multicriteria RS optimization, there is no clear guideline on how to perform this type of evaluation (Jambor and Wang 2010; Ge et al. 2010; Said et al. 2013b). State-of-the-art methods in multicriteria evaluation require a tradeoff between the different metrics to be evaluated; this can then be applied to offline evaluation. Online evaluation using several criterions requires elaborate qualitative analyses of long-term results of recommendation approaches.

Additional challenges related to offline evaluation focus on aspects of the underlying data which is used for both training and evaluation of RSs. The concept known as the *magic barrier* of RS (Herlocker et al. 2004) concerns the fact that user interactions are not concise, i.e., they contain noise and other irregularities which make the data not fully trustworthy. The effect of this is that when optimizing toward a certain metric, there is an upper level, a threshold beyond which optimization is useless (Said et al. 2012). This threshold

is unknown; at best, it can be estimated assuming there are enough interactions provided by each user (Bellogín et al. 2014).

There exist multiple other factors making evaluation difficult, and at some points even flawed. Most RSs are used for the purpose of alleviating users in finding interesting information; this usually creates a bias in terms of popularity, i.e., the fact that some items are more popular than others create synthetic similarities between users based solely on their interaction with popular items. If not taken into consideration, the evaluation of an RS can point to an algorithm performing significantly better than it would in a real-world situation simply due to the fact that it will recommend mostly popular items (Zhao et al. 2013), i.e., items that the user does not need to get recommended as there is a high probability that she already knows of the item.

Replication and Reproducibility

Looking back on the accumulated amount of work in the research community, there have emerged several recommendation frameworks: *Apache Mahout* (Mahout) (Owen et al. 2011), *LensKit* (Ekstrand et al. 2011a), *MyMediaLite* (Gantner et al. 2011), *RankSys* (Vargas 2015), etc. Even though the frameworks provide basically the same recommendation algorithms, they have differences in their implementations, data management, and evaluation methods. Additionally, the frameworks provide basic evaluation packages able to calculate some of the most common evaluation metrics; however, due to the differences in implementation of the same algorithm across frameworks, even when using the same dataset, it becomes uncertain whether a comparison of results from different frameworks is possible.

In Said and Bellogín (2014), a standalone evaluation of recommendation results was performed, allowing fine-grained control of a wide variety of parameters within the evaluation methodology. In that work, a cross-system and cross-dataset benchmark of some of the most common recommendation and rating prediction algorithms was presented. The results obtained there highlights the differences in recommendation accuracy between implementations of the same algorithms on different frameworks, distinct levels of accuracy in different datasets, and variations of evaluation results on the same dataset and in the same framework when employing various evaluation strategies. It is important to note, thus, that inter-framework comparisons of recommendation quality can potentially point to incorrect results and conclusions, unless performed with great caution and in a controlled, framework-independent environment.

In summary, the issues of replication – obtaining the exact same results in the same setting – and reproducibility – obtaining comparable results using a different setting – are very difficult challenges at the moment. They force researchers to reimplement the baseline algorithm they want to compare their approach against, or to pay extra attention to every algorithmic and evaluation detail, ignoring if the observed discrepancies with respect to what already was published come from omitted details from the original papers (parameters, methodologies, protocols, etc.) or a wrong interpretation of any of these intermediate steps.

Key Applications

In general, evaluation of recommender systems is applied whenever an RS is used. Services that use recommender systems, whether for entertainment (Netflix, Spotify, Pandora), e-learning (Coursera, EdX), networking (LinkedIn, Facebook, Twitter), shopping (Amazon, eBay), etc. needs to be evaluated in order to identify whether the recommender system in use is of practical utility to the users and to the service operators. However, evaluation is not only applied in real-world systems. Recommender systems research is contingent on the evaluation of these systems. When a novel recommendation algorithm is created by researchers, the usefulness of the approach needs to be established. In the vast majority of the cases, this is done by comparing the evaluation of the new approach to a state-of-the-art baseline algorithm. Evaluation is thus applied in order to benchmark recommender systems in academia as well as industry.

There are pitfalls related to evaluation. In a closely related research topic, information retrieval, Armstrong et al. (2009) reported that there was little measurable improvement in certain types of retrieval tasks over the course of 10 years, even though there was a large body of scientific work suggesting so. The reason for this appears to have been lack of guidelines, use of weak baselines, and not enough comparisons to results obtained in previous works. Drawing a lesson from this, we stress the importance of proper evaluation, in terms of methods, metrics, and general execution.

Future Directions

The empirical evaluation of RS is acknowledged to be an open problem in the field, with open issues yet to be addressed (Gunawardana and Shani 2015). Many experimental approaches and metrics have been developed over the years, which the community is well acquainted with, but key aspects and details in the design and application of available methodologies are open to configuration and interpretation, where even apparently subtle details may create a considerable difference. This results in a significant divergence in experimental practice, hindering the comparison and proper assessment of contributions and advances to the field.

In this context, the replication and reproduction of experiments is one of the desirable requirements for experimental research still to be met in the field, as it was already described in the technical challenges section. The discussion and definition of the basic elements of the experimental conditions (and their requirements) are critical to support continuous innovation in any discipline. The offline evaluation of RS requires an implementation of the algorithm or technique to be evaluated, a set of quality measures for comparative evaluation, and an experimental protocol establishing how to handle the data and compute metrics in detail. Online evaluation similarly requires an algorithm implementation and a population of users to survey (by means of an A/B test, for instance). Here again, perhaps even more importantly than in offline evaluation, an experimental protocol needs to be established and adhered to. However, even when a set of publicly available resources (data and algorithm implementations) exist in the RS community, very often research studies do not report comparable results for the same methods *under the same conditions* (Said and Bellogín 2014). This is due to the high number of experimental design options and parameters in RS evaluation, and the huge impact of the experiment configuration on the outcomes. In order to seek reproducibility and replication, several strategies can be considered, such as source code sharing, standardization of agreed evaluation metrics and protocols, or releasing public experimental design software, all of which have difficulties of their own. Furthermore, for online evaluation, an extensive analysis of the population of test users should be provided. While the problem of reproducibility and replication has been recognized in the community, the need for a solution remains largely unmet.

Another open problem when evaluating RS is the relation between online and offline experiments. Several authors have explored this issue in some domains but no conclusive results have been obtained (Garcin et al. 2014; Beel et al. 2013; de Souza Pereira Moreira et al. 2015). The main question is how to align the offline evaluation to the online (usually, with A/B tests) results. Some possibilities include designing a good evaluation methodology or using a sensible evaluation metric to the problem at hand. An interesting output that could be produced by a better understanding of this issue is that offline evaluation would predict which methods, parameters, or configurations will work better when integrated and tested in an online evaluation.

Additionally, there should be an increased effort in making metrics meaningful. Evaluation should be realistic and, at the same time, provide test setups and reproducible baselines. For this, we need an honest measure of the preference (predicted items may not be correct just because they were consumed), we should capture the value of the recommendation (it is hard to say if a recommender that is useful in the short term may be just too obvious), and not neglect contextual

side-data. With this goal in mind, evaluation metrics from other areas – specially from information retrieval, considering the increasing importance item rankings have in recommendation nowadays – could be incorporated and its potential analyzed. As an example, the bpref metric (Buckley and Voorhees 2004) was specifically defined to be robust to incomplete judgments sets – a very typical scenario in recommendation; however, this metric is seldom use in recommendation tasks. An interesting problem that is still unsolved is how to properly combine evaluation metrics measuring orthogonal dimensions. A paradigmatic example is the tradeoff between accuracy and coverage (Gunawardana and Shani 2015), where we may prefer recommenders that can provide recommendations to a wider range of users, despite their lower global accuracy. Another example arises when several dimensions are considered and optimized at the same time (see, e.g., Ribeiro et al. (2012)); in these cases, multiobjective evaluation measures should be used, like the one proposed in Said et al. (2013b).

Finally, the evaluation of RSs deserves a closer analysis and more attention to the differences in the experimental conditions, and their implications on the explicit and implicit principles and assumptions on which the metrics are built. As it was detected and examined in Bellogín (2012), different statistical biases – e.g., test sparsity and item popularity – do interfere in recommendation experiments, even though they do not arise in evaluation scenarios on other domains (like IR).

Cross-References

- ▶ Community Detection and Recommender Systems
- ▶ Emotions and Personality in Recommender Systems
- ▶ Friends Recommendations in Dynamic Social Networks
- ▶ Recommender Systems Based on Linked Open Data
- ▶ Recommender Systems Based on Social Networks
- ▶ Recommender Systems Using Social Network Analysis: Challenges and Future Trends
- ▶ Recommender Systems, Basics of
- ▶ Recommender Systems, Semantic-Based
- ▶ Recommender Systems: Models and Techniques
- ▶ Social Recommendation in Dynamic Networks
- ▶ Social-Based Collaborative Filtering
- ▶ Spatiotemporal Personalized Recommendation of Social Media Content
- ▶ Spatiotemporal Recommendation in Geo-Social Networks
- ▶ User Behavior

References

Abel F (2015) We know where you should work next summer: Job recommendations. In: Werthner et al (2015), p 230, https://doi.org/10.1145/2792838.2799496

Amatriain X, Basilico J (2012) Netflix recommendations: beyond the 5 stars (part 1) – the netflix tech blog. http://techblog.netflix.com/2012/04/netflix-recommendations-beyond-5-stars.html. Retrieved July 27, 2016

Armstrong TG, Moffat A, Webber W, Zobel J (2009) Improvements that don't add up: ad-hoc retrieval results since 1998. In: Cheung DW, Song I, Chu WW, Hu X, Lin JJ (eds) Proceedings of the 18th ACM conference on information and knowledge management, CIKM 2009, Hong Kong, China, November 2–6, 2009, ACM, pp 601–610, https://doi.org/10.1145/1645953.1646031

Baeza-Yates RA, Ribeiro-Neto BA (2011) Modern information retrieval – the concepts and technology behind search, Second edn. Pearson Education Ltd., Harlow, England. http://www.mir2ed.org/

Balabanovic M, Shoham Y (1997) Content-based, collaborative recommendation. Commun ACM 40(3):66–72. https://doi.org/10.1145/245108.245124

Basu C, Hirsh H, Cohen WW (1998) Recommendation as classification: using social and content-based information in recommendation. In: Mostow J, Rich C (eds). AAAI/IAAI, AAAI Press/MIT Press, pp 714–720

Beel J, Genzmehr M, Langer S, Nürnberger A, Gipp B (2013) A comparative analysis of offline and online evaluations and discussion of research paper recommender system evaluation. In: Bellogín A, Castells P, Said A, Tikk D (eds) Proceedings of the international workshop on reproducibility and replication in recommender systems evaluation, RepSys 2013, Hong Kong, China, October 12, 2013, ACM, pp 7–14, https://doi.org/10.1145/2532508.2532511

Bellogín A (2012) Recommender system performance evaluation and prediction: An information retrieval

perspective. PhD thesis, Universidad Autónoma de Madrid

Bellogín A, de Vries AP (2013) Understanding similarity metrics in neighbour-based recommender systems. In: Kurland O, Metzler D, Lioma C, Larsen B, Ingwersen P (eds) International conference on the theory of information retrieval, ICTIR'13, Copenhagen, Denmark, September 29–October 02, 2013, ACM, p 13, https://doi.org/10.1145/2499178.2499186

Bellogín A, Cantador I, Castells P (2010) A study of heterogeneity in recommendations for a social music service. In: Proceedings of the 1st international workshop on information heterogeneity and fusion in recommender systems, ACM, New York, NY, USA, HetRec'10, pp 1–8, https://doi.org/10.1145/1869446.1869447

Bellogín A, Castells P, Cantador I (2011) Precision-oriented evaluation of recommender systems: an algorithmic comparison. In: Mobasher B, Burke RD, Jannach D, Adomavicius G (eds) Proceedings of the 2011 ACM conference on recommender systems, RecSys 2011, Chicago, IL, USA, October 23–27, 2011, ACM, pp 333–336, https://doi.org/10.1145/2043932.2043996

Bellogín A, Cantador I, Díez F, Castells P, Chavarriaga E (2013) An empirical comparison of social, collaborative filtering, and hybrid recommenders. ACM TIST 4(1):14. https://doi.org/10.1145/2414425.2414439

Bellogín A, Said A, de Vries AP (2014) The magic barrier of recommender systems – no magic, just ratings. In: Dimitrova V, Kuflik T, Chin D, Ricci F, Dolog P, Houben G (eds) User modeling, adaptation, and personalization – 22nd international conference, UMAP 2014, Aalborg, Denmark, July 7–11, 2014. Proceedings, Springer, Lecture Notes in Computer Science, vol 8538, pp 25–36, https://doi.org/10.1007/978-3-319-08786-3_3

Bennett J, Lanning S, Netflix N (2007) The netflix prize. In: In KDD Cup and Workshop in conjunction with KDD

Berkovsky S, Freyne J, Coombe M (2012) Physical activity motivating games: be active and get your own reward. ACM Trans Comput-Hum Interact 19(4):32. https://doi.org/10.1145/2395131.2395139

Bistaffa F, Filippo A, Chalkiadakis G, Ramchurn SD (2015) Recommending fair payments for large-scale social ridesharing. In: Werthner et al (2015), pp 139–146, https://doi.org/10.1145/2792838.2800177

Bollen DGFM, Knijnenburg BP, Willemsen MC, Graus MP (2010) Understanding choice overload in recommender systems. In: Amatriain et al (2010), pp 63–70, https://doi.org/10.1145/1864708.1864724

Breese JS, Heckerman D, Kadie CM (1998) Empirical analysis of predictive algorithms for collaborative filtering. In: Cooper GF, Moral S (eds) UAI'98: Proceedings of the fourteenth conference on uncertainty in artificial intelligence, University of Wisconsin Business School, Madison, July 24–26, 1998, Morgan Kaufmann, pp 43–52. https://dslpitt.org/uai/displayArticleDetails.jsp?mmnu=1&smnu=2&article_id=231&proceeding_id=14

Buckley C, Voorhees EM (2004) Retrieval evaluation with incomplete information. In: Sanderson et al (2004), pp 25–32, https://doi.org/10.1145/1008992.1009000

Burke RD (2004) Hybrid recommender systems with case-based components. In: Funk P, González-Calero PA (eds) Advances in Case-Based Reasoning, 7th European conference, ECCBR 2004, Madrid, Spain, August 30 – September 2, 2004, Proceedings, Springer, Lecture Notes in Computer Science, vol 3155, pp 91–105, https://doi.org/10.1007/978-3-540-28631-8-8

Campos PG, Díez F, Cantador I (2014) Time-aware recommender systems: a comprehensive survey and analysis of existing evaluation protocols. User Model User-Adapt Interact 24(1–2):67–119. https://doi.org/10.1007/s11257-012-9136-x

Castells P, Hurley NJ, Vargas S (2015) Novelty and diversity in recommender systems. In: Ricci et al (2015), pp 881–918, https://doi.org/10.1007/978-1-4899-7637-6_26

Celma Ò, Herrera P (2008) A new approach to evaluating novel recommendations. In: Pu P, Bridge DG, Mobasher B, Ricci F (eds) Proceedings of the 2008 ACM conference on recommender systems, RecSys 2008, Lausanne, October 23–25, 2008, ACM, pp 179–186, https://doi.org/10.1145/1454008.1454038

Cremonesi P, Koren Y, Turrin R (2010) Performance of recommender algorithms on top-n recommendation tasks. In: Amatriain et al (2010), pp 39–46, https://doi.org/10.1145/1864708.1864721

Cremonesi P, Garzotto F, Negro S, Papadopoulos AV, Turrin R (2011) Comparative evaluation of recommender system quality. In: Tan DS, Amershi S, Begole B, Kellogg WA, Tungare M (eds) Proceedings of the international conference on human factors in computing systems, CHI 2011, Extended Abstracts Volume, Vancouver, May 7–12, 2011, ACM, pp 1927–1932, https://doi.org/10.1145/1979742.1979896

de Souza Pereira Moreira G, de Souza GA, da Cunha AM (2015) Comparing offline and online recommender system evaluations on long-tail distributions. In: Castells P (ed) Poster proceedings of the 9th ACM conference on recommender systems, RecSys 2015, Vienna, September 16, 2015, CEUR-WS.org, CEUR Workshop Proceedings, vol 1441. http://ceur-ws.org/Vol-1441/recsys2015_poster4.pdf

Deshpande M, Karypis G (2004) Item-based top-N recommendation algorithms. ACM Trans Inf Syst 22(1):143–177. https://doi.org/10.1145/963770.963776

Ekstrand MD, Ludwig M, Konstan JA, Riedl J (2011a) Rethinking the recommender research ecosystem: reproducibility, openness, and lenskit. In: RecSys, pp 133–140

Ekstrand MD, Riedl J, Konstan JA (2011b) Collaborative filtering recommender systems. Foundations and Trends in Human-Computer Interaction 4(2):175–243. https://doi.org/10.1561/1100000009

Elahi M, Ge M, Ricci F, Massimo D, Berkovsky S (2014) Interactive food recommendation for groups. In: Chen L, Mahmud J (eds) Poster proceedings of the 8th ACM conference on recommender systems, RecSys 2014, Foster City, October 6–10, 2014, CEUR-WS.org, CEUR Workshop Proceedings, vol 1247. http://ceur-ws.org/Vol-1247/recsys14_poster2.pdf

Elahi M, Ge M, Ricci F, Fernández-Tobías I, Berkovsky S, Massimo D (2015) Interaction design in a mobile food recommender system. In: O'Donovan J, Felfernig A, Tintarev N, Brusilovsky P, Semeraro G, Lops P (eds) Proceedings of the joint workshop on interfaces and human decision making for recommender systems, IntRS 2015, co-located with ACM conference on recommender systems (RecSys 2015), Vienna, September 19, 2015, CEUR-WS.org, CEUR Workshop Proceedings, vol 1438, pp 49–52. http://ceur-ws.org/Vol-1438/paper9.pdf

Elsweiler D, Harvey M, Ludwig B, Said A (2015) Bringing the "healthy" into food recommenders. In: Ge M, Ricci F (eds) Proceedings of the 2nd international workshop on decision making and recommender systems, Bolzano, October 22–23, 2015, CEUR-WS.org, CEUR Workshop Proceedings, vol 1533, pp 33–36. http://ceur-ws.org/Vol-1533/paper8.pdf

Filippone M, Sanguinetti G (2010) Information theoretic novelty detection. Pattern Recogn 43(3):805–814. https://doi.org/10.1016/j.patcog.2009.07.002

Gantner Z, Rendle S, Freudenthaler C, Schmidt-Thieme L (2011) Mymedialite: A free recommender system library. In: RecSys, https://doi.org/10.1145/2043932.2043989

Garcin F, Faltings B, Donatsch O, Alazzawi A, Bruttin C, Huber A (2014) Offline and online evaluation of news recommender systems at swissinfo.ch. In: Kobsa et al (2014), pp 169–176, https://doi.org/10.1145/2645710.2645745

Ge M, Delgado-Battenfeld C, Jannach D (2010) Beyond accuracy: evaluating recommender systems by coverage and serendipity. In: Amatriain et al (2010), pp 257–260, https://doi.org/10.1145/1864708.1864761

Goldberg KY, Roeder T, Gupta D, Perkins C (2001) Eigentaste: a constant time collaborative filtering algorithm. Inf Retr 4(2):133–151. https://doi.org/10.1023/A:1011419012209

Gunawardana A, Shani G (2015) Evaluating recommender systems. In: Ricci et al (2015), pp 265–308, https://doi.org/10.1007/978-1-4899-7637-6_8

Guy I (2015) Social recommender systems. In: Ricci et al (2015), pp 511–543, https://doi.org/10.1007/978-1-4899-7637-6_15

Herlocker JL, Konstan JA, Terveen LG, Riedl J (2004) Evaluating collaborative filtering recommender systems. ACM Trans Inf Syst 22(1):5–53. https://doi.org/10.1145/963770.963772

Jambor T, Wang J (2010) Optimizing multiple objectives in collaborative filtering. In: RecSys, ACM, New York, pp 55–62, https://doi.org/10.1145/1864708.1864723

Jannach D, Lerche L, Jugovac M (2015) Adaptation and evaluation of recommendations for short-term shopping goals. In: Werthner et al (2015), pp 211–218, https://doi.org/10.1145/2792838.2800176

Järvelin K, Kekäläinen J (2002) Cumulated gain-based evaluation of IR techniques. ACM Trans Inf Syst 20(4):422–446. https://doi.org/10.1145/582415.582418

Kohavi R, Longbotham R, Sommerfield D, Henne RM (2009) Controlled experiments on the web: survey and practical guide. Data Min Knowl Discov 18(1):140–181. https://doi.org/10.1007/s10618-008-0114-1

Luo L, Li B, Berkovsky S, Koprinska I, Chen F (2016) Who will be affected by supermarket health programs? tracking customer behavior changes via preference modeling. In: Bailey J, Khan L, Washio T, Dobbie G, Huang JZ, Wang R (eds) Advances in knowledge discovery and data mining – 20th Pacific-Asia conference, PAKDD 2016, Auckland, April 19–22, 2016, Proceedings, Part I, Springer, Lecture Notes in Computer Science, vol 9651, pp 527–539, https://doi.org/10.1007/978-3-319-31753-3_42

Marlin BM (2003) Modeling user rating profiles for collaborative filtering. In: Thrun S, Saul LK, Schölkopf B (eds) Advances in neural information processing systems 16 [neural information processing systems, NIPS 2003, December 8–13, 2003, Vancouver and Whistler, BC, Canada], MIT Press, pp 627–634. http://papers.nips.cc/paper/2377-modeling-user-rating-profiles-for-collaborative-filtering

Massa P, Avesani P (2007) Trust-aware recommender systems. In: Konstan JA, Riedl J, Smyth B (eds) Proceedings of the 2007 ACM conference on recommender systems, RecSys 2007, Minneapolis, October 19–20, 2007, ACM, pp 17–24, https://doi.org/10.1145/1297231.1297235

McLaughlin MR, Herlocker JL (2004) A collaborative filtering algorithm and evaluation metric that accurately model the user experience. In: Sanderson et al (2004), pp 329–336, https://doi.org/10.1145/1008992.1009050

McNee SM, Riedl J, Konstan JA (2006) Being accurate is not enough: how accuracy metrics have hurt recommender systems. In: Olson GM, Jeffries R (eds) Extended abstracts proceedings of the 2006 conference on human factors in computing systems, CHI 2006, Montréal, April 22–27, 2006, ACM, pp 1097–1101, https://doi.org/10.1145/1125451.1125659

Owen S, Anil R, Dunning T, Friedman E (2011) Mahout in Action. Manning Publications Co., Greenwich, CT, USA. Rennie JDM, Srebro N (2005) Fast maximum margin matrix factorization for collaborative prediction. In: Raedt LD, Wrobel S (eds) Machine learning, proceedings of the twenty-second international conference (ICML 2005), Bonn, August 7–11, 2005, ACM, ACM international conference proceeding series, vol 119, pp 713–719, https://doi.org/10.1145/1102351.1102441

Resnick P, Iacovou N, Suchak M, Bergstrom P, Riedl J (1994) Grouplens: An open architecture for collaborative filtering of netnews. In: Smith JB, Smith FD, Malone TW (eds) CSCW'94, Proceedings of the conference on computer supported cooperative work, Chapel Hill, NC, USA, October 22–26, 1994, ACM, pp 175–186, https://doi.org/10.1145/192844.192905

Ribeiro MT, Lacerda A, Veloso A, Ziviani N (2012) Pareto-efficient hybridization for multi-objective recommender systems. In: Cunningham P, Hurley NJ, Guy I, Anand SS (eds) Sixth ACM conference on recommender systems, RecSys'12, Dublin, September 9–13, 2012, ACM, pp 19–26, https://doi.org/10.1145/2365952.2365962

Ricci F, Rokach L, Shapira B (eds) (2015) Recommender Systems Handbook. Springer, New York. https://doi.org/10.1007/978-1-4899-7637-6

Said A (2013) Evaluating the accuracy and utility of recommender systems. PhD thesis, Technische Universität Berlin

Said A, Bellogín A (2014) Comparative recommender system evaluation: benchmarking recommendation frameworks. In: Kobsa et al (2014), pp 129–136, https://doi.org/10.1145/2645710.2645746

Said A, Jain BJ, Narr S, Plumbaum T (2012) Users and noise: The magic barrier of recommender systems. In: Masthoff J, Mobasher B, Desmarais MC, Nkambou R (eds) User modeling, adaptation, and personalization – 20th international conference, UMAP 2012, Montreal, July 16–20, 2012. Proceedings, Springer, Lecture Notes in Computer Science, vol 7379, pp 237–248, https://doi.org/10.1007/978-3-642-31454-4_20

Said A, Fields B, Jain BJ, Albayrak S (2013a) User-centric evaluation of a k-furthest neighbor collaborative filtering recommender algorithm. In: Bruckman A, Counts S, Lampe C, Terveen LG (eds) Computer supported cooperative work, CSCW 2013, San Antonio, February 23–27, 2013, ACM, pp 1399–1408, https://doi.org/10.1145/2441776.2441933

Said A, Jain BJ, Albayrak S (2013b) A 3d approach to recommender system evaluation. In: Bruckman A, Counts S, Lampe C, Terveen LG (eds) Computer supported cooperative work, CSCW 2013, San Antonio, February 23–27, 2013, Companion Volume, ACM, pp 263–266, https://doi.org/10.1145/2441955.2442017

Said A, Bellogín A, Lin JJ, de Vries AP (2014a) Do recommendations matter?: news recommendation in real life. In: Fussell SR, Lutters WG, Morris MR, Reddy M (eds) Computer supported cooperative work, CSCW'14, Baltimore, February 15–19, 2014, Companion Volume, ACM, pp 237–240, https://doi.org/10.1145/2556420.2556510

Said A, Tikk D, Cremonesi P (2014b) Benchmarking – a methodology for ensuring the relative quality of recommendation systems in software engineering. In: Robillard MP, Maalej W, Walker RJ, Zimmermann T (eds) Recommendation Systems in Software Engineering. Springer, Berlin, pp 275–300. https://doi.org/10.1007/978-3-642-45135-5-11

Smyth B, McClave P (2001) Similarity vs. diversity. In: Aha DW, Watson ID (eds) Case-Based Reasoning Research and Development, 4th international conference on case-based reasoning, ICCBR 2001, Vancouver, July 30 – August 2, 2001, Proceedings, Springer, Lecture Notes in Computer Science, vol 2080, pp 347–361, https://doi.org/10.1007/3-540-44593-5_25

Swearingen K, Sinha R (2001) Beyond algorithms: An HCI perspective on recommender systems. In: ACM SIGIR. Workshop on recommender systems, vol 13, no 5–6, pp 393–408

Tkalcic M, Quercia D, Graf S (2016) Preface to the special issue on personality in personalized systems. User Model User-Adapt Interact 26(2–3):103–107. https://doi.org/10.1007/s11257-016-9175-9

Tomlinson S (2012) Measuring robustness with first relevant score in the TREC 2012 microblog track. In: Voorhees EM, Buckland LP (eds) Proceedings of the twenty-first text retrieval conference, TREC 2012, Gaithersburg, November 6–9, 2012, National Institute of Standards and Technology (NIST), vol Special Publication 500–298. http://trec.nist.gov/ pubs/trec21/papers/OpenText.microblog.final.pdf

Vargas S (2015) Novelty and diversity evaluation and enhancement in recommender systems. PhD thesis, Universidad Autónoma de Madrid

Vargas S, Castells P (2013) Exploiting the diversity of user preferences for recommendation. In: Ferreira J, Magalhães J, Calado P (eds) Open research areas in information retrieval, OAIR'13, Lisbon, May 15–17, 2013, ACM, pp 129–136. http://dl.acm.org/citation.cfm?id=2491776

Vargas S, Castells P (2014) Improving sales diversity by recommending users to items. In: Kobsa et al (2014), pp 145–152, https://doi.org/10.1145/2645710.2645744

Vargas S, Baltrunas L, Karatzoglou A, Castells P (2014) Coverage, redundancy and size-awareness in genre diversity for recommender systems. In: Kobsa et al (2014), pp 209–216, https://doi.org/10.1145/2645710.2645743

Yao YY (1995) Measuring retrieval effectiveness based on user preference of documents. JASIS 46(2):133–145. https://doi.org/10.1002/(SICI)1097-4571(199503)46:2<133::AID- ASI6>3.0.CO;2-Z

Zhao X, Niu Z, Chen W (2013) Opinion-based collaborative filtering to solve popularity bias in recommender systems. In: Decker H, Lhotská L, Link S, Basl J, Tjoa AM (eds) Database and expert systems applications – 24th international conference, DEXA 2013, Prague, August 26–29, 2013. Proceedings, Part II, Springer, Lecture Notes in Computer Science, vol 8056, pp 426–433, https://doi.org/10.1007/978-3-642-40173-2_35

Zhao X, Zhang W, Wang J (2015) Risk-hedged venture capital investment recommendation. In: Werthner et al (2015), pp 75–82, https://doi.org/10.1145/2792838.2800181

Zhou T, Kuscsik Z, Liu JG, Medo M, Wakeling JR, Zhang YC (2010) Solving the apparent diversity-accuracy

dilemma of recommender systems. Proc Natl Acad Sci 107(10):4511–4515. https://doi.org/10.1073/pnas.1000488107

Ziegler C, Lausen G (2009) Making product recommendations more diverse. IEEE Data Eng Bull 32(4): 23–32. http://sites.computer.org/ debull/A09dec/ziegler-paper1.pdf

Ziegler C, McNee SM, Konstan JA, Lausen G (2005) Improving recommendation lists through topic diversification. In: Ellis A, Hagino T (eds) Proceedings of the 14th international conference on World Wide Web, WWW 2005, Chiba, Japan, May 10–14, 2005, ACM, pp 22–32, https://doi.org/10.1145/1060745.1060754

Recommender Systems Using Social Network Analysis: Challenges and Future Trends

Fabrice Muhlenbach, Christine Largeron and Johann Stan
Univ. Lyon, UJM-Saint-Etienne, CNRS, Laboratoire Hubert Curien UMR 5516, Saint Etienne, France

Synonyms

Collaborative filtering; Content-based filtering; Information filtering; Recommendation system; Recommender engine; Recommender platform

Glossary

Online Social Networking Services (OSNS)	Web platform having as a goal to build social connections among people who share similar interests or relationships of any kind
Recommender System (RS)	Special type of information filtering system that provides a prediction that assists the user in evaluating items from a large collection that the user is likely to find interesting or useful
Status Update (Micropost)	Short message, shared in an online social platform, expressing an activity, state of mind, or opinion

Definition

Recommender systems (RSs) are software tools and techniques dedicated to generate meaningful suggestions about new items (products and services) for particular customers (the users of the RS). These recommendations will help the users to make decisions in multiple contexts, such as what items to buy, what music to listen to, what online news to read (Ricci et al. 2015), or, in the social network domain (Can 2014), which user to connect to (Arias et al. 2012) or which users to consider as a trustful adviser (Victor et al. 2011a).

Introduction

Recommender systems and social platforms mutually benefit each other. A social network can usually be defined as a set of entities interconnected, and it can be represented as a graph where the entities are described by nodes and their relationships by links. However, in real-world settings, networks are often complex. They can be *heterogeneous* when they contain different types of objects and links or *attributed* when the vertices are described by attributes. Moreover, to take into account their dynamicity, several graphs can be used, one for each timestamp. It should also be noticed that this concept is not limited to the case of online social networking services (OSNSs) such as *Facebook*, *LinkedIn*, or *Twitter*, the main focus of our work. A common characteristic of these networks, and more specifically modern OSNSs, is that they are composed of (i) users (with a user profile, activities, and connections) and (ii) social objects representing the intermediations, e.g., topics of user interactions, shared videos, and photos.

The user profile generally includes static personal information, such as the name, e-mail, and address, as well as more dynamic information about the interests and information needs of the user. The role of the user profile is essential in online communities. Generally user profiles are different from one application to another, as users present themselves differently, based on the targeted population of the given application

(which are sometimes very specific). Another dimension of users is represented by the activities they perform in the social platform. This includes content sharing, media uploading, and content description (such as photo tagging). Finally, the third dimension of users is represented by the social connections they establish with others in the network. Users in these OSNSs are generally connected to different communities, belonging to different social spheres (e.g., friends, family, coworkers).

Another important user characteristic is related to trust. Indeed, the different applications on social content sites allow users to be closer to their communities and to be aware of peer activities and opinions. This brings new dimensions to trust and allows users to have higher confidence in the recommendations, suggestions, and sentiment of friends.

Shared social objects influence interactions between users. An object in this context has a concrete and perceptible, physical and/or numeric, manifestation. Some objects are the source of conversational interactions and keepers of collective attention. They constitute a conversation support. In our actual digital context, objects are mainly multimedia ones as articles (*Wordpress*, *Wikipedia*), videos (*YouTube*, *Dailymotion*), pictures (*Flickr*, *Picasa*, *Instagram*), or specific status updates shared by users (*Facebook*).

In such systems, users can employ different types of annotations to describe social objects: structured annotations (in this case, the terms employed in the annotation are regulated by a common domain vocabulary that must be used by the members of the system) and semi-structured annotations (these annotations are generally freely selected keywords without a vocabulary in the background, and a collection of these annotations is called a *folksonomy*). The last category of such annotations is unstructured, which is the most frequently used in social platforms, and therefore, we describe it in more detail.

This can be found in the majority of social networks and microblogging systems and primarily consists of free texts in the form of short messages describing a resource, a finding, an impression, a feeling, a recent activity, a mood, or a future plan. A common practice is either to express an opinion about the resource (e.g., web page) or to provide its short summary for the community.

The limitations of this kind of content sharing from the viewpoint of information retrieval and knowledge management are similar to that of social tagging, as users have complete freedom in the formulation of these messages. More concretely, it is difficult to extract interesting topics or named entities from such messages, given the fact that there is an ambiguous, frequently changing underlying vocabulary.

Key Points

Nowadays, the wide use of Internet around the world allows a lot of people to connect. This explosion of the Web 2.0 (blogs, wikis, content sharing sites, social networks, etc.) gives rise to a growing need for RSs based on social and information network mining methods. For such systems, the underlying social structure, also called social network or virtual community, can be leveraged.

The substantial growth of the social web poses both challenges and new opportunities for research in RSs. The main reason for this is the fact that the social web transforms information consumers into active contributors, allowing them to share their status, comment or rate web content. Finding relevant and interesting content at the right time and in the right context is challenging for existing recommender approaches.

At the same time, the major added value of social platforms is to encourage interaction between users. Each interaction can be extracted and used as an input for the RS, as it helps to better understand the user interests and information needs. Also, the structure of the underlying social network in a social platform can contribute to generate recommendations that are more trusted by users (e.g., by considering the social distance in the recommendation process, as generally we trust more recommendations from closer connections). Some aspects related to users and available

in OSNS, e.g., preferences, opinions, behavior, or feedbacks, can be used in a user-centered recommendation approach for improving the RS by making better personalized recommendations (Colace et al. 2015). For a more detailed description on how information from social networks can be exploited to improve accuracy of recommendations, we recommend reading the survey of Yang et al. (2014).

Moreover, people are more responsive to the recommendations made by people they trust rather than automated recommender systems, resulting in new areas of study such as "social recommender systems," "trust-enhanced recommendation systems" (Victor et al. 2011a, 2011b), or even "social network-based recommender systems" (Schall 2015). Therefore, we can conclude that the social web provides a huge opportunity for improving RSs (Fig. 1).

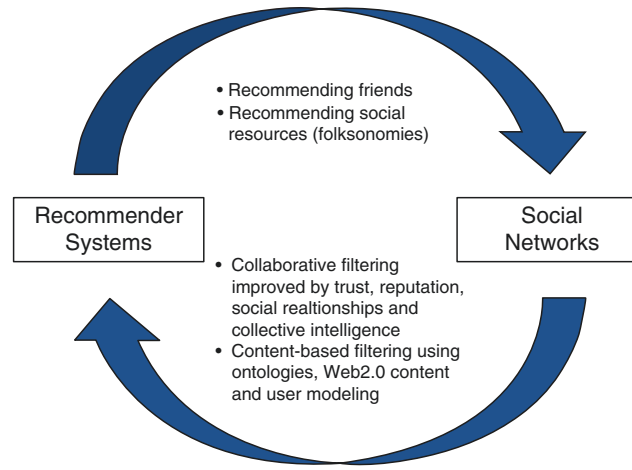
On the other hand, RSs can clearly help to improve user participation in social systems, as they can recommend new friends or interesting content (Arias et al. 2012). Thus, the user will be more motivated to keep ongoing participation in the social platform, because the more content he/she shares, the more relevant connections the system can recommend, having a precise profile about him/her.

Using this connection between social platforms and RSs, new scenarios can be defined for advanced applications, such as people recommendation or various content recommendations (e.g., tags for photo annotation).

Historical Background

Since the origin of humanity, the human being is part of a social relationship network. On the one hand, a human being often makes choices in his everyday life. These choices can be critical, e.g., for our hunter-gatherer ancestors: Is this plant edible or poisonous? How to catch this animal in order to feed me? Requesting advices and recommendations from others has thus proved to be a fundamental necessity. These others providing recommendations can be individuals recognized for their expertise in a given field, or people that this human being knows because they share some common elements with him, and who he can trust.

On the other hand, as human being is a social animal, he feels a natural tendency to belong to a particular group. Usually to recognize this human being as a member of the group, a given social group puts in place a system of filtering by tests, often ritualized, e.g., the rites of passage like the adulthood for indicating a significant change of status in some societies, the brotherhoods and guilds of the Middle Ages considered as a way to transmit knowledge and skills for being recognized as an expert in a given trade, or the defense of a PhD dissertation to obtain the highest



Recommender Systems Using Social Network Analysis: Challenges and Future Trends, Fig. 1 Reciprocal contributions made by recommender systems to social networks

university degree and being recognized as a doctor, which is the terminal educational degree in many fields. Belonging to a group of experts or the membership of a given social group may facilitate the exchange of information, more confidence to the recommendations made by people are given as the links that we can have with these people in the social network.

More recently, automated recommender systems appeared soon after the arrival of the World Wide Web, in the 1990s. In 1992, John Riedl and Paul Resnick started to work on a collaborative filtering system for *Usenet* news and proposed *GroupLens* recommender system in 1994 (Resnick et al. 1994): by taking into account the ratings manually made by *Usenet* readers, *GroupLens* was able to predict how much other readers would like an article before they read it (an history of recommender systems, focusing more on the collaborative filtering approach, its methods and ways of evaluating such systems, can be found in Ekstrand et al. 2011). Rapidly, in 1997 (Terveen et al. 1997), the idea came that within a collaborative filtering system the role specializations of the users are not the same and that this role could be used to improve the recommendations.

Thus, social network and recommender systems began to be linked and were even more intensely connected with the arrival in the 2000s of the OSNSs still heavily used today, e.g., *LinkedIn* (in 2002), *Facebook* (in 2004), *Twitter* (in 2006), or *ResearchGate* (in 2008).

Some OSNSs operate or have operated with filtering criterion or co-option process, as can do social networkings in real life, e.g., *Facebook* membership was initially restricted to students of Harvard College, or for being member of *ResearchGate* – an OSNS for scientists and researchers who want to share papers, ask and answer questions, and find collaborators – you need to have an e-mail address at a recognized institution or you have to be manually confirmed as a published researcher in order to sign up for an account.

Since the early 2010s, a strong entanglement has been observed between social networking and recommender systems with the introduction of the recruitment by recommendation seen as an alternative way for seeking diverse candidate profiles to the usual job boards. There is now a kind of "crowdsourced recruitment," a method used by human resources management services to find and hire employees by taking into account the recommendations of a given candidate made by his personal network (i.e., human contacts) on professional OSNSs (e.g., *LinkedIn*).

Social Network Analysis

Social network analysis (Can 2014) and social mining can be very useful in this context where RSs can take benefit from social networks and conversely, where the formation and evolution of the network can be affected by the recommendations. In order to illustrate this point, we can mention three well-known tasks in social network analysis and social network mining:

- The first one is the identification of key actors which play a particular role or which have a particular position in the network. Different indicators, such as the centrality or the prestige, were initially introduced mainly in order to highlight the "most important" actors in the network (Wasserman and Faust 1994). With the appearance of OSNSs, these measures were recently revisited to detect actors called, depending on the authors, "mediators," "ambassadors," or "experts" (Bonchi et al. 2016; Benyahia and Largeron 2015; Sol'e-Ribalta et al. 2016). Among the actors who have received a lot of attention appears notably the influencer who can be defined as an actor who has the ability to influence the behavior or opinions of the other members in the social network (Anagnostopoulos et al. 2008). The identification of the influencers can be seen as an optimization problem better known as "influence maximization" (or "spread maximization") that is NP-complete, but approximated solutions can be determined thanks to greedy algorithms like "Cost-Effective Lazy Forward" (*CELF*) algorithm or its extensions *Newgreedy*, *Mixedgreedy*, or *Celf++* (Kempe et al. 2003).

- Another well-known problem in the context of social networks is that of community detection. This problem has mainly been studied in the literature in the case where the community structure is described by a partition of the network actors where each actor belongs to one community (Schaeffer 2007; Lancichinetti and Fortunato 2009), and among the core methods, we can mention those that optimize a quality function to evaluate the goodness of a given partition, like the modularity, the ratio cut, the minmax cut, or the normalized cut, and the hierarchical techniques like divisive algorithms based on the minimum cut, spectral methods, or Markov clustering algorithm and its extensions. However, in real networks, an actor can often belong to several groups, and these overlapping communities can be detected using, e.g., the clique percolation algorithm implemented in *CFinder* or *OSLOM* (Order Statistics Local Optimization Method). Other recent works have attempted to detect communities, taking into account the profile of the users and their relationships (Combe et al. 2015). These methods can be applied to determine groups of users with similar characteristics or the same interests, and consequently, they can be integrated in neighborhood-based collaborative systems.
- The evolution of the network is another challenge. Indeed, in many networks, the structure of the network, in other words the actors as well as their relationships, changes quickly over time. The identification of evolving communities or their detection over time is also a subject of recent research which can be integrated in systems to improve recommendations, but the dynamic analysis of the network is also related to the link prediction problem which aims to determine the appearance of new links or the deletion of links in the network (Namata and Getoor 2010; Liben-Nowell and Kleinberg 2007; Getoor 2010; Hasan and Zaki 2011) and, recently, new generators have been designed for dynamic attributed networks with community structure (Benyahia et al. 2016). It is obvious that link prediction can be useful for people recommendation, and, conversely, recommendation approaches can allow to predict the evolution of the network. This temporal dimension is notably important in the context of mobile applications in which moving actors are interacting with each other.

Recommender Systems

The field of social network analysis is a complex and rapidly changing area. To understand the mutual contributions of social networks in recommender systems (and vice versa), it is necessary to clarify the basic principles of these systems.

RSs are dedicated to the help of the users when they must make a decision, taking as basis the fact that in ordinary life, people often make decisions based on the recommendation of others. At work, employers count on recommendation letters when they want to recruit new employees; with friends, we talk about books that we loved to read, music or movies that we liked, purchases that have given us satisfaction, or products that disappointed us; and more generally, we trust reviews of specialists before seeing a TV show, an art exhibit, or purchasing an item. This behavior is based both on the belief that our friends have similar tastes to ours, and on the trust that we can provide to the expert opinion. The recommendations provided by automated systems are trying to mimic those two principles, depending on the available information, and they are supplied to the users in the form of a prediction or a list of items.

The information used for the recommendation process can be extracted from the content available from the users and the items, or it can be inferred from the explicit ratings when the users are asked to rate the items. Depending on the way of how the information is used, the RS is considered to be a content based, a collaborative filtering, or a hybrid RS (where both information, collaborative and content based, are used) (Adomavicius and Tuzhilin 2005).

Whatever approach is used, the key elements of an RS are (i) the users, (ii) the items, and (iii) the transactions. The users of an RS, which may have very diverse goals and characteristics, are both those who benefit from the system and

those who supply it with information. Items are the objects (products or services) that are recommended, and they may be characterized by their complexity and their value or utility for a given user. Transactions are the recorded interactions between a user and the RS, especially the relation between a user and a given item, which can be an explicit feedback, e.g., the rating of a user for a selected item.

In the content-based approach, which has its roots in information retrieval and information filtering research, an item is recommended to a user based upon a description of the item and a profile of the user interests (Ricci et al. 2015). This family of RSs has some advantages (user independence, transparency, easy recommendation of new items) but also some drawbacks: content analysis is limited and the system suffers from over-specialization that leads to homophily (a person is only recommended by people who think like he or she).

In the collaborative filtering approach, an item is recommended to a given user by following another way: the collaborative filtering methods produce user-specific recommendations of items based on patterns of ratings without need for exogenous information about either items or users (Ricci et al. 2015). The preferences of the users are explicit: the users are asked to rate the items (e.g., in terms of l–5 star scale or "I like" / "I don't like"). This approach needs only a set of ratings of users on sets of items: a list of n users, a list of m items, and a rating $r_{x,t}$ indicates the rating of user x on the item t. In a typical collaborative filtering scenario, it is very rare (if not impractical) for a user x to rate all the m items, so the R matrix of all ratings *users* $\times$ *items* is sparse. To result in recommendation, the collaborative filtering can be either neighborhood based (memory based) or model based (Melville and Sindhwani 2010; Ricci et al. 2015). The model-based approaches try to propose a model able to predict the unknown rating of a user x for an item t by discovering the underlying preference class of users and the category class of the items. In neighborhood-based collaborative filtering, the rating matrix R is directly used to predict ratings for new items, either when the neighborhood derives from a similarity between the users (for user-based systems), or when the neighborhood derives from a similarity between the items (for item-based systems); e.g., two items are considered as neighbors if several users have rated these items in a similar way. In most cases, the similarity estimated between users or items in these approaches are Pearson correlation or vector cosine-based similarity.

The efficiency of an RS is measured in terms of relevance of the recommendations and forecast accuracy, in particular seeking to narrow the difference between the predicted ratings made by the system and the real ratings made by the users. Moreover, the system has to be a good filtering system and not present to users uninteresting items while not missing interesting items (e.g., in the case of commercial RS, for increasing the number of items sold). It is important to propose to the users items that might be hard to find without a precise recommendation. Many systems suffer from novelty discovery, i.e., they fail to find serendipitous items. All these properties will increase the user satisfaction and the fidelity to the use of the system.

The latest trends in RS domain seek to take into account how human beings function with their peers, especially in their interpersonal behaviors, which brings it closer to the field of social network analysis. Some users try to find credible recommenders so they can follow them; it is thus interesting to investigate the most influential members. It is also important to develop a method to better understand each user of the system and improve the understanding of their profiles, to identify what they like and dislike or are expecting from the system. The RS must seek to enable individual mechanisms that users can work together, because some users like to contribute to the system with their ratings and express their opinions and beliefs or can be happy to help the others by contributing with information. However, it should be cautious as there are malicious users who seek to influence others in the system just to promote or penalize certain items. A detailed overview of these properties is presented in the different chapters of the collective book edited by Ricci et al. (2015).

Social Search Systems

Frameworks that specifically target recommendation services based on user profiles are mostly in the category of people recommendation and question answering systems. Such systems explore either the topology of the network or the content of the exchanges between communities and peers. The main difference to content-based social search is the fact that the result of a recommendation is not a document but another user or group of users. In this way, the person can interact directly with the recommended user, which provides a more secure and trusted environment for the communication process. Also, such people-to-people interactions are more interesting for the service provider, as they can contribute to the growth of the social platform, which is generally measured by the number of users and connections between them.

Guy et al. (2009) present a people recommendation strategy specially adapted for the enterprise ecosystem. The recommendation engine uses information from an organization Intranet for computing similarity scores between employees. Such information include: (i) paper or patent coauthorship, (ii) commenting of each others' blogs or profiles, and (iii) mutual connection in other social networks, internal to the organization. Based on an aggregated score computed for each relationship, people are recommended to be added in an employee internal messenger system. For each recommendation, an explanation is generated, considered an important component of such systems (Herlocker et al. 2000). A limitation of this approach can be considered the fact that the recommendation only uses statistical information to infer the social proximity between users. More concretely, the content of interactions and exchanges is not taken into account to measure the similarity of interests or information needs. We also mention here the fact that most people recommendation strategies in popular social networks, such as *Facebook* or *Orkut*, are also based on this statistical similarity schema.

Lin et al. (2009) also target the issue of expertise location in the enterprise environment. The proposed system, *SmallBlue* (Lin et al. 2009), similarly to (Guy et al. 2009), employs data mining and statistical data analysis techniques to extract profile information for employees. More specifically, the system uses company e-mail as a source of information. Keywords are extracted from each e-mail, and a bag-of-words-based profile is constructed for employees. An innovative feature of the system is the social explanation of people recommendations, by displaying the social path that connects the user to the recommended person on a specific topic.

Hannon et al. (2010) go beyond the previous approach and build a recommendation strategy using the content of interactions (e.g., status updates) as input. Designed for recommending people to follow in *Twitter*, the *Twittomender* system allows users to expand their network by connecting to people that they do not know directly, but with whom they share similar interests. Each user in the system is represented by a vector, comprised of terms extracted from their shared messages. A kind of *social expansion* of this basic profile is performed, by taking into account messages shared by people connected to the user. This is based on the observation that connected people share close interest. The computation of profile similarities is achieved by the traditional *tfidf* weighting schema in information retrieval and cosine similarity. The *Twittomender* system is original and different from existing collaborative filtering approaches, as it takes into account the structure of the underlying social network to better approximate the interests of the user. It is however a considerable limitation in the system that no disambiguation or semantic expansion of profile terms are considered. More concretely, the user profile is composed of keywords that might have multiple meanings and this could be a considerable drawback for the relevance of recommendations.

A new generation of social search engines is represented by so-called *question answering systems*. The main difference to the previous approaches is the fact that in this case, the system builds a user profile from some kind of user activity (content production or consumption) and uses it to match them with a question formulated by another user.

Aardvark (Horowitz and Kamvar 2010) is certainly the most promising social search engine. *Aardvark* introduced several innovations in the field of social search. First of all, it is the first system that models the users based on their generated content. For this reason, users provide topics of interest to the system when they subscribe. Then, a crawler extracts further topics from the user's profiles and status updates in social platforms to expand the initially entered profile items. The extraction of topics from social updates is achieved by linear classifiers, such as support vector machines and probabilistic classifiers. *Aardvark* is not built on top of existing social platforms and lacks a global approach for conceptualizing user profiles.

In another recent social search engine, *CQA* (Li and King 2010), the objective is similar to that of *Aardvark*: route a question to the right person in a community of answerers. In their paper, Li and King (2010) introduce two important dimensions for such systems: (i) the consideration of the answerer availability and (ii) the question of the quality of answers. The quality of answers is estimated by taking into account statistical information about the length of the answer, the time the user took to send it, and the feedback of other users. In the case of availability, the system monitors the user log-ins and performs a prediction of whether the user will be available at a specific time and date in the future.

We can conclude that in current social search systems that offer a people recommendation service, the issue of recommendation explanation is still not well tackled (which is also strongly related to privacy management). Also, few frameworks benefit from semantic web technologies on a data storage or data enrichment level.

Another possibility to build an RS is to leverage the content shared by users in the social network. More specifically, we consider the content productions of users in order to better understand their interests and information needs and more concretely build expertise profiles. In such way, the recommender engine is able to recommend people that have similar or complementary interests. From a conceptual viewpoint, such a recommender engine is composed of two parts: (i) the identification of semantic data (e.g., entities extracted from status updates) that will compose the profile and (ii) the scoring of the said semantic data (measuring the user expertise).

We consider X the domain of all n users involved in the social platform. T_x represents the set of items correlated with user x, i.e., $T_x = t|$ *Weight* $(t, x) > 0$. Therefore, user x and item t are correlated when *Weight* $(t, x) > 0$, *Weight* being the weight of the item in the profile.

An item in the user profile can be represented by a keyword or a concept. The main difference is that concepts have URIs that provide them the exact semantic meaning. Generally, such URIs can be retrieved from so-called semantic knowledge graphs, such as *DBpedia*. Each profile item is an entity (keyword, named entity) extracted from at least one content production of the user and connected to at least one semantic concept present in at least one semantic knowledge base. The main arguments for this choice are that this kind of representation is richer and less ambiguous than a keyword-based or item-based model. It provides an adequate grounding for the representation of coarse to fine-grained user interests. A semantic knowledge base provides further formal, computer-processable meaning on the concepts (who is coaching a team, an actor filmography, financial data on a stock) and makes it available for the system to take advantage of knowledge base-originated semantic concepts that are more precise and reduce the effect of the ambiguity caused by simple keyword terms.

Normally in a conversation, we depend essentially on the context of the conversation to disambiguate a word. Similarly, in order to associate keywords or entities in a social update to the right concept in linked data, contextual cues are necessary to allow restricting the semantic field of the social update. In traditional documents, generally there are sufficient contextual cues to overcome such ambiguous situations, where the meaning of a term is not straightforward.

In the case of social platforms, the short nature of posts requires to find these cues elsewhere, so we may consider two main additional sources of contextual cues:

- The first contextual cue is user related, which consists in building incrementally a *vocabulary* from all social updates of the user. The assumption behind this first additional context is that there is a probability that the user previously shared some content in a related semantic field (e.g., a user who posted about "Apple" might have shared before about other Apple products, such as the "iPhone").
- The second additional contextual cue is community related. On social platforms users are members of different communities, which influence each other in terms of interests. Users participate in a group or a community because they are interested in what community members say, and as a consequence of this participation, users have intention of using commonly known keywords to make his/her contents easily understandable by the community. This second contextual cue is used only if the user-related one is not yet available or not sufficiently rich (e.g., user has shared few messages but has lots of friend connections). More specifically, it is a solution for the so-called cold-start situation and consists of aggregating the most recent messages of friends connected to the user and constructing a vocabulary from the content of these messages.

After the construction of the vector containing also such items that represent the context of the keyword, several similarity measures can be used to compare it with the description of candidate concepts in the knowledge base, and the best matching concept selected. A further, optional step is to leverage the semantic neighborhood of the concept to better describe the user expertise (e.g., include more general concept into the profile). This could be interesting in case of profile extracted from status updates, as such messages are short and therefore we have little available information about the user information needs or interests.

Key Applications

The special RSs making use of the knowledge that can be obtained from social networks (i.e., explicit or implicit social interaction, social influence, trust, and social behavioral patterns) to improve the recommendation process are called "social network-based recommender systems" (Cabacas et al. 2014). Depending on the application, some properties of these social network-based RSs may be more important than others: these RSs can be assessed in terms of (i) robustness (i.e., the stability of the RS when there is fake information), (ii) trust (i.e., the confidence in a human user based on his expertise and behavior for a particular context at a given time), (iii) serendipity (i.e., how surprising the successful recommendations are), (iv) diversity (i.e., how different the recommended items are with respect to each other), and (v) privacy preservation (i.e., users do not want to provide personal information that could be misused).

Some key applications of the benefits that the social network analysis can provide to the field of recommendation systems – and more specifically collaborative filtering RSs – can be associated to the following tasks identified by Herlocker et al. (2004):

- Find credible recommender: users do not automatically trust a recommender, the social network is a way for recommender to appear more trustworthy.
- Improve profile: by contributing manually to the system (i.e., in collaborative filtering recommendation, by rating items), users believe that they are improving their profile and thus improving the quality of the recommendations that they will receive.
- Help others: some users are happy to express themselves to the social network using the recommender system and are pleased to know that the community will benefit from their contribution. Users of recommender systems can also influence others, by listening a particular music, viewing a specific film, or purchasing particular items.

In conventional commercial recommender systems, the most important function is to increase the number of items sold, whatever these items are. Another perspective is to focus on users rather

than on the items, so recommender systems can be seen as useful tools for improving the behavior of OSNSs. The RSs provide suggestions to online users to make better choices, that is why personalized recommendations – by taking into account the maximal relevant content available – are sought by users. For example, in his practical book, Russell (2013) presents some technical skills for being able to use APIs for mining the data from some of the most popular social websites. We will illustrate some key applications of recommender systems to OSNSs through three popular social networks:

- In *LinkedIn*, you can ask the connections of your personal network/social network to write a recommendation of your work that you can display on your profile. Moreover your connections can recommend some of your skills. This information can be used by RSs for providing suggestions of connections (e.g., a colleague, a coworker, a business partner) to add to your network, for presenting new job opportunities (matching with your skills). With recommendations and data mining techniques, it is possible to cluster your colleagues according to a similarity measurement and identify, e.g., which of your connections are the most similar based upon a criterion like job title, or which of your connections have worked in companies you want to work for.
- In *Facebook*, the nodes of the network are people or organizations, and they are connected as a friendship mode. You can share information (e.g., posts and photos) on *Facebook* by selecting an audience (a public information or to the friends only). In *Facebook*, the "people you may know" feature recommends (for extending your "friendship" network) a personalized list of people who are similar to you based on your friends, or friends of friends, but also with other criteria like your geographical location, groups, or liked pages. By using *Facebook*'s API, it is possible to mine the social network for analyzing the likes, the fan pages, and the clique of friends, and represent the resulting graphs with some specific visualization libraries.
- *Twitter* is a directed graph dedicated to the spread of ideas through the broadcasting of small messages (i.e., tweets). You can follow the messages of some people or organizations of interest, and you can be followed by people or institutions (the followers). In *Twitter*, you can be recommended to follow some users. With data mining and natural language processing techniques, it is possible to mine *Twitter* text data and social graph for exploring the trending topics, discovering what people are talking about, find new people to follow that it would be appropriate to forward message to your followers (i.e., retweet).

Future Directions

Over the last two decades, some major advances have been achieved in the area of RSs using techniques of social network analysis and mining.

In this section we present some current challenges and open questions, which we think, will be a major preoccupation for scientific communities, but also the industry in the upcoming years. We will consider two practical future directions and list the corresponding open challenges that need to be considered.

Recommender Systems in the Enterprise

Nowadays, more and more companies show increasing interest towards the integration of RSs in the Intranet in order to further improve communications and internal knowledge management. Several reasons push companies to invest in such infrastructures:

- It can improve social interactions between employees (e.g., a people recommender in the enterprise may help in finding the best expert for a specific problem (Joly et al. 2010), which may reduce costs and increase efficiency).
- It can provide new means for the dynamic composition of teams for a specific project, as the expertise of employees can be easily retrieved. Also, internal documents, videos can be recommended for a project or learning.

- Such a system may provide specific tools for employees in order to keep motivation and a good atmosphere in the company, e.g., associating specific tags to colleagues, such as expertise tags and specific badges, when being an active contributor in providing help to colleagues or other scenarios.
- With such a system, an implicit internal social network can be built that links employees with similar interests and activities. This can help the company in improving its organization and also optimize human resources management (changing dynamically teams, etc.).

The deployment of an RS in a company faces several challenges, and its design depends on several criteria, such as the type of activity the company performs or the degree of sensibility of the information they share. A first challenge, but also the most important, is what kind of internal content to use as input for the RS. Company e-mails are a rich source for learning more about each employee's expertise and interests, but they may have privacy and security concerns. Another, more acceptable source for such an RS may be represented by content shared by employees on internal or web-based social networks, such as *Twitter* (Stan et al. 2011) or *Yammer*. Such content is shorter and generally does not contain confidential information. Furthermore, the content of web pages employees read may also represent an additional source for such systems for the construction of the expertise profiles (Joly et al. 2010).

Among the challenges for building such a system, the most important are technical and related to human-computer interaction (HCI). More concretely, technical challenges include the implementation of content extraction tools from internal mail servers, microblogging platforms, and web browsers. All the extracted content must be aggregated and stored in a secure database. Challenges related to HCI include the design of user interfaces that allow users to control what content to share with the system (e.g., there may be e-mails for private usage).

An important issue when designing RSs is to generate an explanation for each recommendation. Such explanations could be useful as they increase trust. They can be of several types: (i) the explanation of the social path between the two users, i.e., by showing part of the social graph and the paths in the employee social network that connect them or (ii) a semantic explanation that includes areas of expertise of the recommended employee. According to the social distance, such areas of expertise may be shown with different levels of granularity, by using hierarchical paths of concepts in semantic knowledge bases, such as *DBpedia* (e.g., expert in *Twitter* is more specific than expert in microblogging platforms).

In a nutshell, the following questions should be considered for building a successful RS specifically targeted to an enterprise:

- How to extract the named entities from short, unstructured messages, status updates? In other words, how to transform each social interaction that occurs in the company or that employees share into useful knowledge for the RS.
- How to combine structural and semantic analysis for recommendation ranking.
- What are the next generation privacy protection mechanisms that would allow an easy adoption of such a system in a company.
- How to generate useful and meaningful explanations for a recommendation.
- How to make good recommendations without violating privacy concerns.

The use of such RSs in an enterprise may be useful also for generating a profile for the entire company, e.g., by aggregating all individual user profiles. Such a profile may be useful for the next-generation enterprise social networks, where each node in the network is a company. Such a network could facilitate collaboration between companies, e.g., by finding the best company for a collaborative European project.

Recommendation in Mobile Social Networks: A Multiagent Approach

A second scenario for RSs concerns mobility and ubiquity, as more and more users have

smartphones, capable of sensing context. The most widely used context in such a scenario is the user location, which may significantly improve recommendation (other context data may include available networks (Wifi, Bluetooth) or other physical data). By integrating location in an RS, a preliminary filtering of items can be performed, by selecting only a subset that is in a well-defined perimeter. Such items may include other users with similar interests (e.g., looking for people who like similar artists in a given location), as well as restaurants, cinema, or other services the city provides. The deployment of such an RS faces several challenges, depending on its design. A first important design principle which needs to be fixed early is whether the system is centralized or decentralized. Clearly, a centralized system would face important performance and scalability issues. A decentralized system is more interesting, as a local server can be associated to each location in the city, which could support this recommendation service.

A further step towards decentralization can be considered, by integrating multiagent principles to the RS, i.e., to design and implement a customizable approach where different autonomous decision-making entities (agents) have to communicate, exchange knowledge and cooperate in order to achieve individual and/or collective objectives. It allows the creation of different communities, with different possible functions and modes of exchanges. Such an approach aims to meet several challenges, such as decentralization of the community management, personalized automatic management and discovery of communities, and flexibility so that any agent can create its own community. In addition, it should cover all levels of abstractions (agent, environment, and organization) that are required for the development of sophisticated multiagent system. In this design, each smartphone is equipped with an agent, capable of exchanging knowledge with other agents, using the local server associated to a given location in the city.

Using a multiagent approach for an RS in mobility, agents can act as a personal assistant on the behalf of each user, present in a given location. The agent perceives knowledge from the communities of individual interests and acts upon the communities to meet their goals. Thus, agents can bring the appropriate people having common goals or interests together share their knowledge with each other at ease.

Other scientific challenges for such advanced RSs include the traditional coldstart problem, i.e., how to provide recommendations to users with little information about their profile, or how to recommend items with few ratings. Also, an important general challenge is how to make recommendation users trust, i.e., how to provide users an easy way for giving feedback on recommendations. With regards to trust, recent works try to integrate the notion of distrust, i.e., how to deal with users or items that cannot be trusted.

Cross-References

- ▶ Actionable Information in Social Networks, Diffusion of
- ▶ Collective Intelligence: Overview
- ▶ Combining Link and Content for Community Detection
- ▶ Computational Trust Models
- ▶ Emotions and Personality in Recommender Systems
- ▶ Friends Recommendations in Dynamic Social Networks
- ▶ Group Representation and Profiling
- ▶ Incentives in Collaborative Applications
- ▶ Recommender Systems Based on Linked Open Data
- ▶ Recommender Systems Based on Social Networks
- ▶ Recommender Systems Evaluation
- ▶ Recommender Systems, Basics of
- ▶ Recommender Systems, Semantic-Based
- ▶ Recommender Systems: Models and Techniques
- ▶ Social Recommendation in Dynamic Networks
- ▶ Spatiotemporal Personalized Recommendation of Social Media Content
- ▶ Trust in Social Networks

References

Adomavicius G, Tuzhilin A (2005) Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. IEEE Trans Knowl Data Eng 17(6):734–749. https://doi.org/10.1109/TKDE.2005.99

Anagnostopoulos A, Kumar R, Mahdian M (2008) Influence and correlation in social networks. In: Li Y, Liu B, Sarawagi S (eds) Proceedings of the 14th ACM SIGKDD international conference on knowledge discovery and data mining, 24–27 Aug 2008. ACM, Las Vegas, pp 7–15. https://doi.org/10.1145/1401890.1401897

Arias JJP, Vilas AF, Redondo RPD (eds) (2012) Recommender systems for the social web, Intelligent systems reference library, vol 32. Springer, Berlin. https://doi.org/10.1007/978-3-642-25694-3

Benyahia O, Largeron C (2015) Centrality for graphs with numerical attributes. In: Pei J, Silvestri F, Tang J (eds) Proceedings of the 2015 IEEE/ACM international conference on advances in social networks analysis and mining, ASONAM 2015, Paris, 25–28 Aug 2015, pp 1348–1353. https://doi.org/10.1145/2808797.2808844

Benyahia O, Largeron C, Jeudy B, Zaiane OR (2016) Dancer: dynamic attributed network with community structure generator. In: Machine learning and knowledge discovery in databases European conference, ECML PKDD 2016, Riva del Garda, 19–23 Sept 2016. Proceedings. Lecture notes in computer science. Springer. http://perso.univ-st-etienne. fr/largeron/DANC_Generator/

Bonchi F, Morales GDF, Riondato M (2016) Centrality measures on big graphs: Exact, approximated, and distributed algorithms. In: Bourdeau J, Hendler J, Nkambou R, Horrocks I, Zhao BY (eds) Proceedings of the 25th international conference on world wide web, WWW 2016, Montreal, 11–15 Apr 2016, Companion Volume, pp 1017–1020. https://doi.org/10.1145/2872518.2891063

Cabacas R, Wang Y, Ra I (2014) Qualitative assessment of social network-based recommender systems based on essential properties. In: Kim YS, Ryoo YJ, Jang M, Bae Y (eds) Advanced intelligent systems, Advances in intelligent systems and computing, vol 268. Springer, Berlin, pp 1–11. https://doi.org/10.1007/978-3-319-05500-81

Can F, Ozyer T, Polat F (eds) (2014) State of the art applications of social network analysis. Lecture notes in social networks, Springer, Cham/Heidelberg, https://doi.org/10.1007/978-3-319-05912-9

Colace F, Santo MD, Greco L, Moscato V, Picariello A (2015) A collaborative user-centered framework for recommending items in online social networks. Comput Hum Behav 51:694–704. https://doi.org/10.1016/j.chb.2014.12.011

Combe D, Largeron C, G´ery M, Egyed-Zsigmond E (2015) I-louvain: an attributed graph clustering method. In: Advances in intelligent data analysis XIV 14th international symposium, IDA 2015, Saint Etienne, 22–24 Oct 2015, Proceedings, pp 181–192. https://doi.org/10.1007/978-3-319-24465-516

Ekstrand MD, Riedl J, Konstan JA (2011) Collaborative filtering recommender systems. Found and Trends in Hum-Comput Interact 4(2):175–243. https://doi.org/10.1561/1100000009

Getoor L (2010) Link mining and link discovery. In: Encyclopedia of machine learning. Springer, Berlin, pp 606–609. https://doi.org/10.1007/978-0-387-30164-8_480

Guy I, Ronen I, Wilcox E (2009) Do you know?: recommending people to invite into your social network. In: Conati C, Bauer M, Oliver N, Weld DS (eds) Proceedings of the 2009 international conference on intelligent user interfaces (IUI), 8–11 Feb 2009. ACM, Sanibel Island, pp 77–86. https://doi.org/10.1145/1958824.1958867

Hannon J, Bennett M, Smyth B (2010) Recommending twitter users to follow using content and collaborative filtering approaches. In: Amatriain X, Torrens M, Resnick P, Zanker M (eds) Proceedings of the 2010 ACM conference on recommender systems, RecSys 2010, 26–30 Sept 2010. ACM, Barcelona, pp 199–206. https://doi.org/10.1145/1864708.1864746

Hasan MA, Zaki MJ (2011) A survey of link prediction in social networks. In: Aggarwal CC (ed) Social network data analytics. Springer, New York, pp 243–275. https://doi.org/10.1007/978-1-4419-8462-3_9

Herlocker JL, Konstan JA, Riedl J (2000) Explaining collaborative filtering recommendations. In: Proceedings of the 2000 ACM conference on computer supported cooperative work, CSCW'00, 2–6 Dec 2000. ACM, Philadelphia/New York, pp 241–250. https://doi.org/10.1145/358916.358995

Herlocker JL, Konstan JA, Terveen LG, Riedl J (2004) Evaluating collaborative filtering recommender systems. ACM Trans Inf Syst 22(1):5–53. https://doi.org/10.1145/963770.963772

Horowitz D, Kamvar SD (2010) The anatomy of a large-scale social search engine. In: Rappa M, Jones P, Freire J, Chakrabarti S (eds) Proceedings of the 19th international conference on world wide web, WWW 2010, 26–30 Apr 2010. ACM, Raleigh, pp 431–440. https://doi.org/10.1145/1772690.1772735

Joly A, Maret P, Daigremont J (2010) Contextual recommendation of social updates, a tag-based framework. In: An A, Lingras P, Petty S, Huang R (eds) Proceedigs of the 6th international conference on active media technology, AMT 2010, 28–30 Aug 2010, Lecture notes in computer science, vol 6335. Springer, Toronto, pp 436–447. https://doi.org/10.1007/978-3-642-15470-6_45

Kempe D, Kleinberg JM, Tardos E´ (2003) Maximizing the spread of influence through a social network. In: Getoor L, Senator TE, Domingos P, Faloutsos C (eds) Proceedings of the 9th ACM SIGKDD international

conference on knowledge discovery and data mining, 24–27 Aug 2003. ACM, Washington, pp 137–146. https://doi.org/10.1145/956750.956769
Lancichinetti A, Fortunato S (2009) Community detection algorithms: a comparative analysis. Phys Rev E 80(5):056,117. https://doi.org/10.1103/PhysRevE.80.056117
Li B, King I (2010) Routing questions to appropriate answerers in community question answering services. In: Huang J, Koudas N, Jones GJF, Wu X, Collins-Thompson K, An A (eds) Proceedings of the 19th ACM conference on information and knowledge management, CIKM 2010, 26–30 Oct 2010. ACM, Toronto, pp 1585–1588. https://doi.org/10.1145/1871437.1871678
Liben-Nowell D, Kleinberg J (2007) The link-prediction problem for social networks. J Am Soc Info Sci Technol 58(7):1019–1031. https://doi.org/10.1002/asi.20591
Lin CY, Cao N, Liu S, Papadimitriou S, Sun J, Yan X (2009) SmallBlue: social network analysis for expertise search and collective intelligence. In: Ioannidis YE, Lee DL, Ng RT (eds) Proceedings of the 25th international conference on data engineering, ICDE 2009, 29 Mar 2009 2 April 2009. IEEE, Shanghai, pp 1483–1486. https://doi.org/10.1109/ICDE.2009.140
Melville P, Sindhwani V (2010) Recommender systems. In: Sammut C, Webb GI (eds) Encyclopedia of machine learning. Springer, New York/London, pp 829–838. https://doi.org/10.1007/978-0-387-30164-8_705
Namata G, Getoor L (2010) Link prediction. In: Encyclopedia of machine learning. Springer, New York/London, pp 609–612. https://doi.org/10.1007/978-0-387-30164-8_481
Resnick P, Iacovou N, Suchak M, Bergstrom P, Riedl J (1994) Grouplens: an open architecture for collaborative filtering of netnews. In: Smith JB, Smith FD, Malone TW (eds) CSCW '94, Proceedings of the conference on computer supported cooperative work, 22–26 Oct 1994. ACM, Chapel Hill, pp 175–186. https://doi.org/10.1145/192844.192905
Ricci F, Rokach L, Shapira B (eds) (2015) Recommender systems handbook, 2nd edn. Springer, New York. https://doi.org/10.1007/978-1-4899-7637-6
Russell MA (2013) Mining the social web, data mining Facebook, Twitter, LinkedIn, Google+, GitHub, and more, 2nd edn. O'Reilly Media, Sebastopol
Schaeffer SE (2007) Graph clustering. Comput Sci Rev 1 (1):27–64. https://doi.org/10.1016/j.cosrev.2007.05.001
Schall D (2015) Social network-based recommender systems. New York: Springer. https://doi.org/10.1007/978-3-319-22735-1
Solé-Ribalta A, Domenico MD, Gómez S, Arenas A (2016) Random walk centrality in interconnected multilayer networks. Phys D: Nonlinear Phenom 323–324:73–79. https://doi.org/10.1016/j.physd.2016.01.002
Stan J, Do VH, Maret P (2011) Semantic user interaction profiles for better people recommendation. In: International conference on advances in social networks analysis and mining, ASONAM 2011, 25–27 July 2011. IEEE Computer Society, Kaohsiung, pp 434–437. https://doi.org/10.1109/ASONAM.2011.21
Terveen LG, Hill WC, Amento B, McDonald DW, Creter J (1997) PHOAKS: a system for sharing recommendations. Commun ACM 40(3):59–62. https://doi.org/10.1145/245108.245122
Victor P, Cornelis C, De Cock M (2011a) Trust networks for recommender systems, Atlantis computational intelligence systems, vol 4. Atlantis Press, Amsterdam/Paris. https://doi.org/10.2991/978-94-91216-08-4
Victor P, De Cock M, Cornelis C (2011b) Trust and recommendations. In: Ricci F, Rokach L, Shapira B, Kantor PB (eds) Recommender systems handbook. Springer, New York, pp 645–675. https://doi.org/10.1007/978-0-387-85820-3_20
Wasserman S, Faust K (1994) Social network analysis: methods and applications, Structural analysis in the social sciences, vol 8, 4th edn. Cambridge University Press, New York
Yang X, Guo Y, Liu Y, Steck H (2014) A survey of collaborative filtering based social recommender systems. Comput Commun 41:1–10. https://doi.org/10.1016/j.comcom.2013.06.009

Recommender Systems, Basics of

Marco de Gemmis, Pasquale Lops and Marco Polignano
Department of Computer Science, University of Bari Aldo Moro, Bari, Italy

Synonyms

Information filtering systems; Recommendation systems

Glossary

Information filtering	Information seeking process that typically refers to selection of relevant information, or rejection of irrelevant information, from a stream of incoming data

Information overload	Cognition problem that refers to the difficulty a person can have making decisions, caused by the presence of too much information
Item	Object or information that can be provided by an information source
Personalization system	System that delivers items to user according to their preferences
Rating	Any relevance feedback provided by users on items
Recommendation list	A ranked list of suggested items provided by a recommender system
User profile	A structured representation of user interests

Definition

Recommender Systems (RSs) are filtering tools that guide the user in a personalized way to interesting or useful objects (items) in a large space of possible options (Burke 2002). They collect information about user preferences, either explicitly by asking users to provide ratings on items or implicitly by analyzing their actions on items (download, print, view). The collected data are then exploited to build a user profile (a model of user preferences) or to discover users having similar interests, with the aim of finding novel items that might be interesting to them. Nowadays, those systems can be seen as decision support tools, because they help people to make better choices in their everyday life: what products to buy, what documents to read, which people to have in their social networks.

Introduction

The Web has become the main provider of information. People use the Web for retrieving information instead of conventional sources like books, magazines, encyclopedias, and libraries. Leading commercial search engines created by companies such as Google, Microsoft, and Yahoo! are used by millions of people to search for information that helps them in their work and day-to-day life. This is the Information Society, characterized by very huge knowledge repositories, as well as a rapid increase of information. This information explosion has led to a situation where users felt inundated with information and have difficulty in sifting through the reams of material, much of which is not relevant to them.

This scenario is commonly referred to as the problem of *information overload* (Maes 1994), originated by the gap between the overwhelming amount of information and human limitations.

Finding information relevant to a user's interests is a challenging task. The first obstacle is research, where you must first identify the appropriate information sources and then retrieve the relevant data. Then, you have to sort through this data to filter out the unfocused and unimportant information. Lastly, in order for the information to be truly useful, you must take the time to figure out how to organize and abstract it in a manner that is easy to understand and analyze. To say the least, all of these steps are extremely time consuming. This *relevant information problem* leads to a clear demand for automated methods able to support users in searching large repositories in order to retrieve relevant information with respect to their preferences.

The problem of filtering information in a personalized way consists in designing systems which are able to provide information that matches user needs in a consistent and timely manner. Information Filtering systems, which pursue this goal by providing personalized suggestions, adapt their behavior to individual users by learning their preferences during user interaction in order to construct a profile of the user that can be later exploited to select relevant items (Hanani et al. 2001).

RSs represent the main area where principles and techniques of Information Filtering are applied (Ricci et al. 2015; 2011). They have the effect of guiding users in a personalized way to interesting or useful objects in a large space of possible options, by providing a list of suggested

items that fits their interests (Burke 2002). For example, Netflix, a provider of on-demand Internet streaming video and flat rate DVD-by-mail in the United States, adopts a recommendation algorithm to predict user interests for films, based on feedback provided by users on previously watched items.

Key Points

According to Adomavicius and Tuzhilin (2005), an RS can be seen as a function that predicts a relevance score *r* for each (*user, item*) pair in the system. The relevance score is basically an estimation of the user rating or, more generally, of the user satisfaction with that item. It can be used to produce a ranked list of recommendations for the target user. In order to accomplish this task, RSs have to:

- collect user preferences on some items,
- use the collected data either to infer a *general* model of user interests or to group users having similar tastes, and
- estimate the user rating for unseen (unrated) items, based on the inferred model or groups.

User preferences can be acquired implicitly or explicitly. Implicit ratings are typically collected by monitoring the user's browsing behavior. If the user put an item in her shopping cart or download its web page, an RS could *interpret* this behavior as a positive orientation toward the item and assign it with a positive score. The advantage of this strategy is that it does not require any cognitive effort from the side of the user, but the collected ratings can be noisy, because one cannot be sure whether the user actions are correctly interpreted. An item in the user's shopping cart does not mean necessarily that she will like that item, or perhaps she might have bought that item for someone else.

Explicit ratings are usually collected on a five-point Likert response scale, ranging from "Strongly dislike" to "Strongly like." The advantage of this strategy is that knowledge about user preferences is probably more precise than that implicitly acquired. On the other hand, the problem with this strategy is that not all the ratings are equally informative of the user preferences; therefore, the system has to further explore user preferences, by asking to rate additional and particular items, so that recommendation accuracy can be improved in the future. The explicit elicitation of preferences on selected items exposes users to a cognitive effort. As a consequence, the need to balance the trade-off between cost of rating elicitation and effectiveness of the system has originated strategies whose goal is to narrow the size of the item set shown to the user, as well as to maximize the "informativeness" of the proposed items. Augmenting RSs with Active Learning strategies is one of the most promising directions to address this issue (Rubens et al. 2015). In general, AL methods are used to identify which ratings for items should be obtained by the system to have good training data for improving the accuracy of the learned user model.

Ratings provided by the user are then given to an algorithm that computes rating predictions on unrated items. The techniques adopted to estimate this relevance score for candidate suggestions fall in two main areas, namely, content-based filtering and collaborative filtering, described in section "Main Recommendation Techniques."

Historical Background

Research in the area of RSs has its roots in several communities such as Information Retrieval, Information Filtering, Machine Learning, and User Modeling. Given the long history of research in these areas, it is not possible to provide a comprehensive overview of the historical evolution of the field. Indeed, we focus on Information Filtering (IF), which gave the main contribution.

IF is an information seeking process that typically refers to selection of relevant information, or rejection of irrelevant information, from a stream of incoming data. IF systems are designed to support access to large volumes of information generated by *highly dynamic* information sources, such as news portals or product catalogs, focusing on *long-term and stable information need* of

users, also called *interests*. Based on the user profile, which is learned from the user's previously expressed opinions on example items, a filtering system processes a new item and takes appropriate actions that either ignore it or bring it to the user attention. Feedback on items is usually given in the form of rating on a discrete scale.

IF has become very attractive since 1990s, motivated by the demand of personalized on-line information services, like news filtering or movie recommendation. Early systems were Tapestry (Goldberg et al. 1992), that introduced the idea (and terminology) of collaborative filtering, the GroupLens system (Resnick et al. 1994), the Ringo music RS at Massachusetts Institute of Technology (MIT) (Shardanand and Maes 1995), the Bellcore Video Recommender (Hill et al. 1995), and the FindMe system (Burke et al. 1996).

In 1997, a special issue of *Communication of the ACM* on Recommender Systems (Resnick and Varian 1997) actually ratified the birth of RSs as an independent research area. In the following years, many approaches were developed with the contribution by people coming from several disciplines: User Modeling, Data Mining, Artificial Intelligence, and Information Retrieval. New algorithms were developed and compared, thanks to the availability of large datasets, such as MovieLens (Harper and Konstan 2015). The main goal was to improve predictive accuracy, as witnessed by the Netflix Prize, an open competition for the best collaborative filtering algorithm to predict user ratings for movies. In 2009, the grand prize of US $ 1,000,000 was given to the BellKor's Pragmatic Chaos team, which outperformed Netflix's own algorithm for predicting ratings by 10.06.

In 2007, Joseph Konstan (University of Minnesota) organized the first ACM Conference on Recommender Systems, which nowadays is the premier international forum for the presentation of new research results in this area.

Main Recommendation Techniques

In the following sections, we describe the two main paradigms that RSs adopt to estimate the user rating for an item.

Content-Based Filtering

Content-based RSs try to recommend items similar to those a given user has liked in the past. Items are represented by a set of features, also called *attributes* or *properties*. For example, features adopted to describe movies might be: title, director, cast, genre, and plot summary. Systems adopting this filtering paradigm process both item descriptions and corresponding ratings provided by a user and then build user profile, which is a structured representation of her interests, adopted to recommend items she had not yet considered. The recommendation process consists of matching up the attributes of the user profile against the attributes of items. The result is a relevance judgment that represents the user's level of interest in that item (Lops et al. 2011).

Collaborative Filtering

Systems designed according to the collaborative recommendation paradigm identify users whose ratings are similar to those of the target user and recommend items they have liked (Ekstrand et al. 2011). Collaborative RSs differ from content-based ones in that only user opinions are used, without any content processing. There are two main techniques for collaborative filtering (CF): User-User CF and Item-Item CF.

To provide recommendations to the target user X, a User-User CF system firstly computes the neighborhood of that user, i.e., the subset of users that show tastes similar to X. Similarity in taste is measured by computing the closeness of ratings for items that were rated by both users. Then, the system recommends items that users in X's neighborhood indicated to like, provided that they have not yet been rated by X (Konstan et al. 1997).

Item-Item CF, also called item-based CF is one of the most widely deployed CF techniques today, because it overcomes the scalability problems that affect User-User as the user base grows. This technique computes similarities between the rating patterns of items, rather than between users. Given a target item (for which the system must compute a relevance score for the target user), Item-item CF finds the most similar items, i.e.,

items that tend to have the same ratings from user (Linden et al. 2003; Sarwar et al. 2001).

Many researchers have tried to combine different techniques in different ways in order to obtain *hybrid RSs*.

Hybrid Techniques

Each type of filtering paradigm has its own weaknesses and strengths. For instance, the *new user problem* is common to both filtering methods. The user has to rate a sufficient number of items before the system can really understand her preferences and provide reliable recommendations. Therefore, a new user, having very few ratings, would not be able to get accurate recommendations. *Over-specialization* is a typical problem of content-based RSs. Since those systems recommend items that score highly against a user's profile, the user is limited to being recommended items similar to those already rated. The main advantage of CF over content-based methods is that any items regardless of content can be recommended. Movies, images, art, and text items are all represented by the users' opinions and thus can be recommended by the same system. This also means that CF can recommend things from different genres. On the other hand, collaborative systems require a lot of computational resources with the increasing of the number of users and items, and they act as *black boxes*, computerized oracles which give advice but cannot be questioned. A user is given no indicators to consult in order to decide when to trust a recommendation and when to doubt one (lack of transparency problem).

In an attempt to compensate the drawbacks of each single approach, hybrid systems were proposed (Balabanović and Shoham 1997; Burke 2007; Cremonesi et al. 2011; Degemmis et al. 2007). Robin Burke proposed a very analytical classification of hybridization strategies (Burke 2002):

- **Weighted.** In weighted hybrid RSs, the relevance score of a recommended item is computed from the results of all of the available recommendation techniques available in the system. This means that the scores of several recommendation techniques are combined together to produce a single recommendation. The simplest combined hybrid would be a linear combination of recommendation scores.
- **Switching.** A switching hybrid uses some criterion to switch between recommendation techniques. Switching hybrids introduce additional complexity into the recommendation process since the switching criteria must be determined, and this introduces another level of parameterization.
- **Mixed.** Recommendations from several different recommendation techniques are presented at the same time. This may be possible where it is practical to compute a large number of recommendations simultaneously. Where conflicts occur, some type of arbitration between methods is required.
- **Feature Combination.** Features from different recommendation sources are thrown together into a single recommendation algorithm. For instance, content and collaborative techniques might be merged, treating collaborative information as simply additional feature data associated with each example and using content-based techniques over this augmented data set.
- **Cascade.** The cascade hybrid involves a staged process because one recommender refines the suggestions given by another. This means that, one recommendation technique is employed first to produce a coarse ranking of candidates, while another technique refines the recommendations from among the candidate set.
- **Feature Augmentation.** Output from one technique is used as an input feature to another. This means that one technique is employed to produce a rating or classification of an item, and then that information is incorporated into the processing of the next recommendation technique. Augmentation is attractive, because it offers a way to improve the performance of a core system without modifying it.
- **Metalevel.** The model learned by one recommendation technique is used as input to another. This differs from feature augmentation: in an augmentation hybrid, a learned model is used to generate features for input to a second algorithm; in a metalevel hybrid, the entire model becomes the input.

Evaluation

RSs are mainly evaluated using offline experiments that estimate the prediction error of recommendations using benchmark datasets of user ratings, such as Movie-Lens (Harper and Konstan 2015). User studies (small group of users experiment with the system and report on the experience) and online experiments (where real user populations interact with the system) are also used. RSs are usually evaluated on their ability to accurately predict the user's rating for a target item. A popular accuracy metric is MAE (mean absolute error), the average deviation between the computed recommendation score for a *(user, item)* pair and the actual rating given by the user to that item. Some authors compute the root mean square error (RMSE) to put more emphasis on larger deviations (Sarwar et al. 2001).

Another approach to evaluation is adopted whether the recommendation task is seen as a ranking or classification problem: the system has to identify the *n* most relevant items for the target user. Precision and Recall, commonly adopted to evaluate retrieval systems, are used to measure the accuracy of the ranked list of recommended items given to the user (Baeza-Yates and Ribeiro-Neto 1999). According to some authors (McLaughlin and Herlocker 2004), those metrics reflect the real user experience better than MAE does, because in most cases, users actually receive a recommendation list instead of predictions for ratings of specific items.

However, it is now widely agreed that accurate predictions are not enough to provide the user with *useful* items (Ge et al. 2010). Authors explored other evaluation dimensions, such as *novelty*, *diversity,* and *unexpectedness*.

The *novelty* of a piece of information generally refers to how different it is with respect to "what has been previously seen" by a user or a community. Novelty occurs when an RS suggests to the active user an unknown item that she might have autonomously discovered (Herlocker et al. 2004; Vargas and Castells 2011).

Diversity represents the variety present in a list of recommendations (Adomavicius and Kwon 2012; Fleder and Hosanagar 2009; Zhang et al. 2012). Methods for the diversification of suggestions are generally used to avoid homogeneous lists, in which all the items suggested are very similar to each other. This may reduce the overall quality of the recommendation list because none of the alternative suggestions will be liked, in case the user wants something different from the usual.

While computing relevance is a well-established issue, there is no general consensus on the method to assess *unexpectedness*. Usually, in offline experiments on benchmark datasets, unexpectedness is measured as the deviation from a *standard prediction criterion* which is more likely to produce expected recommendations, as suggested by Murakami et al. (Murakami et al. 2008). For example, if the standard prediction criterion is based on a nonpersonalized recommendation algorithm based on popularity, the most popular items will be the most expected recommendations, while the items in the long tail will be the most unexpected ones.

More details about metrics and experimental protocols can be found in (Gunawardana and Shani 2011; Jannach et al. 2011).

Recently, a key issue is reproducibility of experiments, because it is difficult to compare results that come from different options in design and implementation of evaluation strategies. In Said and Bellog´ın (2014), the authors compare common recommendation algorithms as implemented in three popular recommendation frameworks, in a fair evaluation environment (same dataset, data splitting, evaluation strategies, and metrics). The authors observed that the same baselines may perform orders of magnitude better or worse across frameworks, thus pointing out the necessity of clear guidelines when reporting evaluation of RSs to ensure reproducibility and fair comparison of results. More details about evaluation issues can be found in Gunawardana and Shani (2011) and Said and Bellog´ın (2014) and in a recent special issue of *ACM Transactions on Intelligent Systems and Technology* (Cremonesi et al. 2016).

Key Applications

Research in RSs has many practical applications. Here we discuss emerging applications or new

trends in traditional domains, usually aiming at improving commercial systems.

IP television (IPTV) services provide users with multimedia content (movies, news programs, documentaries, TV series) via broadband Internet networks. In Bambini et al. (2011), the authors describe the adoption of both content and collaborative recommendation methods in the production environment of the one of the largest European IPTV providers. Differently from traditional e-commerce RSs, the IPTV domain has some special requirements, such as very strict time constraints or high dynamism of the catalog. The research showed that the integration of the RS had a positive effect on customers, with an increase of the number of watched movies.

News recommendation is a domain where content-based methods find their natural application. MESH (Multimedia sEmantic Syndication for enHanced news Services) is a research project that designed a framework for intelligent creation and delivery of semantically-enhanced multimedia news information (Picault et al. 2011). MESH adopted a hybrid content-collaborative recommendation method exploiting news metadata expressed through ontological concepts. CF methods are also adopted in this domain. For instance, in Liu et al. (2010), the authors describe a personalized news recommendation system based on profiles learned from user activity in Google News. Log analysis is used to infer user interests based on click behavior on the news website, without using any form of explicit rating. The method for predicting user's interests was combined with a CF method to generate personalized news recommendations.

Recommendation of financial investment strategies is a complex and knowledge-intensive task where RSs are recently used. Typically, financial advisors have to discuss at length with their wealthy clients and have to sift through several investment proposals before finding one able to completely meet investors' needs and constraints. As a consequence, a recent trend in wealth management is to improve the advisory process by exploiting recommendation technologies. In Musto et al. (2015), the authors propose a framework for recommendation of asset allocation strategies which combines case-based reasoning with a novel diversification strategy to support financial advisors in the task of proposing diverse and personalized investment portfolios.

RSs for e-commerce are recently focusing on the problem of providing nonobvious recommendations, in order to suggest items in the long tail and to diversify the recommendation list. For instance, with the *Discover* project, eBay is proposing products the user will be interested in and that are at the same time a complete surprise to her, by avoiding very popular items (Woyke 2011). In order to identify those items, the algorithm implemented in *Discover* takes into account several factors, such as how much people interact with a listing (clicking or returning to it, forwarding it to friends), or the textual analysis (a longer description of the product indicates more passion about the item). Similarly, Amazon tries to lead the user toward unexpected discoveries by using *Statistically Improbable Phrases (SIPs)*, the most distinctive phrases in the text of books, to suggest new items, which are not strictly related to those the user already knows. The ACM Conference on Recommender Systems regularly hosts industry sessions where major companies using personalization technologies describe research results and open issues in their respective domains. In 2015 and 2016, to cite the most recent editions of the conference, industry talks were given by Netflix, Pandora, Expedia, Bloomberg, Spotify, Foursquare, IBM Research, LinkedIn, Booking.com, Yahoo! Labs, among others.

Future Directions

In this section, we describe some of the several research lines in the RSs community.

Semantics-Aware Recommender Systems

The ever increasing interest in *semantic technologies* and the availability of several open knowledge sources, such as Wikipedia, DBpedia, Freebase, and BabelNet have fueled recent progress in the field of content-based RSs. Novel research works have introduced semantic techniques that shift from a *keyword*-based to a

concept-based representation of items and user profiles. These observations make very relevant the integration of proper techniques for deep content analytics borrowed from Natural Language Processing (NLP) and Semantic Technologies, which is one of the most innovative lines of research in *semantic RSs* (de Gemmis et al. 2015a; Musto et al. 2016a).

We roughly classify semantic techniques into *top-down* and *bottom-up* approaches. Top-down approaches rely on the integration of external knowledge, such as machine readable dictionaries, taxonomies (or IS-A hierarchies), thesauri or ontologies (with or without value restrictions and logical constraints), for annotating items and representing user profiles in order to capture the semantics of the target user information needs. The main motivation behind top-down approaches is the challenge of providing RSs with the linguistic knowledge and common sense knowledge, as well as the cultural background which characterize the human ability of interpreting documents expressed in natural language and reasoning on their meaning.

On the other side, bottom-up approaches exploit the so-called geometric metaphor of meaning to represent complex syntagmatic and paradigmatic relations between words in high-dimensional vector spaces. According to this metaphor, each word (and each document as well) can be represented as a point in a vector space. The peculiarity of these models is that the representation is learned by analyzing the context in which the word is used, in a way that terms (or documents) similar to each other are close in the space. For this reason, bottom-up approaches are also called distributional models. One of the great virtues of these approaches is that they are able to induce the semantics of terms by analyzing their use in large corpora of textual documents using unsupervised mechanisms, as evidenced by the recent advances of machine translation techniques (Halevy et al. 2009; Mikolov et al. 2013).

In de Gemmis et al. (2015a), the authors describe a variety of semantic approaches, both top-down and bottom-up, and show how to leverage them to build a new generation of semantic content-based RSs, that they call *semantics-aware content-based RSs*. Many recommendation scenarios may benefit from semantic-based approaches. For instance, in the context of sentiment analysis, concept-based approaches proved to be superior to purely syntactical techniques (Cambria et al. 2013), hence RSs which rely on the analysis of opinions written in natural language for extracting user preferences and affective states might effectively adopt semantic-based approaches to provide better suggestions. In the next future, we foresee some major challenges in adopting semantic technology to improve RSs:

- Definition of recommendation methods able to reason on the graph structure of the Linked Open Data cloud to discover latent connections among items and user profiles (Musto et al. 2016b). Those emerging relations could be exploited for cross-domain recommendations or diversification of suggestions. As an example, the Linked Open Data-enabled Recommender Systems Challenge of the 11th European Semantic Web Conference has shown how Linked Open Data and semantic technologies can boost the creation of a new breed of knowledge-enabled and content-based RSs. In particular, one of the tasks of the challenge was devoted to the design of Linked Open Data-enabled RSs whose effectiveness was evaluated by considering a combination of both accuracy of the recommendation list and the diversity of items belonging to it.
- Definition of content-based methods for mining microblogging data and deep analysis of text reviews. In particular, aspect-based opinion mining and sentiment analysis techniques can support the design of recommendation methods that take into account the evaluation of aspects of items expressed in text reviews. As an example, "Aspect Based Sentiment Analysis" was one of the tasks of SemEval 2014 and was devoted to evaluate methods for automated detection of both aspects and the sentiment expressed towards each aspect in text reviews of laptops and restaurants. These methods could be exploited for implicit rating of aspects and can support the development of multicriteria recommendation techniques.

Emotions-Aware Recommender Systems

Affective Computing is the study and development of systems and devices that can recognize, interpret, process, and simulate human affects. It is an interdisciplinary field spanning computer science, psychology, and cognitive science (Picard 1997). In recent years, the maturity of methods for the unobtrusive acquisition of affective information has grown to a level that allows its incorporation in personalized systems. In order to achieve true emotion-aware personalized systems, psychological theories and computational models need to become a part of user models and personalization algorithms (Tkalčič et al. 2016).

RSs can exploit affective information in order to better model the recommendation process as a decision-making process which is influenced by the emotional state of the user (Tkalčič et al. 2016). In fact, emotions have an impact on actions that we undertake: for instance, when the user wants to choose a song to listen, her decision is driven by music preferences, as well as by her current affective state (mood, emotions, feelings). Acquiring information about the user's affective state might help to improve the recommendation process, which will be grounded on a deeper user model. The literature about this topic distinguishes two main approaches to the problem of including affective information in the recommendation process:

- adoption of cognitive features as additional dimensions of a Context-Aware Recommender System (CARS) (Zheng et al. 2013).
- design of new recommendation models which take into account cognitive aspects of the human decision process. Those models are used to build "affective-aware" RSs which provide users with items compliant with their emotional states (Tkalčič et al. 2013).

The first approach basically considers emotions as external factors influencing the recommendation algorithm. CARS use a multidimensional rating space, i.e., they have multiple two-dimensional tables *Users* × *Items* → *Ratings*, in order to model possible alternative situations in which the user can consume an item (Adomavicius and Tuzhilin 2011). For instance, music preferences of users might depend on the task which they are performing; therefore, the situation in which a song is listened (and rated) is a relevant contextual factor. Gonzales was one of the first researchers who used cognitive features (González et al. 2004). The author defined a "Smart User Model" that associates item features to a finite set of emotional states (Markedly Negative, More Negative, Neutral, More Positive, and Markedly Positive). This association is used to identify those item features which lead the user in a positive mood and therefore they can be useful for item preselection or post-selection. More recently, Zheng demonstrates the effectiveness of cognitive features both in context-aware splitting and differential context modeling (Zheng et al. 2016). He clearly showed that emotions are worth considering as contextual features for recommendation, regardless of the type of context-aware approach that is chosen.

On the other hand, Affective RSs embed emotional information into the recommendation model. Some cognitive theoretical studies tried to define a predictive model of user choices. Appraisal models (Arnold 1960; Lazarus 1966; Reisenzein 1994) are the most widely used in computer science. They describe emotions as functions of internal mental states (appraisal variables) which users refer to when they evaluate choosing options. In literature, most commonly used appraisal variables are: pleasure/valence, arousal, and dominance/control (Arnold 1960; Mehrabian 1996; Reisenzein 1994). Pleasure or valence tells how important is an option for the user. Sometimes, pleasure is also associated with the desire of obtaining that item. The arousal indicates the state of excitement induced by an option on the user (it can be positive or negative). Finally, the dominance or control evaluates how much the user feels herself confident about choosing an option. Tkalčič uses a valence-arousal-dominance (VAD) emotive space for describing the users' reactions to images and store them as additional metadata of the item description (Tkalčič et al. 2013).

Future research might be oriented toward the design of RSs that show emotional intelligence, "a

type of social intelligence that involves the ability to monitor one's own and others' emotions, to discriminate among them, and to use the information to guide one's thinking and actions" (Gonzalez et al. 2007). A system showing emotional intelligence is able: to perceive emotions, to use emotions to facilitate thought, to understand emotions, and to manage emotions (Mayer et al. 2003). An RS might connect emotional information to the feedback the user gave on items, to become more and more closer to her, covering the role of a personal assistant (Hauswald et al. 2015).

Inducing Serendipity in Recommender Systems

One problem with RSs is that the user might be provided with items within her existing range of interests and her tendency towards a certain behavior is reinforced by creating a self-referential loop. This drawback is known as *over-specialization*or *serendipity problem* (McNee et al. 2006) and stems from the fact that the goal of the system is to find items that best match the model of user preferences in order to improve accuracy, regardless of the actual usefulness of the suggestions. The importance of taking into account factors, other than accuracy, which contribute to the perceived quality of recommendations is emphasized in recent research (Castells et al. 2011; Hurley and Zhang 2011; Zhang et al. 2012). One of these factors is *serendipity* that can be seen as the experience of receiving unexpected suggestions helping the user to find surprisingly interesting items she might not have otherwise discovered, or that would have been really hard to discover (Herlocker et al. 2004). Serendipity has been recognized as a goal that often conflicts with accuracy (Fleder and Hosanagar 2009); therefore, it is important that systems were designed and evaluated by taking into account the need of properly balancing these two factors (Gemmis et al. 2015b). As an extreme case, let us consider random recommendations, which improve serendipity but cause a drastic loss in accuracy, making the system actually ineffective. Finding unexpected recommendations is one of the most common strategies for inducing serendipity into filtering systems. One of the first attempts in that direction was the development of MAX (Campos and de Figueiredo 2001), a software agent that mimics the browsing behavior of users navigating the Web just for the sake of wandering. Among more recent systems, Oku and Hattori propose a fusion-based RS which suggests items that have the mixed features of two user-input items (Oku and Hattori 2011), while graph-based methods are adopted in the TANGENT recommendation algorithm (Onuma et al. 2009). The idea of using external knowledge sources to provide serendipitous recommendations is suggested in (Maccatrozzo 2012). Some collaborative approaches have been developed as well (Kawamae 2010; Kawamae et al. 2009). The concept of serendipity is very interesting also for Web companies. For instance, Facebook acquired Glancee, in order to make easier to meet interesting people around you, empowering serendipity and pioneering social discovery. Clever Sense (Lawton 2011), recently acquired by Google, was a startup company that developed a serendipity engine able to learn interests and preferences of users based on their interactions with various sources, including Facebook and Twitter, in order to provide more surprising recommendations.

Cross-References

- ▶ Emotions and Personality in Recommender Systems
- ▶ Recommender Systems Evaluation
- ▶ Recommender Systems, Semantic-Based
- ▶ Recommender Systems: Models and Techniques

References

Adomavicius G, Kwon Y (2012) Improving aggregate recommendation diversity using ranking-based techniques. IEEE Trans Knowl Data Eng 24(5):896–911

Adomavicius G, Tuzhilin A (2005) Toward the next generation of recommender systems: a survey of the state-

of-the-art and possible extensions. IEEE Trans Knowl Data Eng 17:734–749
Adomavicius G, Tuzhilin A (2011) Context-aware recommender systems. In: Recommender Systems Handbook. Springer, New York/London, pp 217–253
Arnold MB (1960) Emotion and personality. Columbia University Press, New York
Baeza-Yates R, Ribeiro-Neto B (1999) Modern information retrieval. Addison-Wesley Longman Publishing Co., Inc., Boston
Balabanović M, Shoham Y (1997) Fab: content-based, collaborative recommendation. Commun ACM 40:66–72
Bambini R, Cremonesi P, Turrin R (2011) A recommender system for an IPTV service provider: a real large-scale production environment. In: Recommender systems handbook. Springer, New York/London, pp 299–331
Burke R (2002) Hybrid recommender systems: survey and experiments. User Model User-Adap Inter 12(4):331–370
Burke R (2007) Hybrid web recommender systems. In: The adaptive web. Springer, Berlin/Heidelberg, pp 377–408
Burke RD, Hammond KJ, Young BC (1996) Knowledge-based navigation of complex information spaces. In: Proceedings of the thirteenth national conference on artificial intelligence - vol 1, AAAI'96, AAAI Press, 1996. Menlo Park, California. pp 462–468
Cambria E, Schuller B, Liu B, Wang H, Havasi C (2013) Knowledge-based approaches to concept-level sentiment analysis. IEEE Intell Syst 28(2):12–14
Campos J, de Figueiredo AD (2001) Searching the unsearchable: inducing serendipitous insights. In: Proceedings of the workshop program at the fourth international conference on case-based reasoning, ICCBR, 2001, pp 159–164
Castells P, Wang J, Lara R, Zhang D (2011) Workshop on novelty and diversity in recommender systems – DiveRS. In: Proceedings of the 5th ACM conference on recommender systems, RecSys 2011, ACM, 2011, pp 393–394
Cremonesi P, Turrin R, Airoldi F (2011) Hybrid algorithms for recommending new items. In: Proceedings of the 2nd international workshop on information heterogeneity and fusion in recommender systems, HetRec '11, ACM, New York, 2011, pp 33–40
Cremonesi P, Said A, Tikk D, Zhou MX (2016) Introduction to the special issue on recommender system benchmarking. ACM TIST 7(3):38
Degemmis M, Lops P, Semeraro G (2007) A content-collaborative recommender that exploits wordnet-based user profiles for neighborhood formation. User Model User-Adapt Interact 17(3):217–255
Ekstrand MD, Riedl J, Konstan JA (2011) Collaborative filtering recommender systems. Found Trends Human-Comput Interact 4(2):175–243
Fleder D, Hosanagar K (2009) Blockbuster culture's next rise or fall: the impact of recommender systems on sales diversity. Manag Sci 55(5):697–712
Ge M, Delgado-Battenfeld C, Jannach D (2010) Beyond accuracy: evaluating recommender systems by coverage and serendipity. In: Proceedings of the 2010 ACM conference on recommender systems, RecSys 2010, ACM, Barcelona, 26–30 Sept 2010, pp 257–260
de Gemmis M, Lops P, Musto C, Narducci F, Semeraro G (2015a) Semantics-aware content-based recommender systems. In: Ricci F et al (eds) Recommender systems handbook. Springer, Heidelberg, pp 119–159
de Gemmis M, Lops P, Semeraro G, Musto C (2015b) An investigation on the serendipity problem in recommender systems. Information Processing and Management 51(5):695–717
Goldberg D, Nichols D, Oki BM, Terry D (1992) Using collaborative filtering to weave an information tapestry. Commun ACM 35(12):61–70
González G, López B, De La Rosa JL (2004) Managing emotions in smart user models for recommender systems. ICEIS 5:187–194
Gonzalez G, De La Rosa JL, Montaner M, Delfin S (2007) Embedding emotional context in recommender systems. In: Data engineering workshop, 2007 I.E. 23rd international conference on, IEEE, 2007. Los Alamitos, California. pp 845–852
Gunawardana A, Shani G (2011) Evaluating recommender systems. In: Ricci F, Rokach L, Shapira B, Kantor PB (eds) Recommender systems handbook. Springer, New York/London, pp 265–308
Halevy AY, Norvig P, Pereira F (2009) The unreasonable effectiveness of data. IEEE Intell Syst 24(2):8–12
Hanani U, Shapira B, Shoval P (2001) Information filtering: overview of issues, research and systems. User Model User-Adap Inter 11:203–259
Harper FM, Konstan JA (2015) The movielens datasets: history and context. ACM Trans. Interact Intell Syst 5(4):19:1–19:19
Hauswald J, Laurenzano MA, Zhang Y, Li C, Rovinski A, Khurana A, Dreslinski RG, Mudge T, Petrucci V, Tang L et al. (2015) Sirius: an open end-to-end voice and vision personal assistant and its implications for future warehouse scale computers. In: ACM SIGPLAN notices, vol 50, ACM, 2015. New York, US. pp 223–238
Herlocker JL, Konstan JA, Terveen LG, Riedl JT (2004) Evaluating collaborative filtering recommender systems. ACM Trans Inf Syst 22:5–53
Hill W, Stead L, Rosenstein M, Furnas G (1995) Recommending and evaluating choices in a virtual community of use. In: Proceedings of the SIGCHI conference on human factors in computing systems, CHI '95, ACM Press/Addison-Wesley Publishing Co., New York, 1995, pp 194–201

Hurley N, Zhang M (2011) Novelty and diversity in top-N recommendation – analysis and evaluation. ACM Trans Internet Technol 10(4):14

Jannach D, Zanker M, Felfernig A, Friedrich G (2011) Recommender systems: an introduction. Cambridge University Press, New York, US

Kawamae N (2010) Serendipitous recommendations via innovators. In: Proceedings of the 33rd international ACM SIGIR conference on research and development in information retrieval, ACM, 2010. New York, US. pp 218–225

Kawamae N, Sakano H, Yamada T (2009) Personalized recommendation based on the personal innovator degree. In: Proceedings of the ACM conference on recommender systems, ACM, 2009. New York, US. pp 329–332

Konstan JA, Miller BN, Maltz D, Herlocker JL, Gordon LR, Riedl J (1997) Grouplens: applying collaborative filtering to usenet news. Commun ACM 40(3):77–87

Lawton G (2011) In the News. Simplifying Mobile Recommendation Technology with AI. *IEEE Intell Syst* 26(3):8–9

Lazarus RS (1966) Psychological stress and the coping process. McGraw-Hill, New York

Linden G, Smith B, York J (2003) Amazon.com recommendations – item-to-item collaborative filtering. IEEE Internet Comput 7(1):76–80

Liu J, Dolan P, Pedersen ER (2010) Personalized news recommendation based on click behavior. In: Proceedings of the 15th international conference on intelligent user interfaces, IUI '10, ACM, New York, 2010, pp 31–40

Lops P, de Gemmis M, Semeraro G (2011) Content-based recommender systems: state of the art and trends. In: Ricci F, Rokach L, Shapira B, Kantor PB (eds) Recommender systems handbook. Springer, Berlin, pp 73–105

Maccatrozzo V (2012) Burst the filter bubble: using semantic web to enable serendipity. In: Proceedings of the 11th international conference on the semantic web – volume part II, ISWC'12, Springer, Berlin/Heidelberg, 2012, pp 391–398

Maes P (1994) Agents that reduce work and information overload. Commun ACM 37(7):31–40

Mayer JD, Salovey P, Caruso DR, Sitarenios G (2003) Measuring emotional intelligence with the msceit v2. 0. Emotion 3(1):97

McLaughlin MR, Herlocker JL (2004) A collaborative filtering algorithm and evaluation metric that accurately model the user experience. In: SIGIR 2004: proceedings of the 27th annual international ACM SIGIR conference on research and development in information retrieval, Sheffield, 25–29 July 2004, pp 329–336

McNee SM, Riedl J, Konstan JA (2006) Being accurate is not Enough: How accuracy metrics have hurt recommender systems. In: Extended abstracts proceedings of the 2006 conference on human factors in computing systems, ACM, 2006. New York, US. pp 1097–1101

Mehrabian A (1996) Pleasure-arousal-dominance: a general framework for describing and measuring individual differences in temperament. Curr Psychol 14(4):261–292

Mikolov T, Le QV, Sutskever I (2013) Exploiting similarities among languages for machine translation. CoRR, abs/1309.4168

Murakami T, Mori K, Orihara R (2008) Metrics for evaluating the serendipity of recommendation lists. In: New frontiers in artificial intelligence, volume 4914 of lecture notes in computer science, Springer, 2008. Berlin, Heidelberg. pp 40–46

Musto C, Semeraro G, Lops P, de Gemmis M, Lekkas G (2015) Personalized finance advisory through case-based recommender systems and diversification strategies. Decis Support Syst 77:100–111

Musto C, Lops P, Basile P, de Gemmis M, Semeraro G (2016a) Semantics-aware graph-based recommender systems exploiting linked open data. In: Vassileva J, Blustein J, Aroyo L, D'Mello SK (eds) Proceedings of the 2016 conference on user modeling adaptation and personalization, UMAP 2016, ACM, Halifax, 13–17 July 2016, pp 229–237

Musto C, Narducci F, Lops P, de Gemmis M, G. Semeraro G (2016b) Explod: a framework for explaining recommendations based on the linked open data cloud. In: Proceedings of the 10th ACM conference on recommender systems, ACM, Boston, 15–19 Sept 2016, pp 151–154

Oku K, Hattori F (2011) Fusion-based recommender system for improving serendipity. In: Proceedings of the ACM RecSys 2011 workshop on novelty and diversity in recommender systems (DiveRS), volume 816 of CEUR workshop proceedings, CEUR-WSorg, 2011. Aachen, Germany. pp 19–26

Onuma K, Tong H, Faloutsos C (2009) TANGENT: a novel, 'surprise me', recommendation algorithm. In: Proceedings of the 15th ACM SIGKDD international conference on knowledge discovery and data mining, ACM, 2009, pp 657–666

Picard RW (1997) Affective computing. MIT Press, Cambridge, MA

Picault J, Ribière M, Bonnefoy D, Mercer K (2011) How to get the recommender out of the lab? In: Recommender systems handbook. Springer, New York/London, pp 333–365

Reisenzein R (1994) Pleasure-arousal theory and the intensity of emotions. J Pers Soc Psychol 67(3):525

Resnick P, Varian HR (1997) Recommender systems. Commun ACM 40(3):56–58

Resnick P, Iacovou N, Suchak M, Bergstrom P, Riedl J (1994) Grouplens: an open architecture for collaborative filtering of netnews. In: Proceedings of the 1994 ACM conference on computer supported cooperative work, CSCW '94, ACM, New York, 1994, pp 175–186

Ricci F, Rokach L, Shapira B, Kantor PB (2011) Recommender systems handbook. Springer, New York/London

Ricci F, Rokach L, Shapira B (eds) (2015). Recommender systems handbook, 2nd edn. Springer US
Rubens N, Elahi M, SugiyamaM, Kaplan D (2015) Active learning in recommender systems. In: Ricci F, Rokach L, Shapira B (eds) Recommender systems handbook. Springer US, pp 809–846
Said A, Bellog A (2014) Comparative recommender system evaluation: benchmarking recommendation frameworks. In: Proceedings of the 8th ACM conference on recommender systems, RecSys '14, ACM, New York, 2014, pp 129–136
Sarwar B, Karypis G, Konstan J, Reidl J (2001) Item-based collaborative filtering recommendation algorithms. In: Proceedings of the 10th international conference on World Wide Web, ACM, New York, 2001, pp 285–295
Shardanand U, Maes P (1995) Social information filtering: algorithms for automating "word of mouth". In: Proceedings of the SIGCHI conference on human factors in computing systems, CHI '95, ACM Press/Addison-Wesley Publishing Co., New York, 1995, pp 210–217
Tkalčič M, Burnik U, Odić A, Košir A, Tasič J (2013) Emotion-aware recommender systems – a framework and a case study. In: ICT Innovations 2012. Springer, Berlin, Heidelberg. pp 141–150
Tkalčič M, Košir A, De Carolis B, de Gemmis M, Odić A (2016) Emotions and personality in personalized services: methods, evaluation and applications. Springer International Publishing Switzerland
Vargas S, Castells P (2011) Rank and relevance in novelty and diversity metrics for recommender systems. In: Proceedings of the ACM conference on recommender systems, RecSys 2011, ACM, 2011, New York, US. pp 109–116
Woyke E (2011) Serendipitous shopping. Forbes Magazine
Zhang YC, Séaghdha DÓ, Quercia D, Jambor T (2012) Auralist: introducing serendipity into music recommendation. In: Proceedings of the fifth international conference on web search and data mining, ACM, 2012. New York, US. pp 13–22
Zheng Y, Mobasher B, Burke RD (2013) The role of emotions in context-aware recommendation. Decisions@ RecSys 2013:21–28
Zheng Y, Mobasher B, Burke R (2016) Emotions in context-aware recommender systems. In: Emotions and personality in personalized services. Springer International Publishing Switzerland. pp 311–326

Recommended Reading

Jannach D, Zanker M, Felfernig A, Friedrich G (2011) Recommender systems: an introduction. Cambridge University Press
Ricci F, Rokach L, Shapira B (eds) (2015) Recommender systems handbook, 2nd edn. Springer
Tkalčič M, Košir A, De Carolis B, de Gemmis M, Odić A (eds) (2016) Emotions and personality in personalized services: methods, evaluation and applications. Springer

Recommender Systems, Semantic-Based

Fatih Gedikli and Dietmar Jannach
Department of Computer Science, TU Dortmund, Dortmund, Germany

Synonyms

Social recommender system; Tag-based recommendation; Web 2.0 recommender systems

Glossary

Cold start problem	The ramp-up phase of a recommender where preference data is missing
Collaborative filtering	A recommendation method which is based on rating information of the user community
Content-based filtering	A recommendation method which is based on the characteristics of the recommended items as well as individual user feedback
Hybrid recommender system	A recommender system that combines different recommendation approaches or data sources
Rating matrix	A grid containing the users' implicit or explicit item rating

Definition

Recommender systems (RS) are software tools that are predominantly used on e-commerce sites and for other online services as a means to help the online customer find the most relevant shopping items or pieces of information quickly. Today, such systems can be found for a variety of different domains such as books, movies, music, hotels, restaurants, or news.

The particularity of RS is that they are able to provide personalized recommendations, which are based on the past and current behavior or the explicit preferences of individual users, on the preferences of a user community as a whole, or on various other forms of available information.

The main task of an RS usually is to predict as precisely as possible which of the recommendable items will be of interest for and accepted by the user (Jannach and Adomavicius 2016). Since the mid-1990s, when RS started to emerge as a research field of their own, a large variety of methods have been proposed to increase the quality of the recommendations, measured, e.g., in terms of their accuracy.

In their early years, RS were mainly considered as a tool for e-commerce sites or for personalized information filtering. However, the emergence of the Social and Semantic Web – or, more generally, the Web 2.0 – soon had a strong impact on the field of RS. One aspect is related to the amount of information we know about the users, which is crucial for the recommendation quality. The Web 2.0 is participatory: People connect on social networks and share information about their personal profile and their interests, they actively contribute content on blogs and microblogs, and they share, rate, and review all types of resources online. Overall, much more information about the user is theoretically available than in the past, when explicit and implicit rating data and the past transaction history were often the only available knowledge sources.

Besides an increased engagement of users, the Web 2.0 also brought new application fields for RS technology. Today, we can find systems on Social Web platforms that recommend people to connect with or people to follow systems that generate personalized information feeds based on the user's interests and systems that recommend potentially interesting Web resources such as images, Web pages, or blog posts. Even the choice of an appropriate set of tags and annotations for user-contributed content can be driven by an RS.

In this entry, we will focus on RS that exploit (semantic) knowledge sources that have become available in the Social and Semantic Web. In particular, we will first focus on the role of user-provided tagging data in the recommendation process and then briefly shed light on other, more structured information sources and discuss recent developments with respect to learning techniques.

Introduction

With the continuously growing amount of information on the Web, the availability of appropriate tools that help the online user retrieve or discover interesting items becomes more and more important. Recommender systems are one type of such tools which are capable of generating personalized lists of shopping items, reading lists, or, more generally, action alternatives (Jannach et al. 2010).

The recommendations of an RS can be based on different types of information. In most cases, the quality of the recommendations and the corresponding effect on the users are directly related to the amount and quality of the available information on which the recommendations are based on. Today, the most popular class of recommendation methods is called collaborative filtering (CF). CF methods rely on the existence of item ratings which are provided by an implicit online community. Amazon.com is an example of an online retailer who relies among others on such methods in their recommendation engines (Linden et al. 2003).

The other major type of systems is based on what is called "content-based filtering." While CF recommender systems recommend items similar users liked in the past, the task of a content-based recommender system is to recommend items that are similar to those the target user liked in the past.

We illustrate the basic rationale of a content-based recommendation method with an example from the movie domain. Table 1 represents an excerpt from an example movie database which also provides plot keywords for each movie, i.e., an item's content description is represented by a set of plot keywords. Table 2, on the other hand, shows an excerpt from the user database.

A simple content-based recommender computes recommendations for *Alice* by selecting

Recommender Systems, Semantic-Based, Table 1 Movie data set with content description

Movie	Plot keywords			
Heat	Detective	**Criminal**	Thief	**Gangster**
Scarface	**Gangster**	**Criminal**	Drugs	Cocaine
Amélie	**Love**	Waitress	France	Happiness
Eat Pray Love	Divorce	India	**Love**	Inner peace

movies Alice is not aware of and which are similar to those movies she watched before. In this example, similarity between movies could be defined by the number of overlapping keywords. The unseen movie *Amélie*, for example, has one keyword in common with *Eat Pray Love* ("love"). Therefore, we can assume some degree of similarity between both movies. Since *Alice* liked *Eat Pray Love* in the past, the movie *Amélie* could be recommended to her.

Note that for a content-based recommender, no user community is required for generating recommendations. However, the target user has to provide an initial list of "like" and "dislike" statements or ratings on a given scale. Alternatively, customer actions such as viewing or purchasing an item can be interpreted as positive signals. New items, on the other hand, can be incorporated in the recommendation process because similarity to existing items can be computed without the need for any rating data.

In the example discussed above, the importance of each keyword was not taken into account, that is, each keyword gets the same importance. However, it appears intuitive that keywords which appear more often in descriptions are less representative. Therefore, the *TF-IDF* encoding format was proposed and gained popularity in particular in the field of information retrieval and is also the basis for various approaches that exploit Social Web data. *TF-IDF* stands for *term frequency–inverse document frequency* and is used to determine the relevance of terms in documents of a document collection. For convenience, we will assume in the following that the underlying item set consists of text documents; e.g., the plot keywords for each movie in Table 1 can be seen as one document. As the name suggests, the *TF-IDF* measure is composed of two frequency measures. The idea of the *term frequency* measure

Recommender Systems, Semantic-Based, Table 2 User database

User	Preference profile
Alice	Eat Pray Love, What Women Want
Bob	Scarface, Carlito's Way, Terminator II

$TF(i, j)$ is to estimate the importance of a term i in a given document j by counting the number of times a given term i appears in document j. Additionally, a normalization is possible, e.g., by dividing the absolute number of occurrences of term i in document j by the absolute number of occurrences of the most frequent word in document j. Several other schemes are however possible.

On the other hand, the idea of the *inverse document frequency* measure $IDF(i)$ is to capture the importance of a term i in the whole set of available documents. Therefore, $IDF(i)$ can be seen as a global measure which reduces the weight of words that appear in many documents (e.g., stop words such as "a," "by," or "about"), since they are usually not representative and helpful to differentiate between documents. Formally, inverse document frequency is usually computed as $\mathrm{IDF}(i) = \log \frac{N}{n(i)}$ where N is the size of the document set and $n(i)$ is the number of documents in which the given term i appears. We assume that each term appears in at least one document, i.e., $n(i) \geq 1$. If $n(i) = N$ the logarithm function returns indicating that term i is of no importance for discriminating documents as it appears in all documents.

Finally, the *TF-IDF* measure which represents the weight for a term i in document j is defined as the combination of these two measures: $\mathrm{TF} - \mathrm{IDF}(i, j) = \mathrm{TF}(i, j)^{*}\mathrm{IDF}(i)$.

With the help of the *TF-IDF* measure, text documents, or generally speaking the textual

description of items, can be encoded as *TF-IDF* weight vectors.

One way of computing n recommendations is to find the n most similar items to the user's average *TF-IDF* weight vector of the user's liked documents. The cosine similarity metric is often used for computing the proximity between items.

Next, we will view user-provided tags as content descriptors and describe the role of tagging data in the recommendation process.

Recommendations Based on Social Web Tagging Data

The advent of the Social Web opened new ways of promoting and sharing user-generated content. Web site visitors turned from passive recipients of information into active and engaged contributors. The Social Web allows users to create and share a large amount of different types of content such as pictures, videos, bookmarks, blogs, comments, or tagging data. It allows users to collaborate with other users on new types of Web applications called Social Web platforms such as Delicious (http://www.delicious.com) and Flickr (http://www.flickr.com). Leveraging useful data from the large amount of user-contributed data available in the Social Web represents a challenging topic which however also opens new opportunities for recommender system research.

For example, user-contributed tags are today a popular means for users to organize and retrieve items of interest in the Social Web. As the application areas of tags are manifold, they play an increasingly important role in the Social Web. They can be used to categorize items, express preferences about items, retrieve items of interest, and so on.

Collaborative tagging or social tagging describes the practice of collaboratively annotating items with freely chosen tags (Golder and Huberman 2006) which plays an important role in sharing content in the Social Web (Ji et al. 2007). In a social tagging system such as Delicious and Flickr, users typically create new content (items), assign tags to these items, and share them with other users (Cantador et al. 2010). The result of social tagging is a complex network of interrelated users, items, and tags often referred to as a community-created *folksonomy.* The term folksonomy is a neologism introduced by the information architect Thomas Vander Wal (http://vanderwal.net/folksonomy.html) and is composed of the terms *folk* as in people and *taxonomy* which stands for the practice and science of classification. A folksonomy is defined as a tuple $F: = (U, T, R, Y)$ where U, T, and R are finite sets, whose elements are called users, tags, and resources, and Y is a ternary relation between them, i.e., $Y \subseteq U \times T \times R$ called tag assignments.

In contrast to typical taxonomies including formal Semantic Web ontologies, social tagging represents a more lightweight approach, which does not rely on a predefined set of concepts and terms that can be used for annotation.

Tagging data also gained importance in the field of RS. User-generated tags not only convey additional information about the items; they also tell something about the user. For example, if two users use the same set of tags to describe an item, we can assume a certain degree of similarity between those. Therefore, tagging data can be used to augment the basic user–item rating matrix.

In the following, a possible categorization of building tag-based RS is given.

Using Tags as Content Maybe the easiest way to use tagging data for RS is to consider tagging data as an additional source of content. Several works exist that view tags as content descriptors for content-based systems; see, for example, in Firan et al. (2007), Li et al. (2008), or Vatturi et al. (2008).

Similarly, in de Gemmis et al. (2008), tagging data is used for an existing content-based recommender system in order to increase the overall predictive accuracy of the system. Machine-learning techniques are applied both on the textual descriptions of items (static data) and on the tagging data (dynamic data) to build user profiles and learn user interests. The user profile consists of three parts: the static content, the user's personal tags, and the social tags which build the collaborative part of the user profile. Thus, in this work, tags are seen as an additional source of information used for learning the profile of a particular

user. The authors compare their tag-based approach with a pure content-based recommender in a user study. The results show that the recommendations made by the tag-augmented recommender are slightly more accurate than the recommendations of the pure content-based one.

In Firan et al. (2007), tags are also seen as content descriptors for different content-based systems. Tags are used for building user profiles for the popular music community site Last.fm. To address the so-called cold start problem (when new users or items enter in the system), the user profiles are inferred automatically, e.g., from the music tracks available on the computer of each user, thus reducing the manual effort from the user's side to express his or her preferences. The authors show that tag-based profiles can lead to better music recommendations than conventional user profiles based on song and track usage.

In Cantador et al. (2010), tags are considered as content features that describe both user and item profiles. The authors propose weighting functions which assess the importance of a particular tag for a given user or item and similarity functions which compute the similarity between a user profile and an item profile. These weighting and similarity functions are then combined in different content-based recommendation models.

In that work, user interests and item characteristics are modeled as vectors $u\,m = (\,u\,m_{,1}, \ldots, u\,m,L\,)$ and $i\,n = (\,u\,n_{,1}, \ldots u\,n,L)$ of length L, respectively, where L is the number of tags in the folksonomy, $u\,m, l$ is the number of times user $u\,m$ has annotated items with tag $t\,l$, and $i\,n,l$ is the number of times item $i\,n$ has been annotated with tag $t\,l$. After modeling users and items as vectors accordingly, the authors can adopt the *TF-IDF* vector space model.

The evaluation results on the Delicious and Last.fm data sets show that the recommendation models focusing on user profiles outperform the models focusing on item profiles.

Tagging data can also be incorporated in search engines to personalize the search results. According to Pitkow et al. (2002), two basic approaches to Web search personalization can be differentiated. In the first approach, a user's original query is modified and adapted to the profile of the user. For example, the query "eclipse" might be extended to "eclipse software development environment" if we know that the user has an interest in software development. In the second approach, the query is not modified, but the returned list of search results is re-ranked according to the user profile.

An example for the latter approach is given in Noll and Meinel (2007). The authors propose a pure tag-based personalization method to re-rank the Web search results which is independent from the underlying search engine. The basic idea is to use bookmarks and tagging data to re-rank the documents in the search result list. The authors propose a concept called *tagmarking* which translates the keywords in the search query to tags and assign them to the bookmarked Web page that is associated with the query. Bookmarks and tags are aggregated in a binary *tag–document* matrix where each column (vector) represents a bookmark of a document with its components set to 1 when the corresponding tag is associated with the document and otherwise. The user profile is modeled as a vector which contains the weights assigned to each tag. The *tag–user* matrix and the document profile are built analogously. Finally, in the personalization step, the documents are re-ranked according to a similarity matrix which combines both the user profile and the document profile. Table 3 shows an example of how personalization affects Google's result list for the search query "security"; see also Noll and Meinel (2007). The ranking of the Web site of the US Social Security Administration (ssa.gov), for instance, has increased because – according to the authors – the user who submitted the query also showed interest in insurance matters.

Clustering Approaches Many tag-based clustering approaches have been proposed in the literature which cluster users and items according to topics of interest by exploiting additional tagging data; see, for example, Li et al. (2008), Xu et al. (2011a), or Zanardi and Capra (2011).

In Li et al. (2008), the authors propose a system called *Internet Social Interest Discovery* (ISID) and show its application for the social bookmarking system Delicious. The ISID system, as the name suggests, is a system specifically

Recommender Systems, Semantic-Based, Table 3 Re-ranking Google's result list (Noll and Meinel 2007)

Rank	Δ	Rank	URL
1	•		securityfocus.com
2	↑	+7	cert.org
3	•		microsoft.com/technet/security/def
4	↑	+4	w3.org/Security
5	↑	+2	ssa.gov
6	↑	+4	nsa.gov
7	↓	−5	microsoft.com/security
8	↓	−2	windowsitpro.com/WindowsSecurity
9	↓	−4	whitehouse.gov/homeland
10	↓	−6	dhs.gov

designed to reveal common user interests based on user-provided tags. The basic assumption, which is then justified in the work, is that user-provided tags are more effective at reflecting the users' understanding of the content than the most informative keywords extracted from the corpus of a Web page. Therefore, tags are seen as good candidates for capturing user interests.

Similarly, in Xu et al. (2011b), co-occurring tags are used to build topics of interests. In the resource–tag matrix, each tag is described by a set of resources to which this tag has been assigned. Afterward, the authors obtain the tag similarity matrix by computing the cosine similarity between the tag vectors in the resource–tag matrix. Based on this similarity matrix, a graph is constructed where the tags represent the nodes and the edges represent the similarity relationships between the tags. Afterward, a clustering algorithm is used to cluster the tags and to extract the topics of interests. Finally, the authors present the topic-oriented tag-based recommendation system (*TOAST*). TOAST applies preference propagation on an undirected graph called the "topic-oriented graph" which consists of three kinds of nodes: users, resources, and topics. In their recommendation strategy, the authors propagate a user's preference through transitional nodes such as users, resources, and topics, to reach an unknown resource node along the shortest connecting path.

In Shepitsen et al. (2008), the authors focus on a recommendation scenario where a user selects a tag and expects a recommendation of related resources. They thus present a recommendation approach which recommends items for a given user–tag pair (u, t). Tag clusters are presumed to act as a bridge between users and items. The idea behind tag clusters is to account for the effects of unsupervised tagging such as redundancy and ambiguity. The authors first determine the items which have some similarity to the query tag t. These items are then re-ranked according to the user profile. The ranking algorithm first calculates the user's interest with respect to each tag cluster as well as the nearest clusters of each item. The nearest clusters are determined by counting the number of times the item was annotated with a tag from the cluster divided by the total number of times the item was annotated. Both measures are then combined in the final personalized rank score used to re-rank the item sets. The results show that data sparsity has a big influence on the quality of the clusters which, on the other hand, corresponds with the accuracy of the recommendations.

Hybrid Approaches Hybrid approaches in general combine different sources of information or different algorithms to make recommendations. In the context of semantic-based recommenders, social data such as tagging data can be combined with other types of information such as content data (Seth and Zhang 2008) or data from the Semantic Web (Durao and Dolog 2010).

In Seth and Zhang (2008), a Bayesian model-based recommender that leverages content and social data is presented. In Durao and Dolog (2010), on the other hand, a tag-based recommender which recommends Web pages is extended such that also semantic similarities between tags are discovered which are usually

Recommender Systems, Semantic-Based, Table 4 Exploiting semantic relations between tags

Web page	Tags
P1	Programming, Web 2.0, Framework
P2	PHP, Scripting, Web 2.0
P3	C++, Programming, Framework

not taken into account in syntax-based similarity approaches. Consider the example in Table 4.

If we assume a syntax-based similarity measure, the Web pages *P1* and *P3* will be considered more similar than *P1* and *P2* as *P1* and *P3* have two tags in common ("Programming" and "Framework"), whereas *P1* and *P2* only share one tag ("Web 2.0"). However, if we analyze the tags in more detail, we see that *P1* is closer to *P2* than to *P3* because *P1* and *P2* are about Web technologies, whereas *P3* focuses on C++ which is a programming language that is usually not associated with Web technologies. In a semantic-based similarity approach which takes lexical and social factors of tags into account, these semantic relations can be made explicit. For example, "Web 2.0" would be considered together with "Scripting," and "Programming" together with "PHP." The authors try to overcome this problem of ignoring the semantic term relations by hybridizing syntax-based approaches such as tag popularity with a new semantic-based approach. In particular, they also make use of external semantic sources such as the WordNet dictionary and different ontologies from Linked Open Data (LOD) available on the Web to identify semantic relations between tags. These semantic relations are then considered in the similarity calculations. Their experimental results show increases of precision when semantic relations are exploited as additional knowledge sources.

Tag-Enhanced Collaborative Filtering A substantial number of papers have been published in recent years on tag-enhanced CF recommender algorithms in which tagging data is used for improving the performance of traditional collaborative filtering recommender systems. In general, tagging data can be incorporated into existing collaborative filtering algorithms in different ways in order to enhance the quality of recommendations (Durao and Dolog 2010; Tso-Sutter et al. 2008 or Zhen et al. 2009).

In Tso-Sutter et al. (2008), for example, the authors incorporate tags into standard collaborative filtering algorithms. The idea is to reduce the three-dimensional relation $\langle$ *user, item, tag* $\rangle$ to three two-dimensional relations, namely, $\langle$ *user, tag* $\rangle$, $\langle$ *item, tag* $\rangle$, and $\langle$ *user, item* $\rangle$. The projection is based on viewing the tags as items ("user tags") and users ("item tags"), respectively. For example, the $\langle$ *user, tag* $\rangle$ relation tags are viewed as items in the user–item rating matrix. These so-called user tags represent tags that are used by the users to tag items. On the other hand, item tags in the $\langle$ *item, tag* $\rangle$ relation correspond to tags that describe the items. Considering the ternary relation as three two-dimensional relations enables the authors to apply standard collaborative filtering techniques. The authors also propose a fusion method which recombines the individual relations. The results of their empirical analysis show that the predictive performance of their proposed fusion method which incorporates tags outperforms the standard tag-unaware collaborative filtering algorithms.

Exploiting tagging data without reducing the three-dimensional $\langle$ *user, item, tag* $\rangle$ relation was the next logical step. In recent years, recommendation methods were proposed which can directly exploit the ternary relationship in tagging data (Symeonidis et al. 2008; Rendle et al. 2009; Rendle and Schmidt-Thieme 2010).

In Hotho et al. (2006), the authors present a *graph-based* tag recommender algorithm called *FolkRank*. As the name suggests, the FolkRank algorithm is based on Google's PageRank algorithm. The main idea of PageRank is that pages are important when linked by other important pages. Therefore, PageRank views the Web as a graph and uses a weight-spreading algorithm to calculate the importance of the pages. FolkRank adopts this idea and assumes that a resource is important if it is tagged with important tags from important users.

A major problem of FolkRank is that it does not scale to larger problem sizes, which is crucial for real-world scenarios. Therefore, in Kubatz

et al. (2011), *LocalRank* – a graph-and-neighborhood-based tag recommendation approach – is presented. Rank computation and weight propagation in LocalRank are done in a similar way to FolkRank but without iterations. As the name suggests, LocalRank computes the rank weights based only on the local "neighborhood" of a given user and resource. Unlike the FolkRank algorithm which considers all elements in the folksonomy, LocalRank focuses on the relevant ones only. Thus, LocalRank can significantly reduce the time needed for computing the recommendations while maintaining or slightly improving recommendation quality.

Tensor factorization (TF) represents another method to directly exploit the ternary relationship in tagging data. In Rendle et al. (2009), the authors see the ternary relationship as a three-dimensional tensor (cube) and apply the idea of computing low-rank approximations for tensors on a tag recommender algorithm. The evaluation results show that their TF-based method achieves even better accuracy results than the tag recommender algorithm FolkRank (Hotho et al. 2006). However, the TF-based model comes with the problem of a cubic runtime in the factorization dimension for prediction and learning. This problem is addressed in the work of Rendle and Schmidt-Thieme (2010). The authors present a *pairwise interaction tensor factorization* (PITF) model with a linear runtime in the factorization dimension. The PITF model explicitly models the pairwise interactions between users, items, and tags.

In Sen et al. (2009), the authors propose tag-based recommender algorithms which they call "tagommenders." The idea is to utilize *tag preference data* in the recommendation process in order to generate better recommendation rankings than state-of-the-art baseline algorithms. The authors define a user's preference for a tag as the user's level of interest in items, e.g., movies, exhibiting the concept represented by the tag. Thus, a user can, for example, indicate that he or she likes *animated* movies, but dislikes movies about *serial killers*. However, since no tag preference data is available, the tag preferences of the target user have to be estimated before the algorithm can predict a user's preference for the target item. To that purpose, the authors evaluate a variety of tag preference inference algorithms. Such algorithms estimate the user's attitude toward a tag, that is, if and to which extent a user likes items that are annotated with a particular tag. Their results show that a linear combination of all preference inference algorithms performed best, that is, algorithms that exploit a variety of signals such as implicit and explicit user data work best.

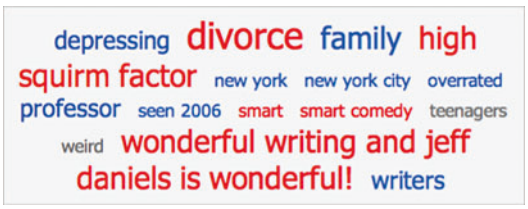
The proposed tagommender algorithms rely on "global" tag preferences: A tag is either liked or disliked by a user, independent of a specific item. In contrast, in Gedikli and Jannach (2013) and Vig et al. (2010), the concept of *item-specific* tag preference data was introduced. The intuition behind this idea is that the same tag may have a positive connotation for the user in one context and a negative in another. For example, a user might like *action movies* featuring the actor *Bruce Willis*, but at the same time, the user might dislike the performance of *Bruce Willis* in *romantic movies*. Based on such an approach, users are able to evaluate an item in various dimensions and are thus not limited to the one single overall vote anymore. According to the study presented in Vig et al. (2010), users particularly appreciated this new feature, a fact that was measured in increased user satisfaction. In Gedikli and Jannach (2013), the authors present first recommendation schemes that take item-specific tag preferences into account when generating rating predictions. The results show that the accuracy can be further improved by exploiting item-specific tag preference data.

Tag-Based Explanations Tagging data is not only a means to enhance existing recommender algorithms, but it can also serve as a means to strengthen and improve explanations for recommendations. Explanations are one of the current research topics in recommender system research. They play an increasingly important role as they can significantly influence the way a user perceives the system.

In Vig et al. (2009), tag-based explanation interfaces which the authors call "tagsplanations" are described and evaluated. The authors propose explanation interfaces which use *tag relevance*

and *tag preference* as two key components. Tag relevance measures the strength of the relationship of the tag to the item, while tag preference indicates the strength of the relationship between a user and the tag. Consider, for example, the tag "love" for a given user–item (movie) pair. Tag preference measures how well the tag "love" describes the particular movie, while tag preference indicates the user's interest in movies about love, that is, how much the user likes/dislikes movies about love in general, independent from a particular movie.

In Gedikli et al. (2011, 2014), the authors introduce explanation interfaces based on personalized and nonpersonalized tag clouds. They compare tag cloud-based explanations with keyword-style explanations proposed in previous work. In order to personalize the explanations, the personalized tag cloud interface makes use of tag preference data proposed in Gedikli and Jannach (2013). These item-specific tag preference values are then mapped to colors which indicate whether the user will like, dislike, or feel neutral about the item features represented by the tags in the cloud. In the example tag cloud in Fig. 1, blue is used as a color for users, for which the system knows or assumes that the user has positive feelings about, e.g., the tag "family." Red tags such as "divorce," on the other hand, represent aspects the user will probably not like. Tags which are marked as neutral are printed in black. The results of their user study showed that users can make better decisions faster when using the tag cloud interfaces rather than the keyword-style explanations. In addition, users generally favored the tag cloud interfaces over keyword-style explanations.



Recommender Systems, Semantic-Based, Fig. 1 Personalized tag cloud explanation

Perspectives

Tag-Based Recommenders Exploiting tagging data for recommendations has become an active research topic in the field of RS. Tag-based computing can further improve the quality of RS and leads to new possibilities but also to a number of new research questions. For example, opinion mining based on folksonomies represents one challenging topic which is currently being addressed in literature. The task of opinion mining is to extract the users' sentimental orientations or attitudes to items based on different information sources such as reviews, blogs, and comments (Dong et al. 2016).

Recently, user-provided tags are recognized as one such information pool as the tagging of items also tells something about the user. The hybridization of these information sources also plays an increasingly important role. In Liang et al. (2012), e.g., the authors combine a user-provided folksonomy and an expert-driven taxonomy to assess a user's opinion about an item and to make personalized recommendations. They show that by taking the expert's viewpoint into account, the accuracy of item recommendations can be further improved. Future work might aim to integrate tagging data with further information sources such as reviews or blogs.

Structured and Semi-Structured Data Sources In this chapter, we have mainly focused on user-generated content. In recent years, however, a number of other data sources have become available that can be leveraged in the recommendation process. Wikipedia, for example, is a typical example of a semi-structured public data source which was, for example, used as a means to improve a content-based recommendation system in Katz et al. (2011). Linked Open Data (LOD) and DBPedia, on the other hand, represent examples of more structured data sources that have been successfully used to build recommender systems (Otsiuni et al. 2013; Di Noia and Ostuni 2015). Besides these data sources that typically provide us additional information about item features or the relationship (e.g., similarity) between items, further knowledge sources and online services exist today that designers of

recommendation algorithms can use to design more semantic-aware approaches. These sources in particular include tools and mechanisms like WordNet that help us to automatically process and interpret the semantics and meaning of written or spoken language (de Gemmis et al. 2015).

Learning Techniques Finally, substantial advancements were also made in recent years in terms of algorithmic approaches to automatically derive item feature and user models and to generate personalized recommendations. Gabrilovich and Markovitch (2009), for example, used a method called "Explicit Semantic Analysis" (in analogy to Latent Semantic Analysis) to derive fine-grained item feature models based on natural language understanding and Wikipedia as a knowledge source. On the other hand, deep learning approaches based on neural networks have been recently applied for content-based recommendation in a variety of ways. Recent examples include the use of content features for music recommendation (van den Oord et al. 2013), the application of an embedding-based method for video recommendation on YouTube (Covington et al. 2016), or the exploitation of user reviews in a "collaborative deep learning" approach as proposed in Wang et al. (2015).

Cross-References

- ▶ Analysis and Mining of Tags, (Micro)Blogs, and Virtual Communities
- ▶ Folksonomies
- ▶ Human Behavior and Social Networks
- ▶ Tag Clouds

References

Cantador I, Bellogin A, Vallet D (2010) Content-based recommendation in social tagging systems. In: RecSys'10, Barcelona, pp 237–240

Covington P, Adams J, Sargin E (2016) Deep neural networks for YouTube recommendations. In: Proceedings of the 10th ACM conference on recommender systems. ACM, New York

Di Noia T, Ostuni VC (2015) Recommender systems and linked open data. In: Reasoning web. Web logic rules: 11th international summer school 2015. Springer International Publishing, pp 88–113

Dong R, O'Mahony MP, Schaal M, McCarthy K, Smyth B (2016) Combining similarity and sentiment in opinion mining for product recommendation. J Intell Inf Syst 46(2):285–312

Durao F, Dolog P (2010) Extending a hybrid tag-based recommender system with personalization. In: SAC'10, Sierre, pp 1723–1727

Firan CS, Nejdl W, Paiu R (2007) The benefit of using tag-based profiles. In: LA-WEB'07, Santiago de Chile, pp 32–41

Gabrilovich E, Markovitch S (2009) Wikipedia-based semantic interpretation for natural language processing. J Artif Intell Res (JAIR) 34:443–498

Gedikli F, Jannach D (2013) Improving recommendation accuracy based on item-specific tag preferences. ACM Trans Intell Syst Technol 4(1):1–19

Gedikli F, Ge M, Jannach D (2011) Understanding recommendations by reading the clouds. In: EC-Web'11, Toulouse, pp 196–208

Gedikli F, Jannach D, Ge M (2014) How should I explain? A comparison of different explanation types for recommender systems. Int J Hum-Comput Stud 72(4):367–382. Elsevier BV, 2014

de Gemmis M, Lops P, Semeraro G, Basile P (2008) Integrating tags in a semantic content-based recommender. In: RecSys'08, Lausanne, pp 163–170

de Gemmis M, Lops P, Musto C, Narducci F, Semeraro G (2015) Semantics-aware content-based recommender systems. In: Ricci et al. (ed) Recommender systems handbook, 2nd edn. Springer US, New York, NY, USA, pp 119–159

Golder SA, Huberman BA (2006) Usage patterns of collaborative tagging systems. J Inf Sci 32(2):198–208

Hotho A, Jäschke R, Schmitz C, Stumme G (2006) Information retrieval in folksonomies: search and ranking. In: ESWC'06, Budva, pp 411–426

http://vanderwal.net/folksonomy.html

http://www.delicious.com

http://www.flickr.com

Jannach D, Adomavicius G (2016) Recommendations with a purpose. In: Proceedings of the 10th ACM conference on Recommender Systems (RecSys 2016), Boston, pp 7–10

Jannach D, Zanker M, Felfernig A, Friedrich G (2010) Recommender systems – an introduction. Cambridge University Press, Leiden

Ji A-T, Yeon C, Kim H-N, Jo G-S (2007) Collaborative tagging in recommender systems. In: AUS-AI'07, Gold Coast, pp 377–386

Katz G, Ofek N, Shapira B, Rokach L, Shani G (2011) Using Wikipedia to boost collaborative filtering techniques. In: Proceedings of the fifth ACM conference on Recommender Systems (RecSys '11). ACM, New York, pp 285–288

Kubatz M, Gedikli F, Jannach D (2011) LocalRank – neighborhood-based, fast computation of tag recommendations. In: EC-Web'11, Toulouse, pp 258–269

Li X, Guo L, Zhao YE (2008) Tag-based social interest discovery. In: WWW'08, Beijing, pp 675–684
Liang H, Xu Y, Li Y (2012) Mining users' opinions based on item folksonomy and taxonomy for personalized recommender systems. In: ICDM'10, Sydney
Linden G, Smith B, York J (2003) Amazon.com recommendations: item-to-item collaborative filtering. IEEE Internet Comput 7(1):76–80
Noll MG, Meinel C (2007) Web search personalization via social bookmarking and tagging. In: ISWC'07/ASWC'07, Busan, pp 367–380
van den Oord A, Dieleman, S., Schrauwen, B. (2013) Deep content-based music recommendation, Proceedings advances in Neural Information Processing Systems (NIPS), pp 2643–2651
Ostuni VC, Di Noia T, Di Sciascio E, Mirizzi R (2013) Top-N recommendations from implicit feedback leveraging linked open data. In: Proceedings of the 7th ACM conference on Recommender systems (RecSys '13). ACM, New York, pp 85–92
Pitkow J, Schütze H, Cass T, Cooley R, Turnbull D, Edmonds A, Adar E, Breuel T (2002) Personalized search. Commun ACM 45(9):50–55
Rendle S, Schmidt-Thieme L (2010) Pairwise interaction tensor factorization for personalized tag recommendation. In: WSDM'10, New York, pp 81–90
Rendle S, Balby Marinho L, Nanopoulos A, Lars S-T (2009) Learning optimal ranking with tensor factorization for tag recommendation. In: SIGKDD'09, Paris, pp 727–736
Sen S, Vig J, Riedl JT (2009) Tagommenders: connecting users to items through tags. In: WWW'09, Madrid, pp 671–680
Seth A, Zhang J (2008) A social network based approach to personalized recommendation of participatory media content. In: ICWSM'08, Seattle
Shepitsen A, Gemmell J, Mobasher B, Burke R (2008) Personalized recommendation in social tagging systems using hierarchical clustering. In: RecSys'08, Lausanne, pp 259–266
Symeonidis P, Nanopoulos A, Manolopoulos Y (2008) Tag recommendations based on tensor dimensionality reduction. In: RecSys'08, Lausanne, pp 43–50
Tso-Sutter KHL, Marinho LB, Schmidt-Thieme L (2008) Tag-aware recommender systems by fusion of collaborative filtering algorithms. In: SAC'08, Fortaleza, pp 1995–1999
Vatturi PK, Geyer W, Dugan C, Muller M, Brownholtz B (2008) Tag-based filtering for personalized bookmark recommendations. In: CIKM'08, Napa Valley, pp 1395–1396
Vig J, Sen S, Riedl JT (2009) Tagsplanations: explaining recommendations using tags. In: IUI'09, Sanibel Island, pp 47–56
Vig J, Soukup M, Sen S, Riedl JT (2010) Tag expression: tagging with feeling. In: UIST'10, New York, pp 323–332
Wang H, Wang N, Yeung D-Y (2015) Collaborative deep learning for recommender systems. In: Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining, pp 1235–1244
Xu G, Gu Y, Dolog P, Zhang Y, Kitsuregawa M (2011a) Semrec: a semantic enhancement framework for tag based recommendation. In: AAAI'11, San Francisco, pp 1267–1272
Xu G, Gu Y, Zhang Y, Yang Z, Kitsuregawa M (2011b) Toast: a topic-oriented tag-based recommender system. In: WISE'11, Sydney, pp 158–171
Zanardi V, Capra L (2011) A scalable tag-based recommender system for new users of the Social Web. In: DEXA'11, Toulouse, pp 542–557
Zhen Y, Li W-J, Yeung D-Y (2009) Tagicofi: tag informed collaborative filtering. In: RecSys'09, New York, pp 69–76

Recommended Reading

Jannach D, Zanker M, Felfernig A, Friedrich G (2010) Recommender systems – an introduction. Cambridge University Press, Leiden
Ricci F, Rokach L, Shapira B, Kantor PB (eds) (2011) Recommender systems handbook. Springer, New York

Recommender Systems: Models and Techniques

Francesco Ricci
Faculty of Computer Science, Free University of Bozen-Bolzano, Bozen-Bolzano, Italy

Synonyms

Advisory systems; Recommendation systems

Glossary

Context	Situational factors influencing the evaluation of a user for an item
Experience	The interaction of a user with an item that is resulting in an evaluation
Evaluation Prediction	The system's prediction of the user's evaluation for an item
Information Filtering	Technique for providing only relevant information to a user

Item	Information content that can be recommended by a RS
Personalization	Providing a user with content adapted or suited to their needs and wants
Preferences	A structured representation of the user preferences for items
Recommendations	System's selected items that are suggested to a user
RSs	Recommender systems
Situation	Conditions under which an item is evaluated by a user
Tag	Metadata in the form of freely chosen keyword

Definition

RSs are information search and filtering tools that provide suggestions for items to be of use to a user. They have become common in a large number of Internet applications, helping users to make better choices while searching for news, music, vacations, or financial investments. RSs exploit data mining and information retrieval techniques to predict to what extent an item suits the user needs and wants and recommend those items with the largest predicted fit score.

Introduction

The explosive growth and variety of information available on the Web and the rapid introduction of new e-business and social services (buying products, product comparison, auction, forums, social networking, multimedia fruition) have created such a richness of choices that, instead of producing a benefit, this overabundance risks to backfire. While choice is good, more choice is not always better. Indeed, choice, with its implications of freedom, autonomy, and self-determination, can become excessive, creating a sense that freedom may come to be regarded as a kind of misery-inducing tyranny (Schwartz 2004). Moreover, if dozens of different types of jams are likely to confuse and paralyze a buyer, as it is illustrated in Schwartz (2004), thousands or even millions of songs are simply impossible to scan if the ultimate goal is just to play some of them.

Such a scenario motivated the introduction of recommender systems (RSs) (Ricci et al. 2011; Konstan and Riedl 2012). RSs are information search and filtering tools that provide suggestions for items to be of use to a user. They have become common in a large number of Internet applications, helping users to make better choices while searching for news, music, vacations, or financial investments. "Item" is the general term used to denote what the system recommends to its users, and a specific RS normally focuses on one type of items (e.g., movies or news). Accordingly, its core algorithmic component and its graphical user interface are customized to provide useful and effective suggestions for that specific type of items. Recommender systems play an important role in highly rated Internet sites, such as Amazon.com, YouTube, Netflix, Yahoo, Tripadvisor, Last.fm, and IMDb. More recently, social networks, such as Linkedin and Facebook, have introduced recommender systems to suggest groups to join or people to relate with.

In their simplest form, personalized recommendations are offered as customized lists of items. In performing this selection the system tries to predict what the most suitable products or services (items) are, based on the user's characteristics and preferences. In order to complete such a computational task, a RS must elicit from users such characteristics and preferences, either along the full history of previous interactions with the users or exploiting information entered by the users at the time the recommendation is requested. Moreover, such information either can be explicitly expressed, e.g., as ratings for products, or can be inferred by interpreting user actions. For instance, the navigation to a particular page can be interpreted as an implicit sign of preference for the items shown on that page.

The study of recommender systems is relatively new compared to research in other classical information system tools and techniques (e.g., databases or search engines). Recommender systems emerged as an independent research area in

the mid-1990s (Goldberg et al. 1992; Resnick et al. 1994), and it is still fast growing. Research works on RSs are published in major conferences on machine learning (ICML, KDD, NIPS), information retrieval (SIGIR, WISDM, CIKM), intelligent user interfaces (IUI), and personalization (UMAP). A specific ACM conference on recommender systems has been launched on 2007, and every year it attracts more and more submissions and attendees. In total, thousands of papers are published every year on this subject.

In this paper we provide a description of the general computational model of a recommender system. We aim at modeling in a compact but rigorous presentation the core functionality of a recommender system. We will decompose it into three fundamental tasks: user's preferences elicitation, prediction of user's evaluations for items, and recommendations generation and presentation. In the last section we will conclude this short article with a discussion of some challenges for RSs.

Recommendation Model and Techniques

A general computational model for recommender systems was previously described in Adomavicius and Tuzhilin (2005). In this model, a RS is defined as a machinery implementing a real-valued function defined on the product space of the users and items $r^*: U \times I \rightarrow \mathcal{R}$ that predicts how a pair consisting of a user $u \in U$ and an item $i \in I$ is mapped to the evaluation $r^*(u, i)$ of the user u for the item i. They call this number $r^*(u, i)$ the predicted "utility" of the item for the user. We prefer here not to use the term "utility," as in the RS literature no special assumption is made on the characteristics of the evaluation function, while a utility function does satisfy specific constraints. For that reason we call it "predicted evaluation." Then, having predicted evaluations of users for items, a RS recommends to a user u the items i with the largest predicted evaluations $r^*(u, i)$. Evaluations are called ratings of users for items in Collaborative Filtering RSs (see next section on Evaluation Prediction Techniques). A RS computes the prediction $r^*(u, i)$ on the base of a collection of observations: these are interactions between users and items, and they provide the system with information about the users' preferences. In many cases these interactions produce explicit evaluations performed by some users on some items. In some other cases, more complex types of relationships are observed, for instance, the relative preference of a user for an item when it is compared to another item.

In this survey we introduce a generalization of this two-dimensional model (Users × Items) that is inspired by the multidimensional model introduced in Adomavicius et al. (2005), and it is motivated by recent researches on social networks and tagging recommender systems (Marinho et al. 2011). Moreover, we delineate a model for RSs that decomposes its behavior into three fundamental tasks: preference elicitation, item evaluation prediction, and recommendation generation.

Preference Elicitation

The first task that a recommender system must implement is the elicitation of user preferences; this means, in RS terminology, that the system must be able to collect from users a set of evaluations for items.

More formally, let us assume that there exist three sets U, I, and M. U is the set of users, I is the set of items, and M is the set of possible situations under which the items can be experienced. For instance, M can be the user's location when the item, e.g., a restaurant, was searched and selected or the location of the user when a recommendation for a restaurant is required (more on this is discussed later on).

Then the experience of a user for an item is a quadruple $(u, i, m, r(u, i, m)) \in U \times I \times M \times \mathcal{R}$ where u is the user who interacted with item i in the situation m and evaluated the item $r(u, i, m) \in \mathcal{R}$. We use here the function notation $r(u, i, m)$ to stress that the evaluation is a function of the user, the item, and the situation. $\mathcal{R}$ is an evaluation scale containing the possible (ordered) evaluation values. Evaluations are commonly called ratings, and a popular evaluation scale is the finite set $\{1, 2, \ldots, 5\}$. This scale is used, for instance, by Amazon.com. But, different evaluation scales are

possible and have been used both in the scientific literature and in some deployed RSs; for instance, the simpler evaluation schema provided by positive ("like," +1) vs. negative ("not like," −1) evaluation is used in YouTube.com. It is interesting to note that rating scales are not neutral and they influence the user evaluation process (Kuflik et al. 2012) and consequently the RS behavior.

A recommender system, in order to generate recommendations, must first collect a set of experiences from the users in U, for items in I, in some situations M, i.e., to acquire a set $E(D) = \{(u, i, m, r(u, i, m)): (u, i, m) \in D \subset U \times I \times M\}$. The evaluation $r(u, i, m)$ can be collected either explicitly or implicitly. By "explicitly" we mean that the user is explicitly entering in some form her evaluation $r(u, i, m)$ on the presented item. In "implicit" models the user is not entering evaluations but is acting on the presented items, e.g., is watching a recommended video, or is browsing the presented information about an item. The system is then inferring the user evaluation (a value in $\mathcal{R}$) from the user action. We first discuss the "explicit" approach and then the "implicit" one.

Many recommender systems allow users to "explicitly" evaluate (rate) items as they encounter them while interacting with the system. In addition, many RSs explicitly request the user to evaluate a certain number of items (e.g., 20 books) before providing recommendations (for new books). This may happen in the sign up stage, i.e., when the user registers to the system and obtains the credential to access it. Another popular approach for collecting evaluations is to let the user to "correct" the system predictions by entering evaluations for items that have been recommended, hence possibly fixing erroneous predictions. In this case the system may learn that some of the recommended items are not good options, i.e., the user may enter low evaluations for them.

In more sophisticated approaches the system may implement a precise preference elicitation strategy by applying active learning techniques (Rubens et al. 2011). So, for instance, the system may identify the most popular items and request the user to rate them, with the objective of maximizing the probability that the user knows these items and can really evaluate them. Or, in a completely different approach, the system may ask the user to evaluate the items that have received so far the most diverse evaluations, since the opinion of the user on these items can better reveal the specific users' preferences.

In "implicit" feedback approaches the user is not directly entering evaluations for items by choosing values from the evaluation scale $\mathcal{R}$ considered by the system. So, for instance, the user is not entering a star-based score for the books she has read. The underlying assumption is that the user may not want to spend time for this task; hence, the system must infer the evaluations in the target $\mathcal{R}$ scale from another measure in another scale $\mathcal{R}'$. For instance, in the music recommender system presented in Moling et al. (2012) the system is measuring the percentage of a suggested track that is actually listened to by the user to decide what type of track to recommend next. Another popular implicit evaluation scale is the number of times a user visited the item (played a track or browsed a page). Yet another example of the "implicit" approach is offered by systems that do not ask the user to evaluate individual items, but allow them to compare two items or to criticize them (McGinty and Reilly 2011). In this case the system first presents some recommended items. These are primarily generated for letting the user to specify the characteristics of the preferred items. In response, the user can inform the system that the presented items do not completely conform to her preferences and select one item that is almost good but is still lacking a preferred feature (e.g., it should be cheaper). In these cases the system uses the user input, which is composed by the selected item and the "critique," to update the evaluation prediction function for that user $r^*(u, \cdot, \cdot)$.

Item Evaluation Prediction

The second task, item evaluation prediction, uses a data set of user-provided evaluations to predict new evaluations. In fact, this is the most important task of a recommender system: exploiting a data set of experiences, its background knowledge, $E(D) = \{(u, i, m, r(u, i, m)): (u, i, m) \in D \subset U \times I \times M\}$ to generate predictions of the user evaluations for

other experiences E^* (D^*) = {(u, i, m, $r^*(u, i, m)$): (u, i, m) $\in D^* \subset U \times I \times M$}, where $r^*(\cdot, \cdot, \cdot)$ is the evaluation prediction function that is estimated with a specific recommendation technology. In the next section, we will present some techniques, e.g., collaborative filtering or content-based, which have been introduced to compute the evaluation prediction function $r^*(\cdot, \cdot, \cdot)$. D^* is the set containing the user, item, and situation combinations for which the recommender system can generate evaluation predictions. D^* is disjoint from D and may be equal to $(U \times I \times M) \setminus D$ or smaller. It is smaller when the RS is not able to generate evaluation predictions for all the items and situations combinations that the user has not evaluated yet. In fact, recommender systems find it difficult, for instance, to make recommendations for new users and new items. These are users who never entered any evaluation in the system or items that were never evaluated by any user (users and items not present in D) (Ricci et al. 2011; Konstan and Riedl 2012).

We note that I, U, and M are sets of objects and may, or may not, have an internal structure, i.e., they may have features. For instance, in plain collaborative filtering systems, which will be described later on, items' and users' features, even if available, are not exploited, and the situation space is ignored. For that reason the evaluation function is modeled as a matrix $\mathbf{R} = [r_{u,i}]_{m \times n}$, where u and i are simple indexes ranging from 1 to m and n, respectively, and $r_{u,i} \in \mathcal{R}$. Whereas, in content-based systems, items have an internal structure and are described with features. Items' features are used to generate user-specific classifiers that can predict the user evaluation for (unseen) items (Lops et al. 2011).

It is worth noting that some recommender systems do not collect user evaluations for items in order to make predictions, i.e., they can make recommendations even though $E(D)$ is empty. For instance, case-based recommender systems let the user to enter a partial description of the preferred item q, as query, and then, using this input, they generate predictions $r^*(u, i, q)$ of the user evaluation for item i. They accomplish this task by exploiting the similarity of q and i, based on some description of i. Hence, in CBR RSs, the situations set M is the space of all possible user queries (Bridge et al. 2006).

Evaluation Prediction Techniques

In order to implement the item evaluation prediction function, RSs can exploit a range of techniques. This has been the major topic of research in RSs. We will here briefly indicate the most important types of techniques, referring the interested reader to Ricci et al. (2011) for more examples, references, and details.

Recommendation techniques vary in terms of the addressed domain, the knowledge used, and the recommendation algorithm, i.e., essentially how the item evaluation prediction is actually computed. We provide here an overview of the different types of RS techniques by quoting a taxonomy introduced by Burke (2007) that has become classical for distinguishing between recommender systems and referring to them. Burke (2007) lists six different types of recommendation approaches.

Content-Based

The system implements for each user a "classifier" that learns to evaluate (classify) higher the items that are similar to the ones that the user evaluated higher in the past. The similarity of items, or more in general the item classification rule, is calculated based on the features associated with the compared items. For example, if a user has systematically positively rated movies that belong to the action genre, then the system can learn that other movies from this genre should have a high value for that user (Lops et al. 2011).

Collaborative Filtering

The simplest and original implementation of this approach predicts that the active user, i.e., the user asking for recommendations, will evaluate higher the items that other users with similar tastes liked in the past (Desrosiers and Karypis 2011). The similarity in taste of two users is calculated based on the similarity of the evaluations' history of the users. Collaborative filtering is probably the most popular and widely implemented technique in RSs. The latest approaches to CF use latent factor models, such as matrix factorization (e.g.,

using Singular Value Decomposition, SVD). These methods map both items and users to the same latent factor space. Then the predicted evaluation of a user for an item is basically computed by the dot multiplication of their representative vectors, which gives a kind of similarity between the user and the item in this common representation space (Koren and Bell 2011).

Demographic

These techniques predict item evaluations based on the demographic profile of the user. The assumption is that different recommendations should be generated for different demographic niches. Many Web sites adopt simple and effective personalization solutions based on demographics. For example, users are dispatched to particular Web sites based on their language or country. Or suggestions may be customized according to the age of the user.

Knowledge-Based

Knowledge-based systems predict item evaluations based on specific domain knowledge about how certain item features meet user's needs and preferences and ultimately how the item is useful for the user. Notable knowledge-based recommender systems are case-based (Bridge et al. 2006). In these systems a similarity function estimates how much the user needs (problem description) match the recommendations (solutions of the problem). Here the similarity score can be directly interpreted as the predicted item evaluation of the user. Another group of knowledge-based systems uses constraints, to represent user preferences and to find relevant items (Jannach et al. 2010).

Community-Based

In this type of systems, item evaluation predictions are based on the preferences of the user's friends. Evidence suggests that people tend to rely more on recommendations from their friends than on recommendations from similar but anonymous individuals. This observation, combined with the growing popularity of open social networks, is generating a rising interest in community-based systems or social recommender systems (Golbeck 2006). This type of RS techniques acquires and exploits information about the social relations of the users and the preferences of the user's friends. The item evaluation predictions are based on ratings that were provided by the user's friends.

Hybrid Recommender Systems

These RSs are based on the combination of the abovementioned techniques. A hybrid system combining techniques A and B tries to use the advantages of A to fix the disadvantages of B. For instance, CF methods suffer from new-item problems, i.e., they cannot generate evaluation predictions for items that have no ratings. This does not limit content-based approaches since the prediction for new items is based on their description (features) that are typically easily available. Given two (or more) basic RS techniques, several ways have been proposed for combining them to create a new hybrid system (see Burke 2007 for the precise descriptions).

Recommendation Generation

RSs, after having generated evaluation predictions for items, can generate recommendations, i.e., absolve their primary role. The classical and common approach for recommendation generation is to recommend to user u in situation m the N items i that have the largest predicted evaluation in that situation, i.e., the set $TopN(u, m) \subset I$, where $|TopN(u, m)| = N$, and if $i \in TopN(u, m)$, then $r^*(u, i, m) \geq r^*(u, j, m)$, for all $j \notin TopN(u, m)$ (Adomavicius and Tuzhilin 2005). N is normally a small value, such as 5 or 10. Reducing the number of recommendations is essential to address the main goal of a recommender system, as we mentioned in the Introduction, namely, to filter irrelevant items and simplify the user's decision-making process.

Similarly, the system can rank all the items for which the recommender can generate an evaluation prediction and present to the user the ranked list of these items, ordered by decreasing value of the predicted evaluation $r^*(u, i, m)$. It is worth noting that this ranked list is not generally equal to the full set of items I, since, as we mentioned above, the RS may not be able to generate an

evaluation prediction for all the items (in all the situations). Hence, in practice this ranked list contains the $TopN(u, m)$ items for a large value of N.

In fact, there is a growing understanding that this apparently obvious design choice may not be the most appropriate in many cases. For instance, the top N items may be all very similar; hence, it would be more useful to introduce in the recommendation list those items, which may have a lower predicted evaluation but are, all together, providing a more useful information to the user. The essential point that we raise here is that, while the item evaluation prediction function estimates to what extent the user likes an item, the user decides what item to select by browsing the recommendation list. Hence, items presented in the recommendation list must fulfil two, possibly conflicting, criteria: be interesting to the user and help the user to make a decision. As the diversity issue points out, items that can better help the user to make a decision may not be just the best top five items that he may select.

Another related issue is pertaining to the information that is provided together with the recommendations. In practice this has been dealt by including explanations for the recommendations (Tintarev and Masthoff 2012). Explanations may be directed to enhance the transparency or the scrutability of the system, i.e., giving to the users hints about the system internal behavior. Or they may increase the trust, the persuasiveness, and the subjective satisfaction of the user. Or ultimately explanations can improve the efficiency and effectiveness of the decision-making process supported by the recommender.

Key Applications

As we mentioned above, the illustrated recommender system model focuses on three types of entities: users U, items I, and situations M. In the following sections we will illustrate three key applications of this model, where the situation space is representing, the context, or the group of users, or a tag. We must stress that the simpler and more popular model, which is described in Adomavicius and Tuzhilin (2005), does not consider any situation space M. In that case the recommendations are generated for a user, independently from any other additional information that may specify the recommendation situation.

Context-Aware Recommender Systems

The first application of the model described in the previous section refers to context-aware recommender systems (Adomavicius et al. 2011). In this case M contains the possible alternative contextual situations that may be concurrent with the experience that a user makes of an item and can have an impact on the item evaluation. The goal is to make predictions of user evaluations for items in a particular target contextual condition m.

For instance, Baltrunas et al. (2012) describe a place of interests (POIs) recommender system for the tourism domain, where 14 contextual factors are considered. To mention some examples, there are factors that describe the time of the travel, the weather condition, the mood of the user, or the type of group that is accompanying the traveler. We refer the reader to Baltrunas et al. (2012) for a detailed description of these contextual factors. For each factor a finite set of possible values is defined. For instance, the weather factor can take the values: snowing, clear sky, sunny, and rainy. Hence, in this recommender system M is the space of all the possible combinations of the values for these 14 contextual factors: $M = F_1 \times \cdots F_{14}$, where, for instance, $F_9 = \{\text{snowing, clear sky, sunny, rainy}\}$ is the weather factor that we mentioned above. This system uses a collection of users' evaluations for a collection of places of interests in Bolzano (items), in different contextual situations, to predict the users' evaluations for POIs not yet experienced in some possibly new contextual situations.

Context-aware RSs are now an active research area and we refer the reader to Adomavicius et al. (2011) for more examples and techniques. The major technical difficulties are related to the following: understanding the impact of the contextual factors on the personalization process; selecting (dynamically) the right factors, i.e., relevant in a particular personalization task; obtaining sufficient and reliable data describing the user preferences in context; and embedding the contextual information in a more classical recommendation computation technique.

Group Recommender Systems

The second application refers to group recommender systems (Jameson and Smyth 2007). In this case M represents the possible groups of users that u may belong to, and the ultimate system goal is to offer the same set of recommendations for items to the users belonging to a group. The underlying assumption is that the items will be experienced together, e.g., in a travel recommender system, the users will travel together to the recommended place. The data set of the users' experiences $E(D)$ in this case contains quadruples of the type $(u, i, g, r(u, i, g))$, where g is a subset of users in U that contains u, and $r(u, i, g)$ is the evaluation provided by the user u, for item i, when the item was experienced together with the other users in g. The idea is that the user evaluation is influenced by the presence of other users (Masthoff 2011), i.e., $r(u, i, g)$ is in principle different from $r(u, i, h)$ if $h \neq g$.

A group recommender system with such background knowledge must predict the evaluation of u for other items, let us say i' when she is together with the users in g'. It is worth noting that no group recommender system is actually making predictions for the user evaluations as a function of the group g the user belongs to. Conversely, the current leading approach is to first predict the individual evaluations independently from the group, using a standard two-dimensional model, i.e., neglecting the situation in M, which means that $r^*(u, i, h) = r^*(u, i, g) = r^*(u, i)$, for all groups $h, g \in M$. Then, in a second step, group RSs compute the "group evaluation prediction" for an item by aggregating the various evaluation predictions for that item for the groups members. So, for instance, if the average aggregation method is used, the predicted evaluation of the group g for the item i is $AVG_{u \in g} \{ r^*(u, i)\}$, and the items with the largest group aggregated evaluations are recommended.

Other approaches, instead of aggregating evaluation predictions, aggregate the input evaluations of the users u belonging to a group g, hence generating (fictitious) group evaluations $r(g, i, g)$. These group evaluations, which constitute the group model, are again computed by, for instance, averaging the evaluations of the users in the group for a given item. Then, the group evaluation predictions are computed for these fictitious users considering them as normal users. One can describe this approach in the model proposed in this article, by introducing new fictitious users that represent groups g, and then the evaluation prediction function for a group g is $r^*(g, \cdot, g)$.

The group recommendation application is stressing once more the fundamental, but so far neglected, difference between the evaluation prediction and the recommendation generation tasks of a recommender system. As we mentioned above, the best recommendations may not be for the items having the largest predicted evaluation. For group recommendations, a better approach is instead to really try to predict $r(u, i, g)$, i.e., the evaluation of u for item i when experienced together with the other users in g. Then the system must compute recommendations for u when she is together with the users in g not simply by taking the highest predicted items, but trying to generate a recommendation list that is really useful for the group, e.g., by combining items that would help the group to make a joint and agreed decision.

Tag-Based Recommender Systems

The last application refers to social tagging recommender systems (Marinho et al. 2011). In social tagging systems, such as Delicious, BibSonomy, and Last.fm, users can search and browse the items managed by the system, which in these three examples are bookmarks, bibliographic references, and music, respectively. But, in addition to this basic functionality, users can tag the items with tags, i.e., metadata in the form of freely chosen keywords. On top of these systems, various kinds of recommendations are possible. One can recommend items to user, but also tags to be assigned by a user to an item, or even users to users. In tag-based recommender systems, M, the space of situations, is a tag vocabulary V. An evaluation $r(u, i, t)$ in these systems is taking values just in the set $\{1\}$, where $r(u, i, t) = 1$ means that the user u has tagged the item i with the tag t. Hence, in this scenario, a set of collected experiences $E(D) = \{(u, i, t, 1): (u, i, t) \in D \subset U \times I \times V\}$ represents tag assignments performed by users to items. This set of

experiences, actually considering only the triples (u, i, t), i.e., discarding the redundant 1, is called in social tag systems a Folksonomy. So, given a Folksonomy, the evaluation prediction task of a tag-based recommender system can be specialized by saying that it predicts whether a user u will assign the tag t to the item i, or in other words, if the triple (u, i, t) not yet included in the Folksonomy should be added to it.

As we mentioned above, while the basic evaluation prediction task in this case is predicting if $r(u, i, t)$ is 1, i.e., if a user will assign a tag to an item, various specific recommendation generation tasks have been considered. The two more popular are to recommend a set of items to a user and to recommend a set of tags to a user for tagging an item. The first one is the classical (two-dimensional) recommendation task of a RS, i.e., where the recommendations are not supposed to be dependent on the tag that the user is giving to the target item (the situation). In fact, tag assignments are used to generate item recommendations, e.g., by using tag assignments to identify similar items (tagged in a similar way) or similar users (tagging common items) in collaborative filtering techniques. But in this case, the recommendations offered to a user are not depending on the tags that the user may have selected for the recommended item. In this case, the original three-dimensional matrix of observed tag assignments (experiences) $r(u, i, t) = 1$ can be projected into the two classical two-dimension space of users and items evaluations $r(u, i)$. This is performed by simply assigning to $r(u, i)$ the value 1 if there exists a tag t such that $r(u, i, t) = 1$. Then, on this two-dimensional space of $U \times I$, standard recommendation techniques, such as collaborative filtering, can be applied to compute evaluation predictions (i.e., if a user likes or not the items) and generate the final recommendation sets.

Conversely, if the goal is to recommend tags to a user for tagging an item, one must consider the original three-dimensional model. After having identified the triples (u, i, t) that are predicted, i.e., such that $r^*(u, i, t) = 1$, the system simply recommends the tag t for tagging the item i to the user u. In practice, prediction functions are scoring the triples not yet observed, e.g., by assigning a predicted evaluation in [0, 1], and then the N tags, for a given pair (u, i), that are predicted to score higher are recommended to the user u for tagging the item i. A large number of techniques have been proposed to compute this score, and we refer to Marinho et al. (2011) for a survey in this fast-growing research area.

Future Directions

The research on RSs is still very active, and numerous issues and challenges are still open. We want to list here some of them, with the obvious caveat that this list cannot be complete and is influenced by our personal vision. The reader is also referred to Konstan and Riedl (2012) for another discussion on the future of recommender systems.

Group Recommenders

Group recommenders deal with situations when it would be good if the system could recommend information or items that are relevant to a group of users rather than to an individual (Jameson and Smyth 2007). For instance, a RS may select television programs for a group to view or a sequence of songs to listen to, based on preference models of all group members (Masthoff 2011). Recommending to groups is clearly more complicated than recommending to individuals. Assuming that we know precisely what is good for individual users, the issue is how to combine individual user preferences or aggregate recommendations for the group members into group-specific recommendations (Jameson and Smyth 2007). Even if several techniques have been proposed so far (Masthoff 2011), very few experiments have been conducted in live-user studies (Senot et al. 2010), and it is challenging to define measures for assessing the quality of group recommendations.

Proactive Recommender Systems

Proactive recommenders can decide to push recommendations even if not explicitly requested. The largest majority of the recommender systems developed so far follow the "pull" model, where

the user originates the request for a recommendation. In the scenarios emerging today, where computers are ubiquitous and users are always connected, it seems natural to imagine that a RS can detect needs even if they are not explicitly stated by the user with a request. In this scenario, the system therefore must predict what to recommend but also when and how to "push" its recommendations. By accurately estimating when the RS can become proactive without being perceived as disturbing, the perceived utility of the recommendations may greatly increase.

Active Learning

RSs need to actively look for new data during the operational life (Rubens et al. 2011) (Active Learning). This issue is normally neglected on the assumption that there is not much space for controlling what data (e.g., ratings) the system can collect, because these decisions are autonomously taken by the users when visiting the system. Actually, a RS provokes the users with its recommendations. In fact, many systems (e.g., MovieLens.org) explicitly ask for user ratings during the recommendation process. Hence, by tuning the process, users can be pushed to enter into the system a range of different information. Specifically, they can be requested to rate particular items, because the knowledge of the user's opinions about these items is estimated as beneficial for improving a particular aspect of the system performance, e.g., in order to generate more diverse recommendations or just to improve the prediction accuracy. Some recent works have addressed these issues (Harpale and Yang 2008; Rashid et al. 2008), but more research activity is required to assess the proposed techniques in live-user studies and to design more adaptive solutions to the dynamics of the data managed by the system (Elahi et al. 2011; Golbandi et al. 2011).

Privacy Preserving

RSs exploit user data to generate personalized recommendations. In the attempt to build increasingly better recommendations, they collect as much user data as possible. This can clearly have a negative impact on the privacy of the users, and the users may start feeling that the system knows too much about their true preferences (Kobsa 2008). Therefore, there is a need to design solutions that will parsimoniously, sensibly, and cooperatively collect user data. At the same time these solutions will ensure that the acquired knowledge about the users cannot be freely accessed by malicious users.

Diversity

Assuring the diversity of the items recommended to a target user is an important feature of a recommender system (Smyth and McClave 2001). For instance, it is more likely that the user will find a suitable item in a recommendation list, if there is a certain degree of diversity among the recommendations. There is a limited value in having perfect recommendations for a restricted type of products, unless the user has expressed a narrow set of preferences. There are many situations, especially in the early stage of a recommendation process, in which the users want to explore new and diverse directions. In such cases, the user is using the recommender as a knowledge discovery tool. The research on this topic is still in an early stage and is now trying to characterize the nature of this "diversity" (Vargas and Castells 2011), i.e., whether the system must produce diversity among different recommendation sessions, or within a session (Adomavicius and Kwon 2012), and how to achieve simultaneously diversity and accuracy of the recommendations (Vargas and Castells 2011).

Generic User Models

Generic user models (Kobsa 2007) and cross-domain recommender systems are able to mediate user data (item evaluations) through different systems and application domains (Berkovsky et al. 2008). Using generic user model techniques, a single RS can produce recommendations about a variety of items. This is normally not possible for a traditional RS which can combine more techniques in a hybrid approach, but cannot easily benefit from the user preferences collected in one domain to generate recommendations in a different one. Solutions to this problem may further push the adoption

of personalized mobile recommender systems, running on the user personal communication device and offering ubiquitous support in many user's activities (Ricci 2011).

Sequential Recommendations

Recommender Systems may optimize a sequence of recommendations (Shani et al. 2005; Baccigalupo and Plaza 2006), for instance, a sequence of songs broadcast by a personalized radio channel. Sequential recommendations may be generated by systems that manage a structured dialogue between the user and the recommender. These systems are called conversational and have emerged in the attempt to improve the quality of the recommendations that are normally offered by simpler systems based on a onetime request/response. Conversational RSs can be further improved by implementing learning capabilities that can optimize not only the items that are recommended at each conversational step but also how the dialogue between the user and the system must unfold in all possible situations (Mahmood et al. 2009).

Robust Recommender Systems

Finally, the topic of robust recommender systems has become a major issue in the past few years. New research activities have focused on algorithms designed to generate more robust recommendations, i.e., recommendations that are harder for malicious users to influence. In fact, collaborative recommender systems are dependent on the goodwill of their users, i.e., there is an implicit assumption that users will interact with the system with the aim of getting good recommendations for themselves while providing useful data for their neighbors. However, users might have a range of purposes in interacting with RSs, and in some cases, these purposes may be opposed to those of the system owner or those of the majority of its user population. Namely, these users may want to damage the Web site that is hosting the recommender or to bias the recommendations, e.g., to score some items better or worse, rather than to arrive at a fair evaluation (Burke et al. 2011).

Cross-References

- ▶ Collective Intelligence: Overview
- ▶ Computational Trust Models
- ▶ Data Mining
- ▶ Distance and Similarity Measures
- ▶ E-Commerce and Internet Business
- ▶ Folksonomies
- ▶ Group Representation and Profiling
- ▶ Matrix Algebra, Basics of
- ▶ Matrix Decomposition
- ▶ Privacy and Disclosure in a Social Networking Community
- ▶ Recommender Systems using Social Network Analysis: Challenges and Future Trends
- ▶ Social Media
- ▶ Social Recommendation in Dynamic Networks
- ▶ Trust in Social Networks

References

Adomavicius G, Kwon Y (2012) Improving aggregate recommendation diversity using ranking-based techniques. IEEE Trans Knowl Data Eng 24(5):896–911

Adomavicius G, Tuzhilin A (2005) Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. IEEE Trans Knowl Data Eng 17(6):734–749

Adomavicius G, Sankaranarayanan R, Sen S, Tuzhilin A (2005) Incorporating contextual information in recommender systems using a multidimensional approach. ACM Trans Inf Syst 23(1):103–145

Adomavicius G, Mobasher B, Ricci F, Tuzhilin A (2011) Context-aware recommender systems. AI Mag 32(3):67–80

Baccigalupo C, Plaza E (2006) Case-based sequential ordering of songs for playlist recommendation. In: Roth-Berghofer T, Göker MH, Güvenir HA (eds) Advances in case-based reasoning, Proceedings of the 8th European conference on case-based reasoning, ECCBR 2006, Fethiye. Lecture notes in computer science, vol 4106. Springer, pp 286–300

Baltrunas L, Ludwig B, Peer S, Ricci F (2012) Context relevance assessment and exploitation in mobile recommender systems. Personal Ubiquitous Comput 16(5):507–526

Berkovsky S, Kuflik T, Ricci F (2008) Mediation of user models for enhanced personalization in recommender systems. User Model User Adapt Interact 18(3):245–286

Bridge D, Göker M, McGinty L, Smyth B (2006) Case-based recommender systems. Knowl Eng Rev 20(3):315–320

Burke R (2007) Hybrid web recommender systems. In: The adaptive web. Springer, Berlin/Heidelberg, pp 377–408

Burke RD, O'Mahony MP, Hurley NJ (2011) Robust collaborative recommendation. In: Ricci F, Rokach L, Shapira B (eds) Recommender systems handbook. Springer, New York, pp 805–835

Desrosiers C, Karypis G (2011) A comprehensive survey of neighborhood-based recommendation methods. In: Ricci F, Rokach L, Shapira B, Kantor PB (eds) Recommender systems handbook. Springer, New York, pp 107–144

Elahi M, Repsys V, Ricci F (2011) Rating elicitation strategies for collaborative filtering. In: Proceedings of the E-commerce and web technologies – 12th international conference, EC-Web, Toulouse, 30 Aug–1 Sept 2011. Lecture notes in business information processing, vol 85. Springer, pp 160–171

Golbandi N, Koren Y, Lempel R (2011) Adaptive bootstrapping of recommender systems using decision trees. In: Proceedings of the forth international conference on web search and web data mining, WSDM, Hong Kong, 9–12 Feb 2011, pp 595–604

Golbeck J (2006) Generating predictive movie recommendations from trust in social networks. In: Proceedings of the trust management, 4th international conference, iTrust, Pisa, 16–19 May 2006, pp 93–104

Goldberg D, Nichols D, Oki BM, Terry D (1992) Using collaborative filtering to weave an information tapestry. Commun ACM 35(12):61–70

Harpale A, Yang Y (2008) Personalized active learning for collaborative filtering. In: Proceedings of the 31st annual international ACM SIGIR conference on research and development in information retrieval, SIGIR, Singapore, 20–24 July 2008. ACM, pp 91–98

Jameson A, Smyth B (2007) Recommendation to groups. In: The adaptive web. Springer, Berlin, pp 596–627

Jannach D, Zanker M, Felfernig A, Friedrich G (2010) Recommender systems: an introduction. Cambridge University Press, New York

Kobsa A (2007) Generic user modeling systems. In: Brusilovsky P, Kobsa A, Nejdl W (eds) The adaptive web, Lecture notes in computer science, vol 4321. Springer, Berlin, pp 136–154

Kobsa A (2008) Privacy-enhanced personalization. In: Proceedings of the twenty-first international Florida artificial intelligence research society conference, Coconut Grove, 15–17 May 2008. AAAI, p 10

Konstan JA, Riedl J (2012) Recommender systems: from algorithms to user experience. User Model User Adapt Interact 22(1–2):101–123

Koren Y, Bell R (2011) Advances in collaborative filtering. In: Ricci F, Rokach L, Shapira B (eds) Recommender systems handbook. Springer, New York, pp 145–186

Kuflik T, Wecker AJ, Cena F, Gena C (2012) Evaluating rating scales personality. In: Proceedings of the user modeling, adaptation, and personalization – 20th international conference, UMAP, Montreal, 16–20 July 2012, pp 310–315

Lops P, de Gemmis M, Semeraro G (2011) Content-based recommender systems: state of the art and trends. In: Ricci F, Rokach L, Shapira B, Kantor PB (eds) Recommender systems handbook. Springer, New York, pp 73–105

Mahmood T, Ricci F, Venturini A (2009) Improving recommendation effectiveness by adapting the dialogue strategy in online travel planning. Int J Inf Technol Tour 11(4):285–302

Marinho LB, Nanopoulos A, Schmidt-Thieme L, Jäschke R, Hotho A, Stumme G, Symeonidis P (2011) Social tagging recommender systems. In: Ricci F, Rokach L, Shapira B, Kantor PB (eds) Recommender systems handbook. Springer, New York, pp 215–644

Masthoff J (2011) Group recommender systems: combining individual models. In: Ricci F, Rokach L, Shapira B (eds) Recommender systems handbook. Springer, New York, pp 677–702

McGinty L, Reilly J (2011) On the evolution of critiquing recommenders. In: Ricci F, Rokach L, Shapira B (eds) Recommender systems handbook. Springer, New York, pp 419–453

Moling O, Baltrunas L, Ricci F (2012) Optimal radio channel recommendations with explicit and implicit feedback. In: RecSys'12: Proceedings of the 2012 ACM conference on recommender systems, Dublin, pp 75–82

Rashid AM, Karypis G, Riedl J (2008) Learning preferences of new users in recommender systems: an information theoretic approach. SIGKDD Explor Newsl 10:90–100

Resnick P, Iacovou N, Suchak M, Bergstrom P, Riedl J (1994) Grouplens: an open architecture for collaborative filtering of netnews. In: Proceedings ACM conference on computer-supported cooperative work, Chapel Hill, pp 175–186

Ricci F (2011) Mobile recommender systems. Int J Inf Technol Tour 12(3):205–231

Ricci F, Rokach L, Shapira B, Kantor PB (2011) Recommender systems handbook. Springer, New York

Rubens N, Kaplan D, Sugiyama M (2011) Active learning in recommender systems. In: Ricci F, Rokach L, Shapira B, Kantor PB (eds) Recommender systems handbook. Springer, New York, pp 735–767

Schwartz B (2004) The paradox of choice. ECCO, New York

Senot C, Kostadinov D, Bouzid M, Picault J, Aghasaryan A, Bernier C (2010) Analysis of strategies for building group profiles. In: Proceedings of the user modeling, adaptation, and personalization, 18th international conference, UMAP, Big Island, 20–24 June 2010. Springer, pp 40–51

Shani G, Heckerman D, Brafman RI (2005) An mdp-based recommender system. J Mach Learn Res 6:1265–1295

Smyth B, McClave P (2001) Similarity vs diversity. In: Case-based reasoning research and development,

Proceedings of the 4th international conference on case-based reasoning, ICCBR 2001, Vancouver. Springer

Tintarev N, Masthoff J (2012) Evaluating the effectiveness of explanations for recommender systems – methodological issues and empirical studies on the impact of personalization. User Model User Adapt Interact 22(4–5):399–439

Vargas S, Castells P (2011) Rank and relevance in novelty and diversity metrics for recommender systems. In: Proceedings of the 2011 ACM conference on recommender systems, RecSys'11, Chicago, 23–27 Oct 2011. ACM, pp 109–116

Reconnaissance

▶ Reconnaissance and Social Engineering Risks as Effects of Social Networking

Reconnaissance and Social Engineering Risks as Effects of Social Networking

Katina Michael
School of Information Systems and Technology, University of Wollongong, Wollongong, NSW, Australia

Synonyms

Footprinting; Hacker; Penetration testing; Reconnaissance; Risk; Security; Self-disclosure; Social engineering; Social media; Social reconnaissance; Vulnerabilities

Glossary

Social reconnaissance	A preliminary paper-based or electronic web-based survey to gain personal information about a member or group in your community of interest. The member may be an individual friend or foe, a corporation, or the government
Social engineering	With respect to security, is the art of the manipulation of people while purporting to be someone other than your true self, thus duping them into performing actions or providing secret information
Data leakage	The deliberate or accidental outflow of private data from the corporation to the outside world, in a physical or virtual form
Online social networking	An online social network is a site that allows for the building of social networks among people who share common interests
Malware	The generic term for software that has a malicious purpose. Can take the form of a virus, worm, Trojan horse, and spyware

Lead

> . . . not what goes into the mouth defiles a man, but what comes out of the mouth, this defiles a man."

Matthew 15:11 (RSV)

Introduction

For decades we have been concerned with how to stop viruses and worms from penetrating organizations and how to keep hackers out of organizations by luring them toward unsuspecting honeypots. In the mid-1990s Kevin Mitnick's "dark-side" hacking demonstrated, and possibly even glamorized (Mitnick and Simon 2002), the need for organizations to invest in security equipment like intrusion detection systems and firewalls, at every level from perimeter to internal demilitarized zones (Mitnick and Simon 2005).

In the late 1990s, there was a wave of security attacks which stifled worker productivity. During these unexpected outages, employees would take long breaks queuing at the coffee machine, spend time cleaning their desk, and try to look busy shuffling paper in their in- and out-trays. It was clear by the downtime caused by malware hitting servers worldwide that corporations had begun to rely on intranets for content and workflow management so much and that employees would be left with very little to do when they were not connected. Nowadays, everything is online with respect to the service industry, and there is a known vulnerability in the requirement to be always connected. For example, you can cripple an organization if you take away their ability to accept electronic payments online, or render their content management system inaccessible due to denial of service attacks, or hack into a company's webpage.

When the "Melissa" virus caught employees unaware in 1999, and was then followed by the "Explorer.zip" worm in the same year, public folders had Microsoft Office files either deleted or corrupted. At the time, anecdotal stories indicated that some people (even whole groups) lost several weeks of work, after falling victim to the worm that had attacked their hard drive. This led many to seek backup copies of their files, only to find that the backups themselves were not activated (Michael 2003).

The moral of the story is that for decades we have been preoccupied with stopping data (executables, spam, false log-in attempts, and the like) from entering the organization when the real problem since the rise of broadband networks, 3G wireless, and more recently social media has been how to stop data from going out of the organization. While this sounds paradoxical, the major concern is not what data traffic comes into an organization, but what goes out of an organization that matters. We have become our own worst enemy when it comes to security in this online-everything world we live in.

In short, data leakage is responsible for most corporate damage, such as the loss of competitive information. You can secure a bucket and make it water tight, put a lid on it, even put a lock on the lid, but if that bucket has even a single tiny hole, its contents will leak out and cause spillage. Such is the dilemma of information security today – while we have become more aware of how to block out unwanted data, the greatest risk to our organization is that which leaves the organization – through the network, through storage devices, and via an employees' online personal blog, even the spoken word. It is indeed what most security experts call the "human" factor (Michael 2008).

Reconnaissance of Social Networks for Social Engineering

Social Networking

The Millennials, also known as Gen Ys, have been the subject of great discussion by commentators. If we are to believe what researchers say about Gen Ys, then it is this generation that has voluntarily gone public with private data. This generation, propelled by advancements in broadband wireless, 3G mobiles, and cloud computing, is always connected and always sharing their sentiments and cannot get enough of the new apps. They are allegedly "transparent" with most of their data exchanges. Generally, Gen Ys do not think deeply about where the information they publish is stored, and they are focused on convenience solutions that benefit them with the least amount of rework required. They tend not to like to use products like Microsoft Office and would rather work on Google Drive using Google Docs collaboratively with their peers. They are less concerned with who owns information and more concerned with accessibility and collaboration.

Gen Ys are characterized with creating circles of friends online, doing everything digitally they possibly can, and blogging to their heart's content. In fact, Google has recently released a study that has found that 80% of Gen Ys make up a new generation dubbed "Gen C." Gen Cs are known as the YouTube generation and are focused on "creation, curation, connection, and community" (Google 2012). It is generally embraced in the literature that this is the generation that would rather use their personally purchased tools, devices, and equipment for work purposes because of the ease of carrying

their "life" and "work" with them everywhere they go and the ease of melding their personal hobbies, interests, and professional skillsets with their workplace seamlessly (PWC 2012). Bring your own device (BYOD) is a movement that has emerged from this type of mind-set. It all has to do with customization and personalization, with working with settings that have been defined by the user and with lifelogging in a very audiovisual way. Above all the mantra of this generation is Open-Everything. The claim made by Gen Cs is that transparency is a great force to be reckoned with when it comes to accessibility. Gen Cs allegedly define their social network and are what they share, like, blog, and retweet. This is not without risk, despite that some criminologists have played down the fear as related to privacy and security concerns (David 2008).

Despite that online commentators regularly like to place us all into categories based on our age, most people we've spoken to through our research do not feel like any particular "generation." Individuals like to think they are smart enough to exploit the technologies for what they need to achieve. People may generally choose not to embrace social networking for blogging purposes, for instance, but might see how the application can be put to good use within an institutional setting and educational framework. For this reason they might be heavy users of social networking applications like LinkedIn, Twitter, Facebook, and Google Latitude but also understand its shortcomings and the potential implications of providing a real name, gender, and date of birth, as well as other personal particulars like geotagged photos or live streaming.

This ability to gather and interpret cyber-physical data about individuals and their behaviors has a double-edged spur when related back to a place of work. On the one hand, we have data about someone's personal encounters that can be placed in a context back to a place of employment (Dijst 2009). For instance, a social networking update might read: "In the morning, I met with Katina Michael, we spoke about putting a collaborative grant together on location-based tracking, and then I went and met Microsoft Research Labs to see if they were interested in working with us, and had lunch with person@microsoft.com (+person) (#microsoft) who is a senior software engineer." This information is pretty innocent on its own but there are a lot of details in there that might be used for gathering information: (1) a real name, (2) a real e-mail address, (3) an identifiable position in an organization, (4) potentially links to an extended social network, and (5) possibly even a real physical location of where the meeting took place if the individual had a location-tracking feature switched on their mobile social network app. The underlying point here is that you might have nothing to fear by blogging or participating on social networks under your company identity, but your organization might have much to lose.

Social Reconnaissance

Despite that many of us don't wish to admit it, we have from time to time conducted social reconnaissance online for any number of reasons. In the most basic of cases, you might be visiting a location you have not previously been to and you use Google Street View to take a quick look at what the dwelling looks like for identification purposes. You might also browse the web with your own name, dubbed "ego surfing," to see how you have been cited, quoted, and tagged in images or generally what other people are saying about you. But businesses also are increasingly keeping their eye out on what is being said about their brand using automatic web alerts based on hashtags, to the extent that new schemes offering insurance for business reputation have begun to emerge. Now, my point here is not whether or not you conduct social reconnaissance on yourself, or your family, or your best friend, or even strangers that look enticing, but on what hackers out there might learn about you and your life and your organization by conducting both social and technical reconnaissance. Yes, indeed, if you didn't know it already, there are people out there that will (1) spend all their work time looking up what you do (depending on who you are), (2) think about how that information they have gathered can be related back to your place of work, and (3) exploit that knowledge to conduct clever social engineering attacks (Hadnagy 2011).

Chris Hadnagy, founder of social-engineer.com, was recently quoted as saying: "[i]nformation gathering is the most important part of any engagement. I suggest spending over 50 percent of the time on information gathering. . . Quality information and valid names, e-mails, phone number makes the engagement have a higher chance of success. Sometimes during information gathering you can uncover serious security flaws without even having to test, testing then confirms them" (Goodchild 2012).

It is for this reason that social engineers will focus on the company website, for instance, and build their attack plan off that. Dave Kennedy, CSO of Diebold, complements this idea by first-hand experience: "[a] lot of times, browsing through the company website, looking through LinkedIn are valuable ways to understand the company and its structure. We'll also pull down PDF's, Word documents, Excel spread sheets and others from the website and extract the metadata which usually tells us which version of Adobe or Word they were using and operating system that was used" (Goodchild 2012).

Most of us know of people who do not wish to be photographed and who have painstakingly attempted to un-tag themselves from a variety of images on social networks, who have tried to delete their online presence and be judged before an interview panel for the person they are today, not the person they were when MySpace or Facebook first came out. But what about the separate group of people who do not acknowledge that there is a fence between their work life and home life, accept personal e-mails on a work account, and then are vocal about everything that happens to them on a moment-by-moment basis with a disclaimer that reads: "anything you read on this page is my own personal opinion and not that of the organization I work for." Some would say these individuals are terribly naïve and are probably not acting in accord with organizational policies. The disclaimer won't help the company nor will it help them. Ethical hackers, who have built large empires around their tricks of the trade since the onset of social networking, have spent the last few years trying to educate us all – "data leakage is your biggest problem folks" not the fact that you have weak perimeters! You are, in other words, your own worst enemy because you divulge more than you can afford to, to the online world.

No one is discounting that there are clear benefits in making tacit knowledge explicit by recording it in one form or another, or openly sharing our research data in a way that is conducive to ethical practices, and making things more interoperable than what they are today – but the world keeps moving so fast that for the greater part people are becoming complacent with how they store their datasets and the repercussions of their actions. But the repercussions do exist, and they are real.

Social Engineering

Expert social engineers have never relied on very sophisticated ways of penetrating security systems. It is worth paying a visit to the social engineering toolkit (SET) at www.social-engineer.org where you might learn a great deal about ethical hacking (Palmer 2001) and pentesting (Social-Engineer.Org 2012). Here social engineering tools are categorized as physical (e.g., cameras, GPS trackers, pen recorders, and radio-frequency bug kits), computer based (e.g., common user password profilers), and phone based (e.g., caller ID spoofing). In phase 1 of their premeditated attacks, social engineers are merely engaged in the practice of observation of the information we each put up for grabs willingly. And beyond "the information" itself, subjects and objects are also under surveillance by the social engineers as these might give further clues to the potential hack. It is when there is enough information that a social engineer will think about the next phase 2 which could mean dumpster diving and collecting as much hard copy and online evidence as possible (e.g., company website info). Social networks have given social engineers a whole new avenue of investigation. In fact, social networking will keep social engineers in fulltime work forever and ever unless we all get a lot smarter with how to use these applications.

In phase 3, the evidence gathered by the hacker is used to good practice as they claw their way

deeper and deeper into organizational systems. It might mean having a few full names and position profiles of employees in a company and then using their "hacting" (hacking and acting) skills to get more and more data. Think about social engineers, building on steps and penetrating deeper and deeper into the administration of an organization. While we might think executives are the least targeted individuals, social engineers are brazen to 'attack' personal assistants of executives as well as operational staff. One of the problems associated with social networking is that executives casually give over their login and passwords to personal assistants to take care of their online reputations, thus becoming increasingly easier to manipulate and hijack these spaces and use them to as proof for a given action. When social engineers get that level of authority they require to circumvent systems or they are able to use a technical reconnaissance to exploit data found via social reconnaissance (or vice versa), then they can gain access to an organization's network resources remotely, free to unleash cross-site scripting, man-in-the-middle attacks, SQL code injection, and the like.

Organizational Risks

We have thus come full circle on what social reconnaissance has to do with social networks. Social networking sites (SNS) provide social engineers with every bit of space they need to conduct their unethical hacking and their own penetration tests. You would not be the first person to admit that you have accepted a "friend" on a LinkedIn invitation without knowing who they are, or even caring who they are. Just another e-mail in the inbox to clear out, so pressing accept is usually a lot easier than pressing ignore and then delete or even blocking them for life.

Consider the problem of police in metropolitan forces creating LinkedIn profiles and accepting friends of friends on their public social network profile. What are the implications of this from a criminal perspective? Carrying the analogy of police further, what of the personal gadgets they carry? How many police are currently carrying e-mails on personal mobile phones that they should not be for security concerns? Or even worse, police who have their Twitter, Facebook, or LinkedIn profile always connected via their mobile phone? The police can be said to be rapidly introducing new policies to address these problems, but the problems regardless still exist for mainstream employees of large, medium, and even small organizations. The theft does not have to be complex like the stealing of software code or other intellectual property in designs and blueprints but as simple as the theft of competitive information like customer lead lists in a Microsoft Access database, or payroll data stored in MYOB, or even the physical device itself.

Penetration testing done periodically can be used as feedback into the development of a more robust information security life cycle that can aid those in charge of information governance to react proactively to help employees understand the implications of their practices (Bishop 2007). Trustwave 2012 advocates for four types of assessment and testing. The first is straightforward and traditional physical assessment. The second is client-side penetration testing which validates whether every staff member is adhering to policies. The third is business intelligence testing which is investigating how employees are using social networking, location-enabled devices, and mobile blogging to ensure that a company's reputation is not at risk and to find out what data exists publically about an organization. And finally, red team testing is when a group of diverse subject matter experts try to penetrate a system reviewing security profiles independently.

No one would ever want to be the cause behind the ransacking of their organization's online information above and beyond the web scraping technologies becoming widely available (Poggi et al. 2007). It would help if policies were enforceable within various settings but these too are difficult to monitor. How does one get the message across that while blocking unwanted traffic at the door is very important for an organization, what is even more important is noting what goes walkabout from inside the organization out? It will take

some years for governance structures to adapt to this kind of thinking because the security industry and the media have previously been rightly focused on Denial of Service (DoS) attacks and botnets and the like (Papadimitriou and Garcia-Molina 2011). But it really is a chicken and egg problem – the more information we give out using social networking sites, the more we are giving impetus to DoS, DDoS, and the proliferation of botnets (Kartaltepe et al. 2010; Huber et al. 2009).

Conclusion

Possibly this entry may not have convinced employees that greater care should be taken about what they publish online, on personal blogs, or the pictures or footage post on lifelogs or on YouTube, but it may have convinced employees that the biggest problems today in security systems arise from the information that users post publicly in environments that rely on social networks. This information is just waiting to be harvested by people unsuspecting to users that they will probably never meet physically. Employers need to get their staff educated on company policies periodically and even review the policies they create no less than every 2 years. As an employer you should also be considering when the last time was that your organization performed a penetration test that considered new social networking applications. Individuals should extend this kind of pentesting to their own online profiles and review their own personal situation. Sure you might not have nothing to hide, but you might have a lot to lose.

Cross-References

- ▶ Ethical Issues Surrounding Data Collection in Online Social Networks
- ▶ Location-Based Online Social Networks: Location-Based Online Social Media, Location-Based Online Social Services
- ▶ Privacy Issues for SNS and Mobile SNS
- ▶ Social Phishing
- ▶ Social Media Policy in the Workplace: User Awareness

References

Bishop M (2007) About penetration testing. IEEE Secur Privacy 5(6):84–87

David SW (2008) Cybercrime and the culture of fear. Inf Commun Soc 11(6):861–884

Dijst M (2009) ICT and social networks: towards a situational perspective on the interaction between corporeal and connected presence. In: Kitamura R, Yoshii T, Yamamoto T (eds) The expanding sphere of travel behaviour research. Emerald, Bingley

Goodchild J (2012) 3 tips for using the social engineering toolkit, CSOOnline- data protection. http://www.csoonline.com/article/705106/3-tips-for-using-the-social-engineering-toolkit. Accessed 3 Dec 2012

Google (2012) Introducing Gen C: the YouTube generation. http://ssl.gstatic.com/think/docs/introducing-gen-c-the-youtube-generation_research-studies.pdf. Accessed 1 Apr 2013

Hadnagy C (2011) Social engineering: the art of human hacking. Wiley, Indianapolis

Huber M, Kowalski S, Nohlberg M, Tjoa S (2009) Towards automating social engineering using social networking sites. In: IEEE international conference on computational science and engineering, CSE'09, Vancouver, vol 3. IEEE, Los Alamitos, pp 117–124

Kartaltepe EJ, Morales JA, Xu S, Sandhu R (2010) Social network-based botnet command-and-control: emerging threats and countermeasures. In: Applied cryptography and network security. Springer, Berlin/Heidelberg, pp 511–528

Michael K (2003) The battle against security attacks. In: Lawrence E, Lawrence J, Newton S, Dann S, Corbitt B, Thanasankit T (eds) Internet commerce: digital models for business. Wiley, Milton, pp 156–159. http://works.bepress.com/kmichael/263/. Accessed 1 Feb 2013

Michael K (2008) Social and organizational aspects of information security management. In: IADIS e-Society, Algarve, 9–12 Apr 2008. http://works.bepress.com/kmichael/46/. Accessed 1 Feb 2013

Mitnick K, Simon WL (2002) The art of deception: controlling the Human element of security. Wiley, Indianapolis

Mitnick K, Simon WL (2005) The art of intrusion. Wiley, Indianapolis

Palmer CC (2001) Ethical hacking. IBM Syst J 40(3):769–780

Papadimitriou P, Garcia-Molina H (2011) Data leakage detection. IEEE Trans Knowl Data Eng 23(1):51–63

Poggi N, Berral JL, Moreno T, Gavalda R, Torres J (2007) Automatic detection and banning of content stealing bots for e-commerce. In: NIPS 2007 workshop on machine learning in adversarial environments for computer security. http://people.ac.upc.edu/npoggi/publications/N.%20Poggi%20-%20Automatic%20Detection%20and%20Banning%20 of%20Content%20Stealing%20Bots%20for%20E-commerce.pdf. Accessed 1 May 2013

PWC (2012) BYOD (Bring your own device): agility through consistent delivery. http://www.pwc.com/us/

en/increasing-it-effectiveness/publications/byod-agility-through-consistent-delivery.jhtml. Accessed 3 Dec 2012
Social-Engineer.Org: Security Through Education (2012) http://www.social-engineer.org/. Accessed 3 Dec 2012
Trustwave (2012) Physical security and social engineering testing. https://www.trustwave.com/socialphysical.php. Accessed 3 Dec 2012

Reconstruction

► Imputation of Missing Network Data

Regional Networks

► Interorganizational Networks

Regression Analysis

Andreas Artemiou
School of Mathematics, Cardiff University, Cardiff, Wales, UK

Synonyms

Regression line; Regression model

Glossary

Akaike Information Criterion (AIC)	It is a criterion used to check the goodness of fit of a model and thus is used for comparison between models. A model with smaller AIC is usually preferred. It describes the amount of information one loses from the fitted model
Bayesian Information Criterion (BIC)	It is a criterion used to check the goodness of fit of a model and thus is used for comparison between models. A model with smaller BIC is usually preferred. It is based on the likelihood function information
Binary Logistic Regression (also as Binomial Logistic Regression)	It is the set of tools used for regression when the response variable is binary
Binomial Logistic Regression	See binary logistic regression
Coefficient	See slope and/or intercept
Dependent Variable	See response
Error	See residual
Estimated Response	It is the value that the regression equation assigns to an observation, based on the value (s) of the predictor(s)
Independent Variable	See predictor
Interaction Term	It is the term in a regression function which is a result of a multiplication between two or more predictors
Intercept	It is the value of the response variable when all predictors are equal to 0
Logistic Regression	It is the set of tools used for regression when the response variable is categorical
Multiple Linear Regression	It is the set of tools used to find the relationship between multiple (more than 2) variables when one is the response/dependent and all the rest are the predictors/ independent
Multivariate Linear Regression	It is the set of tools used to find the linear relationship between a number of variables where the response is a vector
Nonlinear Regression	It is the set of tools used when the regression model includes nonlinear terms of predictors. One example is polynomial regression
Observed Response	It is the value of the response variable we observe for each observation

Ordinary Least Squares	It is the optimization method most frequently used in regression when we are interested in estimating the conditional mean E(Y | X), with the main objective being minimizing the sum of the square of residuals
Polynomial Regression	It is when the regression function has predictors with higher-order terms
Predictor (also known as the independent variable)	It is the variable which can be manipulated in order to see a possible effect on the response variable
Quantile Regression	It is the set of tools used to find the relationship between variables, when our interest is to estimate a conditional percentile rather than conditional mean
Regression Through the Origin	It is when a model is forced to have intercept equal to 0
Residual	It is the distance between the observed and the estimated response. It is calculated as observed – estimated. It shows the distance an observation will have from the regression line
Response (also known as the dependent variable)	It is the variable which is considered an observed result of a possible effect from the predictor variable(s)
Simple Linear Regression	It is the set of tools used to find the relationship between two variables; one is the predictor/independent and the other is the response/dependent
Slope	It is related to one of the predictors in the model, and it shows the rate with which the response variable is changed when that predictor is increased by one unit and all other predictors are held constant

Definition

Regression analysis is the set of techniques and tools used in statistics to explore the relationship between variables. In the simplest form (simple linear regression), there is one variable that is treated as the response and one variable that is treated as the predictor, and ordinary least squares (OLS) is used to estimate the linear regression line, which is the line that best fits the data.

Regression analysis can be used in several ways. The most common one is to estimate the effect each unit increase on the predictor (or independent) variable(s) has on the response. It can also be used for prediction, especially if it is used with statistical/machine learning methodology.

Introduction

In many science fields, there is a need to assess if there is a relationship between different variables, evaluate that relationship between variables, and use the conclusion for future predictions. Regression analysis was one of the first tools developed in the statistics literature to achieve this goal. In regression analysis, one needs to collect data for a number of variables (for simplicity, we assume two variables) and then determine if there is an actual relationship between those two variables. One of the variables is called the predictor/independent variable, and the other variable is called the response/dependent variable. If a relationship is established, then that equation can be used to predict what will be the value of the response variable if the predictor takes a specific value. For example, if one is interested to examine if the number of hours a student studies per week affects the GPA, the predictor variable is the "hours of study" and the response is the "GPA." Finding an equation between the two variables will help predict in the future the expected GPA of a student (who has the characteristics of the original population) who studies x number of hours per week.

Key Points

Regression analysis is a huge subject and depends on the type of the objects that are used as well as what is of interest. In this article, we are going to discuss the classic form of regression where continuous variables are used, and we are interested in finding a linear relationship for the conditional mean of the predictor given the response. Other types of regression include logistic regression, quantile regression, and multivariate regression, among others (see glossary terms for a brief explanation of other types of regression). There are many good books in the literature; Kutner et al. (2004a) is one of them.

Historical Background

Francis Galton was the first to use the term regression in the nineteenth century for biological phenomenon, but Karl Pearson was the one to develop the idea in a statistical context in the early twentieth century. It is worth noting that the idea of least squares was first developed separately by Legendre and Gauss in the early nineteenth century. Since then regression became one of the most common topics in statistics and is taught in all introductory statistics courses (Staunton 2001).

Simple Linear Regression Model

In simple linear regression, the objective is to find the relationship between two variables, one is the independent variable (or predictor), denoted with X, and the other is the dependent variable, denoted with Y. We further assume that the relationship is linear, and the objective is to find the regression equation of the line (known as regression line), which best fits the data, given the fact that the data will most probably not lie on a straight line but around it. If we assume we have n pairs of observed data points (Y_i, X_i), the simple linear regression model takes the form

$$Y_i = \beta_0 + \beta_1 \boldsymbol{X_i} + \varepsilon_i, i = 1, \ldots, n$$

where ε_i denotes the distance of each observed point from the regression line, known as the error or the residual. Estimating the regression equation is equivalent to estimating the regression coefficients β_0, β_1. The first is known as the intercept and it is the point where the line meets the y-axis, and the second is known as the slope and it shows the type of the relationship and the how quickly it changes. If the slope is positive(negative), then the relationship is called positive(negative), and the absolute value, $|\beta_1|$, means that for each unit increase in the predictor, there is a$|\beta_1|$ increase(decrease) in the expected value of the response variable.

In the sample version, if we observe n pairs (y_i, x_i), we have the equation

$$\widehat{y_i} = b_0 + b_1 \boldsymbol{x_i}$$

where b_0 denotes the estimate for the intercept β_0 and b_1 denotes the estimate for the slope β_1. The most used method to estimate the coefficients in the equation is known as ordinary least squares (OLS), and the idea is to minimize the sum of squared error (SSE). Thus, we minimize

$$\sum_{i=1}^{n} \varepsilon_i^2 = \sum_{i=1}^{n} (y_i - \widehat{y_i})^2 = \sum_{i=1}^{n} (y_i - (b_0 + b_1 \boldsymbol{x_i}))^2$$

This will give the following equations, also known as normal equations, to estimate the coefficients.

$$b_0 = \overline{y} - b_1 \overline{x}$$

$$b_1 = \frac{\sum_{i=1}^{n} (x_i - \overline{x})(y_i - \overline{y})}{\sum_{i=1}^{n} (x_i - \overline{x})^2}$$

Multiple Linear Regression

There are cases where we are interested in the effect on the response variable from more than one variable. If we assume that there are p predictors, the model takes the form

$$Y_i = \beta_0 + \beta_1 \boldsymbol{X_{1i}} + \ldots + \beta_p \boldsymbol{X_{pi}} + \varepsilon_i, i = 1, \ldots, n$$

and the sample version of the equation is

$$\widehat{y_i} = b_0 + b_1 \boldsymbol{x_i} + \ldots + b_p \boldsymbol{x_i}$$

This can be expressed using matrix notations as follows:

$$\widehat{Y} = \boldsymbol{X}\widehat{\boldsymbol{B}}$$

where X is a n × (p + 1) matrix

$$\begin{bmatrix} 1 & X_{11} & \ldots & X_{p1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & Xn_1 & \ldots & X_{np} \end{bmatrix}$$

and $\widehat{\boldsymbol{B}} = \left(b_0, b_1, \ldots, b_p\right)^T \in \mathbb{R}^{p+1}$. The OLS estimator in matrix form is

$$\widehat{\boldsymbol{B}} = \left(\boldsymbol{X}^T \boldsymbol{X}\right)^{-1} \boldsymbol{X}^T Y$$

Assumptions

Other than assuming there is a linear relationship between the predictors and the response variable, there are some other important assumptions in the linear regression case. These are as follows:

(a) The errors are normally distributed.
(b) Homoscedasticity (or constant variance) which implies that the variance of the error term is constant and does not depend on the value of the predictors. If this assumption is violated, one can use a different estimation technique to estimate regression coefficients, with weighted least squares being a very common choice.
(c) Independence of errors, which implies that the errors are uncorrelated between them. If this assumption is violated, there are different estimation techniques with the generalized least squares being a common choice. To test the above assumptions, usually one needs to run the ordinary least squares and create plots of the residuals versus the predictors and the response variable. If the plots indicate that the assumptions are violated, then one needs to repeat the procedure using a different estimation technique.

Other Types of Regression

(a) Nonlinear regression: There are cases of interest where the response variable is related to the predictors in a nonlinear trend, for example, a polynomial link function or an exponential link function.
(b) Logistic regression: In the case where the response variable is categorical, we can apply logistic regression, which models the probabilities of getting a single outcome. A special case of logistic regression is the case where the response is binary, in which case we have the binary logistic regression. In the case where we have multiple categories, it is also known as multinomial logistic regression. To model the probabilities, one uses the logistic function which is written as

$$\pi(x) = \frac{e^{\beta_0 + \sum_{i=1}^{p} \beta_i X_i + \epsilon}}{e^{\beta_0 + \sum_{i=1}^{p} \beta_i X_i + \epsilon} + 1}$$

In order to turn it into a linear regression function, one can use the logit function which is the natural logarithm:

$$ln \frac{\pi(x)}{1 - \pi(x)} = \beta_0 + \sum_{i=1}^{p} \beta_i X_i + \epsilon$$

(c) Functional regression: There are cases where the predictors are not random variable or random vectors, but they are random functions. In those cases, we have functional regression.
(d) Quantile regression: When we are not interested for the conditional mean $E(Y|X)$ but instead we are interested for a conditional percentile, then the type of regression we use is quantile regression (Koenker 2005; Hao and Naiman 2007).

Regression Diagnostics

(a) Coefficient of determination (most commonly known as R-square and written as R^2): It is a metric that shows the linear strength of the regression model. It takes values between

0 and 1. The closer it is to 1, the stronger the relationship is (and the closer the points are to the regression line). The closer it is to 0, the weaker the relationship is. It can also be interpreted as the percentage of variability of the data points that can be explained by the linear regression model of Y on X. To calculate it, we use the formula

$$R^2 = \frac{\text{SSR}}{\text{SST}} = 1 - \frac{\text{SSE}}{\text{SST}}$$

where SSE is the sum of squared error, SSR is the sum of squared regression, and SST is the total sum of squares, and these are calculated by

$$\text{SSR} = \sum_{i=1}^{n} (\widehat{y}_i - \overline{y})^2, \text{SST} = \sum_{i=1}^{n} (y_i - \overline{y})^2$$

where $\bar{y}$ is the mean of the observed responses:

$$\overline{y} = \frac{\sum_{i=1}^{n} y_i}{n}$$

(Also note that SST = SSR + SSE.) In multiple linear regression (the case with multiple predictors), the more predictors you add in a model, the higher the R^2 is, and as a result, a model with all the predictors, even the ones that are almost irrelevant, is the one with the highest R^2. Thus, adjusted R^2 (adj R^2) has been introduced which takes into account the number of predictors in the model:

$$\text{adj } R^2 = 1 - \frac{\text{SSE}/\text{dfe}}{\text{SST}/\text{dft}}$$

where dfe denotes the error degrees of freedom which are equal to n − p − 1 where p denotes the number of predictors in the model and dft denotes the total degrees of freedom which are equal to n − 1.

(b) Existence of relationship: To check if there is a statistically significant relationship, we run hypothesis tests. This can be done in two ways. One is by testing if the correlation between the two variables is different than 0, and the other is by testing if the slope is equal to 0. Since the regression equation gives us directly the slope, we are going to show how to do the second one here. The test statistic takes the form of

$$\frac{\text{estimate} - \text{null value}}{\text{s.e. of the estimator}} = \frac{\widehat{\beta}_1 - 0}{s_{\widehat{\beta}_1}}$$

where s.e. stands for "standard error." The estimate of the slope is given by the equation, and the estimate of its standard error is given by most software outputs, but the easier way to do it in simple linear regression is to find the sum of squares of residuals $\sum_{i=1}^{n} \varepsilon_i^2$ and then divide by n − p − 1 and divide the result by $\sum_{i=1}^{n} (x_i - \overline{x})^2$, that is,

$$s_{\widehat{\beta}_1} = \frac{\dfrac{\text{SSE}}{n-p-1}}{\sum_{i=1}^{n} (x_i - \overline{x})^2}$$

The above test statistic follows a t distribution with n − p − 1 degrees of freedom, where p is the number of predictors in the model.

Variable Selection

While adjusted R^2 can be used to see how well a model linearly fits the data, other metrics are also used in order to perform variable selection. Variable selection is a set of techniques that are used to reduce the number of predictors in the model, mainly for the purpose of reducing the dimensionality of the model and improving the model interpretation. Many types of algorithms are used; the most common ones are as follows:

(a) Forward selection: One starts with an empty model with no predictors. At each iteration, the algorithm selects the best (in the sense that including it will have the biggest improvement in the performance of a metric) variable to include in the model among the variables

that are not already in the model. When including more variables will not improve the existing model, the algorithm stops, and the model is the best one.

(b) Backward elimination: One starts with the full model, which is the model that includes all the predictors. At each iteration, the algorithm selects the worst (in the sense that excluding it will have the biggest improvement on the performance of a metric) variable which is excluded from the model. When excluding more variables will not improve existing model, the algorithm stops and the model is the best one.

(c) Forward/backward selection: It is when any model is used as the starting point, and at each iteration either a variable is included or a variable is excluded, based on which action will improve the model based on the performance of the model.

More complicated algorithms like genetic algorithms and simulated annealing can be used when the number of predictors is huge and an exhaustive estimation of all possible models can be time consuming. During variable selection methods, a variable is included in the model or excluded from the model based on a criterion. A number of criteria are available in the literature with the most famous ones being the Akaike information criterion (AIC) and the Bayesian information criterion (BIC).

Feature Extraction

Like variable selection, feature extraction is used mainly to reduce the dimensionality of a model. In variable selection, one selects a subset among the original variables to be included in the model, while in feature extraction, functions of the original variables are considered to be the new predictors and will be used in the reduced model.

One example of linear feature selection is principal component analysis (PCA) which selects the linear functions of the predictors which show the most variability in the data. Kernel principal component analysis (KPCA) selects the number of functions (not necessarily linear) of the predictors that have the most variability in the data. KPCA is used for nonlinear feature extraction. Although PCA and KPCA are very common tools and can recover useful linear and nonlinear functions of the predictors in most cases, there are cases where the extracted features are not the ones that are more correlated with the response. This is a common problem of unsupervised dimension reduction methods, that is, when no information from the response is used for feature extraction.

Supervised dimension reduction methods such as projection pursuit and sufficient dimension reduction methods have been developed to take advantage of the information of the response in feature extraction. These methods have the advantage of extracting features more correlated with the response, but they are relatively new and they have not yet been extensively used in applications.

Illustrative Example(s)

We will show several instances of what we talked in the main part of this entry through an example where we are interested to predict body fat through some anatomical variables. The following variables are included in the dataset from 252 people: density of the body from underwater weighing; percentage of body fat, as was calculated by Siri (1956) using equation 495/density − 450; indicator of age group (0 up to 45, 1 for greater than 45); weight (lbs); height (inches); neck circumference (cm); chest circumference (cm); abdomen circumference (cm); hip circumference (cm); thigh circumference (cm); knee circumference (cm); ankle circumference (cm); biceps circumference (cm); forearm circumference (cm); and wrist circumference (cm).

Although measuring body fat is done accurately by Siri's equation, it is very expensive and inconvenient to measure density underwater. All other variables are easier to measure, and we are interested to see if one or more of them can predict the percentage of body fat.

To run the code, we used R packages (Fox and Weisberg 2010).

Examples on Simple Linear Regression

Example 1 We use the same procedure to see if height affects the percentage of body fat. In this case, the equation is

$$\text{perbodyfat} = 35.21 - 0.23 \times \text{height}$$

with an $R^2 = 0.01025$ which indicates a very weak (almost inexistent) linear relationship between the two variables.

Looking at the scatterplot of the data (Fig. 1), it is clear that one needs to do a prescreening of the values that are in the dataset. There is one data point which is an unusual observation as this person is very short, around 30 in., while the rest of the data is for people between 60 and 70 in.

It makes sense to remove that observation as otherwise it will be an observation with very high influential value (high Cook's distance value) on the equation (Fig. 2).

Removing that point shows more clearly that there is no relationship between the two variables. The equation of the regression line is

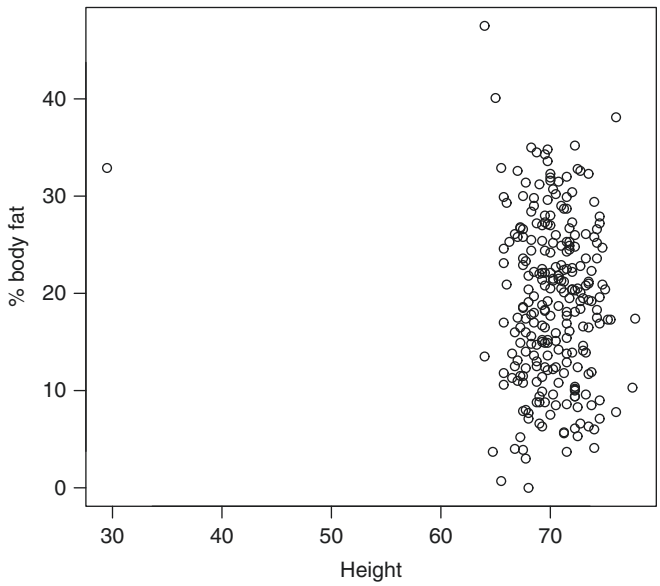
$$\text{perbodyfat} = 24.34 - 0.0746 \times \text{height}$$

with an $R^2 = 0.0005468$ which indicates an extremely weak (essentially inexistent) linear relationship between the two variables.

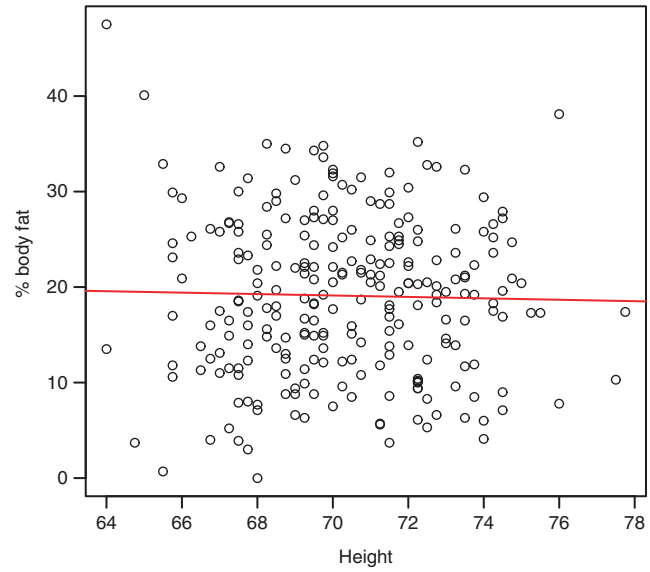
Example 2 We fit a simple linear regression, to see if abdomen circumference is a predictor of the percentage of body fat. The regression equation is the following:

$$\text{perbodyfat} = -39.28 + 0.63 \times \text{abdomen}$$

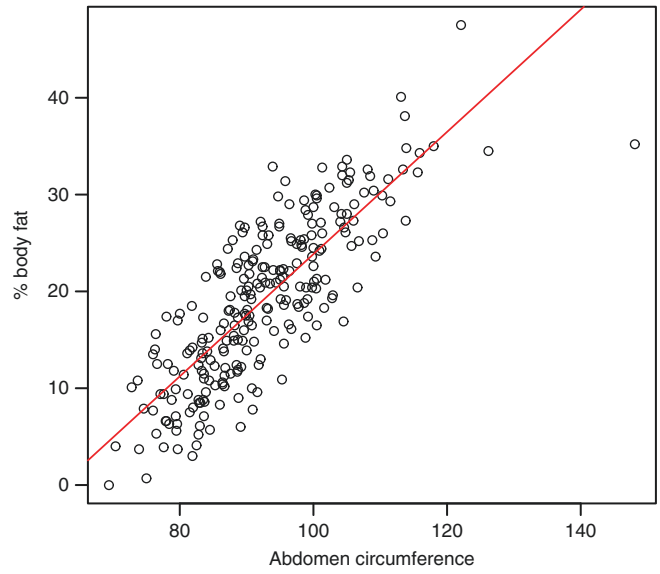
which from the slope we learn that for every cm increase in the abdomen circumference, there is 0.63% increase in the body fat. Also, from the intercept, we can see that someone with 0 cm abdomen circumference has −39.19% body fat which of course does not make any sense. This is called extrapolation, a well-known problem in regression, where we try to make inference for values of the predictors that are beyond the range of the values the predictors have in the available data. The range of values of the abdomen circumference is between 69.4 and 148.1, which indicates that the line should not be interpreted outside that range (because nothing can ensure us the relationship will be the same if data on a different range is collected).



Regression Analysis, Fig. 1 A scatterplot between the height and percentage of body fat indicates that there is a problematic point



Regression Analysis, Fig. 2 The scatterplot with the regression line for percentage of body fat on height when one point is removed from the dataset

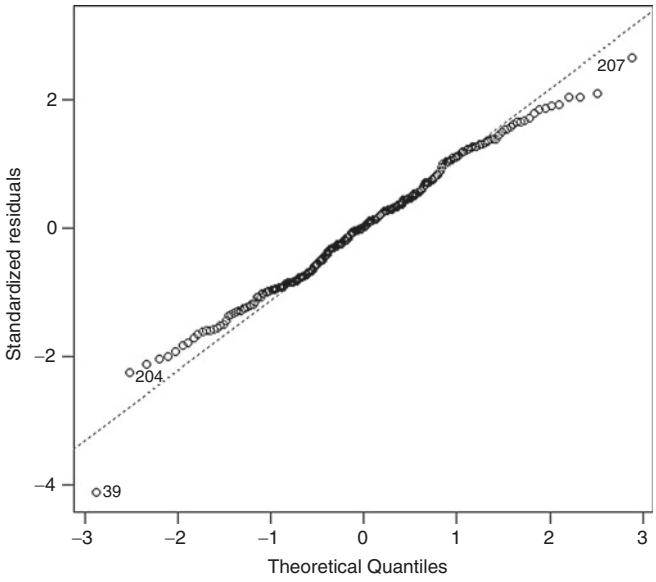


Regression Analysis, Fig. 3 The scatterplot and the regression line of percentage of body fat on abdomen circumference

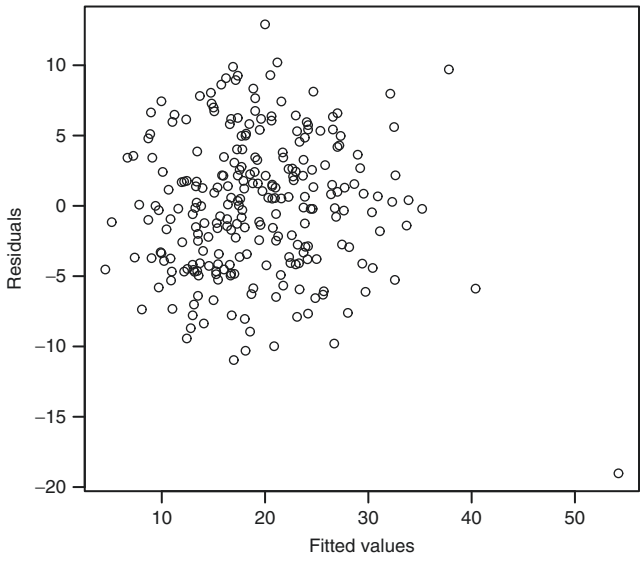
The coefficient of determination, R^2, is 0.6617 which indicates a relatively strong linear relationship between abdomen circumference and percentage of body fat. Another way to interpret this number is by saying that 66.17% of the variability in body fat percentage is explained by its relationship with the abdomen circumference.

The first thing we want to look in this equation is if some assumptions are satisfied and if there is something else (Fig. 3).

The normal probability plot (Fig. 4) is the one that will indicate if the assumption of normality is satisfied. We can see that excluding two points on the left tail the others are very close to, a straight



Regression Analysis, Fig. 4 Normal probability plot for the regression of the percentage of body fat on the abdomen circumference. The points are close to a straight line



Regression Analysis, Fig. 5 The scatterplot of the residuals shows no trend and a constant range throughout the fitted values, which implies independent residuals with constant variance

line indicates that the normality assumption can be assumed. This is the easiest and most frequent way of testing normality although sometimes it can be objective especially if there is curvature but not a clear one.

The second assumption is the independence and homoscedasticity of the residuals. This can be seen in a plot of the residuals with the fitted values. This plot doesn't show any trend; all of them form a pretty nice cloud, which indicates independence. Also, the range of the residuals is constant throughout the range of the predictor (it is obvious with this picture as well that we need to further investigate the point that has residual value of −20 (Fig. 5)).

A way to find unusual observations is by creating the influence plot. Unusual points are points with standardized residuals with high absolute

value or large leverage. Residuals tell us how far away from the line in the y direction are the points, and leverage tells us how far away from the data center in the x direction are the points. A common way to check both is by calculating the Cook's distance which combines the two measures to create a unique measurement that indicates high influential points, that is, points that affect a lot the equation of the line. As a rule of thumb, if Cook's distance is larger than $4/n$, it is considered an influential point, although people use number 1 as a definite cutoff point. As the picture shows, there is a clear indication of point 39 being an influential point and also maybe points 41 and 216. It is important to note that influential points don't necessarily need to be removed from the analysis, because when the analysis is run without current influential points, new highly influential points will appear. So, it is better if the researcher looks carefully at the points that are influential before removing them. We decide to remove point 39, because we suspect there is a typo there (Fig. 6).

Then the equation becomes

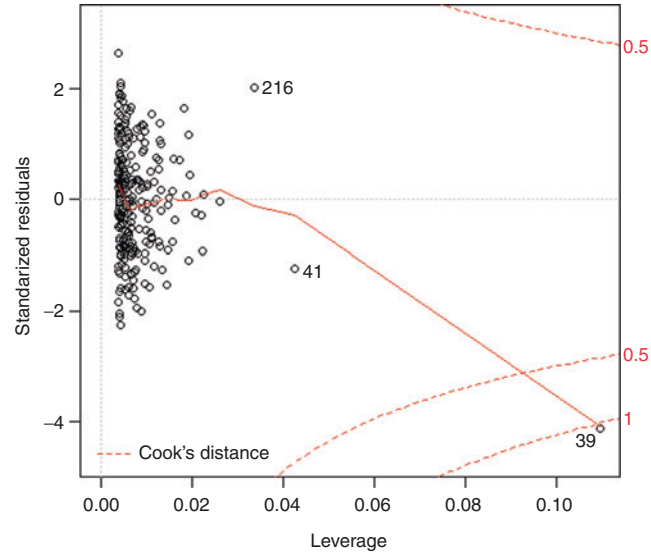
$$\text{perbodyfat} = -42.96 + 0.67 \times \text{abdomen}$$

with an $R^2 = 0.6801$, which is a small difference from the original equation where all the points were included. The overall pictures for the new model though, like normal probability plot and residual plot, make us feel more comfortable about satisfying the assumptions.

Finally, one can perform tests, to check if there is a statistically significant relationship. We use the test statistic for the slope. In our case, the estimate is 0.67, the s.e. of the estimator is 0.02921, and the value of the test statistic is 23.01. The test statistic follows a t distribution with n − 2 degrees of freedom, and the resulting p-value is essentially 0 which indicates that indeed there is a statistically significant evidence that there is relationship between the abdomen circumference and the body fat.

Example on Nonlinear Regression

Example 3: One might claim that the ratio between weight and height is an indication of body fat. This means that we need to fit the model perbodyfat $\sim \frac{\text{weight}}{\text{height}}$. This uses the ratio of two variables, and it is the simple case of nonlinear regression. Like in many cases where nonlinear regression needs to be used, one can create a new variable called ratio and fit the model perbodyfat – ratio. This reduces the problem to the simple linear regression, where we have seen two examples above.



Regression Analysis, Fig. 6 Influence plot, where it is clear that point 39 is very influential and points 41 and 216 can be considered influential as well

Regression Analysis, Table 1 The coefficients and corresponding p-values when fitting the full model for the body fat data

Variable	Coefficient	p-value
Weight	−0.110	0.038
Height	−0.094	0.326
Neck circumference	−0.430	0.066
Chest circumference	−0.017	0.863
Abdomen circumference	1.030	0.000
Hip circumference	−0.230	0.117
Thigh circumference	0.135	0.320
Knee circumference	0.132	0.576
Ankle circumference	0.130	0.559
Biceps circumference	0.205	0.233
Forearm circumference	0.390	0.050
Wrist circumference	−1.272	0.013

Example on Multiple Linear Regression

Example 4: A full model with all anatomic variables is applied to check which of those are statistically significant in predicting the percentage of body fat. We excluded age which is a binary variable. We run ordinary least squares, and we list below the value of the coefficient with the p-value given by the test for statistically significant coefficient different from 0.

Table 1 indicates that a lot of these variables have no real relationship with the percentage of body fat. As a result, it makes sense for someone to eliminate useless variables. This can be done with variable selection. Running a forward selection procedure, it is indicated that the model that has the smallest AIC is the one that includes weight and neck, abdomen, biceps, forearm, and wrist circumferences. The equation is

$$\begin{aligned}\text{perbodyfat} = \; & -32.23 - 0.138 \\ & \times \text{weight} - 0.410 \quad \times \text{neck} \\ & +1.013 \times \text{abdomen} \quad + 0.257 \\ & \times \text{biceps} + 0.426 \quad \times \text{forearm} \\ & -1.236 \times \text{wrist}\end{aligned}$$

All the variables with the coefficient value and the p-value for the test of significance of the coefficient are shown on Table 1. Bold variables are the ones which suggest statistically significant coefficient and with italic are the variables with coefficients that are barely outside the statistically significant region.

Cross-References

- Data Mining
- Eigenvalues: Singular Value Decomposition
- Independent Component Analysis
- Least Squares
- Principal Component Analysis
- Theory of Statistics: Basics and Fundamentals

References

Fox J, Weisberg HS (2010) An R companion to applied regression, 2nd edn. Sage, Los Angeles

Hao L, Naiman DQ (2007) Quantile regression. Sage, Thousand Oaks

Koenker R (2005) Quantile regression. Cambridge University Press, New York, MATH

Kutner M, Nachtsheim C, Neter J (2004a) Applied linear regression models, 4th edn. McGraw-Hill/Irwin, Boston

Siri WE (1956) Gross composition of the body. In: Lawrence JH, Tobias CA (eds) Advances in biological and medical physics, vol IV. Academic, New York

Staunton JM (2001) Galton, pearson and the peas: a brief history of linear regression for statistics instructors. J Stat Educ 3. http://www.amstat.org/publications/jse/v9n3/stanton.html

Recommended Reading

Allison PD (1998) Multiple regression: a primer. Pine Forge, Thousand Oaks

Chatterjee S, Hadi AS (2006) Regression analysis by example. Wiley Interscience, New York, MATH

Cook RD (1998) Regression graphics: ideas for studying regressions through graphics. Wiley Interscience, New York, MATH

Cook RD, Weisberg HS (1999) Applied regression including computing and graphics. Wiley, New York, MATH

Gelman A, Hill J (2006) Data analysis using regression and multilevel/hierarchical models. Cambridge University Press, New York

Kutner M, Nachtsheim C, Neter J, Li W (2004b) Applied statistical models, 5th edn. McGraw-Hill/Irwin, New York

Mendenhall W, Sincich T (2011) A second course in statistics: regression analysis, 7th edn. Prentice Hall, Upper Saddle River

Schroeder DL, Sjoquist DL, Stephan PE (1986) Understanding regression analysis: an introductory guide (quantitative applications in the social sciences). Sage, Newbury Park

Seber GA, Lee AJ (2003) Linear regression analysis, 2nd edn. Wiley, Hoboken, MATH

Seber GA, Wild CJ (2003) Nonlinear regression. Wiley-Interscience, Hoboken

Weisberg HS (2005) Applied linear regression, 3rd edn. Wiley, Hoboken, MATH

Regression Line

▶ Regression Analysis

Regression Model

▶ Regression Analysis

Regulatory Concerns

▶ Legal Implications of Social Networks

Relational Analysis

▶ Origins of Social Network Analysis

Relational Data Mining

▶ Relational Models

Relational Feature

▶ Collective Classification: Structural Features

Relational Learning

▶ Collective Classification
▶ Relational Models

Relational Models

Volker Tresp[1,2] and Maximilian Nickel[2]
[1]Corporate Technology, Siemens AG, München, Germany
[2]Department of Computer Science, Ludwig Maximilian University of Munich, München, Germany

Synonyms

Relational data mining; Relational learning; Statistical relational learning; Statistical relational models

Glossary

Collective classification	A special case of collective learning: The class membership of entities can be predicted from the class memberships of entities in their (social) network environment. Example: Individuals' income classes can be predicted from those of their friends
Collective learning	Refers to the effect that an entity's relationships, attributes, or class membership can be predicted not only from its attributes but also from its (social) network environment
Entities	Are (abstract) objects. We denote an entity by a lowercase *e*. An actor in a social network can be modeled as an entity. There can be multiple types of entities in a domain (e.g., individuals, cities, companies), entity attributes (e.g., income, gender), and relationships between entities (e.g., knows, likes, brother, sister). Entities, relationships, and attributes are defined in the

entity-relationship model, which is used in the design of a formal relational model

Entity resolution The task of predicting if two constants refer to the same entity

Graphical models A graphical description of a probabilistic domain where nodes represent random variables and edges represent direct probabilistic dependencies

Homophily The tendency of an individual to associate with similar others

Latent variables Latent variables are quantities which are not measured directly and whose states are inferred from data

Linked data Linked Open Data describes a method for publishing structured data so that it can be interlinked and can be exploited by machines. Linked Open Data uses the RDF data model

Predicate A predicate R is a mapping of tuples to true or false. $R(t)$ is a *ground* predicate and is true when $t \in I(R)$, otherwise it is false. Note that we do not distinguish between the relation name R and the predicate name R. Example: *knows* is a predicate and *knows (Jack, Mary)* returns *True* if it is true that Jack knows Mary, i.e., that *(Jack, Mary)* $\in I($*knows*$)$. The convention is that relations and predicates are written in lowercase and entities in uppercase

Probabilistic database A (possible) world corresponds to a database instance. In a probabilistic database, a probability distribution is defined over all possible worlds under consideration. Probabilistic databases with potentially complex dependencies can be described by probabilistic graphical models. In a canonical representation, one assigns a binary random variable $X_{R,t}$ to each possible tuple t in each relation R. Then

$t \in I(R) \Leftrightarrow R(t) = \text{True} \Leftrightarrow X_{R,t} = 1$

and

$t \notin I(R) \Leftrightarrow R(t) = \text{False} \Leftrightarrow X_{R,t} = 0$.

The probability for a world x is written as $P(X = x)$ where $X = \{X_{R,t}\}_{R,t}$ is the set of random variables and x denotes their values in the world (see Fig. 1)

Relation A relation or relation instance $I(R)$ is a set of tuples. A tuple t is an ordered list of elements $(e_1, e_2, .., e_{arity})$, which, in the context of this discussion, represent entities. The *arity* of a relation is the number of elements in each of its tuples, e.g., a relation might be unary, binary, or higher order. R is the name or type of the relation. For example, (*Jack*, *Mary*) might be a tuple of the relation instance *knows*, indicating that Jack knows Mary. A database instance (or world) is a set of relation instances. For example, a database instance might contain instances of the unary relations *student*, *teacher*, *male*, and *female* and instances of the binary relations *knows*, *likes*, *brother*, and *sister* (see Fig. 1)

Relationship prediction The prediction of the existence of a relationship between entities, for example, friendship between individuals. A relationship is typically modeled as a binary relation

Triple database A triple database consists of binary relations represented as subject-predicate-object triples. An example of a triple is (*Jack*, *knows*, *Mary*). A triple database

can be represented as a knowledge graph with entities as nodes and predicates as directed links, pointing from the subject node to the object node. The Resource Description Framework (RDF) is triple based and is the basic data model of the Semantic Web Linked Open Data. In social network analysis, nodes would be individuals or actors, and links would correspond to ties

Definition

Relational models are machine learning models that are able to truthfully represent some or all distinguishing features of a relational domain such as long-range dependencies over multiple relationships. Typical examples for relational domains include social networks and knowledge bases. Relational models concern nontrivial relational domains with at least one relation with an arity of two or larger that describes the relationship between entities, e.g., *knows*, *likes*, and *dislikes*. In the following we will focus on nontrivial relational domains.

Introduction

Social networks can be modeled as graphs, where actors correspond to nodes and where relationships between actors such as friendship, kinship, organizational position, or sexual relationships are represented by directed labeled links (or ties) between the respective nodes. Typical machine learning tasks would concern the prediction of unknown relationship instances between actors, as well as the prediction of actors' attributes and class labels. In addition, one might be interested in a clustering of actors. To obtain best results, machine learning should take an actor's network environment into account. Thus, two individuals might appear in the same cluster because they have common friends.

Relational learning is a branch of machine learning that is concerned with these tasks, i.e., to learn efficiently from data where information is represented in the form of relationships between entities.

Relational models are machine learning models that truthfully model some or all distinguishing features of relational data such as long-range dependencies propagated via relational chains and homophily, i.e., the fact that entities with similar attributes are neighbors in the relationship structure. In addition to social network analysis, relational models are used to model knowledge graphs, preference networks, citation networks, and biomedical networks such as gene-disease networks or protein-protein interaction networks. Relational models can be used to solve the aforementioned machine learning tasks, i.e., classification, attribute prediction, and clustering. Moreover, relational models can be used to solve additional relational learning tasks such as relationship prediction and entity resolution. Relational models are derived from directed and undirected graphical models to latent variable models and typically define a probability distribution over a relational domain.

Key Points

Statistical relational learning is a subfield of machine learning. Relational models learn a probabilistic model of a complete networked domain by taking into account global dependencies in the data. Relational models can lead to more accurate predictions if compared to non-relational machine learning approaches. Relational models typically are based on probabilistic graphical models, e.g., Bayesian networks, Markov networks, or latent variable models.

Historical Background

Inductive logic programming (ILP) was maybe the first machine learning effort that seriously focused on a relational representation. It gained attention in the early 1990s and focusses on

learning deterministic or close-to-deterministic dependencies, with representations derived from first-order logic. As a field, ILP was introduced in a seminal paper by Muggleton (Muggleton 1991). A very early and still very influential algorithm is Quinlan's FOIL (Ross Quinlan 1990). ILP will not be a focus in the following, since social networks exhibit primarily statistical dependencies. Statistical relational learning started around the beginning of the millennium with the work by Koller, Pfeffer, Getoor, and Friedman (Friedman et al. 1999; Koller and Pfeffer 1998). Since then, many combinations of ILP and relational learning have been explored. The Semantic Web and Linked Open Data are producing vast quantities of relational data and (Nickel et al. 2012; Tresp et al. 2009) describe the application of statistical relational learning to these emerging fields. Relational learning has been applied to the learning of knowledge graphs, which model large domains as triple databases. Nickel et al. (2016) has a recent review on the application of relational learning to knowledge graphs. An interesting application is the semiautomatic completion of knowledge graphs by analyzing information from the Web and other sources, in combination with relational learning, which exploits the information already present on the knowledge graph (Dong et al. 2014).

Machine Learning in Relational Domains

Relational Domains

Relational domains are domains that can truthfully be represented by relational databases. The glossary defines the key terms such as a relation, a predicate, a tuple, and a database. Nontrivial relational domains contain at least one relation with an arity of two or larger that describes the relationship between entities, e.g., *knows*, *likes*, and *dislikes*. The main focus here is on nontrivial relational domains.

Social networks are typical relational domains, where information is represented by multiple types of relationships (e.g., *knows*, *likes*, *dislikes*) between entities (here: actors), as well as through the attributes of entities.

Generative Models for a Relational Database

Typically, relational models can exploit long-range or even global dependencies and have principled ways of dealing with missing data. Relational models are often displayed as probabilistic graphical models and can be thought of as relational versions of regular graphical models, e.g., Bayesian networks, Markov networks, and latent variable models. The approaches often have a "Bayesian flavor," but a fully Bayesian statistical treatment is not always performed.

The following section describes common relational graphical models.

Non-relational Learning

Although we are mostly concerned with relational learning, it is instructive to analyze the special case of non-relational learning. Consider a database with a key entity class *actor* with elements e_i and with only unary relations; thus, we are considering a trivial relational domain. Then one can partition the random variables into independent disjoint sets according to the entities, and the joint distribution factorizes as

$$\prod_i P\left(\left\{X_{R,e_i}\right\}_R\right)$$

where the binary random variable $X_{R,ei}$ is assigned to tuple e_i in unary relation R (see glossary).

Thus, the set of random variables can be reduced to nonoverlapping independent sets of random variables. This is the common non-relational learning setting with *i.i.d.* instances, corresponding to the different actors.

Non-relational Learning in a Relational Domain

An common approximation to a relational model is to model unary relations of key entities in a similar way as in a non-relational model as

$$\prod_i P\left(\left\{X_{R,e_i}\right\}_R \middle| \mathbf{f}_i\right)$$

where $\mathbf{f}_i$ is a vector of relational features that are derived from the relational network environment

of the actor i. Relational features provide additional information to support learning and prediction tasks. For instance, the average income of an individual's friends might be a good covariate to predict an individual's income in a social network. The underlying mechanism that forms these patterns might be homophily, the tendency of individuals to associate with similar others. The goal of this approach is to be able to use *i.i.d.* machine learning by exploiting some of the relational information. This approach is commonly used in applications where probabilistic models are computationally too expensive. The application of non-relational machine learning to relational domains is sometimes referred to as *propositionalization*.

Relational features are often high dimensional and sparse (e.g., there are many people, but only a small number of them are an individual's friends; there are many items but an individual has only bought a small number of them), and in some domains, it can be easier to define useful kernels than to define useful features. Relational kernels often reflect the similarity of entities with regard to the network topology. For example, a kernel can be defined based on counting the substructures of interest in the intersection of two graphs defined by neighborhoods of the two entities (Lösch et al. 2012) (see also the discussion on RDF graphs further down).

Learning Rule Premises in Inductive Logic Programming

Some researchers apply a systematic search for good features and consider this as an essential distinction between *relational* learning and *non-relational* learning: in non-relational learning, features are essentially defined prior to the training phase, whereas relational learning includes a systematic and automatic search for features in the relational context of the involved entities. Inductive logic programming (ILP) is a form of relational learning with the goal of finding deterministic or close-to-deterministic dependencies, which are described in logical form such as Horn clauses. Traditionally, ILP involves a systematic search for sensible relational features that form the rule premises (Dzeroski 2007).

Relational Models

In this section, we describe the most important relational models in some detail. These are based on probabilistic graphical models, which efficiently model high-dimensional probability distributions by exploiting independencies between random variables. In particular, we consider Bayesian networks, Markov networks, and latent variable models. We start with a more detailed discussion on possible world models for relational domains and with a discussion on the dual structures of the triple graph and the probabilistic graph.

Random Variables for Relational Models

As mentioned before, a probabilistic database defines a probability distribution over the possible worlds under consideration. The goal of relational learning is to derive a model of this probability distribution.

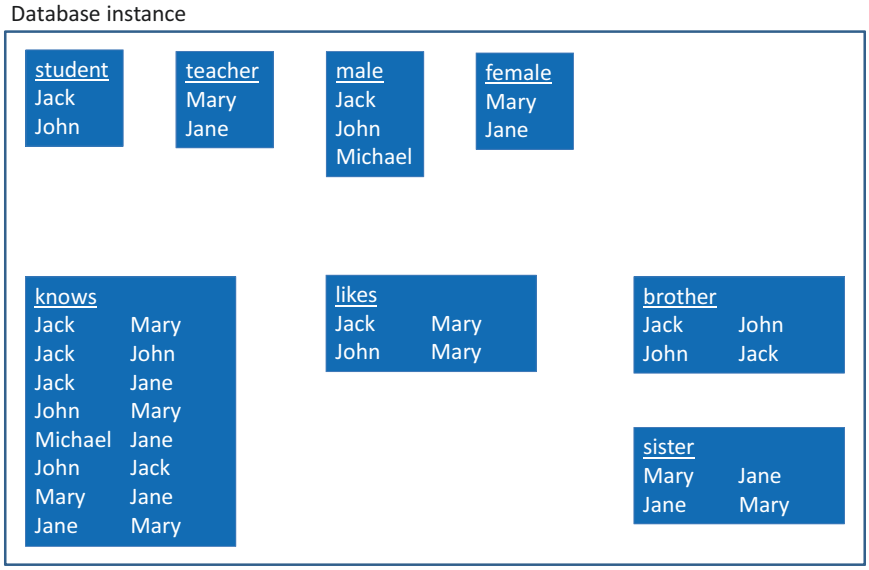
In a canonical representation, we assign a binary random variable $X_{R,t}$ to each possible tuple in each relation. Then

$$t \in I(R) \Leftrightarrow R(t) = \text{True} \Leftrightarrow X_{R,t} = 1$$

and

$$t \notin I(R) \Leftrightarrow R(t) = \text{False} \Leftrightarrow X_{R,t} = 0.$$

The probability for a world x is written as $P(X = x)$, where $X = \{X_{R,t}\}_{R,t}$ is the set of random variables and x denotes their values in the world (see Fig. 1). What we have just described corresponds to a closed-world assumption where all tuples, which are not part of the database instance, map to $R(t) = False$, and thus $X_{R,t} = 0$. In contrast, in an open-world assumption, we would consider the corresponding truth values and states as being unknown and the database instance as being only partially observed. Often in machine learning, some form of a *local* closed-world assumption is applied with a mixture of true, false, and unknown ground predicates (Dong et al. 2014; Krompaß et al. 2015). For example, one might assume that, if at least one child of an individual is specified, it implies that



Relational Models, Fig. 1 A database instance (world) with four unary relations (*student*, *teacher*, *male*, *female*) and four binary relations (*knows*, *likes*, *brother*, *sister*). As examples, (*Jack, Mary*) $\in$ *I*(*knows*). Thus, *knows(Jack, Mary)* = *True*, and $X_{knows,(Jack,\ Mary)} = 1$. (*Jack, Michael*) $\notin$ *I*(*knows*). Thus, *knows(Jack, Michael)* = *False*, and $X_{knows,\ (Jack,Michel)} = 0$

all children are specified (closed world), whereas if no child is specified, children are considered unknown (open world). Another aspect is that type constraints might imply that certain ground predicates are false. For example, only individuals can get married, but neither cities nor buildings. Other types of background knowledge might materialize tuples that are not explicitly specified. For example, if individuals live in Munich, by simple reasoning, one can conclude that they also live in Bavaria and Germany. The corresponding tuples can be added to the database.

Based on background knowledge, one might want to modify the canonical representation, which uses only binary random variables. For example, discrete random variables with N states are often used to implement the constraint that exactly one out of N ground predicates is true, e.g., that an individual belongs exactly to one out of N income classes or age classes. It is also possible to extend the model toward continuous variables.

So far we have considered an underlying probabilistic model and an observed world. In probabilistic databases, one often assumes a noise process between the actual database instance and the observed database instance by specifying a conditional probability:

$$P\left(Y_{R,t} \middle| X_{R,t}\right).$$

Thus, only $Y_{R,t}$ is observed, whereas the real interest is on $X_{R,t}$: One observes a $t \in I^y(R) \Leftrightarrow Y_{R,t} = 1$ from which one can infer for the database instance $P(t \in I(R)) \Leftrightarrow P(X_{R,t} = 1)$. With an observed $Y_{R,t} = 1$, there is a certain probability that $X_{R,t} = 0$ (error in the database), and with an observed $Y_{R,t} = 0$, there is a certain probability that $X_{R,t} = 1$ (missing tuples).

The theory of probabilistic databases is focused on the issues of complex query answering under a probabilistic model. In probabilistic databases (Suciu et al. 2011), the canonical representation is used in tuple-independent databases, while multi-state random variables are used in block-independent disjoint (BID) databases.

Most relational models assume that all entities (or constants) and all predicates are known and fixed (domain closure assumption). In general these constraints can be relaxed, for example, if

one needs to include new individuals in the model. Also, latent variables derived from a cluster or a factor analysis can be interpreted as new "invented" predicates.

Triple Graphs and Probabilistic Graphical Networks

A triple database consists of binary relations represented as subject-predicate-object triples. An example of a triple is *(Jack, knows, Mary)*. A triple database can be represented as a knowledge graph with entities as nodes and predicates as directed links, pointing from the subject node to the object node. Triple databases are able to represent Web-scale knowledge bases and sociograms that allow multiple types of directed links. Relations of higher order can be reduced to binary relations by introducing auxiliary entities ("blank nodes"). Figure 2 shows an example of a triple graph. The Resource Description Framework (RDF) is triple based and is the basic data model of the Semantic Web Linked Open Data. In social network analysis, nodes would be individuals or actors, and links would correspond to ties.

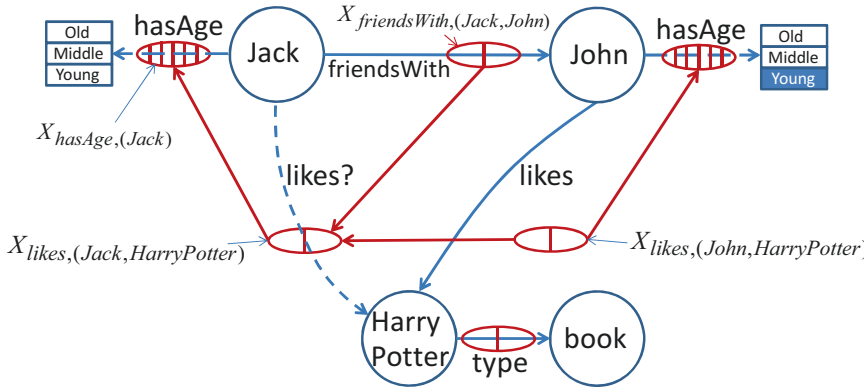
For each triple, a random variable is introduced. In Fig. 2, these random variables are represented as elliptical red nodes. The binary random variable associated with the tripe ($s = i$, $p = k$, $o = j$) will be denoted as $X_{k(i,j)}$.

Directed Relational Models

The probability distribution of a directed relational model, i.e., a relational Bayesian model, can be written as

$$P\left(\{X_{R,t}\}R,t\right) = \prod_{R,t} P\left(X_{R,t} | \operatorname{par}\left(X_{R,t}\right)\right). \quad (1)$$

Here, $\{X_{R,t}\}_{R,t}$ refers to the set of random variables in the directed relational model, while $X_{R,t}$ denotes a particular random variable. In a graphical representation, directed arcs are pointing from

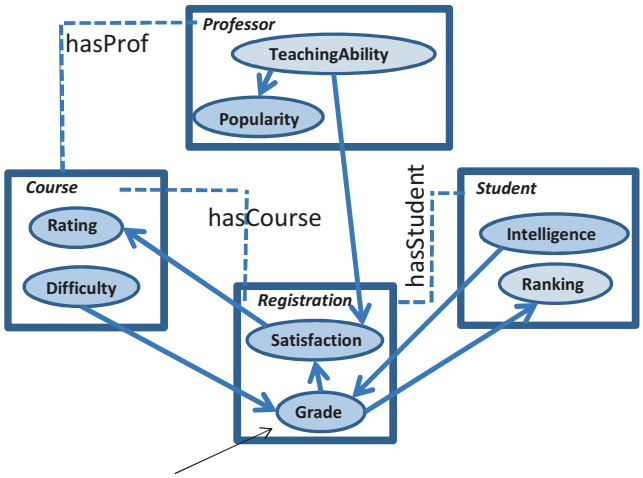


Relational Models, Fig. 2 The figure clarifies the relationship between the triple graph and the probabilistic graphical network. The round nodes stand for entities in the domain, the square nodes stand for attributes, and the labeled links stand for triples. Thus, we assume that it is known that *Jack* is friends with *John* and that *John* likes the book *Harry Potter*. The oval nodes stand for random variables, and their states represent the existence (value 1) or nonexistence (value 0) of a given labeled link; see, for example, the node $X_{likes,(John,HarryP\ otter)}$ which represents the ground predicate *likes (John, Harry Potter)*. Striped oval nodes stand for random variables with many states, which are useful for attribute nodes (exactly one out of many ground predicates is true). The unary relations *hasAgeOld*, *hasAgeMiddle*, and *hasAgeYoung* are represented by the random variable X_{hasAge} which has three states. Relational models assume a probabilistic dependency between the probabilistic nodes. So the relational model might learn that *Jack* also likes *Harry Potter* since his friend *Jack* likes it (homophily). Also $X_{likes,(John,HarryP\ otter)}$ might correlate with the age of John. The direct dependencies are indicated by the red edges between the elliptical nodes. In PRMs the edges are directed (as shown), and in Markov logic networks, they are undirected. The elliptical random nodes and their quantified edges form a probabilistic graphical model. Note that the probabilistic network is dual to the triple graph in the sense that links in the triple graph become nodes in the probabilistic network.

all parent nodes $par(X_{R,t})$ to the node $X_{R,t}$ (Fig. 2). As Eq. 1 indicates, the model requires the specification of the parents of a node and the specification of the probabilistic dependency of a node, given the states of its parent nodes. In specifying the former, one often follows a causal ordering of the nodes, i.e., one assumes that the parent nodes causally influence child nodes and their descendents. An important constraint is that the resulting directed graph is not permitted to have directed loops, i.e., that it is a directed acyclic graph. A major challenge is to specify $P(X_{R,t} | par(X_{R,t}))$, which might require the calculation of complex aggregational features as intermediate steps.

Probabilistic Relational Models

Probabilistic relational models (PRMs) were one of the first published directed relational models and found great interest in the statistical machine learning community (Getoor et al. 2007; Koller and Pfeffer 1998). An example of a PRM is shown in Fig. 3. PRMs combine a frame-based (i.e., object oriented) logical representation with probabilistic semantics based on directed graphical models. The PRM provides a template for specifying the graphical probabilistic structure and the quantification of the probabilistic dependencies for any ground PRM. In the basic PRM models, only the entities' attributes are uncertain, whereas



D, I	A	B	C
H,H	0.5	0.4	0.1
H,L	0.1	0.5	0.4
L,H	0.8	0.1	0.1
L,L	0.3	0.6	0.1

Probabilistic table of *Grade* (with states *A, B, C*) given its parents *Difficulty (H, L)* and *Intelligence (H, L)*

Relational Models, Fig. 3 A PRM with domain predicates *Professor*(*ProfID*, *TeachingAbility*, *Popularity*), *Course*(*CourseID*, *ProfID*, *Rating*, *Difficulty*), *Student* (*StuID, Intelligence, Ranking*), and *Registration* (*RegID*, *CourseID*, *StuID*, *Satisfaction*, *Grade*). Dotted lines indicate foreign keys, i.e., entities defined in another relational instance. The directed edges indicate direct probabilistic dependencies on the template level. Also shown is a probabilistic table of the random variable *Grade* (with states *A*, *B*, *C*) given its parents *Difficulty* and *Intelligence*. Note that some probabilistic dependencies work on multisets and require some form of aggregation: for example, different students might have different numbers of registrations, and the ranking of a student might depend on the (aggregated) average grade from different registrations. Note the complexity in the dependency structure which can involve several entities: for example, the *Satisfaction* of a *Registration* depends on the *Teaching Ability* of the *Professor* teaching the *Course* associated with the *Registration*. Consider the additional complexity when structural uncertainty is present, e.g., if the *Professor* teaching the *Course* is unknown

the relationships between entities are assumed to be known. Naturally, this assumption greatly simplifies the model. Subsequently, PRMs have been extended to also consider the case that relationships between entities are unknown, which is called *structural uncertainty* in the PRM framework (Getoor et al. 2007).

In PRMs one can distinguish parameter learning and structural learning. In the simplest case, the dependency structure is known, and the truth values of all ground predicates are known as well in the training data. In this case, parameter learning consists of estimating parameters in the conditional probabilities. If the dependency structure is unknown, structural learning is applied, which optimizes an appropriate cost function and typically uses a greedy search strategy to find the optimal dependency structure. In structural learning, one needs to guarantee that the ground Bayesian network does not contain directed loops.

In general the data will contain missing information, i.e., not all truth values of all ground predicates are known in the available data. For some PRMs, regularities in the PRM structure can be exploited (encapsulation), and even exact inference to estimate the missing information is possible. Large PRMs require approximate inference; commonly, loopy belief propagation is being used.

More Directed Relational Graphical Models

A Bayesian logic program is defined as a set of Bayesian clauses (Kersting and De Raedt 2001). A Bayesian clause specifies the conditional probability distribution of a random variable given its parents. A special feature is that, for a given random variable, *several* such conditional probability distributions might be given and combined based on various combination rules (e.g., noisy-or). In a Bayesian logic program, for each clause, there is one conditional probability distribution, and for each random variable, there is one combination rule. *Relational Bayesian networks* (Jaeger 1997) are related to Bayesian logic programs and use probability formulae for specifying conditional probabilities. The probabilistic entity-relationship (PER) models (Heckerman et al. 2007) are related to the PRM framework and use the entity-relationship model as a basis, which is often used in the design of a relational database. Relational dependency networks (Neville and Jensen 2004) also belong to the family of directed relational models and learn the dependency of a node given its Markov blanket (the smallest node set that make the node of interest independent of the remaining network). Relational dependency networks are generalizations of dependency networks as introduced by Heckerman (2000) and Hofmann and Tresp (1997). A relational dependency network typically contains directed loops and thus is not a proper Bayesian network.

Undirected Relational Graphical Models

The probability distribution of an undirected graphical model, i.e., a Markov network, is written as a log-linear model in the form:

$$P(X = x) = \frac{1}{Z} \exp \sum_i w_i f_i(x_i)$$

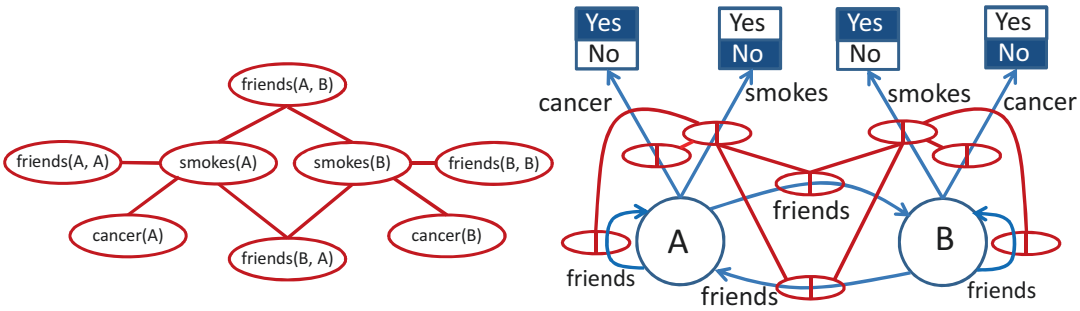
where the feature functions f_i can be any real-valued function on the set $x_i \subseteq x$ and where $w_i \in \mathbb{R}$. In a probabilistic graphical representation, one forms undirected edges between all nodes that jointly appear in a feature function. Consequently, all nodes that appear jointly in a function will form a *clique* in the graphical representation. Z is the partition function normalizing the distribution.

A major advantage is that undirected graphical models can elegantly model symmetrical dependencies, which are common in social networks.

Markov Logic Network (MLN)

A Markov logic network (MLN) is a probabilistic logic which combines Markov networks with first-order logic. In MLNs the random variables, representing ground predicates, are part of a Markov network, whose dependency structure is derived from a set of first-order logic formulae (Fig. 4).

Formally, a MLN L is defined as follows: Let F_i be a first-order formula (i.e., a logical expression containing constants, variables, functions, and predicates), and let $w_i \in \mathbb{R}$ be a weight attached to each formula. Then L is defined as a set of pairs (F_i, w_i) (Domingos and Richardson 2007; Richardson and Domingos 2006).



Relational Models, Fig. 4 *Left*: an example of a MLN. The domain has two entities (constants) A and B and the unary relations *smokes* and *cancer* and the binary relation *friends*. The eight elliptical nodes are the ground predicates. Then there are two logical expressions $\forall x\ smokes(x) \rightarrow cancer(x)$ (someone who smokes has cancer) and $\forall x \forall y\ friends(x, y) \rightarrow (smokes(x) \leftrightarrow smokes(y))$ (friends either both smoke or both do not smoke). Obviously and fortunately, both expressions are not always true, and learned weights on both formulae will assume finite values. There are two groundings of the first formula (explaining the edges between the *smokes* and *cancer* nodes) and four groundings of the second formula, explaining the remaining edges. The corresponding features are equal to one if the logical expressions are true and are zero otherwise. The weights on the features are adapted according to the actual statistics in the data. Redrawn from Domingos and Richardson (2007). Right: The corresponding triple graph for two individuals (*blue*) and the dual ground Markov network (*red*)

From L the ground Markov network $M_{L,C}$ is generated as follows. First, one generates nodes (random variables) by introducing a binary node for each possible grounding of each predicate appearing in L given a set of constants $c_1, \ldots, c_{|C|}$ (see the discussion on the canonical probabilistic representation). The state of a node is equal to one if the ground predicate is true and zero otherwise. The feature functions f_i, which define the probabilistic dependencies in the Markov network, are derived from the formulae by grounding them in a domain. For formulae that are universally quantified, grounding is an assignment of constants to the variables in the formula. If a formula contains N variables, then there are $|C|^N$ such assignments. The feature function f_i is equal to one if the ground formula is true and zero otherwise. The probability distribution of the $M_{L,C}$ can then be written as

$$P(X = x) = \frac{1}{Z} \exp\left(\sum_i w_i n_i(x)\right),$$

where $n_i(x)$ is the number of formula groundings that is true for F_i and where the weight w_i is associated with formula F_i in L.

The joint distribution $P(X = x)$ will be maximized when large weights are assigned to formulae that are frequently true. In fact, the larger the weight, the higher is the confidence that a formula is true for many groundings. Learning in MLNs consists of estimating the weights w_i from data. In learning, MLN makes a closed-world assumption and employs a pseudo-likelihood cost function, which is the product of the probabilities of each node given its Markov blanket. Optimization is performed using a limited memory BFGS algorithm.

The simplest form of inference in a MLN concerns the prediction of the truth value of a ground predicate given the truth values of other ground predicates. For this task, an efficient algorithm can be derived: In the first phase of the algorithm, the minimal subset of the ground Markov network is computed that is required to calculate the conditional probability of the queried ground predicate. It is essential that this subset is small since in the worst case, inference could involve all nodes. In the second phase, the conditional probability is then computed by applying Gibbs sampling to the reduced network.

Finally, there is the issue of structural learning, which, in this context, means the learning of first-

order formulae. Formulae can be learned by directly optimizing the pseudo-likelihood cost function or by using ILP algorithms. For the latter, the authors use CLAUDIAN (De Raedt and Dehaspe 1997), which can learn arbitrary first-order clauses (not just Horn clauses, as in many other ILP approaches).

An advantage of MLNs is that the features and thus the dependency structure are defined using a well-established logical representation. On the other hand, many people are unfamiliar with logical formulae and might consider the PRM framework to be more intuitive.

Relational Markov Networks (RMNs)

RMNs generalize many concepts of PRMs to undirected relational models (Taskar et al. 2002). RMNs use conjunctive database queries as clique templates, where a clique in an undirected graph is a subset of its nodes such that every two nodes in the subset are connected by an edge. RMNs are mostly trained discriminately. In contrast to MLNs and similarly to PRMs, RMNs do not make a closed-world assumption during learning.

Relational Latent Variable Models

In the approaches described so far, the structures in the graphical models were either defined using expert knowledge or were learned directly from data using some form of structural learning. Both can be problematic since appropriate expert domain knowledge might not be available, while structural learning can be very time consuming and possibly results in local optima which are difficult to interpret. In this context, the advantage of relational latent variable models is that the structure in the associated graphical models is purely defined by the entities and relations in the domain.

The additional complexity of working with a latent representation is counterbalanced by the great simplification by avoiding structural learning. In the following discussion, we assume that data is in triple format; generalizations to relational databases have been described (Krompaß et al. 2014; Xu 2006).

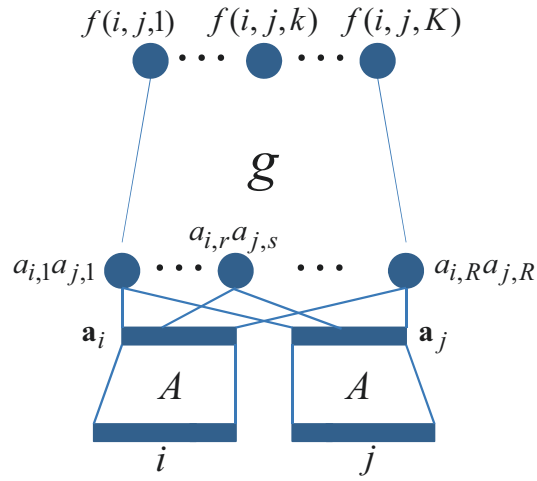
The IHRM: A Latent Class Model

The infinite hidden relational model (IHRM) (Xu 2006) (aka infinite relational model (Kemp et al. 2006)) is a generalization to a probabilistic mixture model where a latent variable with states $1,\ldots, R$ is assigned to each entity e_i. If the latent variable for subject $s = i$ is in state r and the latent variable for object $o = j$ is in state q, then the triple $(s = i, p = k, o = j)$ exists with probability $P(X_{k(i,j)}|r, q)$. Since the latent states are unobserved, we obtain

$$P\left(X_{k(i,j)}|\, i,j\right) = \sum_{r,q} P(r|\, i)P(q|j)P\left(X_{k(i,j)}|\, r,q\right)$$

which can be implemented as the sum-product network of Fig. 5.

In the IHRM, the number of states (latent classes) in each latent variable is allowed to be



Relational Models, Fig. 5 The architecture of RESCAL and the IHRM. In the bottom layer (input), the index units for subject $s = i$ and object $o = j$ are activated (one hot encoding). A is a weight matrix. The second layer calculates the latent representations $\mathbf{a}_i$ and $\mathbf{a}_j$. The following layer forms component-wise products. The output layer then calculates $f(i, j, k) = \sum_{r,q} a(i, r)a(j, q)g(k, r, q)$ for predicate $p = k$. For RESCAL, $P(X_{k(i,j)}|i, j) = sig(f(i, j, k))$. In the IHRM, we identify $P(r|i) = a(i, r)$, $P(q|j) = a(j, q)$, $P(X_{k(i,j)}|r, q) = g(k, r, q)$, and $P(X_{k(i,j)}|i, j) = f(i, j, k)$. For the IHRM, the factors must be nonnegative and properly normalized. R is the number of latent dimension and K is the number of relations

infinite, and fully Bayesian learning is performed based on a Dirichlet process mixture model. For inference, Gibbs sampling is employed where only a small number of the infinite states are occupied in sampling, leading to a clustering solution where the number of states in the latent variables is automatically determined. Models with a finite number of states have been studied as stochastic block models (Nowicki and Snijders 2001).

Since the dependency structure in the ground Bayesian network is local, one might get the impression that only local information influences prediction. This is not true, since latent representations are shared, and in the ground Bayesian network, the latter are parents to the random network variables $X_{k,(i,j)}$. Thus, common children with evidence lead to interactions between the parent latent variables and information can propagate in the network of latent variables.

The IHRM has a number of key advantages. First, no structural learning is required, since the directed arcs in the ground Bayesian network are directly given by the structure of the triple graph. Second, the IHRM model can be thought of as an infinite relational mixture model, realizing hierarchical Bayesian modeling. Third, the mixture model can be used for a cluster analysis providing insight into the relational domain.

The IHRM has been applied to social networks, recommender systems, for gene function prediction and to develop medical recommender systems. The IHRM was the first relational model applied to trust learning (Rettinger et al. 2008).

In Airoldi et al. (2008), the IHRM is generalized to a mixed-membership stochastic block model, where entities can belong to several classes.

RESCAL: A Latent Factor Model

The RESCAL model was introduced in Nickel et al. (2011) and follows a similar dependency structure as the IHRM as shown in Fig. 5. The main differences are that, first, the latent variables do not describe entity classes but are latent entity factors and that, second, there are no nonnegativity or normalization constraints on the factors. The probability of a triple is calculated with

$$f(i,j,k) = \sum_{r,q} a(i,r)a(j,q)g(k,r,q) \qquad (2)$$

as

$$P\left(X_{k(i,j)} \middle| i,j\right) = sig(f(i,j,k))$$

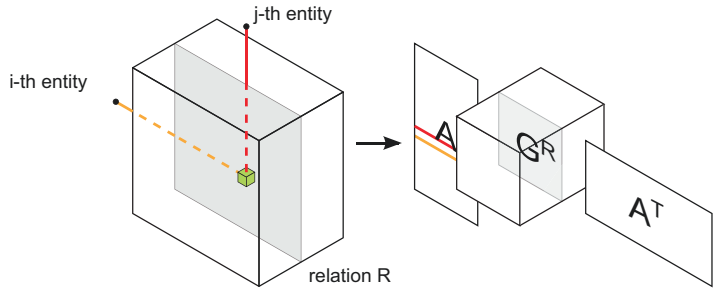
where $sig(x) = 1/(1 + \exp - x)$.

As in the IHRM, factors are unique to entities and this leads to interactions between the factors in the ground Bayesian network, enabling the propagation of information in the network of latent factors. The relation-specific matrix $G^R = g(k,:,:)$ encodes the factor interactions for a specific relation, and its asymmetry permits the representation of directed relationships.

The calculation of the latent factors is based on the factorization of a multi-relational adjacency tensor where two modes represent the entities in the domain and the third mode represents the relation type (Fig. 6). With a closed-world assumption and a squared error cost function, efficient alternating least squares (ALS) algorithm can be used; for local closed-world assumptions and open-world assumptions, stochastic gradient descent is being used.

The relational learning capabilities of the RESCAL model have been demonstrated on classification tasks and entity resolution tasks, i.e., the mapping of entities between knowledge bases. One of the great advantages of the RESCAL model is its scalability: RESCAL has been applied to the YAGO ontology (Suchanek et al. 2007) with several million entities and 40 relation types (Nickel et al. 2012)! The YAGO ontology, closely related to DBpedia (Auer et al. 2007) and the Google Knowledge Graph (Singhal 2012), contains formalized knowledge from Wikipedia and other sources.

RESCAL is part of a tradition on relation prediction using factorization of matrices and tensors. Yu et al. (2006) describe a Gaussian process-based approach for predicting a single



Relational Models, Fig. 6 In RESCAL, Eq. 2 describes a tensor decomposition of the tensor $\mathcal{F} = f(:, :, :)$ into the factor matrix $A = a(:, :)$ and core tensor $\mathcal{G} = g(:, :, :)$. In the multi-relational adjacency tensor on the left, two modes represent the entities in the domain, and the third mode represents the relation type. The i-th row of the matrix A contains the factors of the i-th entity. G^R is a slice in the $\mathcal{G}$-tensor and encodes the relation-type specific factor interactions. The factorization can be interpreted as a constrained Tucker2 decomposition.

relation type, which has been generalized to a multi-relational setting in Xu et al. (2009).

A number of variations and extensions exist. The SUNS approach (Tresp et al. 2009) is based on a Tucker1 decomposition of the adjacency tensor, which can be computed by a singular value decomposition (SVD). The neural tensor network (Socher et al. 2013) combines several tensor decompositions. Approaches with a smaller memory footprint are TransE (Bordes et al. 2013) and HolE (Nickel et al. 2015). The multiway neural network in the Knowledge Vault project (Dong et al. 2014) combines the strengths of latent factor models and neural networks and was successfully used in semiautomatic completion of knowledge graphs. Nickel et al. (2016) have a recent review on the application of relational learning to knowledge graphs.

Key Applications

Typical applications of relational models are in social networks analysis, knowledge graphs, bioinformatics, recommendation systems, natural language processing, medical decision support, and Linked Open Data.

Future Directions

As a number of publications have shown, best results can be achieved by committee solutions integrating factorization approaches with user-defined or learned rule patterns (Dong et al. 2014; Nickel et al. 2014). The most interesting application in recent years was in projects involving large knowledge graphs, where performance and scalability could clearly be demonstrated (Dong et al. 2014; Nickel et al. 2016). The application of relational learning to sequential data and time series opens up new application areas, for example, in clinical decision support and sensor networks (Esteban et al. 2015, Esteban et al. 2016). Tresp et al. (2015) study the relevance of relational learning to cognitive brain functions.

Cross-References

- ▶ Collection and Analysis of Relational Data from Digital Archives
- ▶ Collective Classification
- ▶ Collective Classification: Structural Features
- ▶ Combining Link and Content for Community Detection
- ▶ Data Mining
- ▶ Eigenvalues: Singular Value Decomposition
- ▶ Gibbs Sampling
- ▶ Linked Open Data
- ▶ Machine Learning
- ▶ Markov Graphs
- ▶ Markov Networks
- ▶ Markov Random Fields
- ▶ Matrix Decomposition

- Matrix Factorization
- Multidimensional Embedding
- Multidimensional Social Networks
- Multi-relational Network
- Multirelational Social Networks
- Missing Data
- Network Clustering
- Networked Learning
- Principal Component Analysis
- Probabilistic Graphical Models
- Probabilistic Logic and Relational Models
- RDF
- Recommender Systems: Models and Techniques
- Relational Analysis
- Relational Data Mining
- Relational Learning
- Relational Network Classification and Its Applications in Recommender Systems
- Relationship Mining
- Relationships
- Statistical Research in Networks: Looking Forward

References

Airoldi EM, Blei DM, Fienberg SE, Xing EP (2008) Mixed membership stochastic blockmodels. J Mach Learn Res 9:1981–2014

Auer S, Bizer C, Kobilarov G, Lehmann J, Cyganiak R, Ives ZG (2007) Dbpedia: a nucleus for a web of open data. In ISWC/ASWC, pp 722–735

Bordes A, Usunier N, Garcia-Duran A, Weston J, Yakhnenko O (2013) Translating embeddings for modeling multi-relational data. In: Advances in neural information processing systems, pp 2787–2795

De Raedt L, Dehaspe L (1997) Clausal discovery. Mach Learn 26(2–3):99–146

Domingos P, Richardson M (2007) Markov logic: a unifying framework for statistical relational learning. In: Getoor L, Taskar B (eds) Introduction to statistical relational learning. MIT Press, Cambridge, MA, pp 339–369

Dong X, Gabrilovich E, Heitz G, Horn W, Lao N, Murphy K, Strohmann T, Sun S, Zhang W (2014) Knowledge vault: a web-scale approach to probabilistic knowledge fusion. In: Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining, pp 601–610. ACM

Dzeroski S (2007) Inductive logic programming in a nutshell. In: Getoor L, Taskar B (eds) Introduction to statistical relational learning. MIT Press, Cambridge, MA, pp 57–92

Esteban C, Schmidt D, Krompaß D, Tresp V (2015) Predicting sequences of clinical events by using a personalized temporal latent embedding model. In: Healthcare informatics (ICHI), 2015 International conference on, pp 130–139. IEEE

Esteban C, Tresp V, Yang Y, Baier D, Krompaß S (2016) Predicting the co-evolution of event and knowledge graphs. In: International conference on information fusion

Friedman N, Getoor L, Koller D, Pfeffer A (1999) Learning probabilistic relational models. In: IJCAI, pp 1300–1309

Getoor L, Friedman N, Koller D, Pferrer A, Taskar B (2007) Probabilistic relational models. In: Getoor L, Taskar B (eds) Introduction to statistical relational learning. MIT Press, Cambridge, MA, pp 129–174

Heckerman D, Chickering DM, Meek C, Rounthwaite R, Kadie CM (2000) Dependency networks for inference, collaborative filtering, and data visualization. J Mach Learn Res 1:49–75

Heckerman D, Meek C, Koller D (2007) Probabilistic entity-relationship models, prms, and plate models. In: Getoor L, Taskar B (eds) Introduction to statistical relational learning. MIT Press, Cambridge, MA, pp 201–238

Hofmann R, Tresp V (1997) Nonlinear markov networks for continuous variables. In: NIPS

Jaeger M (1997) Relational bayesian networks. In: UAI, pp 266–273

Kemp C, Tenenbaum JB, Griffiths TL, Yamada T, Ueda N (2006) Learning systems of concepts with an infinite relational model. In: AAAI, pp 381–388

Kersting K, De Raedt L (2001) Bayesian logic programs. CoRR, cs.AI/0111058

Koller D, Pfeffer A (1998) Probabilistic frame-based systems. In: AAAI/IAAI, pp 580–587

Krompaß D, Jiang X, Nickel M, Tresp V (2014) Probabilistic latent-factor database models. Linked data for knowledge discovery, p 74

Krompaß D, Baier S, Tresp V (2015) Type-constrained representation learning in knowledge graphs. In: International semantic web conference, pp 640–655. Springer

Lösch U, Bloehdorn S, Rettinger A (2012) Graph kernels for rdf data. In: ESWC, pp 134–148

Muggleton S (1991) Inductive logic programming. New Gener Comput 8(4):295–318

Neville J Jensen D (2004) Dependency networks for relational data. In: ICDM, pp 170–177

Nickel M, Tresp V, Kriegel H-P (2011) A three-way model for collective learning on multi-relational data. In: ICML, pp 809–816

Nickel M, Tresp V, Kriegel H-P (2012) Factorizing yago: scalable machine learning for linked data. In: WWW, pp 271–280

Nickel M, Jiang X, Tresp V (2014) Reducing the rank in relational factorization models by including observable patterns. In: Advances in neural information processing systems, pp 1179–1187

Nickel M, Rosasco L, Poggio T (2015) Holographic embeddings of knowledge graphs. *arXiv preprint arXiv:1510.04935*
Nickel M, Murphy K, Tresp V, Gabrilovich E (2016) A review of relational machine learning for knowledge graphs. Proc IEEE 104(1):11–33
Nowicki K, Snijders TAB (2001) Estimation and prediction for stochastic blockstructures. J Am Stat Assoc 96 (455):1077–1087
Rettinger A, Nickles M, Tresp V (2008) A statistical relational model for trust learning. In: AAMAS (2), pp 763–770
Richardson M, Domingos P (2006) Markov logic networks. Mach Learn 62(1–2):107–136
Ross Quinlan J (1990) Learning logical definitions from relations. Mach Learn 5:239–266
Singhal A (2012) Introducing the knowledge graph: things, not strings. Technical report, Ofcial Google Blog, May 2012. http://googleblog.blogspot.com/2012/05/introducing-knowledge-graph-things-not.html
Socher R, Chen D, Manning CD, Ng A (2013) Reasoning with neural tensor networks for knowledge base completion. In: Advances in neural information processing systems, pp 926–934
Suchanek FM, Kasneci G, Weikum G (2007) Yago: a core of semantic knowledge. In: WWW, pp 697–706
Suciu D, Olteanu D, Ré C, Koch C (2011) Probabilistic databases, Synthesis lectures on data management. Morgan & Claypool Publishers, San Rafael
Taskar B, Abbeel P, Koller D (2002) Discriminative probabilistic models for relational data. In: UAI, pp 485–492
Tresp V, Huang Y, Bundschus M, Rettinger A (2009) Materializing and querying learned knowledge. In: First ESWC workshop on inductive reasoning and machine learning on the semantic web (IRMLeS 2009)
Tresp V, Esteban C, Yang Y, Baier S, Krompaß D (2015) Learning with memory embeddings. arXiv preprint arXiv:1511.07972
Xu Z, Tresp V, Yu K, Kriegel H-P (2006) Infinite hidden relational models. In: UAI
Xu Z, Kersting K, Tresp V (2009) Multi-relational learning with gaussian processes. In: IJCAI, pp 1309–1314
Yu K, Chu W, Yu S, Tresp V, Xu Z (2006) Stochastic relational models for discriminative link prediction. In: NIPS, pp 1553–1560

Relational Network Classification

▶ Relational Network Classification and Its Applications in Recommender Systems

Relational Network Classification and Its Applications in Recommender Systems

Tanwistha Saha[1], Huzefa Rangwala[2] and Carlotta Domeniconi[2]
[1]Technology Manufacturing Group (TMG), Intel Corporation, Hillsboro, OR, USA
[2]Department of Computer Science, George Mason University, Fairfax, VA, USA

Synonyms

Factorization machines; Relational network classification; Social network-based recommender systems; Tag-based recommender systems

Glossary

Entity	A sample in a relational dataset (also referred as a node in a network)
Matrix factorization	This is the process of factorizing a matrix into a product of two or more matrices
Neighborhood	Nodes which are linked together in a relational dataset form a neighborhood. For nonrelational dataset, samples which are similar to each other based on certain metric, form a neighborhood
Node	A vertex in a graph
Recommender systems	A class of algorithms which recommends items to users depending on the users' past history
Relational network	A dataset represented as a graph in which the nodes correspond to entities and

	edges correspond to relationships between the entities
Support vector machine	A discriminative classifier defined by a hyperplane obtained from a set of points in the feature space of input data. These points are known as *support vectors*

Definition

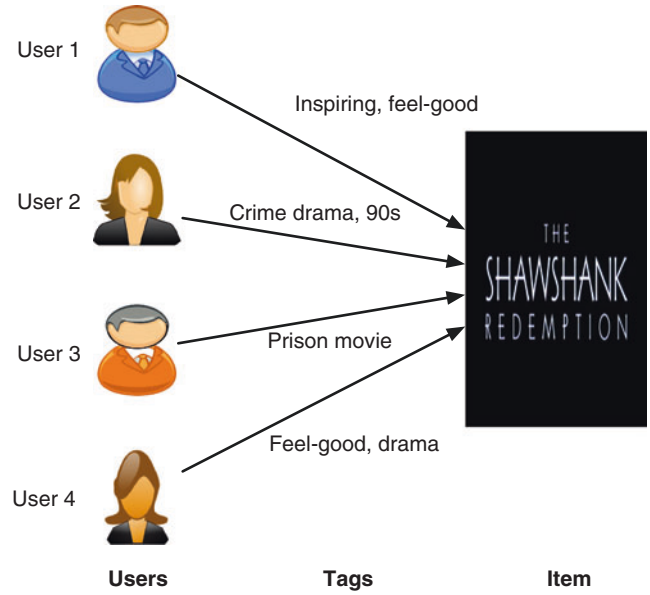
Graph	A diagram showing the relation between variable quantities, typically of two variables, each represented by a vertex
Network	A group or system of interconnected people or things
Vertex	Each angular point of a polygod, polyhedron, or other geometrical figure

Introduction

Relational network classification has emerged into a well-researched topic during the past few decades due to massive rise in the development and use of web services that allow users to share different forms of data like, articles (web-blogs), pictures (Flickr.com), videos (YouTube.com), social life (Facebook), and status updates (Twitter.com). Individuals in social network usually have a variety of interests. Classifying these interests into groups and labeling those can be accomplished using the individuals' social network profile information and their interactions with friends. This classification of individuals can be profitable for advertising agencies because they can use these class labels to recommend new products, thereby providing a personalized experience to users. This gives a whole new perspective for entity classification algorithms in a social network.

Traditional classification methods deal with nonrelational datasets in which samples are independent and identically distributed and there are no explicit relationships between the samples. Hence, classification algorithms that perform well on these nonrelational datasets cannot be directly applied to datasets where samples are somehow related and this relationship is represented by *links* or *edges* between the samples. For this purpose, researchers have come up with *Collective Classification* methods that belong to the class of algorithms which can train a classifier from the characteristic properties as well as topological properties of samples in the relational dataset. These datasets are also known as *relational networks* where the samples are referred as *nodes* and links between the nodes are referred as *edges*. Given a relational network, three possible correlations are relevant while determining the labels of nodes according to Sen et al. (2008): (*i*) the correlation between the label of a node and the observed attributes of the node, (*ii*) the correlation between the label of a node and the observed attributes and labels of neighboring nodes, and (*iii*) the correlation between an observed label of a node and the unobserved labels of its neighboring nodes. Collective classification methods jointly classify a set of related (linked) nodes by exploiting the above underlying correlations (Jensen et al. 2004). Nodes in a relational network can belong to either single class or multiple classes. This has resulted in development of multilabel collective classification algorithms for relational networks, as discussed by Kong et al. (2011) and Saha et al. (2012).

Collaborative filtering-based recommender systems identify and recommend interesting items to a given user based on the user's past rating activity. Some popular real-world applications are Netflix movie recommendation system, Amazon's product recommendation system, Spotify's music recommendation system so on and so forth. These systems improve their performances by identifying user preferences and item-related information that are explicitly available



Relational Network Classification and Its Applications in Recommender Systems, Fig. 1 Tag-based item recommendation systems

from external sources, like reviews written by users, or tags shared by users about these items. These preferences have direct impact to the ratings a user provides. Users often rate items with *tags* which provide insights regarding users preference towards the item. We assume that a user's choice of tags for an item provides additional information about the user's personal preference and characteristic features about the item. Figure 1 shows an example of such system where multiple users rate a movie and express their feelings towards the movie as short texts or tags. Since providing tags is not mandatory, users can rate items *without* tagging it. This results in a scenario when there is only a subset of users who have preference tags associated with the items they have rated. To cope with these circumstances, we have used collective classification to predict tags for the users who have not used any so far, and also for items which have not been associated with any tags.

Key Points

In this entry, we will primarily discuss the following key points:

- Prediction of tags associated with users and items using collective classification algorithms
- Integration of these tags as side information within standard neighborhood-based and matrix factorization-based methods in a collaborative filtering-based recommender system, thereby improving overall performances of both the systems

Historical Background

Collaborative filtering (CF) is a method applied to user-item rating matrix for predicting new items for users, using their past rating history. State-of-the-art collaborative filtering methods, as mentioned in the comprehensive survey by Su and Khoshgoftaar (2009), can be categorized into two types: (*i*) memory based (e.g., neighborhood-based methods) and (*ii*) model based (e.g., latent factor-based methods). Neighborhood-based algorithms can be of two types: (*i*) user based and (*ii*) item based. These approaches recommend products to a target customer either using his/her similarity with other customers (user-based) or using the similarity with the items he/she has rated in the past (item based).

The user-user or item-item similarity can be computed in several ways. Two popular approaches are: (*i*) cosine similarity, (*ii*) Pearson correlation-coefficient. The neighborhood-based methods focus on predicting *top-N* items for a target user in either one of the following two ways:

- *User-based top-N recommendation*: This type of approaches identify K most similar users (K nearest neighbors) to the target user using a similarity metric. After K most similar users are identified, the items they have rated are aggregated from the user-item rating matrix R along with their purchase frequency in this group of users. From this aggregated set of items, *top-N* most frequent items are selected and recommended to the target user.
- *Item-based top-N recommendation*: This algorithm first identifies K most similar items to the items already rated by the target user, using the similarity metrics. Then it creates the set of items I_{sel} by removing the items already rated by the user earlier. It then computes the similarity of each item in I_{sel} to the items that has been rated before by the target user. Based on the similarity scores, *top-N* items from the I_{sel} are finally recommended to the target user.

Model-based methods, on the other hand, focus on learning some characteristics from the user-item rating dataset into a "model" and use that model every time to predict rating for any new item for a target user. The prominent collaborative filtering techniques in the post-Netflix prize competition era are the latent factor models involving matrix factorization-based methods. These models are based on singular value decomposition (SVD) type approaches which decomposes the user preferences and item characteristics into a latent subspace. For example, for $|U|$ users and $|I|$ items, the user-item rating matrix R is given as the product of $|U| \times D$ user coefficient matrix U^T and $D \times |I|$ factor matrix V. SVD finds the low-rank matrix $\widehat{R} = U^T V$ which minimizes the sum squared distance to the target matrix R. Even though it looks like a minor modification, but it results in a nonconvex optimization problem which is difficult to solve, especially when R is very sparse. To address this problem, matrix factorization methods came into prominence during the Netflix grand prize competition. Mnih and Salakhutdinov (2007) proposed a probabilistic extension of SVD (probabilistic matrix factorization or PMF) which models the user-item rating matrix as a product of two low-rank user matrix and item matrix. The sum squared distance objective function for PMF is as follows:

$$f(U,V) = \frac{1}{2}\sum_{i=1}^{|U|}\sum_{j=1}^{|I|} I_{i,j}\left(R_{i,j} - U_i^T V_j\right)^2 + \frac{\lambda_U}{2}\sum_{i=1}^{|U|}\|U_i\|_{Fro}^2 + \frac{\lambda_V}{2} \times \sum_{j=1}^{|I|}\left\|V_j\right\|_{Fro}^2 \quad (1)$$

where $\lambda_U = \sigma^2/\sigma_U^2$ and $\lambda_V = \sigma^2/\sigma_V^2$ and $\|\cdot\|_{Fro}^2$ denotes Frobenius norm.

Both the neighborhood-based methods and matrix factorization approaches work well for single aspect rating prediction for users who have rated lots of items in the system. However, not much has been done to consider users' social network as *side information* into this paradigm for improving the overall performance. A state-of-the-art method by He and Chu (2010) for social network-based recommender systems leverages user's own preferences, the general acceptance of the target item, and the opinions from social friends for item recommendations. But this method only relies on metric-based similarity computation between a user's preference and all his friends' preferences for the item in a latent subspace. Users often associate multiple aspects explicitly while rating an item. These aspects are represented by the user as short text snippets or *tags* which usually signify what the user thinks about that particular item. For example, in Tripadvisor website, consider an user who rates a hotel and tags it as *solo* and *couple*, where the tags are representing the user's perspective about that hotel. If another user also tags the same hotel

with same tags or a subset of these tags, then these two users are similar in the system. Hence, in order to make personalized recommendations, tags can facilitate finding similar users or items with respect to a target user or an item, respectively, as mentioned by Saha et al. (2015).

Latent factor models (e.g., probabilistic matrix factorization, SVD++) have become the state-of-the-art methods in recommender systems. But their formulation as discussed by Mnih and Salakhutdinov (2007) and Salakhutdinov and Mnih (2008) do not involve seamless integration of side information that are sometimes available in the system. To address this problem, factorization machines (FM) were proposed by Rendle (2012) as a new model class of general predictors. These are similar to support vector machines (SVM) in a sense that they can work with any feature vector. However, unlike SVMs, factorization machines can model all pairwise nested interactions between the variables, even for very sparse input data. Factorization machine model requires input as a set of tuples (x, f(x)) where x $\in \mathbb{R}^D$ is a feature vector and f(x) is the corresponding target. A factorization machine that models all interactions up to order $d = 2$ between the D input variables is defined as:

$$f(x) = w_0 + \sum_{j=1}^{D} w_j x_j + \sum_{j=1}^{D} \times \sum_{j'=j+1}^{D} x_j x_{j'} \sum_{f=1}^{F} v_{j,f} v_{j',f} \qquad (2)$$

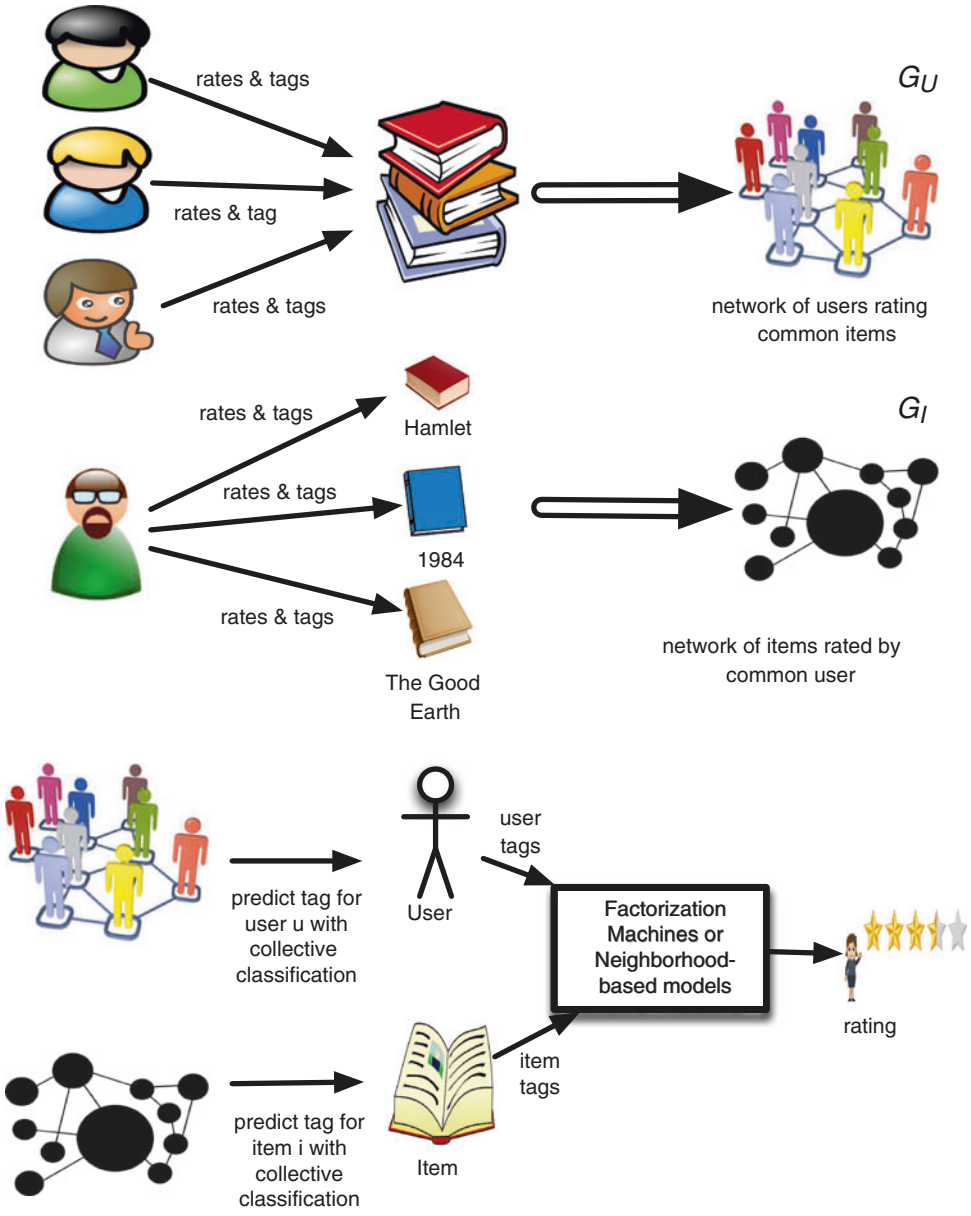
where F is the dimensionality of the factorization, and the model parameters $\Theta = \{w_0, w_1, \ldots, w_D, v_{1,1}, \ldots, v_{D,F}\}$ are: $w0 \in \mathbb{R}$, w $\in$ RD, $V \in \mathbb{R}$D×F. The main difference between the second part of Eq. 2 and polynomial regression, as mentioned by Rendle (2012), is that the interaction is not modeled as an independent parameter wj,j′. Rather, it is modeled as a factor $w_{j,j'} \approx \langle \mathrm{v}_j, \mathrm{v}_{j'} \rangle = \sum_{f=1}^{F} v_{j,f} v_{j',f}$.

In relational learning, *collective classification* methods are quite popular in classifying nodes in network datasets. As described earlier, these methods jointly classify *all* the test nodes in a network by leveraging the complex and implicit correlations between multiple entities and their labels. They are applicable towards networks having either both the node features and the topological features (Sen et al. 2008) or only the topological features (Macskassy and Provost 2007). These methods can also be applied on multilabeled networks according to Kong et al. (2011) and Saha et al. (2012). Recently, researchers have used collective classification in *tag-recommendation* systems with promising results (Preisach et al. 2010). Later on in our work, we have used multilabel collective classification on the implicit user-user network and item-item network (derived from the user-item rating matrix) to predict *tags* (or labels) for users and items. These predicted tags have been used as *user attributes* and *item attributes* in a neighborhood- and a latent factor-based collaborative filtering method (Saha et al. 2015).

Figure 2 provides an overview of the model for tag-based item recommendation system. G_U and G_I are user-user and item-item networks derived from user-item rating matrix. Not all users tag an item while rating it. Hence, the need arises for predicting tags for users who have never used any, and also for items which have not been associated with any tag. As shown in Fig. 2, collective classification is used to predict the preference tags that are associated with such an user, and at the same time can also provide descriptive information for the rated item. These "predicted tags" are then integrated as side information within neighborhood-based and latent factor-based recommendation systems to identify the most relevant items for a user and the rating of an item provided by a user, respectively. The intuition behind this approach is that tag information provides information about user preferences and description/context about items, and utilizing these tags assists in developing more effective recommender systems.

Methodology

Let U be the set of users, I be the set of items and P represent the set of preference tags that a user



Relational Network Classification and Its Applications in Recommender Systems, Fig. 2 Tag-based item recommendation systems

can use to provide additional feedback while rating an item. A four-tuple $\langle u, i, p, R_{u,i} \rangle$ represents that a user $u \in U$ has provided a final rating of $R_{u,i}$ and associated a preference tag $p \in P$ for an item $i \in I$. $R_{u,i}$ can be real valued (if explicit rating information exists) or binary (0 or 1) when there is no explicit rating captured (e.g., tagging or transaction related datasets). $\mathbf{R}$ is the user-item rating matrix which has dimension $|U| \times |I|$.

Let $U_{te} \in U$ be the users in the system for whom we do not have the tag information available. These users are referred as *test users* in the tag classification model. In order to predict one or more tags that denote *preferences* for each of these

test users, multilabel collective classification has been used (Kong et al. 2011; Saha et al. 2014). Let $U_{tr} \in U$, be the set of *training users* for whom we know *all* the tags they have used to rate items ($U_{tr} \cup Ute = U$). For each user $u \in U$, some of his rated items are purposefully hidden during the model training phase. This generates a partially complete user-item rating matrix for training where the number of users and number of items are same as in the original matrix, but the rating entries for an user depend on the items that are not hidden. The task of the recommender system model is to be able to predict ratings for some or all of these hidden items. From the user-item rating matrix for all users in U and the items in I, we can construct:

- A user-user network $G_U = (V_U, E_U)$ where V_U is the set of user nodes ($|V_U| = |U|$) and E_U is the set of edges in G_U. If user x and user y both rate at least one item, then there is a link between these two users in the user network G_U.
- An item-item network $G_I = (V_I, E_I)$ where V_I is the set of item nodes ($|VI| = |I|$) and E_I is the set of edges in G_I. If item w and item v are both bought by at least one user, then there is a link between these two items in the item network G_I.

The label(s) of a training user in the user-user network are the tag(s) he has used to rate items, and the label(s) of an item in an item-item network are the tag(s) that were used previously by users on that item. If $\mathbf{t}_x$ and $\mathbf{t}_y$ be the tag vectors for any pair of users x and y respectively, then pairwise user similarity matrix $\mathbf{S}^U$ can be precomputed as cosine similarity between the tag vectors $\mathbf{t}_x$ and $\mathbf{t}_y$:

$$S^U_{x,y} = \frac{\mathbf{t}_x \mathbf{t}_y{}^T}{\|\mathbf{t}_x\|_2 \|\mathbf{t}_y\|_2} \tag{3}$$

Similarly, the pairwise item similarity matrix $\mathbf{S}^I$ can be precomputed using item tag vectors $\mathbf{t}_w$ and $\mathbf{t}_v$ for items w and v respectively, and using Eq. 3. Once the tag vectors for test users are determined through collective classification, we can employ the following collaborative filtering models depending on the type of dataset we have (i.e., if the ratings are real valued or binary).

Neighborhood-Based Method

According to Su and Khoshgoftaar (2009), these memory-based methods can be either *user based* or *item based*. In their original form, user-based neighborhood models estimate unknown ratings based on pastratings of "likeminded" users. To estimate the rating $\widehat{R}_{u,i}$ for a user u on a specific item i, these approaches identify the neighborhood N_u (users similar to u), who have rated item i. The predicted rating $\widehat{R}_{u,i}$ is given by:

$$\widehat{R}_{u,i} = \frac{\sum_{v \in N_u} S_{u,v} R_{v,i}}{\sum_{v \in N_u} S_{u,v}} \tag{4}$$

where $S_{u,v}$ denotes the similarity between users u and v and is used for determining the neighborhood Nu for computing the weighted average. With only the rating matrix $\mathbf{R}$, choices for computing similarity include cosine similarity and Pearson correlation coefficient. An alternative approach to the user-oriented method is the item-oriented approach (Sarwar et al. 2001), where the predicted rating utilizes the known ratings made by the same user on similar (or neighborhood of) items. The predicted rating $\widehat{R}_{u,i}$ for a user u on an item i is given by:

$$\widehat{R}_{u,i} = \frac{\sum_{j \in N_i} S_{i,j} R_{u,j}}{\sum_{j \in N_i} S_{i,j}} \tag{5}$$

where N_i is the set of items that are "most similar" to item i. $S_{i,j}$ is the similarity between items i and j computed in the user space.

The tags for a user and/or item is used as side information (beyond the rating matrix) within these neighborhood models. For each user u, the predicted tag vector is represented as $\widehat{\mathbf{t}}_u = \left(\widehat{t}_{u1}, \ldots, \widehat{t}_{u|P|}\right)$ where $\widehat{t}_{up} \in \{0, 1\}$ denotes the absence or presence of the p-th tag. Instead of using the rating matrix $\mathbf{R}$ to define the neighborhood for a user u in user-based approach, the neighborhood N_u is computed using the predicted tag vectors for the users.

Given a pair of users x and y with predicted tag vectors $\widehat{\mathbf{t}}_x$ and $\widehat{\mathbf{t}}_y$ respectively, the cosine similarity between the two tag vectors is computed using Eq. 3. Given the neighborhood N_u computed using the user tag vectors, this model identifies all the items rated by users in that neighborhood, denotes the set by I_{N_u}, and estimates their ratings using Eq. 4.

The top-N most popular items are selected from the set I_{N_u}, and this approach is referred as KNN with user tags (**UT-KNN** in the experiments). Analogous to the UT-KNN model, an item-oriented approach is referred as KNN with item tags (**IT-KNN** in the experiments). In this case, every item i is associated with a predicted tag vector denoted by $\widehat{\mathbf{t}}_i$. The final rating is predicted using the neighborhood of candidate items for the target item, following Eq. 5. Going further, the user- and item-associated predicted tag vectors are combined in a neighborhood-based model and is referred as KNN with User and Item Tags (**UIT-KNN** in the experiments). In this case, the similarity score between user u and item i is computed using the cosine similarity between the predicted tag vectors of user u and item i. The rest of the steps are same as Eq. 4. This allows the usage of side information with predicted tags in both the user and item space.

Latent Factor Model

Using the factorization machines proposed by Rendle (2012), it is possible to incorporate the tags associated with users and items as feature vectors of samples used to train the model. According to Saha et al. (2015), these models are described as follows:

- **Factorization Machines with User and Item Tags**: Assuming we have categorical data in the form of user-item rating matrix **R**, then it is easy to represent this information as a feature vector x in the factorization machine framework. Going further, if we want to incorporate the predicted user attribute vector as the tag vector $\widehat{\mathbf{t}}_u = \left(\widehat{t}_{u1}, \widehat{t}_{u2}, \cdots, \widehat{t}_{u|L|}\right)$ for an user u, and the predicted item attribute vector as the tag vector $\widehat{\mathbf{t}}_i = \left(\widehat{t}_{i1}, \widehat{t}_{i2}, \cdots, \widehat{t}_{i|L|}\right)$ for an item i in $d = 2$ way factorization machine, then the feature vector x for the $\langle u, i\rangle$ pair will look like:

$$\mathrm{x} = \Bigg(\underbrace{0, \cdots, 1, \cdots, 0}_{|U|}, \underbrace{0, 0, \cdots 1, \cdots, 0}_{|I|}, \underbrace{\widehat{t}_{u1}, \widehat{t}_{u2}, \cdots, \widehat{t}_{u|L|}}_{\text{attributesofuser}\,u}, \underbrace{\widehat{t}_{i1}, \widehat{t}_{i2}, \cdots, \widehat{t}_{i|L|}}_{\text{attributesofitem}\,i} \Bigg) \tag{6}$$

where the first $|U| + |I|$ features has the value 1 only at two positions corresponding to the active user u and active item i, i.e., the $\langle u, i\rangle$ pair for which we have to predict the rating (during training, we have the ground truth rating value available). In this formulation, the tags given by the users are used as user attributes and tags associated with the items are used as item attributes. Hence, both types of attributes have dimensionality $|L|$ because there are $|L|$ number of total tags. The model for predicting target rating $\widehat{R}_{ui} = f(\mathbf{x})$ corresponding to the user-item pair $\langle u, i\rangle$ with their attribute tag vectors, will have the following representation:

$$\begin{aligned} f(x) = \; & w_0 + w_u + w_i + \sum_{s=1}^{|L|} \widehat{t}_{us} w_s \\ & + \sum_{s=1}^{|L|} \widehat{t}_{us} (\mathbf{v}_i, \mathbf{v}_s) + \sum_{s=1}^{|L|} \sum_{s'>s}^{|L|} \widehat{t}_{us} \widehat{t}_{us'} (\mathbf{v}_s, \mathbf{v}_{s'}) \\ & + \sum_{q=1}^{|L|} \sum_{q'>q}^{|L|} \widehat{t}_{iq} \widehat{t}_{iq'} \left(\mathbf{v}_q, \mathbf{v}_{q'}\right) + \sum_{q=1}^{|L|} \widehat{t}_{iq} w_q \\ & + \sum_{q=1}^{|L|} \widehat{t}_{iq} \left(\mathbf{v}_u, \mathbf{v}_q\right) + \sum_{f=1}^{F} v_{uf} v_{if} \end{aligned} \tag{7}$$

where the feature vector x is replaced by the expression in Eq. 6 and the model parameters are: $\Theta = \{w_0, w_1, \ldots, w_{\mathrm{D}}, v_{1,1}, \ldots, v_{\mathrm{D,F}}\}$ as mentioned in Eq. 2. The fifth term represents "attribute mapping" that uses linear regression to map user attributes to factors. The sixth and seventh terms are two-way interactions between user attributes and item attributes, respectively. The ninth term is to map the item attributes to factors. The fourth, eighth, and the last terms capture one-way interactions. The dimension (i.e., D) of the feature

vector **x** is $|U| + |I| + 2 \times |L|$. This method is referred as **UIT-FM** in the experiments.

- **Factorization Machines with User Tags**: This is derived from the formulation in Eq. 6 except that, only user attributes (tags) are used in the representation. The feature vector **x** looks like:

$$\mathrm{x} = \left(\underbrace{0, \cdots, 1, \cdots, 0,}_{|U|} \underbrace{0, \cdots, 1, \cdots, 0}_{|I|} \underbrace{, \widehat{t}_{u1}, \widehat{t}_{u2}, \cdots, \widehat{t}_{u|L|}}_{\text{attributesofuser } u} \right) \tag{8}$$

The predictive model $f(\mathrm{x})$ is represented as before, except that there are no item attributes (tags) in this model. The dimension of the feature vector x is $|U| + |I| + |L|$. This method is referred as **UT-FM** in the experiments.

- **Factorization Machines with Item Tags**: This is similar to the models described earlier, except that only item attributes (tags) are used in the formulation. The feature vector x looks like:

$$\mathrm{x} = \left(\underbrace{0, \cdots, 1, \cdots, 0,}_{|U|} \underbrace{0, \cdots, 1, \cdots, 0}_{|I|} \underbrace{, \widehat{t}_{i1}, \widehat{t}_{i2}, \cdots, \widehat{t}_{i|L|}}_{\text{attributesofitem } i} \right) \tag{9}$$

The predictive model $f(\mathrm{x})$ is formulated as before, except that there are no user attributes (tags) present in this model. The dimension of the feature vector x for this case is $|U| + |I| + |L|$. This method is referred as **IT-FM** in experiments.

Experimental Setup

Datasets

Table 1 lists all the datasets (along with their properties) used in the experiments. Bibsonomy (http://www.kde.cs.uni-kassel.de/bibsonomy/dumps/) and Delicious (http://grouplens.org/datasets/hetrec-2011/) datasets have tagging information only, and no item ratings are available. Hence, only neighborhood-based methods were tested on these datasets. Tripadvisor (http://nemis.isti.cnr.it/~marcheggiani/datasets/) dataset contains tag as well as rating information, hence latent factor-based models were experimented on this dataset. The performance of the tag-based factorization machines are measured by computing root mean square error (RMSE) on predicted rating on items with respect to the true rating given by all test users for those items. MovieLens is a popular dataset for recommendation systems, obtained from the same source as Delicious dataset. The subset of the original dataset used in this paper contains tags used by users on the movies, in addition to their ratings. LibraryThing (http://www.macle.nl/tud/LT/) is an online catalog where users can tag and rate the books they have read. For both MovieLens and LibraryThing datasets, the aim is similar to that of Tripadvisor dataset, i.e., to be able to predict the rating of an item that the user might like in future, leveraging the tag and rating information provided by that user for other items.

Relational Network Classification and Its Applications in Recommender Systems, Table 1 Description of datasets

Name	$\lvert U\rvert$	$\lvert I\rvert$	$\lvert T\rvert$	#Ratings	Rating density	#Tags per user	#Tags per item	Avg # users tagging or rating an item	Avg # items tagged or rated by user
Bibsonomy	116	361	75	2230	0.05	15.14	8.52	6.18	19.22
Delicious	636	1027	45	4180	0.01	9.76	5.28	4.07	6.57
Tripadvisor	1202	1890	5	4607	≈ 0	1.73	1.74	2.44	3.83
MovieLens	704	3146	58	197,025	0.09	2.57	2.22	62.63	279.86
LibraryThing	1500	3500	21	102,455	0.02	8.74	5.06	29.27	68.30

Results

For the sake of brevity, we discuss part of our results in this entry and refer readers to our original paper for the comprehensive set of experiments (Saha et al. 2015). Performance of neighborhood-based models were validated using top-N hit-rate metric for datasets which only had tagging information (Deshpande and Karypis 2004). For each user, we hide h% of the items for validation purpose. If any of the top-N predicted items fall in the list of h% hidden items for an user, then we consider it as a *hit*. For datasets where we have rating information available, we use the standard metric root mean square error (RMSE) following Su and Khoshgoftaar (2009) in order to compare the results with other baseline methods.

We experimented with two setups for splitting the data into training and testing set: (*i*) **standard split**: for each user, 70% of the randomly chosen items rated by the user were included in training set and 30% ($h = 30$) were hidden for testing; (*ii*) **random split**: for each user, we randomly chose either 70% or 50% or 30% items rated by the user for training set and separated the rest of the items for testing. Based on the percentage of items that were hidden during training (using either one of the two setups), parameter h is set and an updated user-item rating matrix is created. The user-user network G_U and item-item network G_I are constructed using this updated user-item rating matrix. These networks are used within our collective classification framework for predicting the tags for both the users and the items. A total of five independent trials were conducted for both standard split and random split. Table 2 shows the results for datasets which had explicit rating information (Tripadvisor, MovieLens, and LibraryThing), whereas Table 3 shows the results for datasets which did not have explicit rating information (Bibsonomy and Delicious).

From Table 2, it appears that the predicted tag-based method (IT-FM) performs better than the

Relational Network Classification and Its Applications in Recommender Systems, Table 2 Performance of latent factor-based methods with respect to root mean square error (↓) for Tripadvisor/MovieLens/LibraryThing datasets with standard split and random split

Tripadvisor							
Training split	FM	UT-FM	UT-FM-G	IT-FM	IT-FM-G	UIT-FM	UIT-FM-G
Standard	1.06	1.05	1.05	**1.04**	1.03	1.05	1.04
Random	1.07	1.07	1.06	**1.06**	1.05	1.07	1.06
MovieLens							
Standard	0.76	0.77	0.77	**0.75**	0.75	0.76	0.76
Random	0.78	0.79	0.79	**0.77**	0.77	0.78	0.78
LibraryThing							
Standard	0.78	0.79	0.79	0.77	0.77	0.78	0.77
Random	0.79	0.79	0.79	0.78	0.78	0.79	0.78

Relational Network Classification and Its Applications in Recommender Systems, Table 3 Performance of neighborhood-based methods with respect to Top-N Hit rate (↑) for Bibsonomy/Delicious datasets with standard split and random split

Bibsonomy								
Training split	U-CF-Pop	I-CF	UT-KNN	UT-KNN-G	IT-KNN	KNN-GIT-	UIT-KNN	UIT-KNN-G
Standard	**0.32**	0.22	0.28	0.37	0.12	0.32	0.14	0.60
Random	**0.39**	0.27	0.31	0.47	0.22	0.45	0.28	0.69
Delicious								
Standard	**0.07**	0.05	0.04	0.10	0.03	0.07	0.03	0.17
Random	**0.09**	0.06	0.06	0.14	0.03	0.10	0.07	0.22

baseline factorization machine (FM) for all the three datasets (Table 2). There are three ground truth models (UIT-FM-G, IT-FM-G, and UT-FM-G respectively for tag-based factorization machines, and, UIT-KNN-G, IT-KNN-G, and UT-KNN-G respectively for tag-based neighborhood models) where true tags were used instead of predicted tags from collective classification. However, as expected, the tag-based FM methods cannot beat the ground truth models. The best performance of the ground truth models in comparison to all other methods proves that the knowledge of true tags helps in tag-based item recommendation using factorization machines. For predicted tag-based neighborhood methods, we see a different behavior in Table 3. The baseline method U-CF-Pop, which is the standard user-user similarity based collaborative filtering method discussed by Su and Khoshgoftaar (2009) for selecting the "most similar" users with respect to the target user, performs better than tag-based neighborhood methods UIT-KNN, UT-KNN, and IT-KNN for both Delicious and Bibsonomy datasets. However, like before, the ground truth model UIT-KNN-G is the best performer among all the comparative methods in this set. This also proves the hypothesis that knowledge of true tags facilitates the performance of collaborative filtering recommender systems. Looking carefully, it appears that item tag prediction (IT-KNN) is comparatively performing worse than user tag prediction (UT-KNN) for these two datasets, and that is why the overall performance of UIT-KNN is getting affected.

Key Applications and Future Directions

With the increasing popularity of social networks, researchers have taken keen interest to leverage relational learning in the recommender systems paradigm in order to improve performance of item recommendation. Experiments conducted by us on several real-world relational datasets (e.g., Bibsonomy, Tripadvisor, MovieLens) show that predicting preference tags using collective classification has a huge benefit in recommender systems especially in situations when the tags denoting the likings of *all* the users in network are not available (Saha et al. 2015). Accurate tag prediction using collective classification framework is also advantageous in product advertising and marketing industries to provide personalized experience to their users. Collective classification of users' and items' preference tags in relational networks can also be used effectively in trust-aware recommender system paradigm where the trust criteria is used to define the neighborhood of the target user. There has been some work along these lines by Ma et al. (2011), but not much has been done related to circumstances when the preference tags or likings for all users in the social networks are *not* available to the recommender system model. This is an ideal scenario for collective classification of entities in relational data. Hence, it is obvious that leveraging this network classification framework will yield better performance for the recommender systems. Social network analysis and recommender systems have emerged as two important pillars for machine learning-based applications in the industry. Bridging the gap between these two areas of research will become hugely beneficial for the next generation of recommender systems methods.

Conclusion

In this book entry, we have discussed methods of applying collective classification for predicting user preference tags and item descriptive tags as side information for improving state-of-the-art collaborative filtering recommender systems. In general, most recommender systems do not have tag information available for all users/items. Collective classification can be used on implicit user-user and item-item networks to predict tags for those users/items. In this entry, we have shown how preference tags can be easily incorporated as side information into state-of-the-art neighborhood-based methods and latent factor models. When the number of users/items with known tags is very small, there is also scope of using active learning to train the collective classifier by incrementally adding informative samples to training set in each round. Results of such methods on real-world relational datasets have been quite promising (Saha et al.

2014, 2015). However, there is still much to be done in order to make these methods scalable for relational networks with billions of users, which is a challenge in any social network-based recommender system.

Cross-References

- ▶ Collective Classification
- ▶ Learning Networks
- ▶ Matrix Factorization
- ▶ Mining Complex Networks
- ▶ Modeling Social Preferences Based on Social Interactions
- ▶ Recommender Systems Based on Social Networks

References

Deshpande M, Karypis G (2004) Item-based top-n recommendation algorithms. ACM Trans Inform System 22(1):143–177

He J, Chu WW (2010) A social network-based recommender system (SNRS). Springer, New York

Jensen D, Neville J, Gallagher B (2004) Why collective inference improves relational classification. In: Proceedings of the tenth ACM SIGKDD, ACM, pp 593–598

Kong X, Shi X, Philip S (2011) Multi-label collective classification. In: Proceedings of SIAM international conference on data mining (SDM), pp 618–629

Ma H, Zhou D, Liu C, Lyu MR, King I (2011) Recommender systems with social regularization. In: Proceedings of the fourth ACM international conference on web search and data mining, ACM, pp 287–296

Macskassy S, Provost F (2007) Classification in networked data: a toolkit and a univariate case study. J Machine Learning Res 8:935–983

Mnih A, Salakhutdinov R (2007) Probabilistic matrix factorization. Advances in neural information processing systems, In, pp 1257–1264

Preisach C, Marinho LB, Schmidt-Thieme L (2010) Semi-supervised tag recommendation using untagged resources to mitigate cold-start problems. In: Advances in knowledge discovery and data mining. Springer, New York, pp 348–357

Rendle S (2012) Factorization machines with libfm. ACM Trans Systems Technol 3(3):57

Saha T, Rangwala H, Domeniconi C (2012) Multi-label collective classification using adaptive neighborhoods. In: (ICMLA), IEEE, vol 1, pp 427–432

Saha T, Rangwala H, Domeniconi C (2014) Flip: active learning for relational network classification. In: Machine learning and knowledge discovery in databases. Springer, New York, pp 1–18

Saha T, Rangwala H, Domeniconi C (2015) Predicting preference tags to improve item recommendation. In: Proceedings of the 2015 SIAM international conference on data mining, SIAM, pp 854–872

Salakhutdinov R, Mnih A (2008) Bayesian probabilistic matrix factorization using markov chain monte carlo. In: Proceedings of the 25th international conference on machine learning, ACM, pp 880–887

Sarwar B, Karypis G, Konstan J, Riedl J (2001) Item-based collaborative filtering recommendation algorithms. In: Proceedings of the 10th international conference on World Wide Web, ACM, pp 285–295

Sen P, Namata G, Bilgic M, Getoor L, Galligher B, Eliassi-Rad T (2008) Collective classification in network data. AI Mag 29(3):93

Su X, Khoshgoftaar TM (2009) A survey of collaborative filtering techniques. Adv Artificial Intell:4. https://doi.org/10.1155/2009/421425

Relational View

- ▶ Top Management Team Networks

Relationship Extraction

- ▶ Link Prediction: A Primer

Relationship Formation

- ▶ Privacy and Disclosure in a Social Networking Community

Relationship Mining

- ▶ Inferring Social Ties
- ▶ Role Discovery

Relationships

- ▶ Social Network Analysis in an Age of Digital Information

Relative Validity Criteria for Community Mining Algorithms

Reihaneh Rabbany[1], Mansoreh Takaffoli[1], Justin Fagnan[1], Osmar R. Zaïane[1] and Ricardo Campello[1,2]
[1]Computing Science Department, University of Alberta, Edmonton, AB, Canada
[2]Department of Computer Science, University of São Paulo, São Carlos, SP, Brazil

Synonyms

Clustering evaluation; Clustering objective function; Community mining; Evaluation approaches; Graph clustering; Graph partitioning; Quality measures

Glossary

ARI (adjusted rand index)	Measures similarity of two clusterings based on pair counts
Community Structure	Clustering structure underlying a network which models regions of densely connected nodes
External Evaluation	Compares a clustering against the ground-truth clustering
Internal Evaluation	Matches a clustering with the structure of data
Network	A graph of interconnected nodes which models relationships in data
NMI (normalized mutual information)	Measures similarity of two clusterings
Relative Evaluation	Ranks different clusterings of the same dataset

Definition

Grouping data points is one of the fundamental tasks in data mining, which is commonly known as clustering if data points are described by attributes. When dealing with interrelated data, data represented in the form of nodes and their relationships and the connectivity is considered for grouping but not the node attributes; this task is also referred to as community mining. There have been a considerable number of approaches proposed in recent years for mining communities in a given network. However, little work has been done on how to evaluate community mining results. The common practice is to use an agreement measure to compare the mining result against a ground truth; however, the ground truth is not known in most of the real-world applications. In this entry, we investigate relative clustering quality measures defined for evaluation of clustering data points with attributes and propose proper adaptations to make them applicable in the context of social networks. Not only these relative criteria could be used as metrics for evaluating quality of the groupings, but also they could be used as objectives for designing new community mining algorithms.

Introduction

The recent growing trend in the data mining field is the analysis of structured/interrelated data, motivated by the natural presence of relationships between data points in a variety of the present-day applications. The structures in these interrelated data are usually represented using networks, known as complex networks or information networks; examples are the hyperlink networks of web pages, citation or collaboration networks of scholars, biological networks of genes or proteins, trust and social networks of humans, and much more.

All these networks exhibit common statistical properties, such as power law degree distribution, small-world phenomenon, relatively high transitivity, shrinking diameter, and densification power laws (Newman 2010; Leskovec et al. 2005). Network clustering, a.k.a. community mining, is one of the principal tasks in the analysis of complex networks. Many community mining algorithms have been proposed in recent years (for a survey,

refer to Fortunato 2010). These algorithms evolved very quickly from simple heuristic approaches to more sophisticated optimization-based methods that are explicitly or implicitly trying to maximize the goodness of the discovered communities. The broadly used explicit maximization objective is the modularity, first introduced by Newman and Girvan (2004).

Although there have been many methods presented for detecting communities, very little work has been done on how to evaluate the results and validate these methods. The difficulties of evaluation are due to the fact that the interesting communities that have to be discovered are hidden in the structure of the network; thus, the true results are not known for comparison. Furthermore, there are no other means to measure the goodness of the discovered communities in a real network. We also do not have any large enough dataset with known communities, often called ground truth, to use as a benchmark to generally test and validate the algorithms. The common practice is to use synthetic benchmark networks and compare the discovered communities with the built-in ground truth. However, it is shown that the networks generated with the current benchmarks disagree with some of the characteristics of real networks. These facts motivate investigating a proper objective for evaluation of community mining results.

Key Points

Defining an objective function to evaluate community mining is nontrivial. Aside from the subjective nature of the community mining task, there is no formal definition on the term community. Consequently, there is no consensus on how to measure "goodness" of the discovered communities by a mining algorithm. However, the well-studied clustering methods in the machine learning field are subject to similar issues, and yet there exists an extensive set of validity criteria defined for clustering evaluation, such as the Davies-Bouldin index (Davies and Bouldin 1979), Dunn index (Dunn 1974), and Silhouette (Rousseeuw 1987) (for a recent survey, refer to Vendramin et al. 2010). In this entry, we describe how these criteria could be adapted to the context of community mining in order to compare results of different community mining algorithms. Also, these criteria can be used as alternatives to modularity to design novel community mining algorithms.

In the following, we first briefly introduce well-known community mining algorithms and common evaluation approaches including available benchmarks. Next, different ways to adapt clustering validity criteria to handle comparison of community mining results are proposed. Then, we extensively compare and discuss the adapted criteria on real and synthetic networks. Finally, we conclude with a brief analysis of these results.

Historical Background

A community is roughly defined as "densely connected" individuals that are "loosely connected" to others outside their group. A large number of community mining algorithms have been developed in the last few years having different interpretations of this definition. Basic heuristic approaches mine communities by assuming that the network of interest divides naturally into some subgroups, determined by the network itself. For instance, the clique percolation method (Palla et al. 2005) finds groups of nodes that can be reached via chains of k-cliques. The common optimization approaches mine communities by maximizing the overall "goodness" of the result. The most credible "goodness" objective is known as modularity Q, proposed in Newman and Girvan (2004), which considers the difference between the fraction of edges that are within the communities and the expected such fraction if the edges are randomly distributed. Several community mining algorithms for optimizing the modularity Q have been proposed, such as fast modularity (Newman 2006). Although many mining algorithms are based on the concept of modularity, Fortunato and Barthélemy (2007) have shown that the modularity cannot accurately evaluate small communities due to its resolution limit.

Hence, any algorithm based on modularity is biased against small communities. As an alternative to optimizing modularity Q, we previously proposed the Top Leaders community mining approach (Rabbany et al. 2010), which implicitly maximizes the overall closeness of followers and leaders, assuming that a community is a set of followers congregating around a potential leader. There are many other alternative methods. One notable family of approaches mines communities by utilizing information theory concepts such as compression (e.g., Infomap Rosvall and Bergstrom (2008)) and entropy (e.g., entropy base Kenley and Cho 2011). For a survey on different community mining techniques refer to Fortunato (2010).

The standard procedure for evaluating results of a community mining algorithm is the external evaluation of results, particularly when comparing accuracy of different algorithms; which is assessing the agreement between the results and the ground truth that is known for benchmark datasets. These benchmarks are typically small real-world datasets or synthetic networks. On the other hand, there is no well-defined criterion for evaluating the resulting communities for networks without any ground truth, which is the case in most of real-world applications. The common practice is to validate the results partly by a human expert. However, the community mining problem is NP-hard; the human expert validation is limited and rather based on narrow intuition than on an exhaustive examination of the relations in the given network. Alternatively, modularity Q is sometimes reported to show the quality of discovered communities. In this entry, we investigate other potential measures for comparing different (nonoverlapping) community mining results and examine the performance of these measures parallel to the modularity Q. All these new measures are adapted from well-grounded traditional clustering criteria for evaluating data points with attributes. Recently, Vendramin et al. comprehensively compared their performances in Vendramin et al. (2010), based on the idea that the better a criterion, the more correlated is its ranking of different partitions to the ranking of an external index.

The external evaluation requires knowing the true communities. For this purpose, several generators have been proposed for synthesizing networks with built-in ground truth. GN benchmark (Girvan and Newman 2002) is the first synthetic network generator. This benchmark is a graph with 128 nodes, with expected degree of 16, and is divided into four groups of equal sizes; where the probabilities of the existence of a link between a pair of nodes of the same group and of different groups are z_{in} and $1 - z_{in}$, respectively. However, the same expected degree for all the nodes and equal-size communities are not accordant to real social network properties. LFR benchmark (Lancichinetti et al. 2008) amends GN benchmark by considering power law distributions for degrees and community sizes. Similar to GN benchmark, each node shares a fraction $1 - \mu$ of its links with the other nodes of its community and a fraction μ with the other nodes of the network. In this entry, we generate our synthetic networks using LFR benchmark, due to its more realistic structure.

There are recent studies on the comparison of different community mining algorithms in terms of evaluating their performance on synthetic and real networks. For example, refer to studies by Danon et al. (2005) and Lancichinetti and Fortunato (2009). All these studies are based on the agreements of the generated communities with the true one in the ground truth and are using GN and/or LFR benchmarks. Orman et al. (2011) further performed a qualitative analysis of the identified communities by comparing the distribution of resulting communities with the community size distribution of the ground truth. None of these studies, however, considers any different validity criteria other than modularity to evaluate the goodness of the detected communities. In this entry, we plan to examine potential validity criteria specifically defined for evaluation of community mining results. In the future, these criteria can be used not only as a means to measure the goodness of discovered communities but also as an objective function to detect communities.

Community Quality Criteria

In this section, we overview several validity criteria that could be used as relative indexes for comparing and evaluating different partitionings of a given network. All of these criteria are generalized from well-known clustering criteria. The clustering quality criteria are defined with the implicit assumption that data points consist of vectors of attributes. Consequently, their definition is mostly integrated or mixed with the definition of the distance measure between data points. The commonly used distance measure is the Euclidean distance, which cannot be defined for graphs. Therefore, we first review different possible distance measures that could be used in graphs. Then, we present generalizations of criteria that could use any notion of distance.

Distance Between Nodes

Let A denote the adjacency matrix of the graph, and let A_{ij} be the weight of the edge between nodes n_i and n_j. The distance $d(i, j)$ denotes the dissimilarity between n_i and n_j , which can be computed by one of the following measures.

Edge Path (ED)

The distance between two nodes is the inverse of their incident edge weight:

$$d_{ED}(i,j) = \frac{1}{A_{ij}}$$

For avoiding division by 0, when A_{ij} is 0, $1/\epsilon$ is returned where ϵ is a very small number; the same is true for all other formula whenever a division by zero may occur.

Shortest Path Distance (SPD)

The distance between two nodes is the length of the shortest path between them, which could be computed using the well-known Dijkstra's Shortest Path algorithm.

Adjacency Relation Distance (ARD)

The distance between two nodes is the structural dissimilarity between them, that is computed by the difference between their immediate neighborhood.

$$d_{\mathrm{ARD}}(I,j) = \sqrt{\sum_{k\neq j,\, i} \left(A_{ik} - A_{jk}\right)^2}$$

Neighbor Overlap Distance (NOD)

The distance between two nodes is the ratio of the unshared neighbors between them:

$$d_{\mathrm{NOD}}(i,j) = \frac{1|\aleph_i \cap \aleph_j|}{|\aleph_i \cup \aleph_j|}$$

where $\aleph_i$ is the set of nodes directly connected to n_i. Note that there is a close relation between this measure and the previous one, since similarly d_{NOD} could be rewritten as

$$d_{\mathrm{NOD}}(i,j) = 1 - \frac{\sum_{k\neq j,\, i} |A_{ik} + A_{jk}| - \sum_{k\neq j,\, i} |A_{ik} - A_{jk}|}{\sum_{k\neq j,\, i} |A_{ik} + A_{jk}| + \sum_{k\neq j,\, i} |A_{ik} - A_{jk}|}$$

The latter formulation of d_{NOD} in terms of the adjacency matrix can be straightforwardly generalized for weighted graphs.

Pearson Correlation Distance (PCD)

The Pearson correlation coefficient between two nodes is the correlation between their corresponding rows of the adjacency matrix:

$$C(i,j) = \frac{\sum_k (A_{ik} - \mu_i)\left(A_{jk} - \mu_j\right)}{N\sigma_i\sigma_j}$$

where N is the number of nodes, the average $\mu_i = (\sum_k A_{ik})/N$ and the variance $\sigma_i = \sqrt{\sum_k (A_{ik} - \mu_i)^2/N}$. Then, the distance between two nodes is computed as $d_{PCD}(i, j) = 1 - C(i, j)$, which lies between 0 (when the two nodes are most similar) and 2 (when the two nodes are most dissimilar).

ICloseness Distance (ICD)

The distance between two nodes is computed as the inverse of the connectivity between their common neighborhood:

$$d_{\mathrm{ICD}}(i,j) = \frac{1}{\sum_{k \in \aleph_i \cap \aleph_j} ns(k,i)ns(k,j)}$$

where $ns(k, i)$ denotes the neighboring score between nodes k and i that is computed iteratively (for complete formulation, refer to Rabbany and Zaïane 2011).

Community Centroid

In addition to the notion of distance measure, most of the cluster validity criteria use averaging between the numerical data points to determine the centroid of a cluster. The averaging is not defined for nodes in a graph; therefore, we modify the criteria definitions to use a generalized centroid notion, in a way that, if the centroid is set as averaging, we would obtain the original criteria definitions, but we could also use other alternative notions for centroid of a group of data points.

Averaging data points results in a point with the least average distance to the other points. When averaging is not possible, using medoid is the natural option, which is perfectly compatible with graphs. More formally, the centroid of a community can be obtained as:

$$\overline{C} = \arg\min_{m \in C} \sum_{i \in C} d(i,m)$$

Relative Validity Criteria

Here we present our generalizations of well-known clustering validity criteria defined as quality measures for internal evaluation of clustering results. All these criteria are originally defined based on distances between data points, which is in all cases the Euclidean or other inner product norms of difference between their vectors of attributes; refer to Vendramin et al. (2010) for comparative analysis of these criteria in the clustering context. We alter the formulae to use a generalized distance, so that we can plug in our graph distance measures. The other alteration is generalizing the mean over data points to a general centroid notion, which can be set as averaging in the presence of attributes and the *medoid* in our case of dealing with graphs and in the absence of attributes.

In a nutshell, in every criterion, the average of points in a cluster is replaced with a generalized notion of centroid, and distances between data points are generalized from Euclidean/norm to a generic distance. Consider a clustering $C = \{C_1, C_2, \ldots, C_k\}$, which clusters N data points into k clusters, where $\overline{C}$ denotes the centroid of data points belonging to C. The quality of C can be measured using one of the following criteria.

Variance Ratio Criterion (VRC)

This criterion measures the ratio of the between-cluster/community distances to within-cluster/community distances which could be generalized as follows:

$$\mathrm{VRC} = \frac{\sum_{l=1}^{k} |C_l| d(\overline{C}_l, \overline{C})}{\sum_{l=1}^{k} \sum_{i \in C_l} d(i, \overline{C}_l)} \times \frac{N-k}{k-1}$$

where $\overline{C}_l$ is the centroid of the cluster/community C_l, and $\overline{C}$ is the centroid of the entire data/network. The original clustering formula proposed in Calinski and Harabasz (1974) for attribute vectors is obtained if the centroid is fixed to averaging of vectors of attributes and distance to (square of) Euclidean distance.

Davies-Bouldin Index (DB)

This minimization criterion calculates the worst case within-cluster/community to between-cluster/community distances ratio averaged over all clusters/communities (Davies and Bouldin 1979):

$$\mathrm{DB} = \frac{1}{k} \sum l = 1k \max m \neq l \left((\overline{d}_l + \overline{d}_m) / d(\overline{C}_l, \overline{C}_m) \right)$$

$$\overline{d}_l = \frac{1}{|C_l|} \sum_{i \in C_l} d(i, \overline{C}_l)$$

Dunn Index

This criterion considers both the minimum distance between any two clusters/communities and

the length of the largest cluster/community diameter (i.e., the maximum or the average distance between all the pairs in the cluster/community) (Dunn 1974):

$$\text{Dunn} = \min_{l \neq m} \left\{ \frac{\delta(C_l, C_m)}{\max_p \Delta(C_p)} \right\}$$

where δ denotes distance between two communities and Δ is the diameter of a community. Different variations of calculating δ and Δ are available; δ could be single, complete, or average linkage or only the difference between the two centroids. Moreover, Δ could be maximum or average distance between all pairs of nodes or the average distance of all nodes to the centroid. For example, the single linkage for δ and maximum distance for Δ are $(\delta(C_l, C_m) = \min_{i \in C_l, j \in C_m} d(i,j))$ and $\left(\Delta(C_p) = \max_{i,j \in C_p} d(i,j)\right)$ Therefore, we have different variations of Dunn index in our experiments, each indicated by two indexes for different methods to calculate δ (i.e., single(0), complete(1), average (2), and centroid(3)) and different methods to calculate Δ (i.e., maximum(0), average(1), average to centroid(3)).

Silhouette Width Criterion (SWC)

This criterion measures the average of silhouette score for each data point. The silhouette score of a point shows the goodness of the community it belongs to by calculating the normalized difference between the distance to its nearest neighboring community and its own community (Rousseeuw 1987). Taking the average one has:

$$\text{SWC} = \frac{1}{N} \sum_{l=1}^{k} \sum_{i \in C_l} \times \frac{\min_{m \neq l} d(i, C_m) - d(i, C_l)}{\max\left\{ \min_{m \neq l} d(i, C_m), d(i, C_l) \right\}}$$

where $d(i, C_l)$ is the distance of point i to community C_l, which is originally set to be the average distance (called SWC2) (i.e., $1/|C_l| \sum_{j \in C_l} d(i,j)$ or could be the distance to its centroid (called SWC4)) (i.e., $d\left(i, \overline{C_l}\right)$). An alternative formula for Silhouette is proposed in Vendramin et al. (2010):

$$\text{ASWC} = \frac{1}{N} \sum_{l=1}^{k} \sum_{i \in C_l} \frac{\min_{m \neq l} d(i, C_m)}{d(i, C_l)}$$

PBM

This criterion is based on the within-community distances and the maximum distance between centroids of communities (Pakhira and Dutta 2011):

$$\text{PBM} = \frac{1}{k} \frac{\max_{l,m} d\left(\overline{C}_l, \overline{C}_m\right)}{\sum_{l=1}^{k} \sum_{i \in C_l} d\left(i, \overline{C}_l\right)}$$

C-Index

This criterion compares the sum of the within-community distances to the worst and best case scenarios (Dalrymple-Alford 1970). The best case scenario is where the within-community distances are the shortest distances in the graph, and the worst case scenario is where the within-community distances are the longest distances in the graph.

$$\theta = \frac{1}{2} \sum_{l=1}^{k} \sum_{i,j \in C_l} d(i,j)$$

$$\text{CIndex} = \frac{\theta - \min\theta}{\max\theta - \min\theta}$$

The min θ/max θ is computed by summing the m_1 smallest/largest distances between every two points, where $m_1 = \sum_{l=1}^{k} \frac{|C_l|(|C_l|-1)}{2}$.

Z-Statistics

This criterion is similar to C-Index, however, with a different formulation (Hubert and Levin 1976):

$$\text{ZIndex} = \frac{\theta - E(\theta)}{\sqrt{\text{var}(\theta)}}$$

R

$$E(\theta) = \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{N} d(i,j)$$

$$\mathrm{Var}(\theta) = \frac{\left(\sum_{i=1}^{N}\sum_{j=1}^{N} d(i,j)\right)^2 - 2\sum_{i=1}^{N}\left(\sum_{j=1}^{N} d(i,j)\right)^2}{N(N-1)} - \frac{\left(\sum_{i=1}^{N}\sum_{j=1}^{N} d(i,j)\right)^2}{N^2} + \frac{\sum_{i=1}^{N}\sum_{j=1}^{N} d(i,j)^2}{N}$$

Point Biserial (PB)

This criterion computes the correlation of the distances between nodes and their cluster comembership which is dichotomous variable (Milligan and Cooper 1985). Intuitively, nodes that are in the same community should be separated by shorter distances than those which are not:

$$\mathrm{PB} = \frac{M_1 - M_0}{S}\sqrt{\frac{m_1 m_0}{m^2}}$$

where m is the total number of distances, i.e., $N(N-1)/2$, and S is the standard deviation of all pairwise distances, i.e., $\sqrt{\frac{1}{m}\sum_{i,j}(d(i,j)) - \frac{1}{m}\sum_{i,j}(d(i,j))^2}$, while M_1 and M_0 are, respectively, the average of within and between-community distances, and m_1 and m_0 represent the number of within and between community distances. More formally:

$$m_1 = \sum_{l=1}^{k}\frac{N_l(N_l-1)}{2} \quad m_0 = \sum_{l=1}^{k}\frac{N_l(N_l-1)}{2}$$

$$M_1 = 1/2\sum_{l=1}^{k}\sum_{i,j\in C_l} d(i,j)$$

$$M_0 = 1/2\sum_{l=1}^{k}\sum_{\substack{i\in C_l \\ j\notin C_l}} d(i,j)$$

Modularity

Modularity is the well-known criterion proposed by Newman and Girvan (2004) specifically for the context of community mining. This criterion considers the difference between the fraction of edges that are within the community and the expected such fraction if the edges were randomly distributed. Let E denote the number of edges in the network, i.e., $E = \frac{1}{2}\sum_{ij}A_{ij}$, then Q-modularity is defined as

$$Q = \frac{1}{2E}\sum_{l=1}^{k}\sum_{i,j\in C_l}\left[A_{ij} - \frac{\sum_k A_{ik}\sum_k A_{kj}}{2E}\right]$$

The computational complexity of different validity criteria is provided in the previous work by Vendramin et al. (2010).

Comparison Methodology and Results

In this section, we compare the proposed relative community criteria. First, we describe the approach we have used for the comparison. Then, we report the criteria performances in different settings. The following procedure summarizes our comparison approach.

The performance of a criterion could be examined by how well it could rank different partitionings of a given dataset. More formally, consider we have a dataset d and a set of m different possible partitionings, i.e., $P(d) = \{P_{d1}, P_{d2}, \ldots, P_{dm}\}$. Then, the performance of criterion c on dataset d could be determined by how much its values, $I_c(d) = \{c(p_{d1}), c(p_{d2}), \ldots, c(p_{dm})\}$, correlate with the "goodness" of these partitionings. Assuming that the true partitioning (i.e., ground truth) P_d^* is known for dataset d, the "goodness" of partitioning P_{di} could be determined using partitioning agreement measure a, a.k.a. external evaluation. Hence, for dataset d with set of possible partitionings $P(d)$, the external evaluation provides $E(d) = \{a(p_{d1}, p_d^*), a(p_{d2}, p_d^*), \ldots, a(p_{dm}, p_d^*)\}$, where (p_{d1}, p_d^*) denotes the "goodness" of partitioning P_{d1} comparing to the ground truth. Then, the performance score of criterion c on dataset d could be examined by the correlation of its values $I_c(d)$ and the values obtained from the external evaluation $E(d)$ on different possible partitionings. Finally, the criteria are ranked

based on their average performance score over a set of datasets.

```
D ← {d1, d2, ... dn}
for all dataset d ∈ D do
    {generate m possible partitioning}
     P(d) ← {pd1, pd2, pdm}
    {compute external scores}
    E(d) ← {a(pd1, pd*), a(pd2, pd*) ... a
(pdm, pd*})
     for all c ∈ Criteria do
      {compute internal scores}
      Ic(d) ← {c(pd1), c(pd2) ... c(pdm)}
      {compute the correlation}
      scorec (d) ← correlation (E, I)
     end for
   end for
{rank criteria based on their average
scores}
```

$$\text{score}_c \leftarrow \frac{1}{n}\sum_{d=1}^{n} \text{score}_c(d)$$

External evaluation is done with an agreement measure, which computes the agreement between two given partitionings or between a partitioning and the ground truth. There are several choices for the partitioning agreement measure. The commonly used ones are pair counting based, such as adjusted rank index (ARI) (Hubert and Arabie 1985) and Jaccard coefficient (Jaccard 1901), and the information theoretic-based, such as normalized mutual information (NMI) (Kvalseth 1987; Danon et al. 2005) and the adjusted mutual information (AMI) (Vinh et al. 2010). There are also different ways to compute the correlation between two vectors. The classic options are the Pearson product moment coefficient or the Spearman rank correlation coefficient. The reported results in our experiments are based on the Spearman correlation, since we are interested in the correlation of rankings that a criterion provides for different partitionings and not the actual values of that criterion. However, the reported results mostly agree with the results obtained by using the Pearson correlation, which are reported in the supplementary materials available from https://github.com/rabbanyk/CommunityEvaluation. This link also provides implementations for the different measures discussed and the experiment pipeline used in this entry.

Sampling the Partitioning Space

In our comparison, we generate different partitionings for each dataset d to sample the space of all possible partitionings. For doing so, given the perfect partitioning, P_d^*, we randomized different versions of by P_d^* randomly merging and splitting communities and swapping nodes between them. The sampling procedure is described in more details in the supplementary materials.

Results on Real-World Datasets

We first compare performance of different criteria on five well-known real-world benchmarks: Karate Club (weighted) by Zachary 1977, Sawmill Strike dataset (Nooy et al. 2004), NCAA Football Bowl Subdivision (Girvan and Newman 2002), and Politician Books from Amazon (Krebs 2004). Table 1 shows general statistics about the datasets and their generated samples. We can see that the randomized samples cover the space of partitionings according to their external index range.

Figure 1 exemplifies how different criteria exhibit different correlations with the external index. It visualizes the correlation between few selected relative indexes and an external index for one of our datasets listed in Table 1.

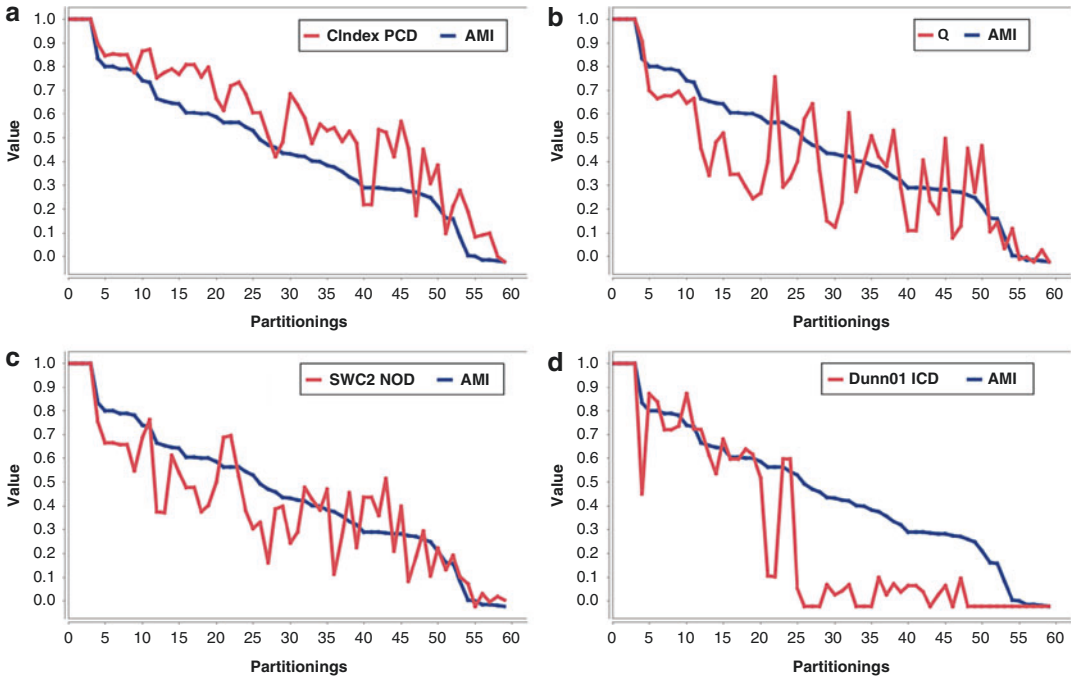
Similar analysis is done for all 4 datasets × 19 criteria × 7 distances × 4 external indexes, which produced over 2,000 such correlations. The top-ranked criteria based on their average performance over these datasets are summarized in Table 2. Based on these results, *CIndex* when used with *PCD* distance has a higher correlation with the external index comparing to the modularity Q. And this is true regardless of the choice of *AMI* as the external index, since it is ranked above Q also by *ARI* and *NMI*.

The correlation between a criterion and an external index depends on how close the randomized partitionings are from the true partitioning of the ground truth. This can be seen in Fig. 1. For example, *Dunn01* (single linkage network diameter and average linkage within community scores)

Relative Validity Criteria for Community Mining Algorithms, Table 1 Statistics for sample partitionings of each real-world dataset. For example, for the Karate Club dataset which has two communities in its ground truth, we have generated 60 different partitionings with average 3.57 ± 1.23 clusters ranging from 2 to 6, and the "goodness" of the samples is on average 0.46 ± 0.27 in terms of their AMI agreement

Dataset	K^*	#	$\overline{K}$	$\overline{AMI}$
Strike	3	60	$3.17 \pm 1 \in [2,5]$	$0.59 \pm 0.27 \in [-0.04,1]$
Polboks	3	60	$3.17 \pm 1.13 \in [2,6]$	$0.44 \pm 0.25 \in [0.04,1]$
Karate	2	60	$3.57 \pm 1.23 \in [2,6]$	$0.46 \pm 0.27 \in [-0.02,1]$
Football	11	60	$10.17 \pm 4.55e[4,19]$	$0.68 \pm 0.16 \in [0.4,1]$

K^*denotes the perfect/true number of clusters



Relative Validity Criteria for Community Mining Algorithms, Fig. 1 Visualization of correlation between an external agreement measure and some relative quality criteria for Karate dataset. The x axis indicates different random partitionings, and the y axis indicates the value of the index. While, the *blue/darker line* represents the value of the external index for the given partitioning and the *red/lighter line* represents the value that the criterion gives for the partitioning. Please note that the value of criteria are not generally normalized and in the same range as the external indexes, in this figure AMI. For the sake of illustration, therefore, each criterion's value is scaled to be in the same range as of the external index. (**a**) CINDEX PCD. (**b**) Q. (**c**) SWC2 NOD. (**d**) Dunn01 ICD

with the *ICD* distance agrees strongly with the external index in samples with higher external index value, i.e., closer to the ground truth, but not on further samples. On the other hand, Q is very well matched for the samples too far or too close to the ground truth, but is not doing as well as others in the middle. With this in mind, we have divided the generated clustering samples into three sets of easy, medium, and hard samples and re-ranked the criteria in each of these settings. Since the external index determines how far a sample is from the optimal result, the samples are divided into three equal length intervals according to the range of the external index. Table 3 reports the rankings of the top criteria in each of these three settings. We can see that these

Relative Validity Criteria for Community Mining Algorithms, Table 2 Overall ranking of criteria on the real-world datasets, based on the average Spearman correlation of criteria with the AMI external index, AMI_{corr}. Ranking based on correlation with other external indexes is also reported

Rank	Criterion	AMI_{corr}	ARI	Jaccard	NMI
1	CIndex PCD	0.907 ± 0.058	1	1	1
2	SWC2 NOD	0.857 ± 0.031	4	4	2
3	Q	0.85 ± 0.083	2	2	3
4	CIndex ARD	0.826 ± 0.162	6	15	5
5	CIndex SPD	0.811 ± 0.126	3	10	4
6	ASWC2 NOD	0.809 ± 0.043	5	11	6
7	CIndex NOD	0.794 ± 0.096	12	3	9
8	SWC2 PCD	0.789 ± 0.103	7	7	8
9	SWC4 NOD	0.778 ± 0.075	9	5	7
10	ASWC2 PCD	0.772 ± 0.088	10	9	10
11	SWC2 SPD	0.751 ± 0.121	8	6	11
12	Dunn01 ICD	0.742 ± 0.111	18	24	12
13	ASWC2 SPD	0.733 ± 0.116	11	8	13
14	Dunn00 PCD	0.721 ± 0.1	21	30	14
15	DBICD	0.712 ± 0.063	24	22	16
16	Dunn00 ICD	0.707 ± 0.133	28	28	15
17	Dunn03 ICD	0.703 ± 0.055	25	23	17
18	SWC4 PCD	0.7 ± 0.072	14	12	21

average results support our earlier hypothesis, i.e., when considering partitionings medium far from the true partitioning, *CIndex PCD* performs significantly better than modularity Q, while their performances are not very different in the near-optimal samples or the samples very far from the ground truth. One may conclude based on this experiment that *CIndex PCD* is a more accurate evaluation criterion comparing to *Q* especially when the results might not be very accurate or very poor.

Synthetic Benchmarks Datasets

Lastly, we compare the criteria on a larger set of synthetic benchmarks. We have generated our dataset using the LFR benchmarks (Lancichinetti et al. 2008) which are the generators widely in use for community mining evaluation. Similar to the last experiment, Table 4 reports the ranking of the top criteria according to their average performance on synthesized datasets of Table 5. Based on which, modularity Q overall outperforms other criteria especially in ranking poor partitionings; while *CIndex PCD* performs better in ranking finer results.

The LFR generator can generate networks with different levels of difficulty for the partitioning task by changing how well separated the communities are in the ground truth. To examine the effect of this difficulty parameter, we have ranked the criteria for different values of this parameter. We observed that modularity Q is the superior criterion for these synthetic benchmarks up to some level of how mixed are the communities, but this changes in more difficult settings. Results for other settings are available in the supplementary materials.

Table 6 reports the overall ranking of the criteria for a difficult set of datasets that have high mixing parameter. We can see that in this setting, *PB* index used with *PCD*, *NOD*, *SPD,* or *ARD* distances is significantly more reliable than modularity Q, particularly considering the much higher variance of the latter.

Key Applications

In short, the relative performances of different criteria depend on the difficulty of the network

Relative Validity Criteria for Community Mining Algorithms, Table 3 Difficulty analysis of the results: considering ranking for partitionings near-optimal ground truth, medium far, and very far. The reported results are based on AMI and the Spearman correlation

Rank	Criterion	AMI_{corr}	ARI	Jaccard	NMI
Near-optimal samples					
1	Q	0.736 ± 0.266	5	5	2
2	CIndex PCD	0.72 ± 0.326	1	1	3
3	SWC2 SPD	0.718 ± 0.389	3	3	4
4	CIndex SPD	0.716 ± 0.14	4	4	1
5	SWC2 ICD	0.713 ± 0.396	2	2	5
6	ASWC2 ICD	0.687 ± 0.334	11	10	7
Medium-far samples					
1	CIndex PCD	0.608 ± 0.202	8	18	1
2	CIndex NOD	0.58 ± 0.053	39	13	2
3	CIndex ARD	0.513 ± 0.313	26	62	5
4	Dunn01 ICD	0.457 ± 0.173	58	83	8
5	SWC2 NOD	0.447 ± 0.19	5	9	3
6	ASWC2 PCD	0.446 ± 0.191	7	3	9
7	SWC2 PCD	0.446 ± 0.19	6	2	10
8	Dunn03 ICD	0.439 ± 0.109	43	37	11
9	Dunn31 SPD	0.437 ± 0.177	56	47	15
10	Dunn01 SPD	0.434 ± 0.205	29	67	7
11	Q	0.409 ± 0.353	4	7	16
12	DB ICD	0.405 ± 0.072	40	38	18
Far-far samples					
1	SWC2 NOD	0.634 ± 0.217	3	13	1
2	ASWC2 NOD	0.583 ± 0.191	5	21	2
3	Q	0.498 ± 0.179	4	38	5
4	CIndex PCD	0.493 ± 0.282	2	4	13
5	CIndex SPD	0.437 ± 0.291	1	11	4
6	SWC3 NOD	0.436 ± 0.344	8	2	25

itself, as well as how far we are sampling from the ground truth. Altogether, choosing the right criterion for evaluating different community mining results depends both on the application, i.e., how well-separated communities might be in the given network and also on the algorithm that produces these results, i.e., how fine the results might be. For example, if the problem is hard and communities are heavily mixed, modularity Q might not distinguish the good and bad partitionings very well. While if we are choosing between fine and well-separated clusterings, it indeed is the superior criterion.

Future Directions

In our experiments, several of these adopted criteria exhibit high performances on ranking different partitionings of a given dataset, which makes them possible alternatives for the Q modularity. However, a more careful examination is needed as the rankings depend significantly on the experimental settings and the criteria should be chosen based on the application. For follow-up works, reader could refer to Rabbany et al. (2013), Rabbany and Zaïane (2015), and Rabbany (2016).

Relative Validity Criteria for Community Mining Algorithms, Table 4 Overall ranking and difficulty analysis of the synthetic results. Here, communities are well separated with mixing parameter of 0.1. Similar to the last experiment, the reported results are based on AMI and the Spearman correlation

Rank	Criterion	AMI_{corr}	ARI	Jaccard	NMI
Overall results					
1	Q	0.894 ± 0.018	1	2	1
2	ASWC2 NOD	0.854 ± 0.056	3	4	2
3	SWC2 NOD	0.854 ± 0.051	4	3	3
4	CIndex PCD	0.826 ± 0.07	2	1	4
5	CIndex SPD	0.746 ± 0.137	8	24	5
6	SWC2 PCD	0.743 ± 0.047	5	5	6
7	ASWC2 PCD	0.739 ± 0.048	6	6	7
8	Dunn00 PCD	0.707 ± 0.11	11	26	8
9	SWC4 NOD	0.699 ± 0.131	7	7	9
10	SWC4 ARD	0.689 ± 0.124	9	8	10
11	ASWC2 ARD	0.683 ± 0.108	15	21	11
12	ASWC2 ED	0.665 ± 0.139	10	11	12
13	SWC2 SPD	0.657 ± 0.124	14	16	13
14	ASWC2 SPD	0.651 ± 0.196	16	17	15
15	Dunn03 NOD	0.645 ± 0.156	23	33	14
Near-optimal results					
1	CIndex PCD	0.729 ± 0.17	1	1	1
2	Q	0.722 ± 0.111	6	5	5
3	SWC2 SPD	0.717 ± 0.185	18	18	2
4	SWC4 NOD	0.709 ± 0.201	5	6	4
5	SWC2 ICD	0.704 ± 0.216	15	15	3
6	SWC4 ARD	0.674 ± 0.183	7	7	6
7	ASWC2 NOD	0.66 ± 0.261	20	19	7
8	SWC2 NOD	0.649 ± 0.264	14	14	9
Medium-far results					
1	SWC2 NOD	0.455 ± 0.191	5	11	3
2	CIndex PCD	0.453 ± 0.245	1	2	5
3	Q	0.45 ± 0.236	2	9	2
4	ASWC2 NOD	0.435 ± 0.187	4	14	1
5	Dunn00 ARD	0.386 ± 0.243	119	111	7
6	Dunn00 PCD	0.38 ± 0.195	58	91	6
7	CIndex NOD	0.373 ± 0.213	7	1	14
8	Dunn01 NOD	0.358 ± 0.146	108	95	15
Far-far results					
1	Q	0.63 ± 0.139	1	4	2
2	ASWC2 NOD	0.596 ± 0.164	2	2	3
3	SWC2 NOD	0.57 ± 0.159	3	3	5
4	CIndex SPD	0.565 ± 0.132	4	25	1
5	CIndex PCD	0.446 ± 0.142	5	1	21
6	CIndex ARD	0.433 ± 0.25	10	106	4
7	ASWC4 NOD	0.397 ± 0.119	15	63	11
8	SWC2 PCD	0.356 ± 0.143	6	6	25

Relative Validity Criteria for Community Mining Algorithms, Table 5 Statistics for sample partitionings of each synthetic dataset. The benchmark generation parameters: 100 nodes with average degree 5 and maximum degree 50, where size of each community is between 5 and 50 and mixing parameter is 0.1

Dataset	K^*	#	$\overline{K}$	$\overline{AMI}$
Network1	4	60	$3.4 \pm 1.17 \in [2,6]$	$0.46 \pm 0.23 \in [0,1]$
Network2	3	60	$3.1 \pm 1.27 \in [2,7]$	$0.49 \pm 0.22 \in [0.13,1]$
Network3	2	60	$3.3 \pm 1.13 \in [2,6]$	$0.47 \pm 0.23 \in [0.11,1]$
Network4	7	60	$5.17 \pm 2.49 \in [2,12]$	$0.57 \pm 0.2 \in [0.18,1]$
Network5	2	60	$3.5 \pm 1.36 \in [2,8]$	0.44 ± 0.222 [0.11,1]
Network6	5	60	$5.8 \pm 2.55 \in [2,12]$	$0.68 \pm 0.2 \in [0.27,1]$
Network7	4	60	$5.2 \pm 2.65 \in [2,12]$	$0.47 \pm 0.19 \in [0.13,1]$
Network8	5	60	$5.37 \pm 2.04 \in [2,10]$	$0.67 \pm 0.21 \in [0.32,1]$
Network9	5	60	$5.5 \pm 2.05 \in [2,10]$	$0.69 \pm 0.19 \in [0.37,1]$
Network10	6	60	$5.33 \pm 2.51 \in [2,11]$	$0.63 \pm 0.19 \in [0.24,1]$

$^{K^*}$denotes the perfect/true number of clusters

Relative Validity Criteria for Community Mining Algorithms, Table 6 Overall ranking of criteria based on AMI and the Spearman correlation on the synthetic benchmarks with the same parameters as in but much higher mixing parameter, 0.7. We can see that in these settings, PB indexes outperform modularity Q

Rank	Criterion	AMI_{corr}	ARI	Jaccard	NMI
1	PBPCD	0.454 ± 0.15	1	1	1
2	PB NOD	0.448 ± 0.146	2	2	2
3	PB SPD	0.445 ± 0.144	3	3	4
4	PBARD	0.44 ± 0.149	4	4	5
5	VRCICD	0.424 ± 0.117	5	5	3
6	Q	0.391 ± 0.381	17	6	12
7	CIndex ARD	0.365 ± 0.173	6	7	6
8	ASWC4 SPD	0.358 ± 0.101	12	12	7
9	DBPCD	0.358 ± 0.108	15	9	10
10	ASWC4 NOD	0.357 ± 0.114	10	10	8

Conclusion

In this entry, we generalized well-known clustering validity criteria originally used as quantitative measures for evaluating quality of clusters of data points represented by attributes. The first reason of this generalization is to adapt these criteria in the context of community mining of interrelated data. The only commonly used criterion to evaluate the goodness of detected communities in a network is the modularity Q. Providing more validity criteria can help researchers to better evaluate and compare community mining results in different settings. Also, these adapted validity criteria can be further used as objectives to design new community mining algorithms. Our generalized formulation is independent of any particular distance measure unlike most of the original clustering validity criteria that are defined based on the Euclidean distance. The adopted versions therefore could be used as community criteria when plugged in with different graph distances.

Cross-References

- ▶ Combining Link and Content for Community Detection
- ▶ Community Detection: Current and Future Research Trends

- ▶ Competition Within and Between Communities in Social Networks
- ▶ Extracting and Inferring Communities via Link Analysis
- ▶ Game-Theoretic Framework for Community Detection

Acknowledgments The authors are grateful for the support from Alberta Innovates Centre for Machine Learning and NSERC. Ricardo Campello also acknowledges the financial support of Fapesp and CNPq.

References

Calinski T, Harabasz J (1974) A dendrite method for cluster analysis. Commun Stat Theory Methods 3:1–27

Dalrymple-Alford EC (1970) Measurement of clustering in free recall. Psychol Bull 74:32–34

Danon L, Guilera AD, Duch J, Arenas A (2005) Comparing community structure identification. J Stat Mech Theory Exp 2005(09):09008

Davies DL, Bouldin DW (1979) A cluster separation measure. IEEE Trans Pattern Anal Mach Intell 1(2):224–227

Dunn JC (1974) Well-separated clusters and optimal fuzzy partitions. J Cybern 4(1):95–104

Fortunato S (2010) Community detection in graphs. Phys Rep 486(3–5):75–174

Fortunato S, Barthélemy M (2007) Resolution limit in community detection. Proc Natl Acad Sci 104(1):36–41

Girvan M, Newman MEJ (2002) Community structure in social and biological networks. Proc Natl Acad Sci 99(12):7821–7826

Hubert L, Arabie P (1985) Comparing partitions. J Classif 2:193–218

Hubert LJ, Levin JR (1976) A general statistical framework for assessing categorical clustering in free recall. Psychol Bull 83:1072–1080

Jaccard P (1901) Étude comparative de la distribution florale dans une portion des alpes et des jura. Bulletin del la Société Vaudoise des Sciences Naturelles 37:547–579

Kenley EC, Cho YR (2011) Entropy-based graph clustering: application to biological and social networks. In: IEEE international conference on data mining, Vancouver

Krebs V (2004) Books about US politics. http://www.orgnet.com/

Kvalseth TO (1987) Entropy and correlation: some comments. IEEE Trans Syst Man Cybern 17(3):517–519. https://doi.org/10.1109/TSMC.1987.4309069

Lancichinetti A, Fortunato S (2009) Community detection algorithms: a comparative analysis. Phys Rev E 80(5):056117

Lancichinetti A, Fortunato S, Radicchi F (2008) Benchmark graphs for testing community detection algorithms. Phys Rev E 78(4):046110

Leskovec J, Kleinberg J, Faloutsos C (2005) Graphs over time: densification laws, shrinking diameters and possible explanations. In: ACM SIGKDD international conference on knowledge discovery in data mining, Chicago, pp 177–187

Milligan G, Cooper M (1985) An examination of procedures for determining the number of clusters in a data set. Psychometrika 50(2):159–179

Newman MEJ (2006) Modularity and community structure in networks. Proc Natl Acad Sci 103(23):8577–8582

Newman M (2010) Networks: an Introduction. Oxford University Press, New York

Newman MEJ, Girvan M (2004) Finding and evaluating community structure in networks. Phys Rev E 69(2):026113

Nooy W, Mrvar A, Batagelj V (2004) Exploratory social network analysis with Pajek. Cambridge University Press, Cambridge

Orman GK, Labatut V, Cherifi H (2011) Qualitative comparison of community detection algorithms. In: International conference on digital information and communication technology and its applications, Dijon 167, pp 265–279

Pakhira M, Dutta A (2011) Computing approximate value of the PBM index for counting number of clusters using genetic algorithm. In: International conference on recent trends in information systems, Kolkata, pp 241–245

Palla G, Derenyi I, Farkas I, Vicsek T (2005) Uncovering the overlapping community structure of complex networks in nature and society. Nature 435 (7043):814–818

Rabbany R (2016) Modular structure of complex networks. PhD thesis, University of Alberta. http://hdl.handle.net/10402/era.43464

Rabbany R, Zaïane OR (2011) A diffusion of innovation-based closeness measure for network associations. In: IEEE international conference on data mining workshops, Vancouver, pp 381–388

Rabbany R, Zaïane OR (2015) Generalization of clustering agreements and distances for overlapping clusters and network communities. Data Min Knowl Disc 29(5):1458–1485

Rabbany R, Chen J, Zaïane OR (2010) Top leaders community detection approach in information networks. In: SNA-KDD workshop on social network mining and analysis, Washington, DC

Rabbany R, Takaffoli M, Fagnan J, Zaïane OR (2013) Communities validity: methodical evaluation of community mining algorithms. Soc Netw Anal Min 3(4):1039–1062

Rosvall M, Bergstrom CT (2008) Maps of random walks on complex networks reveal community structure. Proc Natl Acad Sci 105(4):1118–1123

Rousseeuw P (1987) Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. J Comput Appl Math 20(1):53–65
Vendramin L, Campello RJGB, Hruschka ER (2010) Relative clustering validity criteria: a comparative overview. Stat Anal Data Min 3(4):209–235
Vinh NX, Epps J, Bailey J (2010) Information theoretic measures for clusterings comparison: variants, properties, normalization and correction for chance. J Mach Learn Res 11:2837–2854
Zachary WW (1977) An information flow model for conflict and fission in small groups. J Anthropol Res 33:452–473

Relevance Functions

▶ Retrieval Models

Relevance Ranking

▶ Ranking Methods for Networks

Reliability

▶ Quality of Social Network Data

Reliance

▶ Trust in Social Networks

Report Confirmation

▶ Spatiotemporal Proximity and Social Distance

Repository for Multirelational Dynamic Networks

Aleksei Romanov, Alexander Semenov and Jari Veijalainen
Faculty of Information Technology, University of Jyväskylä, Jyväskylä, Finland

Synonyms

Data storage for networked data

Glossary

Crawler	is a software program that traverses the World Wide Web (WWW or web) information space by following hypertext links and retrieving web documents by HTTP standard
Graph database	is a database used to store graph data. Graph structures with nodes, edges, and attributes are used to represent the data
Monitoring and analysis software	is a piece of software that is deployed at the monitoring site; it collects data from the monitored site(s) over the Internet, stores them at the repository, and analyzes the collected data. Monitoring software observes the site in the same manner as users do
Online social network	is the digital representation of a virtual community at a social media site, adhering to the site ontology. A virtual community is a group of real individuals and an online social network is a group of users' accounts, profiles, etc. and their relationships on the site
Ontology	is an explicit specification of a conceptualization (Gruber 2009). A conceptualization should

	express a shared view between several parties (Staab and Studer 2011). Every knowledge base, knowledge-based system, or knowledge-level agent is committed to some conceptualization, explicitly or implicitly
Relational database	is a database created using a relational model (a set of tables corresponding to a relational scheme). A relational database is managed by a relational database management system (RDBMS)
Repository	is a persistent data collection consisting of heterogeneous multimedia data that are fetched from diverse social media sites for the purpose of storage and analysis
Site ontology	is a conceptual model that captures the concepts, essential relationships between them and manipulation operations offered to the users by a social media site
Temporal database	is a database with built-in time aspects

Definition

An information repository is a way to deploy a tier of data storage that can comprise multiple, networked data storage technologies running on diverse operating systems. They provide centralized management for the deployed data storage resources. Information repositories are self-contained, support heterogeneous storage resources, and support resource management to add, maintain, and recycle media.

Information repository in our case is used to store a persistent data consisting of heterogeneous multimedia data that are fetched from different social media sites for the purpose of subsequent analysis. Information repository appears as a term, for instance in 1992 in a patent (Smiley and Incorporated 1992).

Introduction

Social media sites are rather recent phenomena in the Internet. They have appeared during the last 10 years and have attracted a large numbers of users. Examples of the largest social media sites are Facebook with over 1.6 billion monthly active users (MAU), Twitter with 320 million, Sina Weibo with over 600 million MAUs, etc. One of the main features of social media sites is to allow its users to create and modify the contents of the site utilizing the offered WWW interfaces and mobile applications. In this vein, Kaplan and Haenlein (2010) defined: "Social Media is a group of Internet-based applications that build on the ideological and technological foundations of Web 2.0, and that allow the creation and exchange of User Generated Content." The user-generated content (UGC) and its type varies from site to site, pertaining to the site ontology. It contains concepts that are presented to the users and model the social reality, like user profile with all its attributes, follower or friend, likes, status updates, comments, etc. It also contains operations with which the concepts can be instantiated, such as "publish" (a status update/image/video), "tweet" (a max 140 characters long status update with a possible attachment), "follow" (another profile), "like" (a tweet or status update), "comment" (a status update, image, etc.), and so on. A part of the concepts in the site ontology are only accessible through APIs (e.g., internal unique identifiers of profiles and pieces of contents), but not through mobile applications or browser interfaces. In addition, social media sites have site-internal concepts and models that are only indirectly observable by any user. These are mostly related with the applied business model (which ads should be offered to this user profile?), security (is this profile run by a terrorist?), IPRs (is this content illegally uploaded), etc.

Information existing within social media sites models some part of the social reality, and thus by analyzing this information it is possible to make inferences about the state of the affairs in the real world (Semenov and Veijalainen 2013b). For

instance, two Facebook profiles that are "friends" in the sense Facebook defines and implements this concept might refer to two real friends in the real world often meeting each other face-to-face. On the other hand, they might have never even met in the real world and are thus "Facebook friends," living possibly in different parts of the world. In both cases Facebook models this relationship in the same way. More generally, almost all sites offer in their site ontology a concept and its realization for the individual users ("user profile" or "user account") and some relationships between these concepts, such as a "friend" or a "follower" (Twitter, vk.com), etc. Further UGC is tied to the user profile and it is accessible to various degrees to other users and outsiders. For simplicity, we consider both the UGC produced by the users and the relationships established between them as data. These data are adhering to the particular site ontology the social media site implements, i.e., the data instantiates the ontology and can only be understood and correctly interpreted in the context of the site ontology concepts.

Most social media sites allow observation of the data produced by a person or a set of persons at a particular moment of time, but sometimes also allow access to the data generated in the past. For instance, Twitter allows up to 3200 latest public, nonremoved tweets sent by a particular user to be fetched through a browser or REST API by other users, no matter when the tweets were issued. It also allows all tweets ever sent by a user to be retrieved by him or her (see https://twitter.com/settings/account). Extraction of all this data from social media sites can be done with a reasonable effort, and the information it carries would be obviously nearly impossible to collect directly from people without the digital technology at hand. In most cases it is not possible to extract a full history of the data evolution, though. Typically, after an entity (a message, a tweet, a blogpost, a connection to another user profile, etc.) is removed, the social media platform does not explicitly report the deletion to the users that are not directly affected. Later, the traces of the earlier existence of the deleted entity may vanish altogether from the site for other users and outsiders. Thus, in general, for having the history of the states of the relationships between user profiles, i.e., the social network, and UGC at a site, it is necessary for outsiders to poll the website continuously (or often enough) or to use a streaming API or other APIs offered by the site. In the case of polling, the rate should be higher than the rate of changes at the site (Semenov and Veijalainen 2013b), if it is required that all changes are captured.

Collection of the data from a web site by an external client is usually facilitated using special software for data gathering, web crawlers (Liu and Menczer 2011). They are used by search engines to collect data, and there are many third-party crawlers that can be used for collection of the data from the social media sites. Also, a number of social media sites provide special APIs for data extraction: e.g., Twitter offers a streaming API ("Streaming APIs – Twitter Developers" 2007) that pushes the data continuously to the pieces of the software that requested it. It also offers a REST APIs ("REST APIs – Twitter Developers" 2017) through which one can search users by name, followers or followees of a particular user, and many other data, like the 3200 tweets mentioned above.

One can distinguish three levels of models in the social media monitoring and analysis activity. The social media sites model a certain portion of the immediate social reality (that also includes the social media sites themselves). We call these first-level models, conceptualized as site ontologies and realized at particular social media sites. Each site has its own site ontology and own implementation for it. The ontologies of interest are then remodeled (usually partially) by a monitoring site by the second level model. It should ideally be expressive enough to capture all the site ontologies and to form a homogeneous (meta)modeling level. In our approach this homogeneous level consists of multirelational directed graphs. The third-level model is then the data repository level, where we need to store the second-level model instances persistently. In practice, we must map the above general graph structure onto the data model of a database management system, because using only file system facilities would not offer the functionality needed to analyze the graph and would not provide the needed performance.

We thus argue here that the data hosted by social media sites can often, if not always, be modeled as constantly evolving multirelational directed graphs. In this article we discuss persistent database structures for such graphs, and present and analyze queries performed against the structures. We also estimate the space requirements of the proposed data structures, and compare them with the naive approach consisting of storing each complete. In the sequel we will discuss the architecture of a repository that allows persistent storage for time varying, multirelational directed graphs in a condensed form, i.e., only storing "deltas," but also entire snapshots. We also present an OO interface of the repository and the algorithms for populating the repository and extracting the data from it. Further, we compare this approach with the naïve approach storing persistently complete snapshots of the graph at specific moments of time. In addition, we present the overall architecture of a monitoring and analyzing site, because the repository cannot in a meaningful way support all necessary functionality. We also investigate query performance against our data structure. We present analytical estimation of the repository performance, and simulation results.

Key Points

The central issue in this article is to consider how the changes of relationships between user profiles and the contents generated by the social media site users can be modeled with suitable graph concepts and how the graph instances can be stored into a persistent repository that supports various analysis tasks.

Historical Background

The core issue in this context is how a (specific kind of a) graph modeling social relationships and contents at social media sites could be persistently stored into a repository so that its evolution can be captured and various analysis tasks efficiently performed. In the center of the repository there are one or more database management systems that can support graph snapshots and deltas over time for various analysis tasks. Ideally, the other system components only access the repository interface that implements the necessary operations relying on DBMSs. The implementation can also be changed for various reasons. In theory, the same graph instance might have several parallel implementations in the same or different kind of DBMSs, if this is necessary for the analysis performance reasons. It is obvious that temporal DBMSs are relevant in this context, because we are interested in storing and analyzing graph evolution over time. Another obvious fact is that graph databases are DBMSs that could directly support the graph snapshots and deltas. We will look at some second-level graph models, different from multi-digraph, provided there is a description concerning its storage into a database.

There are a number of papers devoted to social network analysis. A survey of the results can be found in Aggarwal (2011). The article Kazienko et al. (2011) introduces a model of the social network as a multilayered graph, where each type of relations are presented at different levels. The authors Kazienko et al. (2011) discuss multirelational networks, which cover three main dimensions: relation layer, time window, and group. The multirelational model lets the authors to identify social groups considering time aspect. In George and Shekhar (2006) the authors present time aggregated graphs to model spatiotemporal networks. The proposed time-aggregated graphs do not replicate nodes and edges across time; rather they allow the properties of edges and nodes to be modeled as a time series. The authors (Semenov et al. 2011) introduce multidigraph as a solution for the second level models. It is a directed graph, having multiple types of nodes and relations.

Dynamic graph algorithms (Henzinger and King 1999) are relevant for our research as well. This research is related to data stream analysis (Aggarwal et al. 2011). A body of research is devoted to construction of temporal (Jensen 2000) and spatiotemporal (Erwig et al. 1999) databases. There are also industrial solutions which aim at storing various graphs and networks:

graph database Neo4j ("neo4j: World's Leading Graph Database" 2012) and Oracle Network data model ("Network Data Model Overview" 2012). Graph database Neo4j supports the graph evolution by migrations, i.e., through Liquigraph library ("Liquigraph by fbiville" 2016).

A graph data management system that is claimed to efficiently and scalably support tasks on dynamic network analysis over large volumes of data in real-time is presented in the article (Khurana and Deshpande 2013). A distributed graph database is a part of the system that enables temporal and evolutionary queries and analysis. The cornerstone of the system is a hierarchical index structure, DeltaGraph, that enables compact recording of the historical network information, and that supports efficient retrieval of historical graph snapshots. One further element is in-memory graph data structure called GraphPool that can maintain hundreds of historical graph instances in the main memory in a nonredundant manner. Authors also discuss the distribution of query access times by choice of appropriate tradeoff at construction time and memory materialization at runtime. The authors also propose further optimizations like developing techniques for processing different types of temporal queries over the historical trace. The authors use Kyoto Cabinet key-value library as the back-end engine to store the DeltaGraph components. The Kyoto Cabinet library is abandoned by its authors and the last update was in 2011. Many solutions have recently emerged to store key-value pairs efficiently, including Redis.io ("Redis" 2016), Berkeley DB ("Oracle Berkeley DB" 2016), open source in-memory data grid Hazelcast ("Hazelcast.org – The Leading Open Source In-Memory Data Grid" 2016). The efficiency of the presented graph data management system in the article could be enhanced by using other solutions of data storage. In Koloniari and Pitoura (2013), the authors suggest partial view construction of graph snapshot when it is possible to reduce cost for targeted queries in contrast to full snapshot construction in (Khurana and Deshpande 2013a).

HiNGE (Historical Network/Graph Explorer) is described as an analysis and visualization tool for historical networks (Khurana and Deshpande 2013a). HiNGE has been implemented in Java. It uses DeltaGraph and GraphPool libraries (Khurana and Deshpande 2013b) for storage and graph representation respectively and the JUNG ("JUNG – Java Universal Network/Graph Framework" 2016) library for network visualization. The HiNGE tool supports exploration tasks, evolutionary queries and search queries.

Koloniari and Pitoura (2013) use social network model based on the "copy+log" approach the basic components of which are graph snapshots and a log file that maintains the updates on the graph – originally described in detail in (Salzberg and Tsotras 1999). It is claimed that targeted queries are important on a micro-level analysis in social networks. The targeted queries are such queries that only address specific portions (subgraphs) of the social graph modeling social media site contents. In addition, targeted queries can form blocks for global query analysis, such as revealing communities in the global graph. Hence, focusing on targeted queries, it is possible to reduce the evaluation cost of targeted queries by constructing a partial view based on the current graph and the parts of the log file associated with the nodes included in the partial view instead of recreation of the whole graph. That means access only to a specific subgraph, usually centered around one or a few nodes as such queries tend to be egocentric. There are algorithms presented to enable such an approach (i.e., partial view construction and a two-phase greedy static view selection) and the authors also propose maintaining a cache of partial views.

The trade-off between the storage and recreation costs is studied in a rigorous manner in (Bhattacherjee et al. 2015) by formulating several optimization problems that trade off these two in different ways and by showing that most variations are NP-Hard. The authors present several algorithms that are effective at exploring this trade-off: the Local Move Greedy algorithm, tailored to the case when there is a bound or objective on the average recreation cost; two different Modified Prim's algorithms, tailored to the case when there is a bound on the maximum recreation cost; an approximation algorithm called LAST and an algorithm called GitH which is based on

Git repack. The authors also present an extensive experimental evaluation using a prototype version management system that has been built to estimate running times and storage cost for operations with directed and undirected graphs. The authors plan to improve the work by developing online algorithms for making the optimization decisions as new datasets or versions are being created, and also adaptive algorithms that re-evaluate the optimization decisions based on changing workload information.

Overall Architecture of the Monitoring and Analysis System

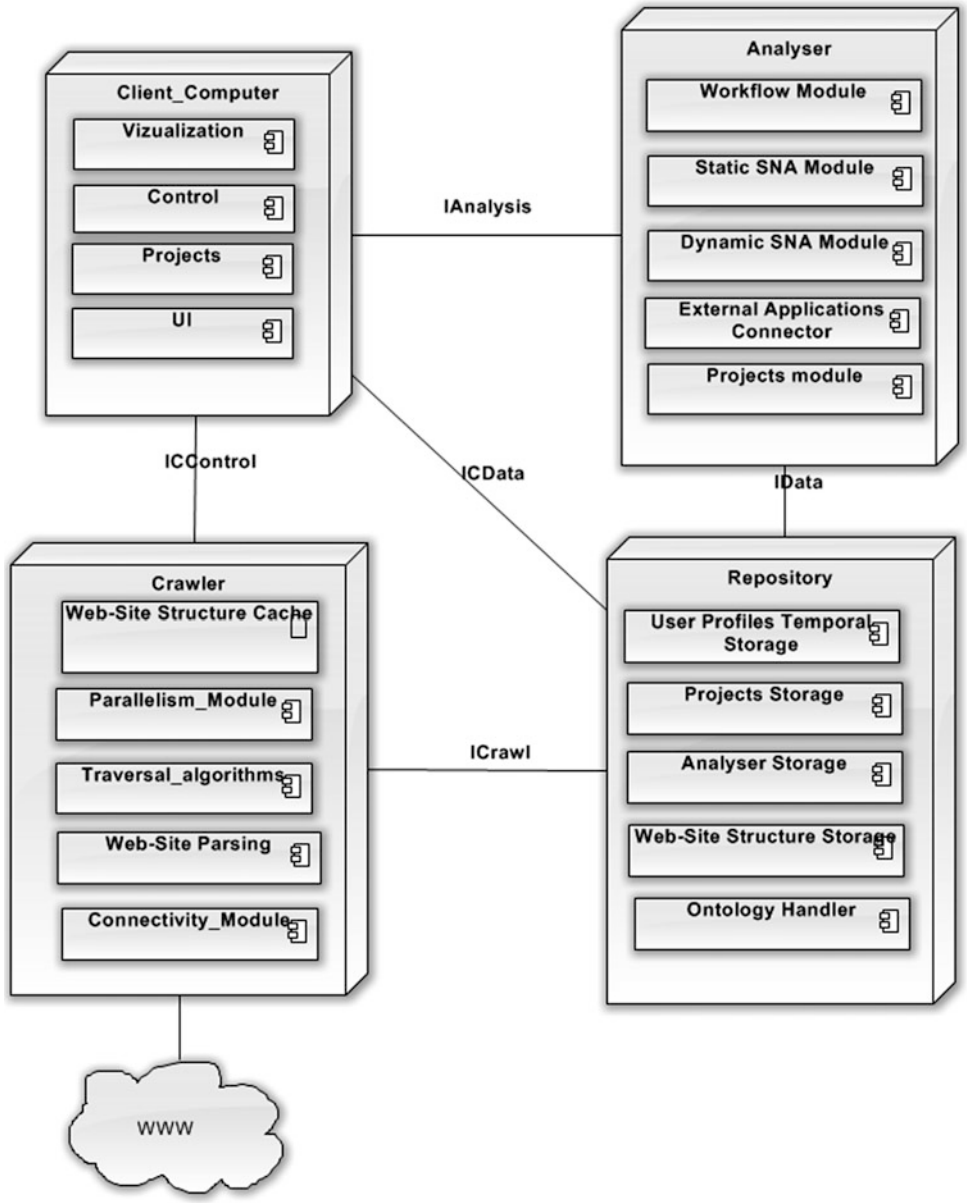
The basic assumption while designing the repository is that it and other necessary software components reside somewhere else (in Internet) than the social media sites it is monitoring and gathering the data from. This means that there is always a communication channel between the monitoring and analysis site and the social media site targeted. The channel capacity and the processing capacity of the monitoring and analysis site set the limits for the amount of data that can be stored and analyzed in a time unit. Further limitations are posed by the APIs of the social media sites through which the data must be fetched by monitoring and analysis software. For instance Twitter receives 500 million tweets per day and none of the free-of-charge API offers the entire stream. Similarly, the API returning the followers of a certain user has the rate limit 5000 followers/15 minutes that alone makes it impossible to remotely monitor all changes in the follower relationships of all hundred millions of users. For a small subset of Twitter users this is possible (if they all allow their followers to be retrieved). The limits for the remote monitoring have been discussed in Semenov and Veijalainen (2013b).

A further assumption related with the above issues is that the repository stores the data fetched by the crawler for the later analysis. That is, it is not primarily designed for real-time data analysis. Still, continuous insertion of data and simultaneous analysis of the data are and must be possible.

The overall architecture is depicted in Fig.1. The Crawler accesses the APIs (or in some cases the browser interface) of the social media sites and fetches the data, i.e., essentially the site ontology instance data as exposed by the APIs. In some cases the corresponding data is extracted from the HTML file returned by the site. The Crawler calls the Repository interface through which the data is stored. The Analyzer module contains individual analyzers that retrieve the data and perform various analysis tasks, such as graph analysis or sentiment analysis. ICControl is used to guide and supervise the functioning of the other modules. It is typically used to notify the Crawler where to gather the data from, for how long, how is the resulting dataset named, etc. It also commands the analysis tasks to be run by Analyzer, how to store the analysis results, etc.

In practice, the Crawler can dump the raw data containing the site ontology instances needed in a particular modeling endeavor and fetched through a social media API into a file or a JSON-enabled database, like MongoDB. The first data extraction operation is then run later, after the data collection, or in parallel, against this intermediate raw dataset and the results stored into a relational or graph database. The data might be also fed piece by piece into the repository without intermediate storage, if the extraction task is simple and can be done at the same pace as the data collection from the site. What kind of raw data is fetched and what data extracted and stored later depends on the modeling and analysis task.

The Repository is constructed based on a view that it must support multirelational directed graphs. The most general repository would be such that upon establishing a new graph model, one could specify in a CREATE operation what attributes the nodes would contain, and all edge types of the graph instance. The repository would then create a suitable database scheme and a corresponding graph instance into the graph or other database when the node and edge instance data would be fed in. Another approach is to try to design a meta-template of multirelational graphs and instantiate any graph instance using it. We follow this path below.



Repository for Multirelational Dynamic Networks, Fig. 1 System architecture, from (Semenov et al. 2011)

Repository Interface

Figure 1 depicts the architecture of the system. The Repository keeps the data of the evolving multirelational graphs. It also provides query interface for the analysis tasks. The collected time-varying multirelational directed graph instances are characterized by unique instance identifiers that relate the parts to each other. Graph operations presented below are a central part of the repository interface (Table 1). The idea is that each graph with a unique <name> hosted by the Repository is a sequence of snapshots corresponding to the data collections. Each snapshot has a unique identifier ID. The Repository returns it whenever addGraph() is called. If the input <name> is already known to the

Repository for Multirelational Dynamic Networks, Table 1 Repository queries

#	Description	Name	Return
1	Add graph instance X with name at time T to the repository	addGraph(X, name, T)	ID of the instance X
2	Add/remove node N of type C to graph ID at time T	addNode(ID, N, T, C) rmNode(ID, T, N)	Node
3	Add/remove edge E connecting nodes N1 and N2 to graph ID at time T	addEdge(ID, T, N1, N2) rmEdge(ID, T, N1, N2)	Edge
4	Add attribute A to node N/edge E of graph ID at time T	addnAttr(ID, A, N) rmnAttr(ID, T, A, N) addeAttr(ID, A, E) rmeAttr(ID, T, A, E)	
5	Extract graph G (slice of the instance ID) at time T, in a predefined format	getGraph(ID, T, F)	Graph G format F
6	Extract validity interval for instance ID	getInt(ID)	time interval [t1,t2]
7	Extract sequence of the graphs {G} during time interval [t1, t2] from instance <name>	getSeq(name, t1, t2)	A list of graphs G
8	Extract inserted/deleted nodes/edges in graph <name> during [t1,t2]	getNewNodes(name, t1, t2) getNewEdges(name, t1, t2) getDelNodes(name, t1, t2) getDelEdges(name, t1, t2)	List of nodes/ edges
9	Extract graph, induced by certain types of nodes/edges from the graph <name> during [t1,t2]	getGraph(name, types)	Graph G
10	Get attribute value of node N/edge E of <name> at time T	getVal(name, N/E, attr, T)	Value
11	Get node by attributes	getNode(attrA, name, type)	Node
12	Get edge by two nodes and type	getEdge(N1, N2, name, type)	Edge

Repository, it just returns a new unique ID. Otherwise it records a new graph <name> and returns ID for the first snapshot.

Operation (1) is called from the control module; then, crawler starts data collection and adds nodes, edges, and attributes to the repository. Node/edge/attribute removal operations can be called from the control module. A query for the entire graph extraction (5) in predefined format (such as GraphML, Pajek format, etc.) can be used for feeding the graph into third-party graph analysis software, such as Pajek or Gephi, or implemented within analyser module algorithms; the same holds for query (7). Query (8) can be used for example by application of stream algorithms or a visualization of graph dynamics. Query (9) can be used for extraction of the graphs, consisting of nodes of predefined types only, and further sending them as input to various analysis submodules. A Node, an Edge, and a Graph (from Table 1) are internal data types, storing attribute values and types of the entities (thus, type is not presented in the list of parameters).

Repository Design: Third-Level Modelling

There are many ways to represent a graph, including adjacency matrix, adjacency list, incidence matrix, and incidence list. These data structures

can be stored in an RDBMS such as PostgreSQL ("PostgreSQL: The world's most advanced open source database" 2016), in plain files, or in a distributed file system like HDFS, accessible by Hadoop (Kang 2012). Graph databases, using their modelling facilities and physical schemes can be used, too. As alluded to earlier, the repository could implement different kinds of graph instances using different means and might offer for the same graph instance several parallel implementations that best fit the analysis tasks at hand. In this context, however, we only look deeper at ordinary relational database implementation. This can be seen as the third-level model of the social reality.

Snapshot/Temporal DB Scheme

When a temporal multirelational directed graph is stored as a sequence of the snapshots, a persistent data storage should contain an individual snapshot of the graph G, for those time moments T= [t_st, t_end], when the data from a social media site corresponding to the instance of the multirelational graph was collected. Since crawling of the website takes place during a time interval rather than in a single moment, we consider T as an interval where t_st marks the collection start and t_end the timestamp of the collection end. Whenever the complete snapshot data is fed into the repository, t_ts = t_end. These two columns represent validity time intervals of the entities (Jensen 2000). The graph can be represented as the following DB scheme:

NODES (i_id, node_id, n_type, t_st, t_end);
EDGES (i_id, node_from, node_to, e_type, t_st, t_end);
N/E_ATTRS (i_id, n_id, a_id, a_value, t_st, t_end);
MDATA (i_id,name, t_st, t_end).

NODES is the table with node data, node_id is the identifier of the node, and node_type is the type of the node, T the timestamp as above. EDGES contains edges of the graph, where node from and node_to are both node identifiers; node_id, which are end points of this particular edge; and edge_type represents the type of the edge in question. Tables N_ATTRS and E_ATTRS contain attributes for the nodes and edges, respectively. Attr_id is the identifier of the attribute, attr_value is the value of the attribute, and node_id and edge_id are identifiers of the node and edge the attribute is related to. T is the timestamp; i_id is the identifier of a crawled site snapshot, relating it to its metadata (a table MDATA associates instance i_id to its text description "name" and also contains the related interval).

While crawling, the Crawler keeps i_id and T in the main memory. When addNode operation is called, its implementation checks the existence of the Node in the N_ATTRS table (by selecting its attribute, e.g., username for LJ), and adds a row, identified by node_id to the NODES table. If the node does not exist in N_ATTRS, its attributes are added to N_ATTRS and addEdge function adds row to EDGES table. Extraction of the graph G at the moment T from a specific snapshot database is straightforward and includes only extraction of a particular snapshot from the DB, identified by T and the instance id (i_id). In addition, queries extracting subgraphs of nodes or edges of various types are also supported. In that case node_type, or edge_type should be specified in the query. Operations getting the difference between the graphs might be implemented using, e.g., SQL EXCEPT statements. Although data insertion and extraction procedures for the snapshots are conceptually simple and can be easily implemented, the problem of this method is the quick growth of the database. Thus, for graph having N nodes there can be $N^*(N-1)$ edges maximum, and from 0 to A_N attributes for each node N, and from 0 to A_E attributes for the edges. Because the number of users can be hundreds of millions, already the number of the rows in the edge table can reach 10^{16} in theory for the biggest sites. But in practice the number of edges is much smaller, because an average user only has a few hundred friends or followers and some sites restrict the number of connections, like Facebook to 5000 friends and vk.com to 10000. This is in practice also a high number. A recent study of the number of friends in Facebook showed that it is in average 100–200 (Backstrom et al. 2012) and in a

big Twitter dataset the median number of followers was 280 (Veijalainen et al. 2015). These are close to the so-called Dunbar's number.

A realistic estimate for the total size of the database in bytes would be:

$$\text{MAXSIZE}(G) = SIZE\_\text{NODE}\, N + \text{SIZE\_EDGE}\, N\, 200 + \text{SIZE\_ATTRIBUTE}\, (N + 200\, N) \quad (1)$$

Putting $N = 10^8$ and each SIZE = 20 B, (1) would yield ~2*200*10^8*20B = 0.8 TB. The EDGE table would have 200*10^8, i.e., 20 billion rows. Collecting the data for a model of this size from any site would already be quite a challenge. Analyzing it in a reasonable time is another challenge. Still, the size of the database would grow linearly w.r.t. the number of profiles in the model. In addition, the same or almost the same graph might be copied into different snapshots, although almost all nodes and edges would remain the same. Thus, the overall space consumption would be MAXSIZE(G) * N_OF_SNAPSHOTS.

The space consumption could easily be reduced by the temporal use of the scheme. While an element is inserted, its validity period should be checked in the Repository. In case the element is valid for the current collection time, its t_end is updated to the current t_end in the input parameter. If the element does not exist in a table, a new row is created with a unique identifier and the current parameter values t_st and t_end inserted to the row. Existence of the node in a table is checked by using its unique identifier or its attribute values. For an edge the unique identifier consists of the pair of unique node identifiers and type. The MDATA table contains metadata.

Complexity of the Main Operations

Let us estimate the complexity of the add graph and get graph operations for both cases: complete snapshot case and the delta case.

For the addGraph(X, name, T) in a snapshot case it is O(n*e), where *n* is the number of nodes and *e* is the number of edges, because two loops are required: one for each node and the second one for each edge to add the whole new snapshot of a graph.

The time complexity of the add operation for the delta case relying on validity intervals is as follows. We assume that a B+-tree indices are kept for NODES and EDGES tables. Detecting an existing node or that it is missing from the table requires O(log n_nodes) and O(log n_edges) index search operations. Inserting a new node to the tables above and updating the corresponding indices can be considered insignificant as compared to the index search. The same holds for updating the validity interval end. Thus the time complexity of add operation is O(log n_nodes) or O(log n_edges). Because typically n-edges < 300*n_nodes, the overall time complexity is at most O(log n_nodes).

The complexity of getGraph (ID, T, F) would be O(log n_edges) ~ O(log n) if we use B+-tree search index for ID and T columns.

Examples

A Simulation Example

We model a hypothetical video sharing site contents.

The schemes of the tables NODES, ATTRIBUTES, and EDGES are as presented above. A MDATA table is not modeled.

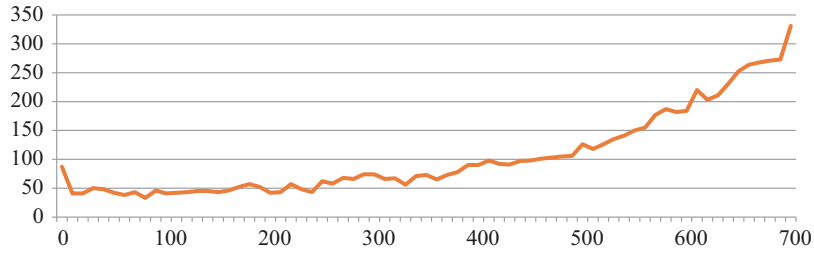
We present the analysis of the data growth, estimated using constructed discrete time simulation model. Simulation model models the network with the following configuration: types of the nodes: {User, Video}; types of the attributes: User: {username, name_of_channel}; Video: {video_url, video_name, header, likes_num, dislikes_num}; types of the edges: {Subscriber, Upload}. The site ontology contains operations presented in Table 2, along the probabilities with which they occur at any discrete time moment.

Additional rules: video can be uploaded by an existing user, but when a user is removed all his or her videos are removed. A "Subscribers" relation exists between the users only. The initial graph is modeled as Erdős–Rényi (Erdős and Rényi 1960) graph with 10 as the number of initial nodes, and 0.1 as a probability of edge establishing between two nodes. We run the model for N = 709 timestamps (for temporal DB). Figure 2 depicts insertions of the rows to NODES table, where

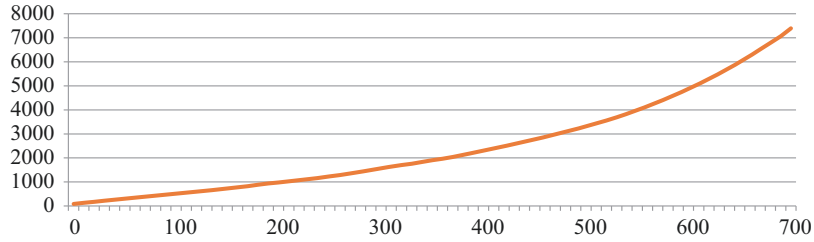
Repository for Multirelational Dynamic Networks, Table 2 Probabilities of the events

Action	Probability	Action	Probability
Add node	0.1	Remove subscriber	0.1
Remove node	0.1	Change name of the channel	0.1
Add video	0.1	Change video name	0.1
Remove video	0.1	Update likes	0.9
Add subscriber	0.1	Update dislikes	0.9

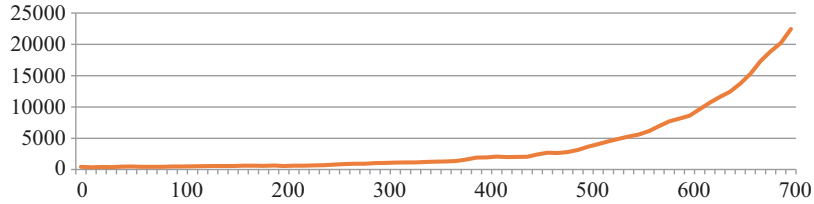
Repository for Multirelational Dynamic Networks, Fig. 2 NODES table insertions



Repository for Multirelational Dynamic Networks, Fig. 3 NODES table growth



Repository for Multirelational Dynamic Networks, Fig. 4 EDGES table insertions

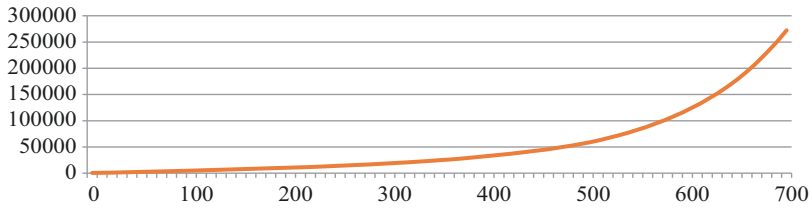


X axis represent time moment, and Y axis represent number of additions for the interval [X, X+10). At the very end 331 rows were added and the table had 7396 rows. Figure 3 depicts the growth of the table NODES.

Figure 4 depicts insertions of the rows to EDGES table, and Fig. 5 depicts the growth of the EDGES table. At the end there were 272456 rows. At the end table N_ATTRIBUTES had 103807 rows.

From the pictures we can see that the growth of the EDGES table is faster than NODES, as one can expect if the simulation is reflecting the behavior of real users.

The graph's evolution simulation in our example is modeled by Erdős–Rényi model, as we mentioned above. The growth rate of the set of nodes is a parameter of the model that can be set by a researcher and may vary. For instance, the numbers of nodes may be fixed or it may grow



Repository for Multirelational Dynamic Networks, Fig. 5 EDGES table growth

slower, which will result in accumulation of the majority of the changes in the network configuration among edges and decrease the node set growth to a logarithmic dependence. In any case, the stored changes in the graph configuration consume less space in the NODES and EDGES tables than in the case of storing the whole new snapshot of the new graph.

As concerns the time complexity of the updates or data retrievals, this was not simulated. It is clear that in a real database installation several indices must be specified for each table. Otherwise the Repository interface operations would need sequential scans of the table data.

A Real Example: Twitter

One example of a data gathering with a Crawler using API can be described for the Twitter social platform. Twitter is a social media platform aimed at microblogging. The maximum length of posts, called tweets, is only 140 characters. A tweet can be "retweeted," i.e., resent by a user to the "followers." A "reply" to the sender is also possible. Tweets dealing with a certain theme can be marked by a hash tag, which is a character string included with the tweet and starting with # sign. A tweet can contain more than one hash tag, and the search engine of the site returns the (top most tweets of the) tweet chains containing the hash tag in the search query. URIs can also be included into tweets, thus guiding other users to more extensive stories – a powerful feature for fast information sharing.

Twitter identity is represented as a user profile, which has a profile identifier as an immutable identifying attribute inside the site. This is only accessible through the APIs, whereas at the web interface the modifiable screen_name identifies the profile. The site ontology allows further user identity cues, such as profile picture and name. The profile also contains information about the followers and followees the user has, the tweets he or she has ever sent or retweeted, and the tweets he or she has liked. In order to send tweets, the user must have a profile and must log in to the site under the profile. Public tweets can be accessed without a login through browsers.

The site ontology offers an operation "follow" through which a user can by default follow any other user without the consent of the latter. A user can, however, select also a profile setting that allows him or her to accept or decline a "follow" requests. A user can also make his or her tweets hidden. In that case only the followers can retrieve them, but the site search engine does not return them. The semantics of the "follow" operation creates an asymmetric relationship between profiles, because the followee will not follow the follower, unless the latter explicitly establishes the relationship. The site shows which other profiles are following a user profile and offers that user the option to follow his/her followers. Operational semantics of "being followed" in the site ontology is that the following user gets access to the tweets of the followed ones, whenever he/she has logged in to the site or also without a login, if the tweets are public. The site can also send notifications on new tweets through email and many smartphone Twitter apps, while the user is logged on.

Twitter site offers several types of APIs, among them REST APIs (see https://dev.twitter.com/rest/public) and Streaming APIs (see https://dev.twitter.com/streaming/overview). Using the APIs requires that the user has registered one or several applications at Twitter website. An application has a unique name and towards the site it is identified by four credentials (strings) generated by the site and adhering to the OAuth authentication model (see https://dev.twitter.com/oauth). Remote access to the API first requires authentication

using the OAuth protocol using the credentials above, after which the established session can continue hours or even months to download data from the site (or upload data to the site). The APIs exhibit many rate limits (see https://dev.twitter.com/rest/public/rate-limiting) that restrict the number of API objects that can be retrieved through them in a time unit. The API objects that can be retrieved and that belong to the site ontology in a wider sense are: Tweets, Users, Entities, and Places. Tweets and users have already been described above. Entities (see https://dev.twitter.com/overview/api/entities) provide metadata and additional contextual information about content posted on Twitter, i.e., hashtags, media, urls, user_mentions, retweets issued. Places are specific, named locations, sometimes with corresponding geo-coordinates (see https://dev.twitter.com/overview/api/places). The metadata attached to a tweet can contain the exact WGS-84 coordinates, from where the tweet was sent, if the smartphone app used to send it was allowed to attach them to the tweet. The metadata could alternatively contain coordinates for the corners of a "bounding box" that shows a larger area on Earth from where the tweet was sent. Exact address, the name of the city, etc., can be included into the attributes of Place. This location information is currently not shown to the ordinary users accessing the site through a browser or an app, but it is highly useful in many monitoring and analysis tasks and accessible through APIs. The site allows, for instance, search based on components of Place through the REST API. Thus, for instance, tweets sent from a certain city can be found from the site or later extracted from the collected tweet data.

A real data Twitter collection and analysis is described in (Semenov et al. 2013) and (Veijalainen et al. 2015). In this case we first collected over 8 million tweets after the Boston bombing that happened on April 15, 2013. We started the collection about an hour after the bombs exploded and continued about 5 days. Another direction was to look at the activity of the perpetrators (Tsarnaev brothers) at various social media sites. This is reported in (Semenov et al. 2013a).

At the graph modeling level we have two different node types (users and tweets), a user node has only one attribute uid (of type long integer). The tweet node has several attributes, like tweet_id (long integer), retweeted_id (long integer), uid (long integer), and time-stamps (time stamp). There are three types of edges. One modeling the "follows" relation among users, "retweet" among tweets, and "sent by" between users and (re)tweets.

The keyword used to select tweets to the original tweet set from Twitter Streaming API was "Boston." There were over 4 million different users who had sent tweets. After that we collected the followers of those users, as far as it was possible. Our main goal was to investigate the influence of various users, especially through retweeting activity, as well as the life time of retweet activity. There were about 8 billion rows in the follower table after many months of data collection. The scheme of the follower table was simply FOLLOWERS (uid1, uid2, timestamp), where id1 and id2 are the unique identifiers of the users inside Twitter and timestamp indicates the data collection time. We extracted several attribute values from the available metadata of the tweets, such as the unique tweet_id, retweeted_id, user_id, screen_name of the original sender and the possible retweeter, the tweet text, timestamps from the metadata of the tweets. This made it possible to construct a table where the tweets and their retweets can be selected using the unique identifiers by SELECT queries. The follower data was related with the tweet data in order to find out how high the retweet cast tree actually is. Only a follower can retweet the original tweet or already retweeted instance of the tweet and the timestamps of the retweets sets an order in which the involved profiles could have retweeted the tweet. Unfortunately, the order cannot be uniquely constructed, if a retweeter follows the original tweeter and some retweeter(s) or several retweeters who appear earlier in the timestamp order. The cast size can be found without this information, by simply selecting all tweets where the retweeted_id is the tweet_id of the original tweet and counting the size of the result set.

Repository for Multirelational Dynamic Networks, Table 3 LJ database, NODES

i_id	node_id	n_type	t_st	t_end
1	1	"profile"	0	1
1	2	"profile"	0	1
1	3	"profile"	0	1
1	4	"profile"	0	1
1	5	"profile"	0	1

Repository for Multirelational Dynamic Networks, Table 4 LJ database, EDGES

i_id	node_id	attr_id	attr_value	t_st	t_end
1	1	"username"	user_1	0	1
1	2	"username"	user_2	0	1
1	3	"username"	user_3	0	1
1	4	"username"	user_4	0	1
1	5	"username"	user_5	0	1

Casting the above example to the Repository structure, NODES would contain user_ids and tweet_ids in node_id, the type information "tweet," "user," and the time interval would be the collection interval. EDGES would contain the FOLLOWER table data, enhanced with the type "follower" collection identifier i_id and retweet identifier data extracted from the tweet table. N/E_ATTRS would contain the selected attribute values collected for the tweets. Time interval information would be taken from the timestamp of the collection.

The EDGES table would in this case contain more rows than the FOLLOWER table. Retrieving the data in an efficient manner is important. Both node_id:s should be indexed. This increases the space occupied by the model in the database. The time complexity of finding a follower for an arbitrary profile would be roughly O(log(Navf*N)), where N is the number of profiles in the dataset and Navf is the average number of followers. In our case Navf was around 2000, but the median was 280.

Assuming that the nodes in EDGES are 20 bytes each, and the attributes of a tweet are 100 B each, the model instance would require, $N_{tw}{}^{*}100B + N_f{}^{*}20B$ where N_{tw} is the number of tweets modeled, N_f is the number of nodes in EDGES. Putting $N_{tw} = 4 * 10^6$ and $N_f = 10^{10}$ yields values in the range of $2^{*}10^{11}$, that is 200 GB or more. This would still be a small model, as compared to the relationships present and the amount of data Twitter hosts. The graph model above does not, for instance, address at all image and video material Twitter hosts.

Thus the tweet set and profile set modeled must be collected in a way that suits the modeling task at hand, locally available resources and one must remember the restrictions of the APIs when the model is designed.

A similar approach can be used for other social networks, like Facebook, APIs of which are documented well (see https://developers.facebook.com/docs/graph-api).

Another Real Example: Livejournal (LJ)

An example of the schema used for real collection of the data from LiveJournal.com website is presented in Tables 3, 4 and 5. Data are anonymized. Here, a NODES table contains data on LJ user profiles, an EDGES table contains "friend" relation between two users, and a N_ATTRIBUTES table contains usernames. All entities have "1" as their instance, since this was one data collection aimed at the analysis of the entire LJ site.

Key Applications

As alluded above, this monitoring and analysis software architecture is primarily designed for non-real-time social media data analysis. It is especially suited for a user relationships capture

Repository for Multirelational Dynamic Networks, Table 5 LJ database, N_ATTRIBUTES

i_id	node_from	node_to	edge_type	t_st	t_end
1	1	3	"friend"	0	1
1	1	4	"friend"	0	1
1	1	2	"friend"	0	1

and analysis of their changes over time. Because the repository is also capable of storing multimedia data, the corresponding data analysis tasks are possible. A simple example is a sentiment analysis in a tweet collection or response time to a customer's complaints and comments through Facebook by a company.

Future Directions

A future research direction is an analytical estimation of a size of storage for the temporal scheme, and an analytical estimation of the procedures for insertion and extraction of the data. Another direction is a more exact estimation of the parameters of the simulation model for achieving more accurate results of the simulation. This includes substitution of the constant probabilities with functions, depending on the network characteristics and estimation of functions' parameters by an analysis of growth of real multirelational graphs. An additional direction is implementation of the dynamic algorithms (which make use of results of the previous solutions) and its adoption to temporal scheme of storage of multirelational graphs. A further issue is to study how different DBMSs could be simultaneously used behind the repository interface to store and retrieve the collected data. Finally, one should be able to import ready models to the repository that are already inserted into database tables either by the Crawler or somewhere else and are ready for analysis.

Cross-References

- ▶ Analysis and Visualization of Dynamic Networks
- ▶ Collecting Qualitative Data to Enhance Social Network Analysis and Data Mining
- ▶ Collection and Analysis of Relational Data from Digital Archives
- ▶ Evolving Social Graph Clustering
- ▶ Exploring Actor-level Dynamics in Longitudinal Networks: The State of the Art
- ▶ Extracting Rich Social Network Data via the Facebook API
- ▶ Graph Classification in Heterogeneous Networks
- ▶ Mining Social Media Data
- ▶ Multigraph
- ▶ Multilayered Social Network
- ▶ Multirelational Social Networks
- ▶ Network Dynamics
- ▶ Network Models
- ▶ Network: Graph
- ▶ Relational Models
- ▶ Research Designs for Social Network Analysis
- ▶ Social Media, Definition, and History
- ▶ Social Search and Querying
- ▶ Social–Spatiotemporal Analysis of Topical and Polarized Communities in Online Social Networks
- ▶ Sociograph Representations, Concepts, Data, and Analysis
- ▶ Temporal Analysis on Static and Dynamic Social Networks Topologies
- ▶ Temporal Networks
- ▶ Topology of Online Social Networks
- ▶ Tracking Dynamic Community Evolution and Events Mobility in Social Networks

References

Aggarwal CC (ed) (2011) Social network data analytics, 1st edn. Springer, New York

Aggarwal CC, Yuchen Zhao, Yu PS (2011) Outlier detection in graph streams. Presented at the 2011 I.E. 27th international conference on data engineering (ICDE), IEEE, pp 399–409. https://doi.org/10.1109/ICDE.2011.5767885

Backstrom L, Boldi P, Rosa M, Ugander J, Vigna S (2012) Four Degrees of separation. In: Proceedings of the 4th annual ACM web science conference, WebSci'12. ACM, NY, pp 33–42. https://doi.org/10.1145/2380718.2380723

Bhattacherjee S, Chavan A, Huang S, Deshpande A, Parameswaran A (2015) Principles of dataset versioning: exploring the recreation/storage tradeoff. Proc VLDB Endow 8:1346–1357. https://doi.org/10.14778/2824032.2824035

Erdős P, Rényi A (1960) On the evolution of random graphs. Publ Math Inst Hung Acad Sci 5:17–61

Erwig M, Güting R, Schneider M, Vazirgiannis M, Erwig M, Güting R, Schneider M, Vazirgiannis M (1999) Spatio-temporal data types: an approach to modeling and querying moving objects in databases. GeoInformatica 3:269–296

George B, Shekhar S (2006) Time-aggregated graphs for modeling spatio-temporal networks. In: Roddick JF, Benjamins VR, Si-said Cherfi S, Chiang R, Claramunt C, Elmasri RA, Grandi F, Han H, Hepp M, Lytras MD, Mišić VB, Poels G, Song I-Y, Trujillo J, Vangenot C (eds) Advances in conceptual modeling: theory and practice. Springer, Berlin/Heidelberg, pp 85–99

Gruber T (2009) Ontology (computer science): definition in encyclopedia of database systems, encyclopedia of database systems. Springer-Verlag, Berlin/Heidelberg

Hazelcast.org (2016) The leading open source in-memory data grid [WWW Document]. http://hazelcast.org/. Accessed 18 Jul 2016

Henzinger MR, King V (1999) Randomized fully dynamic graph algorithms with polylogarithmic time per operation. JACM 46:502–516. https://doi.org/10.1145/320211.320215

Jensen CS (2000) Temporal database management. IEEE Trans Knowl Data Eng 1:1–15

JUNG (2016) Java universal network/graph framework [WWW Document]. http://jung.sourceforge.net/. Accessed 19 Jul 2016

Kang U (2012) Mining tera-scale graphs: theory, engineering and discoveries. Carnegie Mellon University, Pittsburgh

Kaplan AM, Haenlein M (2010) Users of the world, unite! the challenges and opportunities of social media. Bus Horiz 53:59–68. https://doi.org/10.1016/j.bushor.2009.09.003

Kazienko P, Musial K, Kukla E, Kajdanowicz T, Bródka P (2011) Multidimensional social network: model and analysis. In: Jędrzejowicz P, Nguyen NT, Hoang K (eds) Computational collective intelligence: technologies and applications. Springer, Berlin/Heidelberg, pp 378–387

Khurana U, Deshpande A (2013a) Efficient snapshot retrieval over historical graph data. In: 2013 I.E. 29th international conference on data engineering (ICDE), pp 997–1008. https://doi.org/10.1109/ICDE.2013.6544892

Khurana U, Deshpande A (2013b) HiNGE: enabling temporal network analytics at scale. In: Proceedings of the 2013 ACM SIGMOD international conference on management of data, SIGMOD'13, ACM, NY, pp 1089–1092. https://doi.org/10.1145/2463676.2465262

Koloniari G, Pitoura E (2013) Partial view selection for evolving social graphs. In: First international workshop on graph data management experiences and systems, GRADES'13. ACM, NY, pp 9:1–9:6. https://doi.org/10.1145/2484425.2484434

Liquigraph by fbiville (2016) WWW document. https://fbiville.github.io/liquigraph/latest/index.html. Accessed 14 Aug 2016

Liu B, Menczer F (2011) Web crawling. In: Carey MJ, Ceri S (eds) Web data mining. Springer, Berlin/Heidelberg, pp 311–362

neo4j (2012) World's leading graph database [WWW Document]. http://neo4j.org/. Accessed 19 Mar 2012

Network Data Model Overview (2012) WWW document. http://docs.oracle.com/cd/B28359_01/appdev.111/b28399/sdo_net_concepts.htm. Accessed 10 Apr 2012

Oracle Berkeley DB (2016) WWW document. http://www.oracle.com/technetwork/database/database-technologies/berkeleydb/overview/index.html. Accessed 18 Jul 2016

PostgreSQL (2016) The world's most advanced open source database [WWW document]. https://www.postgresql.org/. Accessed 15 Aug 2016

Redis (2016) WWW document. http://redis.io/. Accessed 18 Jul 2016

REST APIs – Twitter Developers (2017) WWW document. https://dev.twitter.com/rest/public. Accessed 15 May 2017

Salzberg B, Tsotras VJ (1999) Comparison of access methods for time-evolving data. ACM Comput Surv 31:158–221. https://doi.org/10.1145/319806.319816

Semenov A, Nikolaev A, Veijalainen J (2013a) Online activity traces around a #x201C;Boston bomber #x201D. In: IEEE/ACM international conference on advances in social networks analysis and mining (ASONAM 2013), pp 1050–1053. https://doi.org/10.1145/2492517.2500316

Semenov A, Veijalainen J (2013b) A modelling framework for social media monitoring. Int J Web Eng Technol 8:217–249. https://doi.org/10.1504/IJWET.2013.057226

Semenov A, Veijalainen J, Boukhanovsky A (2011) A generic architecture for a social network monitoring and analysis system. In: Barolli L, Xhafa F, Takizawa M (eds), IEEE Computer Society, Los Alamitos, CA, pp 178–185. https://doi.org/10.1109/NBiS.2011.52

Smiley PL (1992) Information repository system and method for modeling data. US Patent 5978811 A. Texas Instruments Incorporated. Accessible at http://www.google.ch/patents/US5978811?hl=en

Staab S, Studer DR (eds) (2011) Handbook on ontologies, international handbooks on information systems. Springer, Berlin/Heidelberg. https://doi.org/10.1007/978-3-540-92673-3

Streaming APIs — Twitter Developers (2007) WWW document. https://dev.twitter.com/streaming/overview. Accessed 15 May 2017

Veijalainen J, Semenov A, Reinikainen M (2015) User influence and follower metrics in a large Twitter dataset. In: Proceedings of 11th International Conference on Web Information Systems and Technologies (WEBIST 2015). Presented at the WEBIST 2015, Science and Technology Publications, Lisbon, pp 487–497. https://doi.org/10.5220/0005410004870497

Reproducibility

▶ Recommender Systems Evaluation

Reputation Systems

▶ Fraud Detection Using Social Network Analysis: A Case Study

Research Challenges

▶ Statistical Research in Networks: Looking Forward

Research Design

▶ Process of Social Network Analysis

Research Designs for Social Network Analysis

Katarzyna Musial
Department of Informatics, King's College London, London, UK
Department of Computing and Informatics, Bournemouth University, Poole, UK

Synonyms

Graph analysis; Knowledge discovery in networks; Social network mining

Glossary

SNA	Social network analysis
SN	Social network

Definition

Social network analysis	a set of tools and methods that enable to analyze structures called social networks
Social network	a set of nodes and connections between nodes. Nodes may represent people, organizations, departments within organizations, or other social entities. Connections reflect interactions or common activities between nodes

Introduction

Research design for social network analysis (SNA), as for any other types of research, is a process during which the research question and set of methods that enable to answer the stated question are described. Social network analysis is a multidisciplinary research area, and in consequence a wide range of approaches to analyze network data exists. Nevertheless, each study in the field of social networks contains the following stages: (i) selecting sample, (ii) collecting data, (iii) preparing data, (iv) choosing and applying the method of social network analysis, and (v) drawing conclusions. Each of the elements is equally important, and mistakes made during designing one of them can cause that conclusions drawn from the study may be invalid.

The main goal of this work is to describe each of the enumerated above phases. This will help researchers from different backgrounds to understand the main concepts connected with research design for social network analysis and make them aware that none of the steps can be neglected and that each of the stages should be carefully planned and executed.

Historical Background

The concept of social network, first coined by J. A. Barnes in 1954, has been included in a field of study of modern sociology, anthropology, geography, social psychology, and organizational studies for the last few decades.

The person who built the modern social network theory and designed and conducted the first well-known research experiments using early concepts of social network analysis was Stanley Milgram. He studied the small-world phenomenon, which states that if persons x and y do not know each other, then in order to reach from x to y, one needs to travel through a chain consisting of at most five people (Pool and Kochen 1978; Travers and Milgram 1969). The theoretical model of this small-world phenomenon was created by Pool and Kochen (1978) and served as the basis for Milgram's research that was purely pictorial. Stanley Milgram conducted two experiments – Kansas Study and Nebraska Study – in which he asked many people from one city to forward a letter to a chosen person in another city. The only stipulation was that a sender could only forward this letter to a person whom he or she knew on a first-name basis. Afterward, Milgram analyzed the results of the experiment and concluded that people in the USA create the social network, and they are connected within this network with "six degrees of separation." It means that a message in such a network would be delivered on average through the usage of five intermediaries (Pool and Kochen 1978). Kochen confirmed that this value is relatively stable even if the starter selection criteria is changed (Degenne and Forse 1999). Howard claims that six degrees of separation may by true off-line, while less than three degrees is more likely in an online case (Howard 2008).

Since 1967, social networks have become one of the research areas where scientists from different fields are looking for inspiration and new methods of network analysis have been developed. The concept of social network has been studied in many different contexts, e.g., corporate partnership networks (law partnership) (Lazega 2001), scientist collaboration networks (Newman 2001; DiMicco et al. 2008), movie–actor networks, friendship network of students (Amaral et al. 2000), a set of business leaders who cooperate with one another (Liben-Nowell and Kleinberg 2003; Robins and Alexander 2004), sexual contact networks (Morris 1997), customer networks (Yang et al. 2006; Kazienko and Musial 2006; Golbeck and Hendler 2006), labor markets (Montgomery 1991), public health (Cattell 2001), and psychology (Pagel et al. 1987). Recently, with the expansion of the Internet and the increasing popularity of social and collaborative computing, social networks have emerged as a significant and promising field of study within computer science (Musial and Kazienko 2012). Social computing involves activities such as collecting, extracting, accessing, processing, computing, and visualizing of all kinds of social information.

The fact that social networks have been investigated in many areas, different approaches to network analysis have been developed depending on the focus and research interests of a specific group of scientists. Nevertheless, each of these approaches has one important element in common – they cope with network data, i.e., the emphasis is put on the connections between people rather than on individuals themselves, and this is a very important characteristic of social network analysis.

Research Designs for Social Network Analysis

This section is devoted to the research designs for social network analysis. First, the concept of SNA is introduced, and after that the goals and methods used in the network analysis are presented and described in details.

Social Network Analysis

In social networks some typical phenomena such as small-world effect (Pool and Kochen 1978), clustering (Davis 1967), both strong and weak ties (Granovetter 1983), and many others may be observed. Various human features, extracted from

user profiles, which can have more or less significant influence on the process of formation of a relationship, can be also discovered. In order to identify these phenomena, the appropriate SNA method ought to be applied.

Social network analysis stems from traditional social analysis used by sociologists and anthropologists in the first half of the twentieth century. After introducing mathematical interpretation of social networks, scientists started developing social network analysis.

SNA can be defined as "the disciplined inquiry into the patterning of relations among social actors, as well as the patterning of relationships among actors at different levels of analysis (such as persons and groups)" (Breiger 2004). Another definition of SNA was proposed by Valdis Krebs: "Social network analysis (SNA) is the mapping and measuring of relationships and flows between people, groups, organizations, computers, web sites, and other information/knowledge processing entities. The nodes in the network are the people and groups while the links show relationships or flows between the nodes. SNA provides both a visual and a mathematical analysis of human relationships" (Krebs 2000).

Social Network Data

Each research design for social network analysis starts with defining what kind of data and for what purpose will be gathered. It should be emphasized that the regular social data (Table 1) is quite different than social network data (Table 2). Traditional social data describes actors, whereas social network data mainly describes connections between actors rather than actors themselves (Hanneman and Riddle 2005). In other words, network data analysis puts emphasis not on the individuals themselves but on the relationships among people (Hanneman and Riddle 2005). Because of the fact that the social network analysis focuses on investigation of connections, it does not mean that SNA is not interested in actors. After drawing conclusions network analysis may focus on actors to retrieve additional information and to better understand the network; however, it is not its primary goal.

Social network data can include information about relation type and character, direction, and weight. Also, more than one type of relation between two actors can be distinguished. All of this can serve as an input for social network analysis.

Research Designs for Social Network Analysis, Table 1 Example of simple social data

Name	Surname	Gender	Age	Marital status
Kate	Davis	Female	29	Single
Frank	Martin	Male	37	Divorced
Jason	Smith	Male	56	Married
Ann	Jones	Female	25	Married
Carol	Damon	Female	43	Single

Research Designs for Social Network Analysis, Table 2 Example of social network data. 0 means that person A does not know person B, and 1 means that person A knows person B

Name A/B	Kate	Frank	Jason	Ann	Carol
Kate	–	1	0	1	1
Frank	1	–	0	0	1
Jason	0	0	–	0	1
Ann	1	0	0	–	1
Carol	1	1	1	1	–

Steps in Social Network Analysis

In social network analysis, five main steps can be distinguished:

- Selecting a sample from population
- Collecting data
- Data preparation
- Choosing and applying the method of SNA
- Drawing conclusions

It should be noted that a specific research design can include all these steps or just a subset of them.

Selecting a Sample from Population

In order to identify and investigate the patterns that occur within the network, first the selection of a group of people (or other social entities) that are to be investigated should be done. Sometimes, due to the research question, some network data can be neglected. For example, if one would like to investigate relationships between teenagers, data about adults are not important and will not be included in the study. Also, the possibility of analyzing every node of the network (especially these huge and heterogeneous) is usually limited by the available resources, and because of that the representative group of actors ought to be chosen for data collection and further analysis. This group of actors is called population (Hanneman and Riddle 2005) or sample (Garton et al. 1997).

Selecting a proper sample is especially important when the data is collected using surveys, questionnaires, or observations (please see the next section). In such situation researchers have limited capacity when it comes to collecting data as they can conduct only certain number of interviews, observations, etc.

Another issue occurs when it comes to network data generated in different device-supported social services. These datasets, stored in the databases, are huge and it causes a problem with their efficient analysis. Although sometimes it is possible to analyze the whole available dataset, in vast majority of cases, the tools and computational power that is at the researcher disposal are not sufficient. Thus, sampling large online datasets, which include information about people, is also a challenging task. One of the approaches to sampling is a random approach which randomly selects a group of users.

In both cases the sampling procedure should be defined before the data collection will be performed. Before the next step, data collection, will take place, it has to be determined how and what data will be obtained. This process is common for all social sciences, and it abstracts from the relational nature of the analysis as at this stage the information about social connections is not available. In other words the sample is selected from a given population and not from a social network graph.

Collecting Data

The next step is data collection. As it was briefly mentioned before, many methods of obtaining network data such as questionnaires, interviews, observations, and artifacts exist (Garton et al. 1997). Preparation that needs to be done prior to employing one of these methods is very time-consuming. Moreover, it enables to gather data about small subset of nodes – up to few hundreds. One of the characteristics, which is perceived to be a shortcoming of these approaches, is that people are aware that they take part in the study so they can be biased and give answers or behave in a way that they think is required. These approaches were commonly used in social sciences throughout the twentieth century and are still very popular especially in cases when data cannot be gathered in a more automatic way.

For the last few years, due to the rapid development and explosion of the Internet and World Wide Web services that enable people to communicate, collaborate, and share interests in the virtual world, the approach to data collection has started changing. For the first time in history, we have at our disposal vast amount of data about people, their activities and interactions that are available online. In this case the process of collecting data is very simple; one just needs to query database to get required data. However, data

preparation and cleansing are far more complex than in the case of traditional approaches and can be seen as data-mining task. As people are not aware of the fact that data about their behaviors is used, they act in more natural way, and this provides information about their "real" actions and reactions. The drawback of this approach is that, in contrary to questionnaires and other traditional methods where the amount of gathered data is small, the volume of available data is large which results in information overload. With the development of information technologies, the slow shift in approach to social network analysis can be observed. The focus is changing from investigation of small (up to few hundreds of nodes) samples to data that includes information about millions of users. Nowadays, the analysis of social networks copes with two types of data: the one that represents regular, small communities and the one that represents online world and online society that has been formed. One can argue that the former one still dominates in social sciences, while the latter one took its origins in computer science and with time made its way to social sciences.

Data Preparation

As it was mentioned before, network data differs from conventional sociological data (Hanneman and Riddle 2005). Network data in contrast to traditional data, which consists of rectangular array of measurements, consists of a square array of measurements. The SNA research has identified three types of data also called units of analysis, which should be and are investigated: relations, ties (Garton et al. 1997), and actors. Actors are nodes of the network that have such characteristics as degree, centrality, prestige, clustering coefficient, and others. Relations describe connections between actors and tie is a set of different relations that can link two actors.

Collected data has to be represented in a way that facilitates the application of SNA methods. A very common representation is graph or matrix. The adjacency matrix describes relationships that exist between actors within the network. For example, matrix in Table 2 represents a social network that is undirected and not weighted. This is not the only type of network representation that can be analyzed. In general four types of networks can be distinguished: (i) undirected–unweighted, (ii) undirected–weighted, (iii) directed–unweighted, and (iv) directed–weighted. An analyst is responsible for choosing network type that will represent the available data. The decision made depends on the type of data or on the type of analysis that one is interested in.

The goal of data preparation is to represent the collected data in a form of network, e.g., matrix or graph. This can be done manually by extracting information from surveys, interviews, and observations or in automatic manner using data-mining techniques for data cleansing. The resulting matrix serves as an input to the next phase of research where the appropriate method of SNA is chosen.

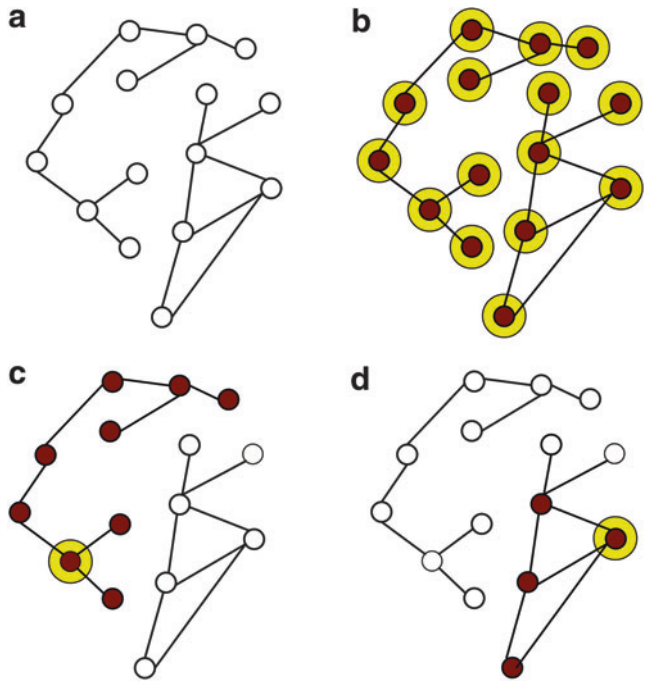
Choosing and Applying the Method of SNA

Once the sample has been selected, appropriate data gathered, and social network extracted, the next step is to perform network analysis on the obtained network. However, before any analysis is done, researcher must decide which part of the extracted network will be used. While "selecting a sample" is concerned with choosing people from the whole population and is common for all social sciences, choosing one of the methods of SNA is concerned with selecting users and their connections from the previously extracted social network and is typical for research where relations are of key importance for the analysis.

The most popular methods, which are currently used to sample social network data, are (Fig. 1):

- Full network method
- Snowball method
- Egocentric method "with alter connections"
- Egocentric method "ego-only" (Hanneman and Riddle 2005; Garton et al. 1997)

The full network method is the most complex one, because all members of the created network and all their possible connections are taken into consideration (Hanneman and Riddle 2005). To analyze the whole network, not only the complete



Research Designs for Social Network Analysis, Fig. 1 Methods of social network analysis. (**a**) Example of social network. (**b**) Full network method. (**c**) Snow ball method. (**d**) Egocentric method

list of connections between people is created but also the links to external environment (Garton et al. 1997). The biggest advantage of this approach is that it provides one full and integrated view on all ties within the network. On the other hand, it is really hard to create such description, because it demands resources and is time-consuming, and sometimes the required information is not available. Additionally, there is always the possibility that some of the connections will be missed, especially in a case of an extensive network with many ties.

An alternative strategy, which is less complex, is the snowball method (Hanneman and Riddle 2005). Firstly, we define a group of actors (nodes) who describe their connections to other people. Next, the same task, i.e., an identification of all outgoing connections, is done for the actors that have been identified in the first step. This recurrence is executed until all ties have been defined, or we have decided to stop creating new ties due to time limits. The biggest shortcoming of this method is the strong possibility that not all connections and not all actors, particularly isolated ones, will be identified. Also, if social network is not connected (i.e., in the undirected representation of network, it is not possible to reach all nodes from every single node), then only some parts of the structure will be analyzed, whereas others can be neglected. One of the approaches is to use random walks to obtain representative group of users.

If there is no need to identify all connections in the network, the egocentric method can be used (Hanneman and Riddle 2005). It focuses on a single individual rather than on groups or pairs. In the first step, one "ego" is chosen. The information about this ego connections is retrieved, together with their target actors and relationships among them. As a result, a subnetwork is created that helps to understand the possibilities and constraints of a given individual. In this approach, we consider the "ego" and their alter direct connections (Hanneman and Riddle 2005). However, the "ego-only" approach can be also exploited. In this case, we are not interested in the connections between the various alters, but we only

concentrate on a single ego and their first level connections (Hanneman and Riddle 2005).

The decision which method to choose should be made based on the type of analysis to be performed. If the research interest is to investigate the global characteristics of network, then full method is the most appropriate, but if the goal is to analyze local structure, e.g., local clustering coefficient, then the egocentric approach will be better. The characteristics that can be investigated on the selected network are centrality, prestige, clustering coefficient, community detection, density, modularity, motifs, and others (Wasserman and Faust 1994). Specific measures and their analysis are described in other sections, so please refer to Cross-References section for more detail.

Drawing Conclusions

The last step that enables to identify the existing within the particular social network patterns is to draw the conclusion from the conducted investigation and answer the research question. Depending on the goal of the analysis, the results can be interpreted only in the context of existing dataset or generalized. In both cases the statistical significance of the obtained results should be investigated. The research outcomes can be compared with the characteristics of existing network models (e.g., random, small-world, or scale-free networks) in order to better understand the phenomena present in a given network.

The issue that has to be emphasized is that collecting network data and picking the right method of analysis are extremely challenging tasks, and it should be done very carefully.

Cross-References

- Centrality Measures
- Clustering Algorithms
- Combining Link and Content for Community Detection
- Community Detection
- Data Mining
- Link Prediction
- Motif Analysis
- Probabilistic Analysis
- Process of Social Network Analysis
- Role Discovery
- Sentiment Analysis

References

Amaral LAN, Scala A, Barthelemy M, Stanley HE (2000) Classes of small-world networks. Proc Natl Acad Sci USA 97(21):11149–11152

Barnes JA (1954) Class and committees in a Norwegian Island parish. Hum Relat 7:39–58

Breiger RL (2004) The analysis of social networks. In: Hardy M, Bryman A (eds) Handbook of data analysis. Sage, London, pp 505–526

Cattell V (2001) Poor people, poor places, and poor health: the mediating role of social networks and social capital. Soc Sci Med 52(10):1501–1516

Davis JA (1967) Clustering and structural balance in graphs. Hum Relat 20:181–187

Degenne A, Forse M (1999) Introducing social networks. Sage, London. MATH

DiMicco J, Millen DR, Geyer W, Dugan C, Brownholtz B, Muller M (2008) Motivations for social networking at work. In: Proceedings of the computer supported cooperative work 2008 conference, San Diego. ACM, pp 711–720

Garton L, Haythorntwaite C, Wellman B (1997) Studying online social networks. J Comput-Mediat Commun 3(1). http://jcmc.indiana.edu/vol3/issue1/garton.html

Golbeck J, Hendler J (2006) FilmTrust: movie recommendations using trust in web-based social networks. In: Proceedings of consumer communications and networking conference, IEEE conference proceedings 1, Las Vegas, pp 282–286

Granovetter MS (1983) The strength of weak ties: a network theory revisited. Sociol Theory 1:201–233

Hanneman R, Riddle M (2005) Introduction to social network methods. Online textbook. Available from Internet: http://faculty.ucr.edu/~hanneman/nettext/ (01 04 2006)

Howard B (2008) Analyzing online social networks. Commun ACM 51(11):14–16

Kazienko P, Musial K (2006) Recommendation framework for online social networks. In: Proceedings of the 4th Atlantic web intelligence conference. Studies in computational intelligence. Beer-Sheva, Israel, pp 111–120

Krebs V (2000) The social life of routers. Internet Protoc J 3:14–25

Lazega E (2001) The collegial phenomenon. The social mechanism of co-operation among peers in a corporate law partnership. Oxford University Press, Oxford

Liben-Nowell D, Kleinberg J (2003) The link prediction problem for social networks. In: Proceedings of the

12th international conference on information and knowledge management, New Orleans. ACM, pp 556–559
Montgomery J (1991) Social networks and labor-market outcomes: toward an economic analysis. Am Econ Rev 81(5):1407–1418
Morris M (1997) Sexual network and HIV. AIDS 11:206–219
Musial K, Kazienko P (2012) Social networks on the internet. World Wide Web J. https://doi.org/10.1007/s11280-011-0155-z
Newman MEJ (2001) The structure of scientific collaboration networks. Natl Acad Sci USA 98:404–409. MATH
Pagel M, Erdly W, Becker J (1987) Social networks: we get by with (and in spite of) a little help from our friends. J Pers Soc Psychol 53(4):793–804
Pool I, Kochen M (1978) Contacts and influence. Soc Netw 1(1):5–51. MathSciNet
Robins GL, Alexander M (2004) Small worlds among interlocking directors: network structure and distance in bipartite graphs. Comput Math Organ Theory 10:69–94. MATH
Travers J, Milgram S (1969) An experimental study of the small world problem. Sociometry 32(4):425–443
Wasserman S, Faust K (1994) Social network analysis: methods and applications. Cambridge University Press, New York
Yang WS, Dia JB, Cheng HC, Lin HT (2006) Mining social networks for targeted advertising. In: Proceedings of the 39th Hawaii international conference on systems science. IEEE Computer Society, Kauai, pp 425–443

Research Dynamic

► Network Science Research Dynamic

Research in Network Visualization

► Arts and Humanities, Complex Network Analysis of

Research Methodology

► Process of Social Network Analysis

Research on Online Health Communities: A Systematic Review

Ronghua Xu[1], Jiaqi Zhou[2], Qingpeng Zhang[2,3] and James A. Hendler[4,5]
[1]College of Economics and Management, Zhejiang University of Technology, Hangzhou, Zhejiang Province, China
[2]Department of Systems Engineering and Engineering Management, City University of Hong Kong, Hong Kong, SAR, China
[3]Shenzhen Research Institute of City University of Hong Kong, Shenzhen, China
[4]Department of Computer Science, Rensselaer Polytechnic Institute, Troy, NY, USA
[5]Computer and Cognitive Science Departments, Tetherless World Constellation, Rensselaer Polytechnic Institute, Troy, NY, USA

Synonyms

Health-related virtual communities; Internet support groups; Online health groups; Online self-help groups

Glossary

CHV	Consumer health vocabulary
LIWC	Linguistic inquiry and word count
NLP	Natural language processing
SNA	Social network analysis. It devotes to analyze the social networks (or social graphs) by using the techniques and methods defined in the field of complex networks, such as centrality measures and community detection
UMLS	Unified Medical Language System
User roles	A set of users' behavioral patterns present in the social context of online communities

Definition

Online health communities (OHCs) are one type of online platforms, which are commonly used as a means to communicate with others, exchange messages, and share health-related information. Most of the users in the OHCs are common patients, and their interactions are sometimes mediated by doctors, nurses, or healthcare providers. These patients get together to share experience, obtain health information, and seek help or support from each other.

Introduction

With the advance of the Internet and Web 2.0 technologies, social media have become a novel platform for Web users to form online health communities (OHCs), such as Patientslikeme.com, breastcancer.org (for breast cancer survivors), and quitsmoking.com (for smokers). On the other hand, mobile-based OHCs have also been growing fast due to the rapid development of mobile devices and the wide adoption of health-related mobile applications. These OHCs, either Web based or mobile based, are commonly used as means to create and update health profiles, communicate with other users, and share and discuss health-related information (e.g., drugs and treatments) (Eysenbach et al. 2004; Neal et al. 2007; Rupert et al. 2014). And they have been playing an increasingly important role in providing healthcare services because of its publicity, broad reach, usability, and immediacy (Zhang et al. 2014). According to the recent report, 35% of US adults have been to online platforms to figure out a medical condition, and 24% have also got information or support from other OHC users with similar health conditions (Fox and Duggan 2013). Interestingly, a significant portion of patients have considered these online sources more helpful than consultation of medical professionals when they seek emotional support and practical advices to cope with day-to-day health situations (Fox 2011).

The main advantage of OHCs is that people can access them for support anytime anywhere without revealing their true identities and concerning about privacy or stigma issues (Moreno et al. 2011; Nature 2013; Hale et al. 2014). By providing medical knowledge and facilitating social support, OHCs can help their users enhance health conditions and lower the use of real-world healthcare services (Coursaris and Liu 2009; Hamm et al. 2014; Choudhury et al. 2014). But there are also a number of concerns, such as the usability issues, misinformation, and the incredibility of the content in OHCs (Stinson et al. 2010; Rupert et al. 2014). One recent example happened in China was the *hemophilia patients community* in *Baidu Tieba*, an online community-based forum service run by *Baidu*. In order for commercial profits, *Baidu* employed the representatives of a notorious drug company as the community managers. This effort triggered huge amount of discussions in China, questioning the ethics of *Baidu* and the trust of the information online. Eventually, *Baidu* claimed that it would stop monetizing its illness-related forums in *Baidu Tieba* (http://www.chinadaily.com.cn/business/2016-01/12/content_23052646.htm). Therefore, in order for the rational use of OHCs by the public, there is a need of a comprehensive review over the feasibility and effectiveness study of OHCs.

Key Points

The analysis and modeling of OHCs can be elaborated from four aspects, the feasibility study, content analysis, social network analysis, and modeling of users' behaviors and roles. In the following of this article, we first introduce some popular OHCs on social media, briefly review the recent research on the analysis and modeling of OHCs following the aforementioned four aspects, and finally discuss the key research questions for future research.

Historical Background

The wide adoption of OHCs provides an ideal data source and a test-bed for researchers to validate hypotheses and theories; existing studies have been conducted from various perspectives,

such as psychology, sociology, and organization science. Through longitudinal observations of online discussions, the ethnographic study described how all aspects of the online community culture were related to each other and the participation in the communities positively influenced users' offline lives by receiving health information and social support (Souza et al. 2004; Maloney-Krichmar and Preece 2005). In sociology, the aspects of social support, social structure, and interaction dynamics among OHC users were extensively examined for better management and assessment of social influence (Nabi et al. 2013; Choudhury et al. 2014). From the perspective of organization science, OHCs were treated as self-organizing communities with similar group dynamics to those in offline face-to-face groups, in terms of the relationship primacy and the structural activity patterns (McNab 2009; Faraj and Johnson 2011; Cain 2011). For instance, the strong group norms of support and reciprocity among OHC users could even make externally governance and management unnecessary, forming the type of self-organized social structures. Besides, the relationships between the social and technological aspects were also studied to facilitate social engagement and interactions among members through better design of the communities (Souza et al. 2004; Vydiswaran et al. 2013; Xu and Zhang 2016).

Online Health Communities on Social Media

There are two major types of social media platforms used in health – the platforms dedicated to healthcare issues and the comprehensive platforms with sub-platforms dedicated to healthcare issues. Table 1 shows some of these platforms and their visions. It is worth noting that the majority of these platforms can also be accessed on mobile devices through specific mobile applications. We do not differentiate whether they are Web based or mobile based in this table. Among the listed platforms, *PatientsLikeMe* and *MedHelp* are typical examples of the first type. In China, there were also a number of such dedicated social media platforms, such as *Xingren* and *Haodf*. Besides these dedicated platforms, people have also created health related sub-platforms on the comprehensive social media platforms. For example, users of *Douban*, a popular Chinese Web 2.0 site, are able to create "interest groups," which are user-created groups consisted of users with the same interests in the specified topic. *Douban* users have created a series of such interest groups for different diseases, including all major illnesses and therapies. Such user-created health groups also exist in other social media platforms, such as *Reddit* and *Facebook*. An example of *Facebook* group is shown in Fig. 1.

Recent Research

Feasibility Study

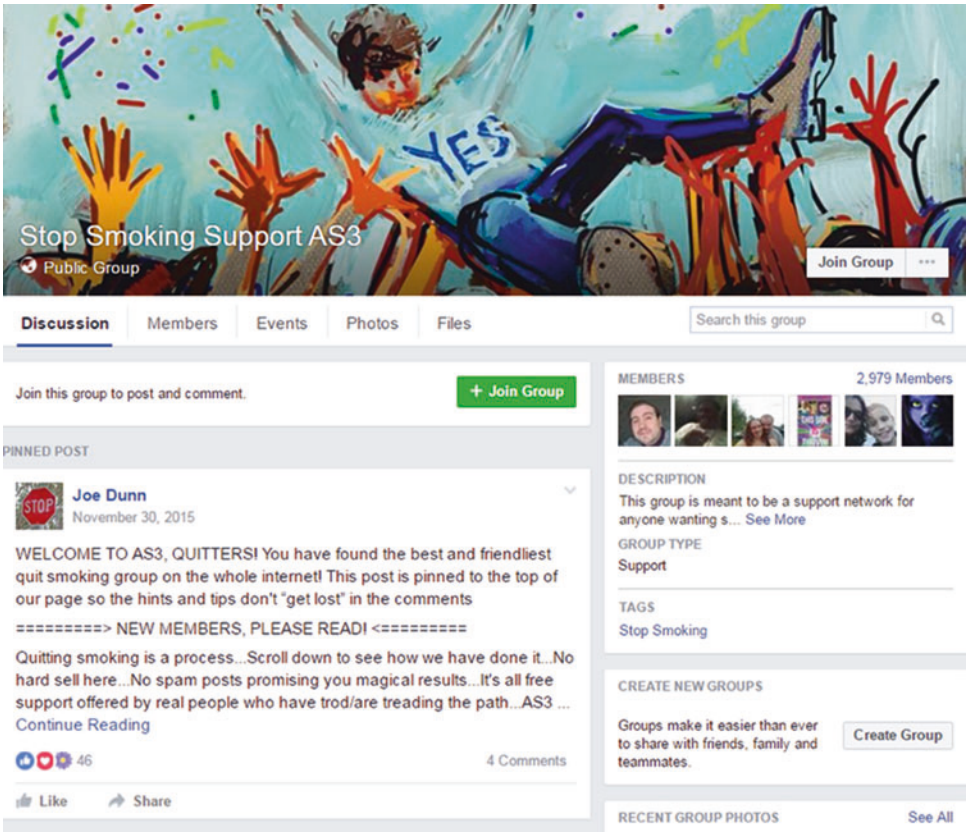
Recent research on the feasibility study of OHCs includes but does not limit to their effects on health condition improvement and the OHC-based health interventions. Hamm et al. conducted a systematic review on the effectiveness of social media (including OHCs) on child health (Hamm et al. 2014). They concluded that the effective attributes of social media included their use as a distraction in younger children and the ability to facilitate communication among adolescent peers (Hamm et al. 2014). In one recent study on the mobile-based OHC of Smarter Pregnancy, Van Dijk et al. analyzed the usage information of 1878 users and stated clear improvements in terms of nutrition and lifestyle behaviors (Van Dijk et al. 2016). In terms of the effects of OHC-based interventions, Ni Mhurchu et al. and Partridge et al. developed a series of mobile-based weight management community programs and demonstrated the positive effectiveness of these mobile-based interventions for weight management (Ni Mhurchu et al. 2014; Partridge et al. 2015). However, the OHC-based interventions were not always effective, especially for patients with some specific diseases. Koufopoulos et al. conducted a 9-week study of online intervention for adherence to asthma medicine but found that joining OHCs did not improve the adherence of asthma patients to the preventer medication (Koufopoulos et al. 2016). Although

Research on Online Health Communities: A Systematic Review, Table 1 Typical OHCs or social media platforms containing OHCs

Name	Vision	Style	Standout feature
MedHelp (www.medhelp.org)	Support and health tracking	Community, information, and tools	To track and understand health condition and share with doctors
PatientsLikeMe (www.patientslikeme.com)	Share, support, and research	Community and health condition record	To track health state, symptom, and treatment and to find similar patients
WebMD (www.webmd.com)	Information sharing	Information service and community	Quality health information
eHealthforum (ehealthforum.com)	Information sharing	Health consultation and community	Professionally moderated social network
HealthBoards (www.healthboards.com)	Support group	Community and blog	Anonymous and peer-to-peer support
Twloha (twloha.com)	Present hope and help	Blog and online events	To encourage, inform, and inspire
Xingren (xingren.com)	Doctor-patient interaction	Blog, clinical reservation, and chat	Patients' self-management moderated by doctors
Name	Vision	Style	Standout feature
Haodf (www.haodf.com)	Doctor-patient interaction	Health consultation and treatment sharing	Professional healthcare service
Facebook (www.facebook.com)	Social networking	Social network	Comprehensive
Twitter (twitter.com)	Events sharing	News and share	Comprehensive
Reddit (www.reddit.com)	Information sharing	Blog and online community	Comprehensive
Douban (www.douban.com)	Information sharing	Blog and online community	Comprehensive
Zhihu (www.zhihu.com)	Question and answer	Forum	Comprehensive

most researchers draw positive conclusions about the feasibility of OHCs to enhance users' health condition, there was still little evidence to support these statements (Hamm et al. 2014). As an example, one recent study on schizophrenia concluded that the effects of OHC-based interventions were still largely unknown and more research were needed to determine their risk and impact on the mental well-being of their users (Välimäki et al. 2016).

On the other hand, quantitative analyses by using statistical models and machine learning methods have also been proposed to evaluate the feasibility and effectiveness of OHCs. The huge amount of data in OHCs can be used to characterize and forecast disclosure of health conditions and health outcomes, such as suicide risk and ideation and mental illness severity (Tamersoy et al. 2015; Chancellor et al. 2016b). Maclean et al. evaluated the effectiveness of OHCs in



Research on Online Health Communities: A Systematic Review, Fig. 1 A screenshot of a typical OHC for quitting smoking on Facebook (https://www.facebook.com/groups/AS3onFB/)

recovering from prescription drug abuse. Activity and linguistic features were used in training a conditional random field (CRF) classifier for labeling users' addiction phrases (three predefined phrases, USING, WITHDRAWING, and RECOVERING). They showed that although relapse of addictions was common, the chances of recovering were still favorable. And there was a significant correlation between the probability that a user was recovering and the user's high community engagement (Maclean et al. 2015). Kiciman et al. developed a logistic regression model to infer the risk of *Reddit* users in transiting from mental health disclosure to suicidal ideation. The factors of linguistic structures, interpersonal awareness, and social interaction and content were used to characterize individual's behavioral and psychological state. From the model of those factors, the authors discovered a number of distinct markers of suicidal ideation (Kiciman et al. 2016). Similar dataset and models were also employed to justify the effectiveness of OHCs in improving addiction-related health outcomes, such as the long-term abstinence from tobacco and alcohol use (Tamersoy et al. 2015). Although some interesting insights have been concluded from the quantitative study on the datasets of the huge amount of samples, there are still some doubt about the quality of the datasets and the accuracy of the statistical models and machine learning methods in concluding such insights. In the future research, it is of urgent need to develop novel techniques specially designed for the datasets of OHCs.

As patients increasingly use the OHCs and access and share personal health information, they also have the concerns of the security and privacy about how their personal information is

R

protected. The fact is that the privacy threats are real and are still not sufficiently addressed by the current legislation and regulations (Huesch 2013). Besides, the adverse online privacy experiences negatively affected patients' information-sharing intentions and impeded the diffusion of health information in these communities (Bansal et al. 2010). But one recent study found that even when people had high concerns of privacy, their attitudes could be positively altered with appropriate message framing in design of the communities (Angst and Agarwal 2009) and these concerns did not deter users from sharing their information, suggesting that the benefits of joining the health communities may outweigh patients' perceived risks to privacy (Vodicka et al. 2013). Moreover, one study of patients' preferences around information sharing showed that patients shared medical details more willingly than identity information, suggesting the need to focus on anonymity (identify information) rather than privacy (medical information) in online communities (Frost et al. 2014). However, the detailed impact of privacy concerns on online behaviors (e.g., usage of OHCs, engagement, and supportive behaviors) has not been well investigated yet, and further research is needed to demonstrate the impact of the privacy concerns.

Content of OHCs

User-generated content in OHCs contains plenty of self-reported clinical and sociodemographic information of users, the huge amount of which is composed of natural data sources suitable for content analysis. Computer-aided automated methods such as natural language processing (NLP), topic modeling, and sentiment analysis are of great interest to understand what was discussed about (e.g., the type of health conditions or medical treatment) and how it was discussed (e.g., opinion exchange or interaction styles) (Blei et al. 2003; Wallace 2012; Ji et al. 2014; Deng et al. 2014). For example, Howes, Purver, and McCabe applied standard topic modeling and sentiment and emotion modeling to analyze online text therapy dialogue for depression and anxiety. Furthermore, linguistic cues of healthcare condition and the expertise of medical advice can also be revealed by the use of Linguistic Inquiry and Word Count (LIWC) (Howes et al. 2014). As an example, it was shown that messages were perceived as more expert if they were longer, with fewer I-pronouns and anxiety-related words and with more long words and negations (Toma and D'Angelo 2014).

Additionally, the standard text analysis methods and user-generated text in OHCs have also been used in knowledge discovery, including identifying medical terms and medical entities and building consumer health vocabulary (CHV, comparable to the professional vocabulary, such as the Unified Medical Language System, UMLS) (Vydiswaran et al. 2014). To identify medical terms automatically, MacLean and Heer proposed both dictionary-based and statistical learning models in extracting medical terms from *MedHelp*, and their performances were comparable with other toolkits, such as MetaMap (a medical entity extractor, mapping text to concepts in UMLS) and Open Biomedical Annotator (OBA, a dictionary-based semantic expansion tool) (MacLean and Heer 2013). To identify medical entities, Bobo et al. combined both the dictionary-based and statistical models in an iterative framework and extracted two entity types, symptoms and conditions, and drugs and treatments, based on the lexico-syntactic patterns. By adding these medical terms and entities found in the content of OHCs, Vydiswaran et al. extended the CHV and thereby bridged the gap between the patient language (consumer terms) and the expert language (professional terms) (Vydiswaran et al. 2014). These extended CHVs formed the foundations for further medical applications elaborated in the next paragraph.

Recently, the content of OHCs has also been incorporated into medical domain to discover new symptoms and adverse drug reactions (ADRs). The symptom clusters were usually defined as three or more concurrent and related symptoms frequently found in patients and represented by the co-occurrence of keywords (Yang 2014). Therefore, statistical learning models were useful to cluster the focused symptoms based on the occurrence patterns of keywords in the content of medical forums. Yang et al. compared the automatically detected symptom clusters from an online community for breast cancer with the

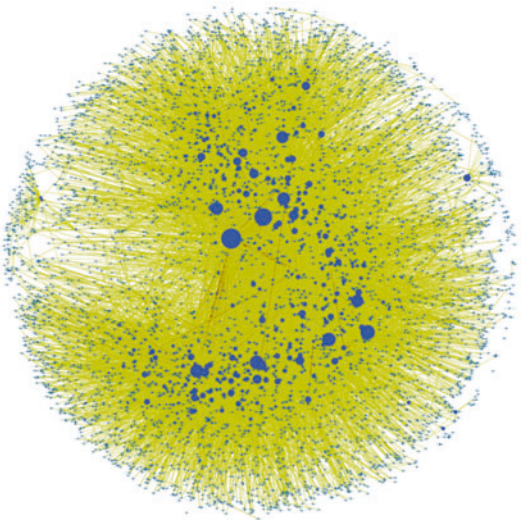
symptom clusters from the clinical study described by breast cancer survivors. They observed the substantial overlap between these two sources in spite of some discrepancies, verifying that OHCs were reliable sources for the discovery of symptom clusters (Yang et al. 2015). To discover ADR, traditional approaches rely on spontaneous and volunteering report. With the development of OHCs, patients now have new channels to talk about ADR. These contents can be used to supplement traditional approaches of drug safety surveillance. Yang, Yang, and Jiang aimed to identify the association between a drug and an ADR implied in the patient-generated text in OHC with the CHV lexicon. They compared the resulting association rules with FDA-alerted ADRs (as the gold standard) and showed that most ADRs detected in OHC could be found in the gold standard, and even more, some unreported ADRs (or side effects) could be observed (Yang et al. 2014).

Although these automatic methods provided new ways to discover new health knowledge from OHC content, there is still not enough evidence to testify and prove the quality of the newly discovered knowledge, and more experiments are still needed before the knowledge can be further utilized in health applications. Besides, these classic text mining methods adopted in the existing research may not perform well in the health-related topics on social media. More research should be conducted to find the characteristics of online health-related topics and develop specified text mining methods for OHC content.

Social Network Analysis

Online health communities rely on computer-mediated communications between participants whose interactions and behaviors can be manifested in a network level. Various types of online communication networks (Sadilek et al. 2012; Su 2014) and social networks (Chomutare et al. 2013) from OHC have been studied. Analyses of these networks help examine the social context and provide a framework for understanding social exchange relationships, social structure, social support, and network effects. However, most of the network models in the recent research were based on one specific type of connections. But in the OHCs, there are various types of connections such as having the same medical symptoms, taking the same medicine, or visiting the same doctor. Future work may focus on developing models of multiple (interdependent) networks and multilayer networks combing two or more types of connections simultaneously.

Social network analysis (SNA) is an important tool to characterize users' interactions and social influences. The social network analysis of a depression-focused OHC *Douban* found that this community possessed both small-world and scale-free properties, with a much higher reciprocity ratio and clustering coefficient value as compared to the networks of other OHCs, social media platforms, and classic network models (Xu and Zhang 2016). The social structure of the group, as was visualized in Fig. 2, was much stickier than those of other social media groups, indicating the tendency of mutual communications and efficient spread of information in the group (Xu and Zhang 2016). To understand the impact of tobacco-related user-generated content on tobacco use and control, Liang et al. analyzed three tobacco-



Research on Online Health Communities: A Systematic Review, Fig. 2 Visualization of the social network of a depression-related community from *Douban*. The size of the node is proportional to the in-degree of the node, and the darkness of the edge represents the edge betweenness centrality value (Xu and Zhang 2016)

related social networks, pro-tobacco, anti-tobacco, and quitting-tobacco networks. By comparing the topological properties of these three networks, they found that Facebook users were exposed to pro-tobacco information. They also found that the user interactions were largely random and different patterns of user interactions were revealed in the tobacco-focused OHCs (Liang et al. 2015). To better understand the mechanism and usefulness of user-generated OHCs in *MedHelp*, friendship networks were constructed and compared with those of site-defined groups. The founders of OHCs possessed the highest centrality measures, indicating that they were the most influential users as well. Members of user-created OHCs tended to have more friends and higher node centralities than those of site-defined groups. Their friends were also more likely to be friends with each other, as suggested by a larger clustering coefficient. In general, members of both user-created OHCs and site-defined groups were more social and more influential (Vydiswaran et al. 2013).

In healthcare, subcommunities may exist through the clustering of peer interactions. These subcommunities may enhance the quality of Web offerings for people with chronic diseases due to their intense interactions and proper information flow to corresponding users (Sadilek et al. 2012; Chomutare et al. 2013; Tamersoy et al. 2015). The comment and interaction behavior can be modeled as directed and weighted networks, and the subcommunity detection algorithms from network science, such as Greedy Optimization and the Affinity Propagation, can be used to deal with large-scale data (Clauset et al. 2004; Frey and Dueck 2007). Chomutare et al. analyzed five diabetes-focused forums in English and Spanish and found that similar users tended to co-occur in the same subcommunities. In addition, the number of years since diagnosis was a significant predictor of their co-occurrence. This finding indicated that the suffering time served as important factor to motivate patients to form subcommunity of similar patients in OHCs. In another study, Carron-Arthur et al. aimed to characterize the users who were associated with the subcommunities in an OHC for mental health. They assessed demographic characteristics, including age, gender, residential location, type of user (consumer or provider), and registration dates. It was found that registration date contributed most to the outcome. Especially, the registration date of the highly engaged and central users, who communicated with many other users, was significantly earlier than the average registration date of other users in the same subcommunity. This finding indicated that the central users played an important role in the formation of subcommunities, as well as in the growth and development of the OHCs (Carron-Arthur et al. 2016). In addition, SNA can also be used to find influential users and the roles associated with users in the social network. This will be elaborated in the following subsection.

Roles of OHC Users

Online users were associated with different roles (e.g., key members, active members, or newcomers) according to their position within a social network and their observed behavior (Rowe et al. 2013). A good understanding of the social positions and participating behaviors of users is helpful for the effective community management and the timely dissemination of intervention information to the members who need it (Morrison et al. 2010). It was also found that the key to succeed for an OHC was the interactions between key members and newcomers, which could generate a friendly social environment of the community and motivate the engagement and contribution of members (Hale et al. 2014).

There were two main methodologies to identify user roles developed in existing research, the interpretative methodology and the structural methodology (Golder and Donath 2004; White et al. 2012). The former methodology rooted in sociological theory and defined social roles of users as a combination of factors constraining their social behaviors. The latter methodology inferred user roles based on the topological properties of the social networks and obtained different roles automatically through unsupervised clustering (Wang et al. 2012; De Choudhury et al. 2013; Brennan et al. 2013; Doran 2015).

With respect to the interpretative methodology, the dynamics of online social support was

examined by modeling the participants' behaviors in providing emotional and informational support. Y. C. Wang, Kraut, and Levine found that (a) self-disclosure was effective in delivering emotional support and (b) answering questions was effective in delivering informational support. Besides, the type and amount of received support were related to the belongingness to OHCs, and more exposure to emotional support was associated with higher probability to stay in (Wang et al. 2015). In a systematic review of participation styles, various metrics were summarized to classify participation patterns according to the 41 identified participation styles, such as influential users and topic-focused responders. But there was still lack of basis for consistent participating patterns and consistent taxonomy of roles across different communities, and therefore some methodological issues existed for comparative research in this area (Carron-Arthur et al. 2015).

With respect to the structural methodology, it commits to reflect the aspects of social forces driving users to embed themselves in the network. Doran summarized the role discovery research in the context of ego networks and proposed a clustering method based on conditional triad census (CRC) (Doran 2015). CRC indicated the dynamics and orientations of three-way relationships in the ego network, representing the circumstances and reasons why a user participates in a social system. Zhao et al. adapted statistical models to identify influential users in OHCs based on the sentiment dynamics in discussion threads (Zhao et al. 2014). The authors proposed a novel metric, the number of influential responding replies (IRR), which revealed a user's ability to affect the sentiment of others. They claimed that the influential users were the ones with highest IRR, which significantly overlapped with the influential users suggested by the community manager (Zhao et al. 2014). In addition, Wang, Zuo, and Zhao employed unsupervised clustering algorithms to classify users into eight roles according to their behaviors of providing/receiving social support embedded in each post (can be divided into five categories). It was found that the types of social support received by a user could facilitate or delay the role transition of users in OHCs (Wang et al. 2015).

Despite the enormous methodologies of detecting user roles in OHCs, there are still many questions that have not been answered yet. For one thing, there are neither consistent definitions nor classifications of user roles, which may confuse and conflict each other and lead to unreliable research results. For another thing, most research only use the users' online behavior such as user-generated content and topologic properties to classify user roles rather than combining users' offline activities. Integration of users' online and offline behavior could help researchers track and evaluate users' health station dynamically and compressively, classify user roles more accurately, and understand the function of each role in depth.

Key Applications

The analysis of OHCs sheds light on the in-depth understanding of how Web users communicate with one another in online health groups. The content analysis helps understand the expression of specific diseases on a large scale, and the results provide important insights for the disease surveillance in public health. The social network analysis and the detection of user roles present novel and efficient information spread patterns of OHCs that can be further adapted by healthcare providers to develop better and more effective functions to facilitate online communications in the design of Health 2.0 applications.

Future Directions

Based on the above review, the research on OHCs is from various perspectives of different research areas. Although existing studies have made significant progress by adopting advanced quantitative models and methodologies to characterize and model different aspects of OHCs, a commonly adopted research framework has been absent across multiple disciplines. In particular, integrating the research methodologies of different disciplines is able to help us obtain a more comprehensive and in-depth understanding of the

user behaviors in OHCs. For example, the content analysis and social network analysis can be integrated into a multivariate statistical model to identify influential users who play an essential role in both creating supportive content and disseminating the content to users of urgent need. Another benefit of integrating the data and methods of different disciplines is that the feasibility and effectiveness of the OHCs can be assessed so that different stakeholders can get comprehensive understanding of their pros and cons. Last but not the least, future research should also concern about the misinformation and malicious marketing of medications. New mechanisms to facilitate meaningful discussions and community growth are critical to the effective use of OHCs to promote the health of the public. The adaptation of new artificial intelligence (AI) methods is also important in utilizing the user-generated content of OHCs for new medical applications, such as personalized medical information retrieval, personalized medication and care, reliable AI-generated decision-making support, etc.

Cross-References

- ▶ Online Communities
- ▶ Online Healthcare Management
- ▶ Social Networks in Healthcare: Case Study
- ▶ User Behavior in Online Social Networks: Influencing Factors

Acknowledgment This work was supported by the National Natural Science Foundation of China (NSFC) Grant Nos. 71402157 and 71672163, the Guangdong Provincial Natural Science Foundation No. 2014A030313753, and the Theme-Based Research Scheme of the Research Grants Council of Hong Kong Grant No. T32–102/14 N.

References

Angst CM, Agarwal R (2009) Adoption of electronic health records in the presence of privacy concerns: the elaboration likelihood model and individual persuasion. MIS Q 33(2):339–370

Bansal G, Zahedi FM, Gefen D (2010) The impact of personal dispositions on information sensitivity, privacy concern and trust in disclosing health information online. Decis Support Syst 49(2):138–150. https://doi.org/10.1016/j.dss.2010.01.010. Elsevier B.V

Blei D, Andrew Y, Jordan M (2003) Latent dirichlet allocation. J Mach Learn Res 3:993–1020

Brennan S, Sadilek A, Kautz H (2013) Towards understanding global spread of disease from everyday interpersonal interactions. In: IJCAI international joint conference on artificial intelligence, vol 1, pp 2783–2789

Cain J (2011) Social media in health care: the case for organizational policy and employee education. Am J Health Syst Pharm 68(11):1036–1040. https://doi.org/10.2146/ajhp100589

Carron-Arthur B, Ali K, Cunningham JA, Griffiths KM (2015) From help-seekers to influential users: a systematic review of participation styles in online health communities. J Med Internet Res 17(12). https://doi.org/10.2196/jmir.4705

Carron-Arthur B, Reynolds J, Bennett K, Bennett A, Cunningham JA, Griffiths KM (2016) Community structure of a mental health internet support group: modularity in user thread participation. JMIR Mental Health 3(2):e20. https://doi.org/10.2196/mental.4961

Chancellor S, Lin Z, Goodman E, Zerwas S, De Choudhury M (2016a) Quantifying and predicting mental illness severity in online pro-eating disorder communities. In: Proceedings of the 19th ACM conference on computer-supported cooperative work and social computing (CSCW 2016). https://doi.org/10.1145/2818048.2819973

Chancellor S, Mitra T, De Choudhury M (2016b) Recovery amid pro-anorexia: analysis of recovery in social media. In: Proceedings of the 2016 CHI conference on human factors in computing systems, pp 2111–2123. https://doi.org/10.1145/2858036.2858246

Chomutare T, Årsand E, Fernandez-Luque L, Lauritzen J, Hartvigsen G (2013) Inferring community structure in healthcare forums: an empirical study. Methods Inf Med 52(2):160–167. https://doi.org/10.3414/ME12-02-0003

Clauset A, Newman MEJ, Moore C (2004) Finding community structure in very large networks. Phys Rev E 70. APS. http://arxiv.org/PS_cache/cond-mat/pdf/0408/0408187.pdf

Coursaris CK, Liu M (2009) An analysis of social support exchanges in online HIV/AIDS self-help groups. Comput Hum Behav 25(4):911–918. https://doi.org/10.1016/j.chb.2009.03.006. Elsevier

De Choudhury M, De S (2014) Mental health discourse on reddit: self-disclosure, social support, and anonymity. In: Proceedings of the eight international AAAI conference on weblogs and social media, pp 71–80

De Choudrhury M, Gamon M, Counts S, Horvitz E (2013) Predicting depression via social media. In: Seventh international AAAI conference on weblogs and social media, vol 2, pp 128–137

Deng H, Han J, Li H, Ji H, Wang H, Yue L (2014) Exploring and inferring user-user pseudo-friendship for sentiment analysis with heterogeneous networks. Stat Anal

Data Mining 7(4):308–321. https://doi.org/10.1002/sam.11223

Dijk V, Matthijs R, Huijgen NA, Willemsen SP, Laven JSE, Steegers EAP, Steegers-Theunissen RPM (2016) Impact of an mHealth platform for pregnancy on nutrition and lifestyle of the reproductive population: a survey. JMIR mHealth and uHealth 4(2):e53. https://doi.org/10.2196/mhealth.5197

Doran D (2015) On the discovery of social roles in large scale social systems. Soc Netw Anal Min 5(1):1–18. https://doi.org/10.1007/s13278-015-0290-0

Eysenbach G, Powell J, Englesakis M, Rizo C, Stern A, and others (2004) Health related virtual communities and electronic support groups: systematic review of the effects of online peer to peer interactions. BMJ: British Medical J 328(7449):1166. https://doi.org/10.1136/bmj.328.7449.1166. BMJ Publishing Group Ltd

Faraj S, Johnson SL (2011) Network exchange patterns in online communities. Organ Sci 22(6):1464–1480. https://doi.org/10.1287/orsc.1100.0600. INFORMS

Fox S (2011) The social life of health information, 2011. Pew Internet & American Life Project, pp 1–33. http://www.who.int/topics/tuberculosis/en/

Fox S, Duggan M (2013) Health Online 2013. http://www.pewinternet.org/2013/01/15/health-online-2013/

Frey BJ, Dueck D (2007) Clustering by passing messages between data points. Science 315(5814):972–976. https://doi.org/10.1126/science.1136800. American Association for the Advancement of Science, New York

Frost J, Vermeulen IE, Beekers N (2014) Anonymity versus privacy: selective information sharing in online cancer communities. J Med Internet Res 16(5):e126. https://doi.org/10.2196/jmir.2684. Edited by Gunther Eysenbach JMIR Publications Inc, Toronto

Golder SA, Donath J (2004) Social roles in electronic communities. Aoir 5:1–25. papers2://publication/uuid/A44A65B0-3B94–4809-A728-DAB3D4674856

Hale TM, Pathipati AS, Zan S, Jethwani K (2014) Representation of health conditions on Facebook: content analysis and evaluation of user engagement. J Med Internet Res 16(8):e182. https://doi.org/10.2196/jmir.3275

Hamm MP, Shulhan J, Williams G, Milne A, Scott SD, Hartling L (2014) A systematic review of the use and effectiveness of social media in child health. BMC Pediatr 14(1):138. https://doi.org/10.1186/1471-2431-14-138

Howes C, Purver M, McCabe R (2014) Linguistic indicators of severity and progress in online text-based therapy for depression. In: Workshop on computational linguistics and clinical psychology: from linguistic signal to clinical reality, no 611733, pp 7–16

Huesch MD (2013) Privacy threats when seeking online health information. JAMA Intern Med 173(19):4–7. https://doi.org/10.1001/jamainternmed.2013.7795

Ji Y, Hong H, Arriaga R, Rozga A, Abowd G, Eisenstein J (2014) Mining themes and interests in the Asperger's and autism community. In: Proceedings of the workshop on computational linguistics and clinical psychology: from linguistic signal to clinical reality, no. JUNE, pp 97–106. http://www.aclweb.org/anthology/W/W14/W14-3212%5Cn, https://www.researchgate.net/publication/272747520_Mining_Themes_and_Interests_in_the_Asperger's_and_Autism_Community_full_paper%5Cn, https://scholar.google.com/citations?view_op=view_citation&hl=en&user=gb8sbdc

Kiciman E, Kumar M, Coppersmith G, Dredze M, De Choudhury M (2016) Discovering shifts to suicidal ideation from mental health content in social media. In: Proceedings of the SIGCHI conference on human factors in computing systems. https://doi.org/10.1145/2858036.2858207

Koufopoulos JT, Conner MT, Gardner PH, Kellar I (2016) A web-based and mobile health social support intervention to promote adherence to inhaled asthma medications: randomized controlled trial. J Med Internet Res 18(6):e122. https://doi.org/10.2196/jmir.4963

Liang Y, Zheng X, Zeng DD, Zhou X, Leischow SJ, Chung W (2015) Characterizing social interaction in tobacco-oriented social networks: an empirical analysis. Sci Rep:10060. https://doi.org/10.1038/srep10060. Nature Publishing Group

MacLean DL, Heer J (2013) Identifying medical terms in patient-authored text: a crowdsourcing-based approach. J Am Med Inform Assoc 20(6):1120–1127. https://doi.org/10.1136/amiajnl-2012-001110

Maclean D, Gupta S, Lembke A, Manning C, Heer J (2015) Forum77: an analysis of an online health forum dedicated to addiction recovery. In: CSCW'15 proceedings of the 18th ACM conference on computer supported cooperative work & social computing, pp 1511–1526. https://doi.org/10.1145/2675133.2675146

Maloney-Krichmar D, Preece J (2005) A multilevel analysis of sociability, usability, and community dynamics in an online health community. ACM Trans Comput-Hum Interact 12(2):201–232. https://doi.org/10.1145/1067860.1067864

McNab C (2009) What social media offers to health professionals and citizens.pdf. Bull World Health Organ. http://www.who.int/bulletin/volumes/87/8/09-066712/en/

Mhurchu N, Cliona RW, McRobbie H, Ball K, Crawford D, Michie J, Jiang Y, Maddison R, Waterlander W, Myers K (2014) Feasibility, acceptability and potential effectiveness of a mobile health (mHealth) weight management programme for New Zealand adults. BMC Obesity 1(1):10. https://doi.org/10.1186/2052-9538-1-10

Moreno MA, Jelenchick LA, Egan KG, Cox E, Young H, Gannon KE, Becker T (2011) Feeling bad on Facebook: depression disclosures by college students on a social networking site. Depress Anxiety 28(6):447–455. https://doi.org/10.1002/da.20805

Morrison D, Mcloughlin I, Hogan A, Hayes C (2010) Evolutionary clustering and analysis of user behaviour in online forums. Artif Intell:519–522

Nabi RL, Prestin A, and So J (2013) Facebook friends with (health) benefits? Exploring social network site use and

perceptions of social support, stress, and well-being. Cyberpsychol Behav Soc Netw 16 (10). Mary Ann Liebert, Inc, 721–727. https://doi.org/10.1089/cyber.2012.0521 140 Huguenot Street, 3rd Floor New Rochelle, NY 10801 USA

Nature (2013) No dishonour in depression. Nature 498 (7453):137–137. https://doi.org/10.1038/498137a

Neal L, Oakley K, Lindgaard G, Kaufman D, Leimeister JM, Selker T (2007) Online health communities. In: CHI'07 extended abstracts on human factors in computing systems, New York. ACM, pp 2129–2132. https://doi.org/10.1145/1240866.1240965

Partridge SR, McGeechan K, Hebden L, Balestracci K, Wong AT, Denney-Wilson E, Harris MF, Phongsavan P, Bauman A, Allman-Farinelli M (2015) Effectiveness of a mHealth lifestyle program with telephone support (TXT2BFiT) to prevent unhealthy weight gain in young adults: randomized controlled trial. JMIR mHealth and uHealth 3(2):e66. https://doi.org/10.2196/mhealth.4530

Rowe M, Fernandez M, Angeletou S, Alani H (2013) Community analysis through semantic rules and role composition derivation. J Web Semantics 18(1):31–47. Elsevier B.V. https://doi.org/10.1016/j.websem.2012.05.002

Rupert DJ, Moultrie RR, Read JG, Amoozegar JB, Bornkessel AS, O'Donoghue AC, Sullivan HW (2014) Perceived healthcare provider reactions to patient and caregiver use of online health communities. Patient Educ Couns 96(3):320–326. https://doi.org/10.1016/j.pec.2014.05.015

Sadilek A, Kautz H, Silenzio V (2012) Modeling spread of disease from social interactions. In: Proceedings of the sixth international AAAI conference on weblogs and social media, pp 322–329

Souza D, Sieckenius C, Preece J (2004) A framework for analyzing and understanding online communities. In: Interacting with computers, vol 16, pp 579–610. https://doi.org/10.1016/j.intcom.2003.12.006

Stinson J, McGrath P, Hodnett E, Feldman B, Duffy C, Huber A, Tucker L et al (2010) Usability testing of an online self-management program for adolescents with juvenile idiopathic arthritis. J Med Internet Res 12(3). https://doi.org/10.2196/jmir.1349

Su WC (2014) Integrating and mining virtual communities across multiple online social networks: concepts, approaches and challenges. In: 2014 4th international conference on digital information and communication technology and its applications, DICTAP 2014, pp 199–204. https://doi.org/10.1109/DICTAP.2014.6821682

Tamersoy A, De Choudhury M, Chau DH (2015) Characterizing smoking and drinking abstinence from social media. In: Proceedings of the 26th ACM conference on hypertext & social media, pp 139–148. https://doi.org/10.1145/2700171.2791247

Toma CL, D'Angelo JD (2014) Tell-tale words: linguistic cues used to infer the expertise of online medical advice. J Lang Soc Psychol 34(1):25–45. https://doi.org/10.1177/0261927X14554484

Välimäki M, Athanasopoulou C, Lahti M, Adams CE (2016) Effectiveness of social media interventions for people with schizophrenia: a systematic review and meta-analysis. J Med Internet Res 18(4):e92. https://doi.org/10.2196/jmir.5385

Vodicka E, Mejilla R, Leveille SG, Ralston JD, Darer JD, Delbanco T, Walker J, Elmore JG (2013) Online access to doctors' notes: patient concerns about privacy. J Med Internet Res 15(9):e208. https://doi.org/10.2196/jmir.2670

Vydiswaran V, Vinod G, Yang L, Zheng K, Hanauer DA, Mei Q (2013) User-created groups in health Forums: what makes them special? In: International AAAI conference on weblogs and social media, pp 515–524. https://doi.org/10.1007/978-3-642-28997-2_19

Vydiswaran V, Vinod G, Mei Q, Hanauer DA, Zheng K (2014) Mining consumer health vocabulary from community-generated text. In: AMIA annual symposium proceedings, pp 1150–1159

Wallace BC (2012) What affects patient (dis) satisfaction? Analyzing online doctor ratings with a joint topic-sentiment model. J Gen Intern Med 18(9):13–19

Wang C, Mao Y, Huberman BA (2012) From user comments to on-line conversations. In: Proceedings of the 18th ACM SIGKDD international conference on knowledge discovery and data mining – KDD'12. ACM Press, New York, p 244. https://doi.org/10.1145/2339530.2339573

Wang YC, Kraut RE, Levine JM (2015a) Eliciting and receiving online support: using computer-aided content analysis to examine the dynamics of online social support. J Med Internet Res 17(4):e99. https://doi.org/10.2196/jmir.3558

Wang X, Zuo Z, Zhao K (2015b) The evolution of user roles in online health communities – a social support perspective. In: Pacific Asia conference on information systems (PACIS 2015)

White A, Chan J, Hayes C, Murphy BT (2012) Mixed membership models for exploring user roles in online fora. In: Proceedings of the 6th annual international conference on weblogs and social media – ICWSM2012, pp 599–602

Xu R, Zhang Q (2016) Understanding online health groups for depression: social network and linguistic perspectives. J Med Internet Res 18(3):e63. https://doi.org/10.2196/jmir.5042

Yang CC (2014). Patient centered healthcare informatics. IEEE Intell Inform Bull 15(1):1–5

Yang CC, Yang H, Jiang L (2014) Postmarketing drug safety surveillance using publicly available health-consumer-contributed content in social media. ACM Trans Manag Inf Syst 5(1):1–21. https://doi.org/10.1145/2576233

Yang CC, Ip E, Avis N, Ping Q, Jiang L (2015) A comparative study of symptom clustering on clinical and social media data. In: Social computing, behavioral-cultural modeling, and prediction. Springer International Publishing, pp 222–231. https://doi.org/10.1007/978-3-319-16268-3_23

Zhang Q, Difranzo D, Hendler JA (2014) Social networking on the world wide web: a review. In: Alhajj R, Rokne J (eds) Encyclopedia of social network analysis and mining. Springer-Verlag, NewYork

Zhao K, Yen J, Greer G, Qiu B, Mitra P, Portier K (2014) Finding influential users of online health communities: a new metric based on sentiment influence. J Am Med Inform Assoc 21:e212–e218. https://doi.org/10.1136/amiajnl-2013-002282

Resource Exchange Networks

▶ Web Communities Versus Physical Communities

Restructuring Social Networks

▶ Transforming Social Networks Data

Retrieval Models

Benno Stein[1], Tim Gollub[1] and Maik Anderka[2]
[1]Bauhaus-Universität Weimar, Weimar, Germany
[2]Bad Arolsen, Germany

Synonyms

Document models; Document representations; Relevance functions

Glossary

Feature	A characteristic property of a document. Usually, a document's terms are used as features, but virtually every measurable document property can be chosen, such as word classes, average sentence lengths, principal components of term-document-occurrence matrices, or term synonyms
Information need	Specifically here: A lack of information or knowledge that can be satisfied by a set of text documents
Query	Specifically here: A small set of terms that expresses a user's information need
Relevance	The extent to which a document is capable to satisfy an information need. Within probabilistic retrieval models, relevance is modeled as a binary random variable

Definition

A retrieval *model* provides a formal means to address (information) retrieval *tasks* with the aid of a computer.

Introduction

A retrieval task is given if an information need is to be satisfied by exploiting an information resource. More specifically, the information need is represented as a term query provided by a user, the information resource is given in the form of a text document collection, and the solution of the retrieval task is a subset of such documents of the collection, which the user considers as relevant with respect to the query. Though a broad range of retrieval tasks can be imagined, including all kinds of multimedia queries and multimedia collections (consider for example "query by humming" or medical image retrieval), the term "retrieval model" is predominantly used in the aforementioned narrow sense. Retrieval models in this sense are based on a linguistic theory and can be considered as heuristics that operationalize the *probability ranking principle* (Robertson 1997): "Given a query q, the ranking of documents according to their probabilities of being relevant to q leads to the optimum retrieval performance." The principle cannot be applied to all kinds of retrieval tasks. In comment ranking, for example, the differential information gain must be considered.

Key Points

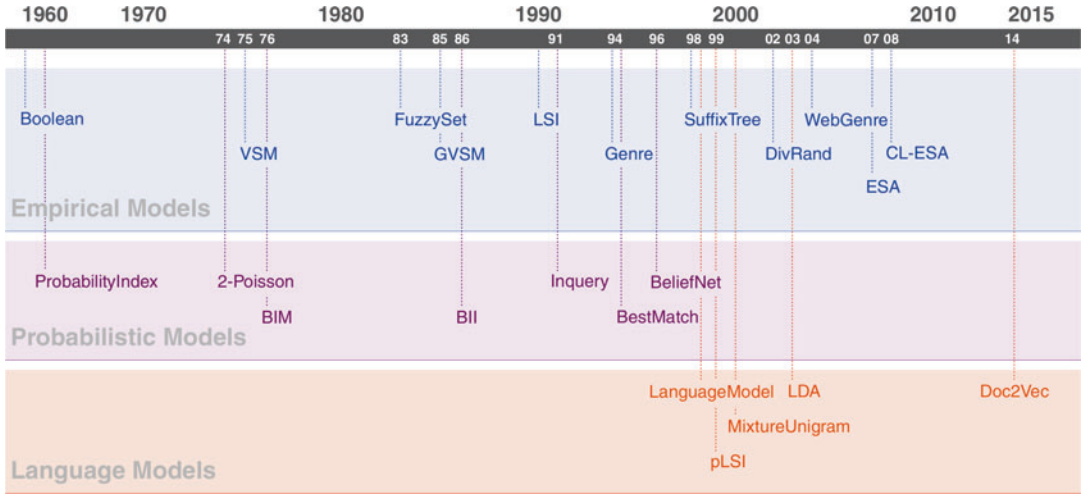
Retrieval models can be classified according to the linguistic theory they are based upon. In the literature a distinction between *empirical models*, *probabilistic models*, and *language models* is often made, which is rooted in the query-oriented understanding of retrieval tasks but which also has historical reasons.

1. Empirical models, sometimes referred to as vector space models, focus on the document representation (Salton and McGill 1983). Both documents and queries are considered as high-dimensional vectors in the Euclidean space, whereas a compatible representation is presumed: a particular document term or query term is always mapped on the same dimension, whereas the term importance is specified by a weight. Usually, the cosine of the angle between two such vectors or simply their dot product is used to quantify their similarity; in particular, the concept of similarity is put on a level with the concept of relevance. Empirical models can be distinguished with regard to the dimensions that are considered (i.e., features that are chosen) and how these dimensions (features) are weighted.
2. Probabilistic models strive for an explicit modeling of the concept of relevance. Statistics comes into play in order to estimate the probability of the event that a document is relevant for a given information need. Most probabilistic models employ conditional probabilities to quantify document relevance given the occurrence of a term.
3. Language models are based on the idea of language generation as it is used in speech recognition systems. A language-based retrieval model is computed specifically for each document in a collection and is usually term-based. Given a query q, document ranking happens according to the generation probability of q under the language model of the respective document.

Historical Background

Figure 1 illustrates the historical development of well-known retrieval models. From each of the three modeling paradigms (empirical models, probabilistic models, language models), selected representatives are in the following characterized along with the respective publications.

The Boolean retrieval model uses binary term weights, and a query is a Boolean expression with terms as operands. Drawbacks of the Boolean



Retrieval Models, Fig. 1 Historical development of retrieval models, organized according to three paradigms: empirical models, probabilistic models, and language models. http://www.uni-weimar.de/medien/webis/research/projects/retrieval-models

model include its simplistic weighting scheme, its restriction to exact matches, and that no document ranking is possible. The vector space model (VSM) and its variants consider documents and queries as embedded in the Euclidean space (see above). Key challenge for these kinds of models is the term weighting. Salton et al. (1975) proposed the *tf* · *idf*-scheme, which combines the term frequency *tf* (the number of term occurrences in a document) with the inverse document frequency *idf* (the inverse of the number of documents that contain this term). The latent semantic indexing (LSI) model was developed to improve query interpretation and semantic-based matching (Deerwester et al. 1990). For example, a document d should match a query even if the user specified synonyms that do not occur in d. The LSI model attempts to achieve such effects by projecting documents and queries into a so-called "semantic space," which is constructed by a singular value decomposition of the term-document-matrix. The explicit semantic analysis (ESA) model was introduced to compute the semantic relatedness of natural language texts (Gabrilovich and Markovitch 2007). The model represents a document d as a high-dimensional vector whose dimensions quantify the pairwise similarities between d and the documents of some reference collection such as Wikipedia. Potthast et al. (2008) demonstrated how the ESA principles are applied to develop an effective cross-language retrieval approach, the so-called CL-ESA model. In contrast to most retrieval models, the suffix tree model represents a document d not as a vector of index terms but as a compressed trie containing all suffixes (i.e., suffixes of all lengths) of a text d. As a consequence, the collocation information of d is preserved, which may render the model superior for particular retrieval tasks (Meyer zu Eißen et al. 2005).

Under the binary independence model (BIM), the documents are ranked by decreasing probability of relevance (Robertson and Sparck-Jones 1976). The model is based on two assumptions which allow for a practical estimation of the required probabilities: documents and queries are represented under a Boolean model, and the terms are modeled as occurring independently of each other. The best match (BM) model computes the relevance of a document to a query based on the frequencies of the query terms appearing in the document and their inverse document frequencies (Robertson and Walker 1994). Three parameters tune the influence of the document length, the document term frequency, and the query term frequency in the model. The best match model belongs to the most effective retrieval models in the Text Retrieval Conference (TREC) series.

The language modeling approach to information retrieval was proposed by Ponte and Croft (1998); the idea is to rank documents by the generation probabilities for a given query (see above). The algorithmic core of the model is a maximum likelihood estimation of the probability of a query term under a document's term distribution. The latent Dirichlet allocation (LDA) model is a sophisticated generative model in the context of probabilistic topic modeling (Blei et al. 2003). Under this model it is assumed that documents are composed as a mixture of latent topics, where each topic is specified as a probability distribution over words. The mixture is generated by sampling from a Dirichlet distribution. More recently, Le and Mikolov (2014) introduced paragraph vector, also known as the Doc2Vec model, which learns continuous distributed vector representations for documents using a neural network classifier.

Relevance Computation

Despite the large variety of retrieval models that have been developed so far, computing the relevance ρ of a document for a query usually boils down to a multiplication of two feature vectors: (1) a feature vector $\mathbf{q}$ representing query q, and (2) a feature vector $\mathbf{d}$ representing document d:

$$\rho(q,d) = \mathbf{q}^T \cdot \mathbf{d} = \sum_{i=1}^{|\mathbf{q}|} \mathbf{q}_i \cdot \mathbf{d}_i$$

What distinguishes retrieval models from each other is the feature set that they employ for representing queries and documents, as well as the computation rule used to calculate the respective feature weights. In the following, the feature

sets and the computation rules for four retrieval models are outlined, starting from the basic *tf*-model to the more sophisticated models *tf* · *idf*, BM25, and ESA.

tf-Model. The *tf*-model (*term frequency* model) is a variant of the vector space model that uses the vocabulary of the given document collection D as feature set. The dimension i of a feature vector is associated with a specific term t_i that occurs in D. As feature weight, the frequency *tf* by which t_i appears in a specific query or document is taken:

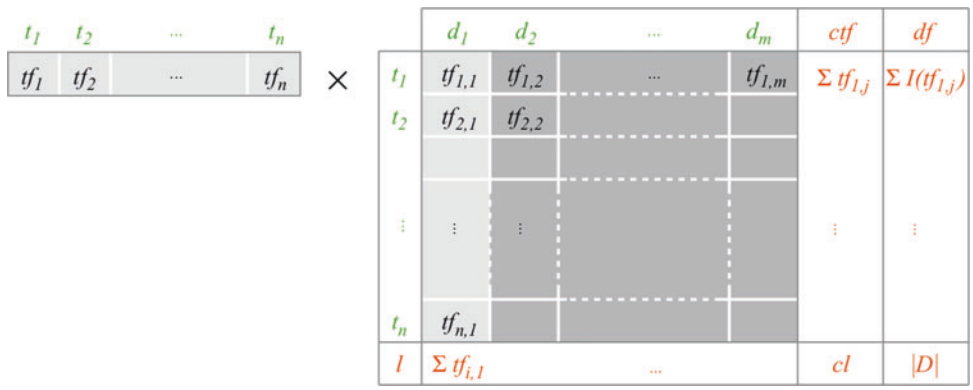
$$\mathbf{q}_i^{tf} = tf(t_i, q)$$
$$\mathbf{d}_i^{tf} = tf(t_i, d)$$

Stacking the *tf* feature vectors of all documents $d \in D$ as columns into a matrix gives the so-called *term-document-matrix*, a data structure from which many retrieval models can be derived. The term-document-matrix (along with additional statistics used later on, shown in red) is depicted in Fig. 2 as shaded area on the right-hand side. On the left-hand side, a canonical query vector is shown. Multiplying the query vector with the term-document-matrix results in a vector containing the relevance scores for all documents in D. A particular property of the *tf* -Model is that the relevance computation for a document d is independent of the collection D from which d is taken. However, the model has certain weaknesses that the following models try to alleviate.

tf · *idf-Model*. An extension of the *tf*-model is the *tf* · *idf*-model, where *idf* stands for *inverse document frequency*. The linguistic intuition behind this extension is that the occurrence of a rare query term in a document is a better indicator for relevance than the occurrence of a frequent term. Considering the query "math for computer science" as an example, the occurrence of the term "for" in a document provides, in comparison to the occurrence of "math" or "computer science," only little evidence about the relevance of a document – a fact which is not exploited in the *tf*-model. A document containing the query term "for" ten times is considered as relevant as a document containing the query term "math" ten times. To address this deficit, the *tf* · *idf*-model incorporates the document frequency *df* of a term into the feature weight computation for documents. The document frequency denotes the number of documents in D that contain a term. In Fig. 2, *df* is illustrated as an additional column of the term-document-matrix. To formalize the computation of *df*, the indicator function $I(\cdot)$ is used, which yields 1 in case $tf(t_i, d_j) > 0$ and 0 otherwise. Since the influence of a term should decrease with its document frequency, an *inverse document frequency* factor is added to the *tf* feature weight computation. Note that several variations for this factor have been proposed. The original formula by Salton et al. (1975) reads as follows:

$$\mathbf{d}_i^{tfidf} = tf(t_i, d) \cdot \log \frac{|D|}{df_i}$$

BM25-Model. The BM25-model is a further advancement of the *tf* · *idf*-model. The



Retrieval Models, Fig. 2 Basic model for relevance computation: the term frequency vector of a query (*left*) is combined with the term frequency vectors of the documents d_j from a document collection D (shaded matrix on the *right*)

development of the model was driven by the observation that the $tf \cdot idf$-model (1) is biased towards long documents, and (2) that it insufficiently favors documents containing all query terms – compared to documents containing only a subset of the query terms. To account for the first observation, the BM25-model introduces a length normalization factor to the feature weight computation. The length of a document is considered as the sum of its term frequencies. In Fig. 2, the document length l is illustrated as an additional row of the term-document-matrix. The sum over all $l \in D$ gives the overall collection length cl. The idea of the length normalization factor is to calculate the average length $\widehat{l}$ of the documents, $\widehat{l} = cl/|D|$, and to penalize documents that are longer than the average, while rewarding shorter documents. To account for the second observation, a term frequency normalization factor is introduced. Given the above example query "math for computer science," the goal of this factor is to consider a document containing both "math" and "computer science" once as more relevant than a document containing one of the terms twice, even if the terms have equal document frequency. The BM25 approach applies a "logarithmic-shaped" function to the term frequency value in order to limit the contribution of a single term to the overall relevance score. The general form of this function is $\frac{tf}{tf+c}$, where c is a constant. The final BM25 formula for the computation of feature weights, which incorporates both normalization factors into a single expression, is the product of extensive empirical evaluation efforts:

$$\mathbf{d}_i^{bm25} = \frac{tf(t_i,d)\cdot(k_1+1)}{tf(t_i,d)+k_1\cdot\left(1-b+b\cdot\frac{l(d)}{\widehat{l}}\right)} \cdot \log\frac{|D|-df_i+0.5}{df_i+0.5}$$

For the two parameters of the function, values of $k_1 = [1.2,2.0]$ and $b = 0.75$ are considered standard choices. The two normalization factors (length normalization and term frequency normalization) are balanced by the parameter b. The last factor of the formula is the BM25 variant of the inverse document frequency.

ESA-Model. The ESA-model represents a class of retrieval models that do not employ terms as features but concepts or topics (hence called "topic models"). Other retrieval models of this kind are LSI and LDA. Topic models aim to further improve the assessment of relevance by taking the semantic relatedness of terms into account. The intuition is that if a document contains terms related to the query terms, like "statistics" or "calculus" which are related to the query term "math," or "programming" and "algorithm" which are related to "computer science," the relevance of this document should be raised. To operationalize this idea, topic models represent queries and documents by a feature vector of topics and provide a means to compute the relevance of a topic for a query or document. In the case of ESA, topics are drawn randomly from the set of Wikipedia articles, and the $tf \cdot idf$-model is employed to represent each drawn article a as a term-based feature vector $\mathbf{a}$. The assumption underlying this approach is that each feature vector $\mathbf{a}$ will contain high $tf \cdot idf$ scores for semantically related terms. To compute the relevance of a topic for a document or query, the cosine similarity between the $tf \cdot idf$ representation of the topic and the $tf \cdot idf$ representation of the query or document is used:

$$\mathbf{q}_i^{esa} = cos\left(\mathbf{q}^{tfidf}, \mathbf{a}_i^{tfidf}\right)$$
$$\mathbf{d}_i^{esa} = cos\left(\mathbf{d}^{tfidf}, \mathbf{a}_i^{tfidf}\right)$$

Key Applications

The key application of retrieval models is to provide keyword-based search capabilities over large collections of natural language text such as digital libraries or the World Wide Web. In many practical settings, the documents of a collection are not completely unstructured but come along with designated meta data such as document titles, abstracts, or markup in the text as in the case of web pages. By taking this additional information into account, e.g., through boosting the relevance

score of documents that contain the query terms in the title, the quality of the search can often be improved significantly over the use of standard retrieval models. In the field of Web search, probably the most prominent approach in this respect is the PageRank score (Page et al. 1999), which exploits the hyperlink graph of the Web for the relevance assessment of web pages. Note, however, that today the PageRank score is only one signal among several hundered other features.

Future Directions

Classical retrieval models provide the formal means of satisfying a user's information need (typically a query) against a large document collection such as the Web. These models can be seen as heuristics that operationalize the probability ranking principle mentioned at the outset. Regarding future directions, a new generation of retrieval models may be capable to support information needs of the following kind: "Given a hypothesis, what is the document that provides the strongest arguments to support or attack the hypothesis?"

Obviously, the implied kind of relevance judgments cannot be made based on the classical retrieval models, as these models do not capture argument structure. In fact, so far the question of how to exploit argument structure for retrieval purposes has hardly been raised, but the research community has picked up this exciting direction (Gurevych et al. 2016).

Cross-References

- Analysis and Mining of Tags, (Micro)Blogs, and Virtual Communities
- Automatic Document Topic Identification
- Data Mining
- Eigenvalues: Singular Value Decomposition
- Knowledge Discovery
- Microtext Processing
- Mining Trends in the Blogosphere
- Social Web Search
- Theory of Probability: Basics and Fundamentals
- Theory of Statistics: Basics and Fundamentals

References

Blei DM, Ng AY, Jordan MI (2003) Latent dirichlet allocation. J Mach Learn Res 3:993–1022

Deerwester S, Dumais S, Landauer T, Furnas G, Harshman R (1990) Indexing by latent semantic analysis. J Am Soc Inf Sci 41(6):391–407

Gabrilovich E, Markovitch S (2007) Computing semantic relatedness using Wikipedia-based explicit semantic analysis. In: IJCAI 2007, proceedings of the 20th international joint conference on artificial intelligence, Hyderabad, 6–12 Jan 2007, pp 1606–1611

Gurevych I, Hovy E, Slonim N, Stein B (2016) Debating technologies (Dagstuhl Seminar 15512). Dagstuhl Reports 5(12):18–46, https://doi.org/10.4230/DagRep.5.12.18. URL http://drops.dagstuhl.de/opus/volltexte/2016/5803

Le QV, Mikolov T (2014) Distributed representations of sentences and documents. In: Proceedings of the 31th international conference on machine learning (ICML2014), Beijing, pp 1188–1196

Meyer zu Eißen S, Stein B, Potthast M (2005) The suffix tree document model revisited. In: Tochtermann K, Maurer H (eds) 5th international conference on knowledge management (IKNOW 05), Know-Center, Graz. Journal of Universal Computer Science, pp 596–603

Page L et al. (1999) The pagerank citation ranking: bringing order to the we

Ponte J, Croft W (1998) A language modeling approach to information retrieval. In: SIGIR'98: proceedings of the 21st annual international ACM SIGIR conference on research and development in information retrieval. ACM Press, New York, pp 275–281. https://doi.org/10.1145/290941.291008

Potthast M, Stein B, Anderka M (2008) A Wikipedia-based multilingual retrieval model. In: Macdonald C, Ounis I, Plachouras V, Ruthven I, White R (eds) Advances in information retrieval. 30th European conference on IR research (ECIR 08). Lecture notes in computer science, vol 4956. Springer, Berlin/Heidelberg/New York, pp 522–530

Robertson S (1997) The probability ranking principle in IR. Morgan Kaufmann Publishers, San Francisco

Robertson S, Sparck-Jones K (1976) Relevance weighting of search terms. J Am Soc Inf Sci 27(3):129–146

Robertson S, Walker S (1994) Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In: SIGIR'94: proceedings of the 17th annual international ACM SIGIR conference on research and development in information retrieval. Springer, New York, pp 232–241

Salton G, McGill M (1983) Introduction to modern information retrieval. McGraw-Hill, New York

Salton G, Wong A, Yang C (1975) A vector space model for automatic indexing. Commun ACM 18(11):613–620

Review Spammer

▶ Identifying Spam in Reviews

Rewarding

▶ Social Interaction Analysis for Team Collaboration

Rich Communication Suite

▶ Social Networking in the Telecom Industry

RIF: The Rule Interchange Format

Michael Kifer
Department of Computer Science, Stony Brook University, Stony Brook, NY, USA

Synonyms

Datalog; Logic programming; Production rules; Rule-based systems

Glossary

Fact	An type of a rule that has no premise
Horn Rule	A type of a rule whose conclusion is a predicate statement. The body of a Horn rule has no negated premises
Production Rule	A type of a rule whose head is an action, which inserts or deletes information
Rule Body	An alternative name for the premise of a rule
Rule Head	An alternative name for the conclusion of a rule
Rule	A statement that has a premise and a conclusion. It states that if the premise is true then so must be the conclusion

Definition

Rule languages and rule-based systems have been playing an important role in the information technology. The applications of rule systems include expert systems, decision-support, deductive databases, and business rules. Most people do not realize that even the ubiquitous database query language SQL is also rule-based. Although the basic idea of a rule is simple: it is just a statement with a premise and a consequent, there is a remarkable variety of dissimilar, incompatible rule-based systems and languages.

With the advent of the Web and the push towards the semantic Web, new opportunities for rule-based applications have emerged. However, to realize these opportunities and make effective use of the Web as a global information system, standards are needed so that the different applications and systems could interoperate.

To help this process along, the World Wide Web Consortium (W3C) created a working group chartered with the task of designing a standardized language for exchanging rules among different and dissimilar rule engines – The Rule Interchange Format or RIF. After RDF (Klyne and Carroll 2004) and OWL (Dean and Schreiber 2004), RIF is the latest installment in a series of semantic Web standards (to which SPARQL was also recently added). It is a suite of documents designed to facilitate rule exchange among systems, especially among Web-enabled rule engines – engines that are aware of such semantic Web standards as IRIs (Duerst and Suignard 2005), RDF (Klyne and Carroll 2004), and can import and process distributed knowledge published as Web documents. Several key components of RIF have become W3C recommendations in June 2010. These documents as well as a number of related specifications are available on

the RIF working group Web site http://www.w3.org/2005/rules/wiki/RIF_Working_Group.

Overview of RIF

RIF is called an "interchange format" to emphasize its purpose – facilitation of inter-operation between the different rule engines, primarily on the Web. Given the diversity of the different uses of the rules in the applications, RIF is not trying to solve the universal interoperation problem, which is believed to be impossible. Instead, it provides a family of languages, called dialects, with rigorously specified syntax and semantics.

The main idea behind rule exchange through RIF is that the different systems will provide syntactic mappings from their native languages to appropriate RIF dialects and back. These mappings are required to be semantics-preserving and thus rule sets and data could be communicated by one system to another provided that the systems can talk through a suitable dialect, which they both support.

The family of RIF dialects was intended to be extensible and uniform. Extensibility here means that it should be possible to add new dialects that various user groups might want to develop. RIF uniformity means that dialects are expected to share much of the syntactic and semantic apparatus.

The current crop of RIF standards is focusing on two kinds of dialects: logic-based dialects and dialects for rules with actions. Generally, logic-based dialects include languages that employ some kind of a logic, such as the first-order logic or non-first-order logics underlying the various logic programming languages (e.g., logic programming under the well-founded or stable semantics (Van Gelder et al. 1991; Gelfond and Lifschitz 1988)). Rules-with-actions dialects include production rule systems, which are typically based on the Rete algorithm (Forgy 1982). Currently, only two logical dialects have reached the status of W3C recommendations: the Core dialect (RIF-Core) and its extension, the Basic Logic Dialect (RIF-BLD) (Boley and Kifer 2010a). The Production Rule Dialect (RIF-PRD) (de Sainte Marie et al. 2010) is currently the only representative of the rules-with-actions group of dialects. Other dialects are expected to be defined by the various user communities.

RIF-BLD is essentially a rule language that is based on Horn rules with a number of syntactic additions such as frames, which are modeled after F-logic (Kifer et al. 1995). RIF-CORE is a minimal logical dialect that is in the intersection of RIF-BLD and the production rules dialect RIF-PRD. This pair of dialects is generally viewed as insufficient for many applications and various extensions based on more advanced semantic theories, such as the well-founded and stable semantics (Van Gelder et al. 1991; Gelfond and Lifschitz 1988)), are expected.

In addition, RIF includes a framework for logic dialects, RIF-FLD (Boley and Kifer 2010b; Kifer 2008, 2010), which provides extensive support and guidelines for the development of future logic RIF dialects. The anticipated extensions of RIF-BLD are expected to be defined in this framework.

Every RIF dialect has a well-defined syntax and semantics. For human consumption and rule-authoring, the presentation syntax is used. For rule exchange among different engines, XML-based syntax is used. The following is an example of an RIF document that contains a rule and a fact.

```
Document(
  Base(<http://example.com/people#>)
  Prefix(cpt <http://example.
  com/concepts#>)
  Prefix(prj <http://example.
  com/projects#>)
  Group (
    Forall?Pers1?Pers2?Proj
      (cpt:coworker(?Pers1?Pers2)
      :-cpt:works(?Pers1?Pers2?Proj)
    )
    cpt:works(<John>  <Mary>  prj:RIF)
  )
)
```

This document contains a rule that says that people working together on a project are coworkers. The statement below the rule is a fact

that says that Mary and John work together on a project called RIF. The example relies on a macrofacility called curi or compact URI (Birbeck and McCarron 2008). A curi such as prj:RIF stands for http://example.com/projects#RIF. That is, prj is expanded into a URL specified in the appropriate Prefix statements in the preamble to the document. The Base statement applies to constants that have no prefix, such as <John>, which are thought of as having the "empty" prefix whose expansion is given in the Base statement. Thus, <John> expands into http://example.com/people#John.

Applications

Applications of RIF are expected to range from question–answering and intelligent planning systems to decision support and business rules. A typical scenarios are consumption and exchange of rules published on the Web in the RIF format. Such rules could be either imported into one's rule language in order to make desired inferences or they can be sent to remote engines for evaluation.

Cross-References

- ▶ Description Logics
- ▶ RDF
- ▶ SPARQL
- ▶ Web Ontology Language (OWL)

References

Birbeck M, McCarron S (2008) CURIE Syntax 1.0: a syntax for expressing compact URIs, W3C working draft. Available at http://www.w3.org/TR/curie/

Boley H, Kifer M (2010a) RIF basic logic dialect. W3C recommendation 3 July, W3C. http://www.w3.org/TR/rif-bld/

Boley H, Kifer M (2010b) RIF framework for logic dialects. W3c recommendation, W3C. http://www.w3.org/TR/rif-fid/

de Sainte Marie C, Paschke A, Hallmark G (2010) RIF Production Rule Dialect. W3c recommendation, W3C. http://www.w3.org/TR/rif-prd/

Dean M, Schreiber G (2004) OWL Web ontology language reference. Recommendation 10 Feb 2004, W3C. http://www.w3.org/TR/owl-ref/

Duerst M, Suignard M (2005) Internationalized resource identifiers (IRIs). http://www.ietf.org/rfc/rfc3987.txt

Forgy C (1982) Rete: a fast algorithm for the many pattern/many object pattern match problem. Artif Intell 19:17–32

Gelfond M, Lifschitz V (1988) The stable model semantics for logic programming. In: Logic programming: proceedings of the fifth conference and symposium. MIT Press, Cambridge, pp 1070–1080

Kifer M (2008) Rule interchange format: the framework. In: Calvanese D, Lausen G (eds) Proceedings of the second international conference (RR 2008) on web reasoning and rule systems, Karlsruhe, 31 Oct–1 Nov 2008. Lecture notes in Computer Science, vol 5341. Springer, Berlin, pp 1–11

Kifer M (2010) Knowledge representation and reasoning with the rule interchange format. In: Domingue J, Fensel D (eds) Handbook of semantic web technologies. Springer, Heidelberg/New York

Kifer M, Lausen G, Wu J (1995) Logical foundations of object-oriented and frame-based languages. J ACM 42:741–843

Klyne G, Carroll JJ (2004) Resource description framework (RDF): concepts and abstract syntax. Recommendation 10 Feb 2004, W3C. http://www.w3.org/TR/rdf-concepts/

Van Gelder A, Ross K, Schlipf J (1991) The well-founded semantics for general logic programs. J ACM 38(3):620–650. httpciteseer.ist.psu.edu/gelder91well founded.html

Risk

- ▶ Crime Prevention, Dataveillance, and the Regulation of Information Communication Technologies
- ▶ Reconnaissance and Social Engineering Risks as Effects of Social Networking

Risks

- ▶ Social Media Policy in the Workplace: User Awareness

Risks Involved in Sensitive Identifiable Personal Information Disclosure in Online Social Networks

▶ Consequences of Publishing Real Personal Information in Online Social Networks

Rival Communities

▶ Competition Within and Between Communities in Social Networks

Role Discovery

Chi Wang[1] and Jiawei Han[2]
[1]Microsoft Research, Redmond, WA, USA
[2]Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, IL, USA

Synonyms

Actors, Nodes; Latent relationship, Hidden role; Relationship mining; Social status, Social location, Position; Structural measure, Link mining metric

Glossary

Position	The location of an actor or class of actors in a system of social relationships
Expectation	An evaluative standard applied to an incumbent of a position, such as rights, duties, norms, and behaviors that a person has to face and to fulfill
Role	A set of expectations that are coupled to the positions
Positional Sector	An element of the relational specification of a position, specified by the relationship of a focal position to a single counterposition
Role Theory	A perspective in sociology or social psychology that considers most of everyday activities to be the acting out of socially defined categories (e.g., mother, manager, teacher)
Role Discovery	Extracting implicit knowledge about roles from behavior data in a social network
Network	A graph that assigns some semantics to the nodes and some kind of interaction to the links
Ego Network	The subgraph that represents all of the direct relationships between a selected entity (the ego) and others (the alters)

Definition

One of the useful applications in social networks is to discover the role of an individual or a set of entities. It is a data mining problem of uncovering the hidden social roles from a network that link people with semantic interactions. Computer scientists have studied the empirical role discovery problems in scattered context, without any significant attempt on defining or delimiting the concept theoretically or hypothetically. We refer to the literature of sociology (Gross et al. 1966; Lorrain and White 1971; Everett 1985; Coulson 2010).

A simple definition of social role is a set of rights, duties, expectations, norms, and behavior a person has to face and to fulfill. Substantial debate exists in the field over the meaning of the "role" in role theory. A role can be defined as a social position, behavior associated with a social position, or a typical behavior. Some theorists have put forward the idea that roles are essentially expectations about how an individual ought to behave in a given situation, while others consider it as means on how individuals actually behave in a given social position. Others have suggested that a role is a characteristic behavior or expected

behavior, a part to be played, or a script for social conduct. Most authors consider the ideas of social location, expectations, and behavior. For instance, Gross et al. (1966) defined a role to be a set of expectations or, in terms of their definition of expectations, a set of evaluative standards applied to an incumbent of a particular position. This definition is dependent on the definition of position as a location of an actor or class of actors in a system of social relationships. Position has two aspects in specification, namely, relational and situational. In a position-centric model, the focal position is specified by its relationship to one or multiple counterpositions. For example, a school superintendent has relationships with many other positions such as school board member, principal, and teacher. Situational specification describes the scope of the social system in which the position is studied. For example, the superintendent position can be studied in a specific community, in the state of Illinois, or in the United States. It is necessary to specify at which level one intends to work.

With these backgrounds, we characterize the role discovery problem in social networks, in a data mining point of view, as to answer the following two typical questions from investigators:

1. What is the role of X?
2. Who has the role R?

The first type of question asks the role of one or multiple actors; X can be an individual or a set of actors. An investigator can specify several predefined roles and ask for the role labels of X, such as to label one's Facebook friends as relatives, schoolmates, colleagues, or other friends. One can also ask for finding unknown roles based on positions with relational and situational specifications, such as finding the different organizational roles of every worker in a company, with the knowledge of the existence of a social hierarchy specified by the manager-subordinate relationship. Note that when the role is not predefined, the expected behavior of each role needs to be discovered as well as the actors who have these roles.

The second type of question asks which actors have a certain role. The answers are often given in a comparative sense – some actors are more likely to have a role than others – except when the evaluative standards are readily defined and computationally easy to measure. Examples of this kind of question include role discovery at different levels, such as finding advisors of researchers, parents of people, finding leaders of a community, finding initiators of discussion topics, and finding influencers of information diffusion in the whole network.

Introduction

The concept of role is pivotal in the analysis of the structures and functions of social systems and explanation of individual behavior. Unfortunately, the knowledge of people's roles is often hidden in the network data. The main goal of role discovery is to recover such hidden knowledge to facilitate social network analysis.

Researchers use different specification of roles in the existent work on role discovery and do not have a unified theoretical framework for it. However, we find that most of the work can fit in our definition. It is beneficial to categorize the work first according to the nature of the task they can perform.

Type 1: What is the Role of X? We consider multiple possible roles, either predefined or automatically discovered. The goal is to determine the role label of a target object X, which can be one actor or a set of actors. The label of X can be one or multiple roles. Representative studies are mining the roles of email communicators (Freeman 1997; Leuski 2004; McCallum et al. 2005; Rowe et al. 2007), inferring social hierarchy in an organization (Memon et al. 2008; Maiya and Berger-Wolf 2009), discovering overlapping roles in synthetic data (Wolfe and Jensen 2004), and mining different types of relationships in online social network (Leskovec et al. 2010; Tang et al. 2011; Wang et al. 2012).

Type 2: Who has the Role R? It focuses on finding actors with a certain role. The expectation of the role is however undefined or hard to evaluate. Representative work includes ranking hubs and authorities (Page et al. 1999; Kleinberg 1999),

mining influential people in viral marketing (Kempe et al. 2003; Chen et al. 2010a), exploring community-based roles and its application in influence maximization (Scripps et al. 2007; Hopcroft et al. 2011), inferring friendship (Eagle et al. 2009; Crandall et al. 2010), mining advising roles in author network (Wang et al. 2010), and detecting topic initiators on the Web (Jin et al. 2010).

Not all of these studies state their problem as "role discovery." In fact, most of them do not. But we include them because they can be regarded as solving role discovery problems in specific contexts and different emphases. Some of them explicitly use the concept role, such as Leuski (2004), Wolfe and Jensen (2004), and McCallum et al. (2005). Some do not use the term role but explicitly specify the role name to detect such as "influencers" (Kempe et al. 2003), "initiators" (Jin et al. 2010), and "key players" (Shaparenko et al. 2005). Some focus on the social positions (Freeman 1997; Rowe et al. 2007; Memon et al. 2008; Maiya and Berger-Wolf 2009; Wang et al. 2010), which is a fundamental concept of social roles. Some have an emphasis on the relationship, which is a necessary concept of defining social positions, such as Diehl et al. (2007), Eagle et al. (2009), Crandall et al. (2010), Leskovec et al. (2010), and Tang et al. (2011).

Most of the studies use heuristically defined role concept, either explicitly or implicitly. Jin et al. (2011) theoretically studied the axiomatic role similarity measure based on theory from sociology (Everett 1985).

Key Points

The role discovery problem asks two types of questions which have slightly different emphasis: what is the role of X and who has the role R? The first emphasizes on differentiating the roles of different nodes, and the second aims to find the similar nodes that share certain roles. Both pay major attention to the accuracy and quality of the results, whereas the challenge for answering the second question is sometimes the efficiency.

Historical Background

The concept of role has assumed a key position in the fields of sociology, social psychology, and cultural anthropology. Yet, despite its frequent use and presumed heuristic utility since the 1920s and 1930s, the conceptualization of role is lack of progress until the 1950s. The debate has been on whether role is a redundant concept in sociology since then. In Coulson (2010), the criticisms are summarized. Even without a consensus of the exact definition of social roles, the operational problem of role analysis has attracted a lot of research interests. Recently, with the emergence of large-scale social network data and analysis, computer scientists can contribute to the empirical role discovery problem with new computing techniques. We have seen such kind of practice in the past decade. However, when different computer scientists tackled with the seemingly distinct tasks in their own context, the same term role is used in an even more arbitrary way than sociologists. No one has explicitly defined the ubiquitous role discovery problem in a generic social network nor connected to the role theory in sociology research community except the very recent work by Jin et al. (2011) and Henderson et al. (2012). We attempt to delimit the concept of role with resort to sociology study and define the role discovery problem toward the interest and view of computer scientists. Our definition of role discovery problem has a clear task-driven flavor and captures the scattered practice by computer scientists. More importantly, we can categorize the existent work within our definition and identify a broad range of tasks awaiting study.

The role discovery problem we defined here should be distinguished from the role mining problem in the database and security community and the semantic role labeling problem in the natural language processing community.

Solutions and Methodology

While role discovery can be related to various traditional data mining tasks and techniques such as clustering, classification, and ranking, there is

no generic solution to the role discovery problem we defined in this essay. We describe the diverse attempt to solving part of this problem, according to the nature of the methodology: (1) link or content analysis via probabilistic modeling, (2) social network analysis with structural measurement, (3) combinatorial optimization, and (4) machine learning to classify or rank.

Link or Content Analysis via Probabilistic Modeling

Probabilistic modeling covers a vast body of approaches to role discovery. Various models based on different stochastic assumptions are proposed for link or content analysis in social networks. We describe one of them in detail and review others briefly.

A stochastic blockmodel is a model for analyzing link data characterized by block structure. It can be used to answer the type 1 question by partitioning nodes of the network into subgroups called blocks (roles). The basic assumption is that the distribution of the ties between nodes is dependent on the blocks to which the nodes belong (Holland et al. 1983).

Formally, let $\langle B_1, \ldots, B_t \rangle$ be a partition of the nodes into mutually exclusive and exhaustive subsets called node blocks. Nodes in the same node block are stochastically equivalent in the following sense. Consider a block B_r and any node j in the network. The likelihood of any given pattern of ties with node j is the same for all nodes in the block B_r. In other words, if i and i' are two nodes (excluding j) belonging to node block B_r, any probability statement about the links in the network can be modified by exchanging the links between i and j and the links between i′ and j, without changing its probability. Stochastic equivalence is a generalization of the algebraic notion of structural equivalence (Lorrain and White 1971).

As an example, we show how to model a network with n nodes and a single relation R between them. For any two nodes i and j in the network,

$$X_{ij} = \begin{cases} 1 & \text{if iRj,} \\ 0 & \text{otherwise.} \end{cases}$$

We assume the matrix $X = (x_{i,j})_{n \times n}$ is generated by the following steps:

- Step 1. Decide the number of blocks, t. In the simplest case, we can assume the number of blocks is known. In the actual analysis, the value of t can be varied to see how it changes the results and selected with certain model selection criteria (e.g., Bayesian information criteria).
- Step 2. Sample the block-size distribution vector

$$\lambda = (\lambda_1, \ldots, \lambda_t)$$

where $\lambda_i > 0$, $\Sigma_i \lambda_{i\cdot} = 1$. The *a priori* probability that a node is in block j is λ_i. λ can be sampled from a Dirichlet distribution or determined according to expected behavior of the model. For example, for a four-block model where we wish to have approximately the same number of nodes in each block, we let $t = 4$ and $\lambda_i = 1/4$; $i = 1, 2, 3, 4$. Such assumptions do not force the node blocks to be equally large but do keep their size similar.

Step 3. Sample the block membership indicator matrix $C = (c_{ij})_{n \times t}$, i.e., $c_{ij} = \begin{cases} 1 & \text{if } i \in B_j \\ 0 & \text{otherwise} \end{cases}$ with $P(c_{ij} = 1 \mid \lambda, t) = \lambda_j$. Here we assume the a priori membership of every node is independent and identically distributed given λ and t.

Step 4. Sample the density parameters $\pi = (\pi_{rs})_{t \times t}$ for every pair of blocks B_r and B_s. Based on the stochastic equivalence assumption, we have $Pr_{rs}(x_{ij} = 1) = \pi_{rs}$ for any two nodes $i \in B_r$ and $j \in B_s \cdot \pi_{rs}$ can be generated from a beta prior $B(\alpha, \beta)$:

$$P(\pi_{rs}) = K \pi_{rs}^{\alpha - 1} (1 - \pi_{rs})^{\beta - 1},$$

where K is a normalizing constant. When β is large and α is small, the prior is concentrated on low values of π. This can be used to model the sparse interaction of a pair of blocks. When

both α and β are small, the prior is "flat" (i.e., more approaching to uniform distribution) to model the unspecified pattern.

Step 5. Sample X with *a priori* blocks and pairwise block patterns,

$$P(x_{ij} = 1 | C, \pi) = \prod_{1 \le r, s \le t} \pi_{rs}^{c_{ir} c_{js}}$$

Given a matrix X generated from this Bayesian model, we can use the Bayes' theorem to compute the posterior probability distribution of the block membership of each node:

$$P(c_{ij} = 1 | X), i = 1, \ldots, n, j = 1, \ldots, t.$$

This n × t array of values gives the probability that each node belongs to each block. If the data fit the blockmodel, these probabilities should be near 0 or 1. If the posterior probabilities are not near 0 or 1, the blockmodel is not informative, and one cannot tell which nodes go into which blocks. If the blockmodel cannot explain the data, the posterior probabilities will be near their *a priori* values (i.e., λ_1 through λ_t).

The parameters of the model can be estimated with the maximum likelihood estimation (MLE) principle or maximum a posteriori (MAP) principle. The difference of them is MLE only fits the data, while MAP fits both data and prior on the parameters. When the amount of data is small, MAP avoids overfitting. For example, if we want to estimate the pairwise density parameter π, the MAP estimation has the form

$$P(\pi | X) = \frac{P(\pi) P(X | \pi)}{P(X)} \propto P(\pi) \sum_{C} P(X | C, \pi) P(C | \lambda, t)$$

Since there are hidden variables C, the log likelihood is not easy to maximize. The MAP can be solved by expectation-maximization (EM) algorithm approximately.

This simple stochastic blockmodel can be generalized in many ways. First, it can be used to analyze multiple relations instead of a single relation, and the paired measurements of each relation can be nonbinary and multivariate. To achieve this, we need to sample more than one relation matrix X, from different distributions rather than Bernoulli. Second, reciprocity can be modeled for asymmetric relations by sampling x_{ij} and x_{ji} together rather than separately and encode the reciprocity in the joint probability of them. Third, each node can have multiple roles, and each observed relation can be associated with multiple blocks rather than a single block. This can be modeled by assigning each node a subset of role labels (Wolfe and Jensen 2004), or a mixed membership of these blocks (Airoldi et al. 2008). Fourth, the number of roles can be learned automatically instead of being predefined or selected by model selection criteria. For example, we can assume t is infinite, and the role membership vectors are generated from a hierarchical Dirichlet process in the nonparametric setting.

We have discussed the stochastic blockmodel in detail. Now we briefly review other probabilistic models:

- Scalable "social" Bayesian network (Goldenberg 2007): This model considered person-event relation instead of direct interaction between people. A person's participation in any event is indicated by a binary variable. For example, if two people coauthored a paper, the participation indicator variables for both of them in this "coauthoring paper" event take 1. It is assumed the roles of people decide the dependency structure of these variables. Such a representation of data casts the problem of learning interaction networks in terms of structural learning of probabilistic graphical models for binary variables. The goal is to learn the underlying dependencies that trigger events. In other words, based on known information about simultaneous participation of people in observed events, we can construct a probabilistic generative model that would describe those events. The advantage of this representation is that the sparseness of social network interaction is helpful to efficiently learn the model. The problem is that the roles are indirectly modeled and it requires further

interpretation of the structure of the learned Bayesian network. This model can be used to answer the type 1 question "what is the role of X?"

- Proximity-based interaction model: Maiya and Berger-Wolf (2009) proposed a probabilistic generative model for weighted social networks governed by a latent social hierarchy among individuals. The goal is to discover the hierarchy from observed interaction data. The hierarchy can be regarded as a special social organization in which each node can find its position. The fundamental hypothesis of the interaction generation from the hierarchy is that for each pair of nodes, there is a probability of interaction, and the highest rates of interaction are assigned between parents and their children or between siblings. The interaction probabilities for all other pairs of nodes either decay with tree distance or remain constant. Based on this hypothesis, the authors defined four specific instantiations of the model. The weight of interaction, which can be explained as the frequency of social interactions, is assumed to be generated from a Bernoulli distribution according to some model of them. Given a weighted social network, they inferred both the hierarchy and the model from which it is generated with the maximum likelihood principle. This model can be used to answer the type 1 question "what is the role of X?"
- Author-recipient-topic (ART) model: McCallum et al. (2005) argued that network properties are not enough to discover all the roles in a social network. As they argued, the email messages in a corporate network can have no obvious traffic patterns, and the role of manager becomes obvious only when one accounts for the language content of the email messages. They proposed a directed graphical model of words in a message generated given their author and a set of recipients. Compared to related topic model latent Dirichlet allocation (LDA) and author-topic (AT) model, ART considers both the author and recipients of a message. Each author-recipient pair has a multinomial topic distribution. For each word in a message, a recipient is chosen uniformly from the observed set of recipients, a topic is chosen from the topic distribution of this author-recipient pair, and then the word is generated from that topic, which is a multinomial distribution over the vocabulary. From the observed email messages, we can infer the topic distribution of each author-recipient pair, as well as the marginal distribution conditioned on an author or a recipient. The topic distribution can then be used to find out the relations between senders and receivers, as well as the roles of each person. This model can be used to answer the type 1 question "what is the role of X?"
- Time-dependent probabilistic factor graph (TPFG): Wang et al. (2010) studied the case when the role of a person can change with time. Specifically, it designed an undirected probabilistic graphical model to predict the role of a researcher based on inferred advisor-advisee relationship from the publication network composed of authors and papers. The number of coauthored publications, as well as the publications of one's own in each year, is used to compute a rough estimate of the likelihood of the advising relationship between two authors and the advising duration. The role of a researcher has an important temporal constraint: one cannot become an advisor before he graduated. This constraint is utilized in TPFG via a factor function which connects the hidden variables representing each author's advisor, and they are jointly predicted throughout the network. The marginal probability of each variable is used to rank the potential advisors for each researcher. This model can be used to answer the type 2 question "who has the role R?"
- Constructing a topical hierarchy in heterogeneous information network (CATHYHIN): Wang et al. (2014) proposed a generative model for text-attached heterogeneous information networks, which link text with multiple types of entities. This model can be used to model general documentary data collections such as scientific publications, enterprise reports, news articles, and social media and construct hierarchies that represent topics at multiple granularities. Each topic is

represented by ranked lists of phrases and entities of each type. The mined hierarchies can be used to answer the two types of role discovery questions in multiple granularity. For example, Danilevsky et al. (2013) present probabilistic methods for answering both types of questions given a heterogeneous topical hierarchy.

- Community role model (CRM): Han and Tang (2015) proposed a generative model that jointly discovers communities and roles. It considers two kinds of latent information, community and role of each node, and three kinds of observed information, links, node attributes, and actions. It assumes that the links of a network are generated from latent community membership, and the attributes are generated from latent roles. Finally, it assumes node actions are generated from both the community membership and the role of each node. This model was used for structure discovery, behavior prediction, and community detection. In principle, it can also be used to analyze the roles, though it is not done explicitly in the work.
- Mixed membership community and role (MMCR) model: Chen et al. (2016) also jointly discovered communities and roles. It has two differences from Han and Tang (2015). First, it models the generation of links only, not the attributes and actions. Second, it considers community-specific roles, while the previous work considers community-independent roles. This model can be used to answer the type 1 question "what is the role of X?"

Social Network Analysis with Structural Measurement

In social network analysis, people have studied numerous structural metrics. With these metrics, social positions and roles can be inferred from the link behavior without referring to content information. In this subsection, we introduce several pieces of work that are most relevant to our role discovery problem:

- Hierarchy-based role: Rowe et al. (2007) proposed a social network analysis algorithm to extract social hierarchy from email flows within an organization. They used Enron email dataset which showcased the internal working of a real corporation over a period between 1998 and 2002. Without taking into account the actual contents of the email messages, they automatically rank the major officers, group similarly ranked and connected users in order to accurately reproduce the organizational structure, and understand relationship strengths between specific sets of users. This can be regarded as a role discovery problem where the role is determined by the social position in a hierarchy.

The algorithm works in two stages. In the first stage, each email user is profiled by two sets of statistics pertaining to the flow of information, both volumetric and temporal. With these individual features, users can be measured against one another for the purpose of ranking and grouping. In the second stage, users are ranked by analyzing cliques and other graph theoretical qualities of the email network. All the statistics are normalized and combined to calculate an overall social score with which the users are ultimately ranked.

Freeman (1997) also studied the hierarchy in an organization with a procedure called canonical analysis of asymmetry. We omit the details here.

Different with above work, Memon et al. (2008) studied the hidden hierarchy detection in terrorist networks which are undirected. The links are not observed communication records but collected from co-mentioning in news articles. They defined the problem as a process of comparing different centrality values of different nodes to identify which node is more powerful, influential, or worthy to neutralize than others. Their analysis was based on centrality measure. Different centrality measures can be used for finding the hierarchical view of a network, each associated with a particular meaning. A review of key centrality concepts can be found in Freeman et al. (1988):

- Community-based role: Roles can be studied with the knowledge of community and community attachment. Community-based roles

provide useful information to analysts in areas such as antiterrorism and law enforcement. In searching for potential terrorist threats, for example, analysts may find it useful to identify suspects with certain roles (mastermind, financier, facilitators, military commander, etc.). They have different interaction characteristics with communities.

Scripps et al. (2007) defined the community-based role of a node according to the number of communities (community score) and links incident to it (degree). Ambassadors provide connections to many different communities. Big fish, named after the clich "big fish in a small pond," are very important only within a community. They have a high degree but a relatively small community score. Contrasting with them are people with a low degree but a high community score. These are called bridges because they serve as bridges between a number of communities. Finally, loners have a low relative degree and low community score. When the community membership is not available, it needs to be inferred from the network in order to define the community-based role. Moreover, overlapped membership must be considered as a natural requirement to measure the number of communities linked to each node. When the notion of community is well aligned with the network topology, the authors show that the approximations become quite reliable.

Combinatorial Optimization

In the previous sections, we have considered the role for an individual node. Now, we consider mining the roles of a group of nodes. Combinatorial optimization techniques are used to find a group of nodes that can holistically play some important role in the whole network:

- Influence maximization: Influence maximization, defined by Kempe et al. (2003), is the problem of finding a small set of seed nodes in a social network that maximizes the spread of influence under certain influence cascade models. It is the abstraction of word of mouth or viral marketing: a small company has a limited budget but wants to select a small number of initial users in the network to market and wishes they would influence their friends' friends and so on and thus affect a large population in the social network to adopt the application. The problem is whom to select as the initial users so that they eventually influence the largest number of people in the network. These users play the role of seeds of information propagation.

  Influence maximization is formulated as a discrete optimization problem. Influence is assumed to be propagated according to a stochastic cascade model, and the goal is to find k nodes such that under the influence cascade model, the expected number of nodes activated by the k seeds (referred to as influence spread) is the largest possible. Kempe et al. proved that the optimization problem is NP-hard, and Chen et al. (2010a) proved that computing the influence spread given a seed set is #P-hard. Kempe et al. (2003) proposed a greedy approximation algorithm guaranteeing the influence spread is within $1–1/e - \epsilon$ of the optimal influence spread, where e is the base of natural logarithm and ϵ depends on the accuracy of their Monte Carlo estimate of the influence spread given a seed set. This algorithm is not scalable to large networks, and several studies have been devoted to address it. Chen et al. (2010a) presented a state-of-the-art scalable algorithm under the independent cascade model and Chen et al. (2010b) under the linear threshold model. However, no algorithm has been able to beat Kempe et al.'s greedy algorithm in terms of the optimality of the solution.
- Community kernel: Hopcroft et al. (2011) defined and explored community kernel detection problem. It is similar to Scripps et al. (2007) in the sense of studying the dependence of community membership and role. The difference is that Hopcroft et al. (2011) distinguished two roles of users in each community. One role is named community kernels, such as influential users on Twitter. The other role is auxiliary communities, such as the followers of the community kernels on Twitter. The authors

gave the formal definition of the two types of roles and cast the problem of finding community kernels as combinatorial optimization. They presented two algorithms. First algorithm GREEDY is based on maximum cardinality search, which is efficient but does not have a bounded error. The second one is based on relaxation of the combinatorial optimization to continuous optimization. And then a near-optimal iterative algorithm weight-balanced algorithm (WeBA) was proposed. WeBA also has a nice property: easy for parallelization.

- Co-profiling attributes and relationships: Li et al. (2014) defined and explored co-profiling problem in an ego network. It is based on the insight that social connections are discriminatively correlated with attributes via hidden relationship type. For example, a user's colleagues are more likely to share the same employer with him than other friends. To achieve co-profiling of attributes and relationship types, a combinatorial optimization framework is developed. Two kinds of dependencies are formulated in the objective function: attribute-circle dependency and circle-connection dependency. The former suggests that friends in a circle formed by a common relationship type share a corresponding attribute value with the ego user, and the latter suggests that friends in a circle do not have many connections to the friends in other circles. The objective is to minimize violation to these two dependencies. An iterative algorithm is presented to solve the problem. It is guaranteed to converge, and every iteration is linear to the number of the nodes in an ego network, which is typically small in hundreds.

Supervised Learning of Roles

In the previous sections, most of the approaches are unsupervised. Now we consider supervised methods with machine learning techniques. Learning to classify and learning to rank are deployed, respectively, to answer type 1 and type 2 questions:

- Learning to classify: Leuski (2004) studied the roles from email content collected in a research group. He tagged the messages with eight speech acts including plan, request advice, request meeting, and so on. With the standard tf-idf feature, a single support vector machine (SVM) classifier for each speech act is trained. Each message is predicted to have one or more speech acts. Then, the roles of professor, graduate student, secretary, researcher, and programmer are distinguished through the speech act patterns in their incoming and outgoing emails.

Wang et al. (2012) presented a model to learn the roles when nodes can be positioned in a hidden hierarchy. As an example of application, they recovered family trees of Kennedy's and Roosevelt's. The model was similar to the unsupervised model in Wang et al. (2010), but allowed to learn feature weights from observed labels. With that model, local features of object attributes, interaction patterns, and rules and constraints for knowledge propagation can be used to infer the hierarchical relationships. They summarized the heterogeneous features in eight different categories:

- Learning to rank: Diehl et al. (2007) proposed a different model for relationship identification. The focus is to find manager-subordinate relationship using the Enron email corpus. Therefore, it falls into the category of type 2 role discovery. It performs the ranking for all the candidate relationships in each ego network. The goal of the learning is to find a scoring function for the relationships that minimizes the number of rank violations committed by the scoring function. For every possible pairing of relevant and irrelevant relationships in an ego network, we desire a scoring function that scores the relevant relationships higher than the irrelevant relationships. A large-margin approach is used to learn the scoring function. Two kinds of features are used in this work to obtain two different rankers. One is traffic statistics and the other is message content. It is found that content-based ranking outperforms traffic-based ranking overall, but for some ego networks, content-based ranking performs

worse. It is suggested that this problem is caused by more complex relationships (e.g., one performed similar tasks for different individuals as performed for the direct manager).

Key Applications

The rich knowledge of roles can provide interesting semantics of a social network and is used to improve organizational role and structure or utilized for economy, security, or education purposes.

For one example, discovery of roles in customer social network helps with viral marketing. Word of mouth or viral marketing differentiates itself from other marketing strategies because it is based on trust among one's close social circle of families, friends, and coworkers. People trust the information obtained from their close social circle far more than the information obtained from general advertisement channels such as TV, newspaper, and online advertisements. Role discovery helps to answer the question whom to select as the initial users so that they eventually influence the largest number of people in the network.

For another example, discovery in roles in adversary social network helps with counterterrorism. It is critical for successful identification who are the commanders, communicators, etc. in the terrorist network where the roles are mostly hidden.

For the third example, finding roles among researchers helps the community detection in academia. Within each community, there are researchers of different roles such as cliquey, bridge, and periphery. Also, communities can be discovered more accurately when the existence of advisor, advisee, coworker, external collaborator, etc. is taken into consideration.

Future Directions

First, there is no generic solution to the role discovery problem defined in this essay. There is a great opportunity for one to develop a systematic solution with some of the described methodology as basis. Second, theoretical study of notions in role theory is another promising direction. For example, role similarity is studied recently by Jin et al. (2011) with axiomatic principles. Third, role discovery in heterogeneous network has been largely unexplored. There is considerable room for advancing the technology to tackle that challenge.

Cross-References

- ▶ Centrality Measures
- ▶ Community Detection: Current and Future Research Trends
- ▶ Counterterrorism, Social Network Analysis in
- ▶ Human Behavior and Social Networks
- ▶ Patterns in Productive Online Networks: Roles, Interactions, and Communication
- ▶ Probabilistic Graphical Models
- ▶ Role Identification of Social Networkers
- ▶ Structural Holes

References

Airoldi E, Blei D, Fienberg S, Xing E (2008) Mixed membership stochastic blockmodels. J Mach Learn Res 9:1981–2014

Chen T, Tang L, Sun Y, Chen Z, Chen H, Jiang G (2016) Integrating community and role detection in information networks. In: SDM, Miami

Chen W, Wang C, Wang Y (2010a) Scalable influence maximization for prevalent viral marketing in large-scale social networks. In: SIGKDD, Washington

Chen W, Yuan Y, Zhang L (2010b) Scalable influence maximization in social networks under the linear threshold model. In: ICDM, Berlin

Coulson M (2010) Role: a redundant concept in sociology? Some educational considerations. In: Jackson JA (ed) Role: sociological studies, vol 4. Cambridge University Press, Cambridge

Crandall D, Backstrom L, Cosley D, Suri S, Huttenlocher D, Kleinberg J (2010) Inferring social ties from geographic coincidences. PNAS 107:22436–22441

Danilevsky M, Wang C, Desai N, Han J (2013) Entity role discovery in hierarchical topical communities. KDD-MDS workshop, Chicago

Diehl C, Namata G, Getoor L (2007) Relationship identification for social network discovery. In: AAAI, Vancouver

Eagle N, Pentland A, Lazer D (2009) Inferring friendship network structure by using mobile phone data. PNAS 106(36):15274–15278

Everett M (1985) Role similarity and complexity in social networks. Soc Netw 7:353–359
Freeman L (1997) Uncovering organizational hierarchies. Comput Math Org Theory 3(1):5–18
Freeman L, Freeman S, Michaelson A (1988) On human social intelligence. J Soc Biol Struct 11:415–425
Goldenberg A (2007) Scalable graphical models for social networks. PhD thesis, Carnegie Mellon University
Gross N, Mason W, Mceachern A (1966) Explorations in role analysis: studies of the school superintendency role. Wiley, New York
Han Y, Tang J (2015) Probabilistic community and role model for social networks. In: SIGKDD, Sydney
Henderson K, Gallagher B, Basu S, Eliassi-rad T, Akoglu L, Koutra D, Faloutsos C, Tong H, Li L (2012) RolX: structural role extraction and mining in large graphs. In: SIGKDD, Beijing
Holland P, Laskey K, Leinhardt S (1983) Stochastic blockmodels: first steps. Soc Netw 5:109–137
Wang L, Lou T, Tang J, Hopcroft J (2011) Detecting community kernels in large social networks. In: ICDM, Vancouver
Jin X, Spangler S, Ma R, Han J (2010) Topic initiator detection on the World Wide Web. In: WWW, Raleigh
Jin R, Lee V, Hong H (2011) Axiomatic ranking of network role similarity. In: SIGKDD, San Diego
Kempe D, Kleinberg J, Tardos E (2003) Maximizing the spread of influence through a social network. In: SIGKDD, Washington
Kleinberg J (1999) Sources in a hyperlinked environment. J ACM 46:668–677
Leskovec J, Huttenlocher D, Kleinberg J (2010) Predicting positive and negative links in online social networks. In: WWW, Raleigh
Leuski A (2004) Email is a stage: discovering people roles from email archives. In: SIGIR, Sheffield
Li R, Wang C, Chang K (2014) User profiling in an ego network: Co-profiling attributes and relationships. In: WWW, Seoul
Lorrain F, White H (1971) Structural equivalence of individuals in social networks. J Math Sociol 1:49–80
Maiya A, Berger-Wolf T (2009) Inferring the maximum likelihood hierarchy in social networks. In: IEEE international conference on computational science and engineering (CSE), Vancouver
McCallum A, Corrada-Emmanuel A, Wang X (2005) Topic and role discovery in social networks. In: IJCAI, Edinburgh
Memon N, Larsen H, Hicks D, Harkiolakis N (2008) Detecting hidden hierarchy in terrorist networks: some case studies. In: IEEE international conference on intelligence and security informatics (ISI), Taipei, pp 477–489
Page L, Brin S, Motwani R, Winograd T (1999) The page rank citation ranking: brining order to the web. Technical report, Stanford Info Lab
Rowe R, Creamer G, Hershkop S, Stolfo S (2007) Automated social hierarchy detection through email network analysis. In: Joint 9th WEBKDD and 1st SNA-KDD workshop, San Jose
Scripps J, Tan P, Esfahanian A (2007) Exploration of link structure and community-based node roles in network analysis. In: ICDM, Leipzig
Shaparenko B, Caruana R, Gehnke J, Joachims T (2005) Identifying temporal patterns and key players in document collections. In: SIGKDD, Chicago
Tang W, Zhuang H, Tang J (2011) Learning to infer social ties in large networks. In: ECMLPKDD, Athens
Wang C, Han J, Jia Y, Tang J, Zhang D, Yu Y, Guo J (2010) Mining advisor-advisee relationships from research publication networks. In: SIGKDD, Washington
Wang C, Han J, Li Q, Li X, Lin W, Ji H (2012) Learning hierarchical relationships among partially ordered objects with heterogeneous attributes and links. In: SDM, Anaheim
Wang C, Danilevsky M, Liu J, Desai N, Ji H, Han J (2014) Constructing topical hierarchies in heterogeneous information networks. Knowl Inf Syst 44(3):529–558
Wolfe A, Jensen D (2004) Playing multiple roles: discovering overlapping roles in social networks. In: ICML, Banff

Role Identification of Social Networkers

Anna Zygmunt
Department of Computer Science, AGH University of Science and Technology, Kraków, Poland

Synonyms

Finding social positions; Key users

Glossary

Key Members	(Users) Those who contribute to the success and health of the community
Social Media	Set of web-based technologies targeted at forming and enabling a potentially massive community of participants to productively collaborate
Categories of Social Media	Blogs (e.g., Wordpress, Blogcatalog), Friendship Networks (e.g., Facebook, MySpace, LinkedIn), Microblogging (e.g.,

	Twitter), Media Sharing (e.g., Flickr, YouTube), Social Bookmarking (e.g., Del.icio.us), Social News (e.g., Digg), Social Collaboration (e.g., Wikipedia, Scholarpedia)
Social Networker	Person building her/his position by creating its own social networks in a variety of social media (categories) (http://www.wikihow.com/Be-a-Social-Networker)

Definition

The concept of "social role" has been the subject of analysis for more than 100 years, which underlines the importance of the problem. The meaning of this term is broad and definitions are largely dependent on the application. Therefore, it is difficult to give one definition which would widely be recognized.

Most of the definitions of roles in network analysis have their origin mainly in sociological and psychological research, where social roles are treated as "cultural objects that are recognized, accepted and used to accomplish pragmatic interaction goals in a community" (Gleave et al. 2009). Sociological research emphasizes the importance of relations to others and expectations for systematic behavior. So roles describe the intersection of behavioral, meaningful, and structural attributes that emerge regularly in particular settings and institutions (Wesler et al. 2011).

In role theory, roles are defined as "those behaviors characteristic of one or more persons in a context" (Biddle 1986). In turn, in network analysis, a role is identified as a position that has a distinct pattern of relations to other positions (Wasserman and Faust 1994). In social media, a definition of a role that seems to be most appropriate treats it as a set of characteristics (relevant metrics) that describe behavior of individuals and their interactions between them within a social context (Junquero-Trabado and Domingues-Sal 2012).

Introduction

A social network consists of a set of actors and a set of relationships between them which describe certain patterns of communication. Most current networks are huge and difficult to analyze and visualize. One of the methods frequently used to analyze social networks is to extract the most important features, namely, to create a certain abstraction, that is, the transformation of a large network to a much smaller one, so the latter is a useful summary of the original one, still keeping the most important characteristics. In the case of a social network, it can be achieved in two ways. One is to find groups of actors and present only them and relationships between them. The other is to find actors who play similar roles and to construct a smaller network in which the connection between the actors would be replaced with connections between the roles.

Work on assigning actors to roles greatly intensified with the advent of the possibility of collecting vast amounts of data capable of being then analyzed using methods of social network analysis. Social media, whose rapid growth can be observed for several years, provide us with new opportunities to define and use roles.

Classifying actors by the roles they are playing in the network can help to understand "who is who." This classification can be very useful, because it gives us a comprehensive view of the network and helps to understand how it is organized and to predict how it could behave in the case of certain events (internal or external).

In the beginning, the analysis of social media (mainly Usenet – the oldest social media developed in 1979) indicated a very unbalanced participation of users, i.e., most messages were written by a small percentage of users. It was thought that the identification of such popular users would help to understand processes taking place in the community. Many approaches, therefore, focused on finding only *leaders* in the community.

Another group of similar topics is finding *influential users*. Referring to the fundamental

article (Keller and Barry 2003), attempts were made to translate influential users characteristics (e.g., recognition, generation activity, novelty, eloquence) described therein into SNA measures.

However, to become important or influential, the group must have users performing different roles (more or less important) that will support such key users and influential users, as well as cause the group formed around these users to be more or less stable.

A lot of studies relate to certain social media and attempt to define their specific roles, so that a different set of roles is characteristic of discussion forums, blogosphere, etc. It is based on the assumption that different communities have different needs and the roles that support these needs vary greatly (Nolker and Zhou 2005). In addition, even for the same social media, different authors define different roles, different naming, and different characteristics.

Roles are also considered as a tool for simplifying patterns of action, distinguishing between different types of users, and understanding human behavior.

Since it was noticed that human activity is determined and restricted by social structures (such as social context, history of actions, structure of interactions, attributes people bring to the interaction) (Gleave et al. 2009), the concept of "social roles" is being broadened by a combination of social psychology, social structures, and behaviors.

Thus, a huge network can be defined as the interaction of relatively small sets of roles that exist in different proportion, making up a diversity of role ecologies.

Key Points

Because of different characteristics of social media, different roles can be identified. There are four main approaches to identifying social roles based on equivalence classes, identification of the core/periphery structure, analysis of basic social network measures, and clustering feature vectors.

Historical Background

Concept of role identification was first introduced in psychology and sociology (Merton 1968). Small, manually prepared networks were analyzed to explain the diversity of functions performed by the actors. Most current networks are huge (because of the popularity of social media) and difficult to analyze. The oldest and one of the most popular approaches to identifying roles is based on the equivalence classes.

Characteristics of Social Media

People use various social media services for different reasons, in different ways, and generally behave accordingly in each of them. For example, they use Flickr to display photos with friends; Twitter to show their status; Facebook to be in touch with friends; blogs to express their opinions, interests, and beliefs; Delicious to bookmark Web pages, etc. (Agarwal et al. 2012).

Members demonstrate different activity levels, but generally only a small percentage of them are active, participating regularly in discussions and with a very large number of followers, as well as the number of friends and social connections. The rest are not very active, which confirms the power law phenomena (Mathioudakis and Koudas 2009).

Some people are active only in few social media sites, while others use most of the sites. Each social medium has a different structure and a different way to use it. However, it must be noted that the vast majority have the structure containing the link information and content information (Agarwal et al. 2012) and are built around the message thread data structure (some messages are sent as a reply to a particular previous message).

By combining information about the roles performed by users on various social networking sites, more extensive (comprehensive) user profile can be obtained. If the user has a similar behavior on various social media, their behavior on other media can be predicted, without the need to analyze them. It can be indicated what types of media

will inspire greater commitment of users and why. Identification of the same users on different social networking sites is not easy, and often an assumption that users behave similarly performing similar roles is used (e.g., in Agarwal et al. 2012, they discovered that the user who is influential on one site has also a tendency to be influential on the other, using similar style of speaking).

Examples of Roles Performed by Social Networkers

A lot of the early studies on finding roles dealt with Usenet discussion groups. After some time, it turned out that some similar roles can be identified in other threaded discussion spaces (e.g., in the majority) where some messages are sent as a reply to a particular previous message, and in this way, a chain of messages is created (Wesler et al. 2007; Fisher et al. 2006).

Based on a combination of visualization of authors' posting behavior, posting behaviors of each author's neighbors and egocentric network graph, three important roles were identified in such spaces: "answer person," "discussion person," and "reply magnet."

The primary mode of interaction for the "answer person" is to provide useful answers to other questions asked by members of the group. "Discussion people" reply to one another about the topics introduced by the topic started by "reply magnet." "Discussion people" are characterized by frequent reciprocal exchanges with a relatively high number of other participants. This social role is the source of most of the discussion content contributed to long threaded conversations. "Reply magnet" is responsible for the majority of messages that initiate long threads. The key behavior of these individuals is creating new threads, usually by posting quoted material from external news sources.

It was noted that the two roles, "answer person" and "discussion person," are critical for many threaded discussion media; their presence contributes to the fact that given social media are living: questions are asked, answers are given, and there are new topics for discussion. Thus, these roles have a positive impact on the community. Unfortunately, negative roles, such as "spammers" or "flammers" may appear in community.

Analyzing Usenet from the perspective of finding important roles for the durability of community, two types of roles were identified: "leaders" (spread knowledge and maintain the cohesiveness of the group) and "motivators" (keep conversation going) (Nolker and Zhou 2005). Both roles were defined on the basis of their behavior, conversations, and member relationships. The third role, "chatters," was introduced, which refers to those who are engaged in a single discussion but rarely get involved in other discussions.

Defining a role only on the basis of an individual's behavioral patterns, other roles with different characteristics were found (Viegas and Smith 2004): "answer persons (pollinators)," "debaters," "spammers," and "conversationalists." For example, "pollinators" are characterized by a high number of days active, mostly responding to threads started by other authors with one or just a small number of messages sent to each thread. In turn, "debaters" with a high number of days active mostly respond to threads started by other authors with large numbers of messages sent to each thread.

Recently, many studies concern Twitter (mainly due to the easy availability of data and the possibility of analysis of the network dynamic). In the literature, many different sets of roles have been proposed, for example, "mainstream news source" (spreads information through the network), "celebrities" (public figures followed by many persons), and "opinion leaders" (spread widely their opinions and exercise a big influence among their persons in the network). A negative role is performed by "social spammers" who use social network to disseminate malware or spread commercial spam messages (Cha et al. 2010; Junquero-Trabado and Domingues-Sal 2012). Another set of roles is presented in Fazeen et al. (2011): "leaders" (who start tweeting, but do not follow others, but they can have many followers), "lurkers" (generally inactive, but occasionally follow some tweets), "spammers" (the unwanted tweeters, also called twammers), and "close associates" (including friends, family members, relatives, colleagues, etc.).

In Wesler et al. (2011) and Gleave et al. (2009), roles in Wikipedia were identified (noting that Wikipedia differs from discussion spaces in that the primary activity of the community is the construction of an artifact): "technical editors" (correct small errors related to style or formatting of articles), "vandal fighters" (revert vandalism and sanction norm violators), "substantive experts" (improve the quality of the content of the articles), and "social networkers" (support community aspects of Wikipedia and contribute little to the content and form of articles directly). As can be seen, there is quite a large flexibility in defining sets of roles.

The Methodology of Finding Roles

The most general approach to finding roles consists of two main stages (Wesler et al. 2007; Junquero-Trabado and Domingues-Sal 2012): an in-depth understanding of the community in order to identify roles which may be detected and then a creation of a role with observed characteristics and rules that will allow the classification of individuals into the predefined roles.

Social roles can be conceptualized at several different levels of abstraction (Gleave et al. 2009; Wesler et al. 2011). A good starting point is to first identify roles at the level of meaningful social action and descend to a lower level of abstraction to identify the key behavioral regularities and distinctive positions in social networks and then make generalizations to abstract theoretical categories that will make it possible to tie these particular roles to general research objectives.

Finding roles should rely on both structural data and detailed qualitative description of the context and meaning of interaction (Gleave et al. 2009). In the past, this approach was very difficult (only one of these aspects could be taken into consideration). Now, we have great opportunities to find significant structural roles and understand their meaning within the social context.

Social roles are often inherently defined in relational terms: a role only exists in relation to others, who are likewise enacting social roles (Wesler et al. 2007). Therefore, it is necessary to adopt a macro-perspective that combines both individual behavior and ecology of the entire social roles (balance and interaction of roles within a given population (Gleave et al. 2009) within a given social space).

Methods of Finding Social Roles

There are four main approaches to identifying social roles.

Approach Based on Equivalence Classes

It is the oldest approach to identifying roles (Wasserman and Faust 1994). It is assumed that a given person is a representative of a group of people who are somehow "equivalent" that are similar to and different from those of the other categories. Categories are defined in the context of the similarity between patterns of relationships among actors. Formally, these categories are presented using a family of algebraic equivalence classes on nodes: structural (the most basic and rigorous), automorphic, regular, and stochastic (Rossi and Ahmed 2015) and their variants, and with the use of blockmodels which are produced by applying these reduction to social networks. The least restrictive reduction is regular equivalence and is best suited for the sociological concept of the role: two nodes are said to be regularly equivalent if they have the same profile of ties with members of other sets of actors that are also regularly equivalent (Hannemann and Riddle 2005; Lerner 2005). Regular equivalence can be treated as clustering actors, according to their social positions (Brynielsson et al. 2012).

There are many algorithms for finding regular equivalence. All are based on searching the neighborhood of actors to find actors of other types. As long as they have similar "types" of actors at similar distances in their k-neighborhoods, they are regularly equivalent.

Approach Based on the Identification of the Core/Periphery Structure

This approach is based on extracting certain areas of varying activities and assigning roles to users based on membership of a particular area.

The concept of the core and the periphery was first introduced in Borgatti and Everett (1999). In the directed graph, two classes of nodes can be distinguished: belonging to a coherent subgraph (the core), in which nodes are connected to each other in some maximal way, and loosely connected nodes (the periphery). The core should have a lot of links with the periphery and should be connected with the core members much more than with the periphery. In turn, the periphery should indicate mainly nodes in the core and those in the periphery, but only to a small extent.

A core/periphery model was used in the analysis of dynamics of community (Yahoo!) (Backstrom et al. 2008). The concept of the core was redefined by introducing a definition of the k-core. Members belonging to the k-core must have an appropriate activity (e.g., replied to and been replied by minimum number of distinct users) during a specified time period. People who do not belong to the k-core are "light" users. Due to the time they are part of, the users can further be classified as "long-core" and "short-core." It was noticed, for example, that it is more likely that the users will be long-core, if they belong to a smaller number of groups (probably because they can then focus on those groups, at the level necessary to become long-term users). Moreover, it was noticed that the core is approximately the same size regardless of the size of the group. The relatively simple model was proposed, which did not take into account the quality of written posts by Yahoo! users (they might as well be spam).

In the case of blogosphere, work on the extracting structure similar to the core/periphery has gone in the direction of discovery A-List blogs, defined as "those that are most widely read, cited in the mass media, and receive the most inbound links from other blogs" (Obradovic and Baumann 2009). Blogs from A-List are strongly connected with one another but poorly with the rest of blogosphere, which is a decisive majority (long tail). Blogs from A-list are often linked to from the long tail, often link to each other, and rarely link to the long tail. To find blogs from the A-list, the modified definition of the k-core as connected components that contain only nodes of a minimum degree of k was used.

Analyzing the dynamics of social media (Flickr, Yahoo!), it was noted (Kumar et al. 2006) that the structure of the periphery is not homogenous and can be distinguished in the "singletons" (nodes not linked to any other network nodes) and "middle region" (isolated groups who interact with one another, but not with the network as such). These isolated groups are in fact influential individualities that act as "stars" combined with a varying number of other users who, in turn, have very few other connections. An analysis of network dynamics shows that such "stars" may be included in the core or disappear as soon as they lose interest in the group.

Approach Based on the Analysis of Basic SNA Measures

This approach takes into consideration the fundamental features of users' structural position such as their number of neighbors and relies on various centrality measures, i.e., local (within social communities) or global (for the entire network) structural features (Newman and Girvan 2004; Zygmunt et al. 2012).

There are three traditional roles based on node network structure: "hubs," "brokers" ("gatekeepers"), and "bridges" ("pustakers") (Nolker and Zhou 2005; Denning 2004). "Hub" is a person who links to many others; "broker" is the only connection between communities; "bridge" links several communities. Such roles can be found by analyzing the basic SNA centrality measures, such as *degree centrality* (the number of conversations that a user is engaged in or the number of users that a user has conversed with), *betweenness centrality* (the number of pairs of other members who can converse with each other directly through a user with shortest distance), and *closeness centrality* (average conversation distance between a user and all the others in the community). In Goldenberg et al. (2009), "hub" was defined as "people with both in- and out-degrees that are larger than three standard deviations above the mean." The presence of such roles in the community promotes the spread of innovations such as: innovations are most likely to spread if "hubs" adopt and recommend them; "brokers" and

"bridges" play important roles in spreading the idea to new groups.

An analysis of the basic measures of SNA has been used in several studies to define social roles of "starters" and "followers" on discussion forums and in blogosphere (Hansen et al. 2010; Mathioudakis and Koudas 2009; Sun and Ng 2012). "Starters" receive messages mostly from people who are well connected to each other and therefore can be identified by low in-degree, high out-degree, and high clustering coefficient in the graph. The distinction between the roles is obtained by combining the difference between the numbers of in-links and out-links of their blogs.

In a similar way an "answer person" was identified on discussion forums (Fisher et al. 2006), noting that it is a person who responds to many other people, but rarely to those who provide answers to the questions raised by the community. So it should have high out-degree and low in-degree. For each author one-degree and two-degree egocentric social networks were constructed through patterns of reply. These networks were then grouped (e.g., on the basis of collective in-degree and out-degree, degree distribution coefficient across groups), and for each distribution of the out-degree of each actor's out-neighbors were calculated.

Wesler et al. (2007) observed that social roles can be described using patterned characteristics of communication between network members, which is called "structural signatures." It was hypothesized that it is possible to recognize the roles that people play by measuring behavioral and structural signature of their participation. The goal is to identify general structural features that are associated with a particular role. Egocentric network and visualization were used to identify structural attributes associated with the role. Thus, for example, "answer persons" are mainly connected with users with low degree, their local networks tend to have small proportions of three cycles (i.e., their neighbors are not neighbors of each other, they seldom send multiple messages to the same recipient (few intense ties), they tend to reply to discussion threads initiated by others, and they generally contribute one or two messages per thread. Ego network for an "answer person" is similar to a star and differs from ego network persons performing other two roles (Gleave et al. 2009).

In Nolker and Zhou (2005), to define roles, a combination of SNA measures with behavior-based measures from the information retrieval (term frequency and inverse document frequency: TF*IDF) was used. TF*IDF indicates the weight of conversation relationships between members. It was observed that *betweenness* plays an important role in predicting the relationship potential that a member has with other members. In turn, high *in-closeness* indicates a member who provides consistency and high *out-closeness* – a member who spreads knowledge.

Approach Based on Clustering Feature Vectors

In this approach, each person in the network is represented by a vector of some of the features that represent its behavior and relationships with the other members of the community. These features can be, for example, the number of peoples the user knows, the number of people that know the user, the number of reciprocal relationships of the user, the number of messages that the user receives, the number of documents that depict the user, etc.

Such vectors can then be clustered so that people with similar characteristics are placed in one group (Maia et al. 2008; Junquero-Trabado and Domingues-Sal 2012; Pal and Counts 2011). Mostly well known in statistics and data mining, *k-means* algorithm was used. The cluster is described with the use of relevant metrics that are important for a given role. Thus, each role can have a different number of relevant metrics. On Twitter, for example, the role of "celebrities" means the most followed and mentioned persons, most connected but generally not the most influential, so the relevant metrics for this role are shown in the number of followers and the number of documents depicting a given person. In turn, the role of "information propagators" is mainly a

source of information, so the relevant metrics are expressed in the number of followers, average and maximum information propagation, the number of publications, and the number of words in tweets that exist in dictionary. It is the user, who on the basis of the results of the clustering, can create a set of roles; therefore, such an approach can be regarded as most universal and one of the few independent of the particular application (Junquero-Trabado and Domingues-Sal 2012).

Key Applications

Marketing, opinion diffusion, and advertising: finding people that will ensure that information about new product will come down to the largest number of other people and will cause, for example, an increase in sales.

Recommendation systems: individuals, who share the same social role, might be expected to share the same taste.

Political campaign: roles, which are necessary to ensure the success of campaign.

Detecting important members of criminal or terrorist groups.

Future Directions

Searching for new roles, creating a uniform formal model describing the community, and taking into account the roles and interactions between them.

Searching roles in several heterogeneous networks and trying to find roles for a user in multiple, heterogeneous networks.

Developing recommendations for the structure of the system of roles (roles ecologies) (qualitative and quantitative) for the effective functioning of communities.

Cross-References

- ▶ Patterns in Productive Online Networks: Roles, Interactions, and Communication
- ▶ Process of Social Network Analysis
- ▶ Role Discovery
- ▶ Social Influence Analysis

References

Agarwal N, Kumar S, Gao H, Zafarani R, Liu H (2012) Analyzing behavior of the influentials across social media. In: Cao L, Yu PS (eds) Behavior computing: modeling, analysis, mining, and decision. Springer, London, pp 3–19

Backstrom L, Kumar R, Marlow C, Noval J, Tomkins A (2008) Preferential behavior in online groups. In: 1st ACM WSDM international conference on web search and data mining, Palo Alto, pp 117–128

Biddle BJ (1986) Recent developments in role theory. Annu Rev Sociol 12:67–92

Borgatti SP, Everett MG (1999) Models of core/periphery structures. Soc Networks 21:375–395

Brynielsson J, Kaati J, Svenson P (2012) Social position and simulation relations. Soc Netw Anal Min 2(1):39–52

Cha M, Haddadi H, Benevenuto F, Gummadi P (2010) Measuring user influence in Twitter: the million follower fallacy. In: 4th international AAAI conference on weblogs and social media (ICWSM), Washington, DC

Denning PJ (2004) Network laws. Commun ACM Publ 47:15–20

Fazeen M, Dantu R, Guturu P (2011) Identification of leaders, lurkers, associates and spammers in a social network: context-dependent and context-independent approaches. Soc Netw Anal Min 1(3):241–254

Fisher D, Smith M, Welser HT (2006) You are who you talk to: detecting roles in usenet newsgroups. In: Proceedings of the 39th annual Hawaii international conference on systems sciences, HICSS'06, Kauai. IEEE Computer Society

Gleave E, Welser HT, Lento TM, Smith MA (2009) A conceptual and operational definition of 'social role' in online community. In: Proceedings of the 42nd Hawaii international conference on systems sciences, HICSS'09, Waikoloa, Big Island. IEEE Computer Society, pp 1–11

Goldenberg J, Han S, Lehmann DR, Hong JW (2009) The role of hubs in the adoption process. J Mark 73(2):1–13

Hannemann RA, Riddle M (2005) Introduction to social network methods. University of California, Riverside

Hansen DL, Shneiderman B, Smith MA (2010) Visualizing threaded conversation networks: mining message boards and email lists for actionable insights. In: Proceedings of the 6th international conference of active media technology (AMT'10), Toronto. Springer, Berlin/Heidelberg, pp 47–62

Junquero-Trabado V, Domingues-Sal D (2012) Building a role search engine for social media. In: Proceedings of

the 21st international conference companion on world wide web (WWW'12), Lyon. ACM, New York, pp 1051–1060
Keller E, Barry J (2003) The infiuentials: one American in ten tells the other nine how to vote, where to eat, and what to buy. Free Press, New York
Kumar R, Noval J, Tomkins A (2006) Structure and evolution of online social networks. In: Proceedings of the 12th ACM SIGKDD international conference on knowledge discovery and data mining (KDD'06), Philadelphia. ACM, New York, pp 611–617
Lerner J (2005) Role assignments. Network analysis. In: Brandes U, Erlebach T (eds) Methodological foundations, LNCS, vol 3418. Springer, Berlin/Heidelberg, pp 216–252
Maia M, Almeida J, Almeida V (2008) Identifying user behavior in online social networks. In: SocialNets, proceedings of the 1st workshop on social network systems (SocialNets'08), Glasgow, Scotland. ACM, New York, pp 1–6
Mathioudakis M, Koudas N (2009) Efficient identification of starters and followers in social media. In: Proceedings of the 12th international conference on extending database technology: advances in database technology (EDBT'09), Saint-Petersburg. ACM, New York, pp 708–719
Merton R (1968) Social theory and social structure. Simon & Schuster, New York
Newman MEJ, Girvan M (2004) Finding and evaluating community structure in networks. Phys Rev E 69(2):026113
Nolker RD, Zhou L (2005) Social computing and weighting to identify member roles in online communities. In: Web intelligence, proceedings of the IEEE/WIC/ACM international conference on web intelligence, Compiegne, pp 87–93
Obradovic D, Baumann S (2009) A journey to the core of the blogosphere. In: Social network analysis and mining, ASONAM'09, international conference on advances in social networks analysis and mining, Athens. IEEE Computer Society, pp 1–6
Pal A, Counts S (2011) Identifying topical authorities in microblogs. In: Proceedings of the 4th international conference on web search and data mining (WSDM'11), Hong Kong. ACM, Hong Kong, pp 45–54
Rossi R, Ahmed N (2015) Role discovery in networks. IEEE Trans Knowl Data Eng 27(4):1112–1131
Sun B, Ng VTY (2012) Identifying influential users by their postings in social networks. In: Proceedings of the 3rd international workshop on modeling social media (MSM'12), Milwaukee, pp 1–8
Viegas FB, Smith MA (2004) Newsgroup crowds and author lines: visualizing the activity of individuals in conversational cyberspace. In: Proceedings of the 37th Hawai'i international conference on systems sciences, Kauai. IEEE, Los Alamitos
Wasserman S, Faust K (1994) Social network analysis: methods and applications. Cambridge University Press, Cambridge/New York
Wesler HT, Gleave E, Fisher D, Smith M (2007) Visualizing the signature of social roles in online discussion groups. J Soc Struct 8(2.) http://www.cmu.edu/joss/content/articles/volume8/Welser/
Wesler HT, Cosley D, Kossinets G, Lin A, Dokshin F, Gay G, Smith M (2011) Finding social roles in Wikipedia. In: Proceedings of the 2011 iConference, Seattle. ACM, Seattle, pp 122–129
Zygmunt A, Bródka P, Kazienko P, Koÿlak J (2012) Key person analysis in social communities within the blogosphere. J UCS 18(4):577–597

Recommended Reading

Bettencourt BA, Sheldon K (2001) Social roles as mechanisms for psychological need satisfaction within social groups. J Pers Soc Psychol 81(6):1131–1143
Golder SA, Donath J (2004) Social roles in electronic communities. AoIR, Brighton
Obradovic D, Rueger C, Dengel A (2011) Core/periphery structure versus clustering in international weblogs. In: Proceedings of the international conference on computational aspects of social networks (CASoN 2011). IEEE, Salamanca

Role-Playing

► Gaming and Virtual Worlds

Roll Call Prediction

► Legislative Prediction with Political and Social Network Analysis

Routine Discovery

► Assessing Individual and Group Behavior from Mobility Data: Technological Advances and Emerging Applications

Routing

► Paths in Complex Networks

R-Project

► R Packages for Social Network Analysis

RSiena

► R Packages for Social Network Analysis

Ruby on Rails

► Server-Side Scripting Languages

Rule-Based Systems

► RIF: The Rule Interchange Format