UNIT 1 INTRODUCTION

1.1 Limitations of the current Web

1.1.1 What's wrong with the Web?

1. **Who is Frank van Harmelen?**

To answer such a question using the Web one would go to the search engine and enter the most logical keyword: *harmelen*. The results returned by Google are shown in Figure 1.1.1 (Note that the results are slightly different depending on whether one enters Google through the main site or a localized version.)

If this question and answer would be parts of a conversation, the dialogue would sound like this:

> Q: *Who is Frank van Harmelen?*
>
> A: *I don't know but there are over a million documents with the word "harmelen" on them and I found them all really fast (0.31s). Further, you can buy Harmelen at Amazon. Free Delivery on Orders Over 15.*

the word Harmelen means a number of things. It's the name of a number of people, including the (unrelated) Frank van Harmelen and Mark van Harmelen. Harmelen is also a small town in the Netherlands (one hit) and the place for a tragic train accident (one hit).

The problem is thus that the keyword *harmelen* (but even the term *Frank van Harmelen*) is polysemous. The reason of the variety of the returned results is that designers of search engines know that users are not likely to look at more than the top ten results. Search engines are thus programmed in such a way that the first page shows a diversity of the most relevant links related to the keyword.

This allows the user to quickly realize the ambiguity of the query and to make it more specific.

Studying the results and improving the query, however, is up to the user.

Figure 1.1.1 Search results for the keyword *harmelen* using Google.

If we browse further in the results we notice that the overwhelming majority of the results are related to prof. Frank van Harmelen of the Vrije Universiteit, but not all of them: there are other people named Frank van Harmelen. None of our queries would return pages about him where he is only mentioned by his first name for example or as *van Harmelen, F.* Not even if for the human reader it would be blatantly obvious that the Frank in question could only be Frank van Harmelen.

2. **Show me photos of Paris**

The most straightforward solution to this search task is typing in “paris photos” in the search bar of our favorite search engine. Most advanced search engines, however, have specific facilities for image search where we can drop the term photo from the query. Some of the results returned by Google Image Search are shown in Figure 1.1.2.

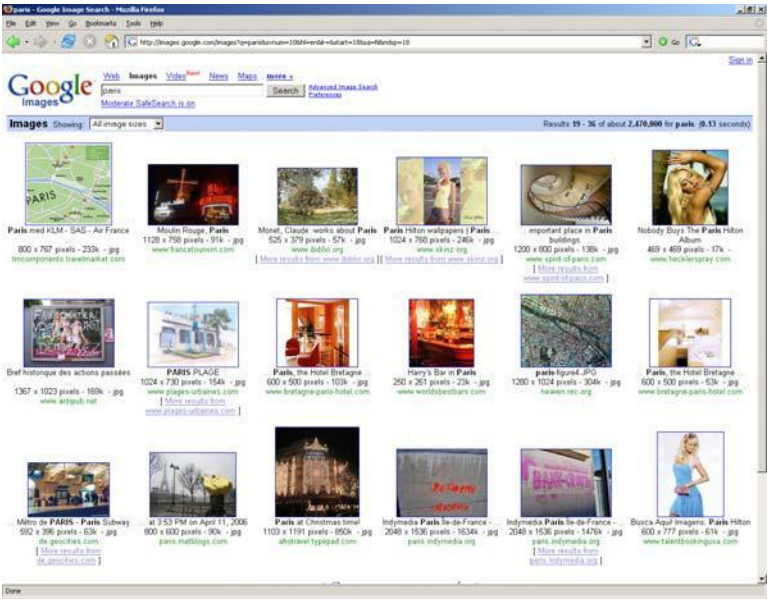


Figure 1.1.2. Search results for the keyword *paris* using Google Image Search.

Again, what we immediately notice is that the search engine fails to discriminate two categories of images: those related to the city of Paris and those showing Paris Hilton, the heiress to the Hilton fortune whose popularity on the Web could hardly be disputed.

For the keyword *Paris* most of us would expect photos of places in Paris or maps of the city. In reality only about half of the photos on the first page, a quarter of the photos on the second page and a fifth on the third page are directly related to our concept of Paris. The rest are about clouds, people, signs, diagrams etc.

The problem is that associating photos with keywords is a much more difficult task than simply looking for keywords in the texts of documents.

3. **Find new music that I (might) like**

As in the previous case, a search engine could avoid the problem of understanding the content of music and look at the filename and the text of the web page for clues about the performer or the genre. We suspect that such search engines do not exist for different reasons: most music on the internet is shared illegally through peer-to-peer systems that are completely out of reach for search engines. Music is also a fast moving good; search engines typically index the Web once a month and therefore too slow for the fast moving world of music releases. (Google News, the news search engine of Google addresses this problem by indexing well-known news sources at a higher frequency than the rest of the Web.)

But the reason we would not attempt to pose this query mostly has to do with formulating the music we like. Most likely we would search for the names of our favorite bands or music styles as a proxy, e.g. *"new release" ("Macy Gray" OR "Robbie Williams")*. This formulation is awkward on the one hand because it forces us to query by example. It will not make it possible to find music that is similar to the music that we like but from different artists. In other words it will not lead us to discover new music.

On the other hand, our musical taste might change in which case this query would need to change its form.

4. **Tell me about music players with a capacity of at least 4GB.**

This is a typical e-commerce query: we are looking for a product with certain characteristics.

One of the immediate concerns is that translating this query from natural language to the Boolean language of search engines is (almost) impossible. We could try the search *"music player" "4GB"* but it is clear that the search engine will not know that 4GB is the capacity of the music player and we are interested in all players with at least that much memory (not just those that have exactly 4GB). Such a query would return only pages where these terms occur as they are. Problem is that general purpose search engines do not know anything about music players or their properties and how to compare such properties. They are good at searching for specific information (e.g. the model number of an MP3 player), but not in searching for descriptions of items.

An even bigger problem is the one our machines face when trying to collect and aggregate product information from the Web. Even if an algorithm can determine that the page describes a music player, information about the product is very difficult to spot. We could teach the computer a heuristic that the price is the number that appears directly after the word "price".

Further, what one vendor calls "capacity" and another may call "memory". In order to compare music players from different shops we need to determine that these two properties are actually the same and we can directly compare their values.

1.1.2 Diagnosis: A lack of knowledge

Namely, in all five cases we deal with a *knowledge gap*: what the computer understands and able to work with is much more limited than the knowledge of the user. The handicap of the computer is mostly due to technological difficulties in getting our computers to understand natural language or to "see" the content of images and other multimedia. Even if the information is there, and is blatantly obvious to a human reader, the computer may not be able to see anything else of it other than a string of characters. In that case it can still compare to the keywords

provided by the user but without any understanding of what those keywords would mean.

This problem affects all of the above queries to some extent. A human can quickly skim the returned snippets (showing the context in which the keyword occurs) and realize that the different references to the word Harmelen do not all refer to persons and even the persons named Harmelen cannot all be the same. In the second query, it is also blatantly obvious for the human observer that not all pictures are of cities. However, even telling cities and celebrities apart is a difficult task when it comes to image recognition.

In most cases, however, the knowledge gap is due to the lack of some kind of *background knowledge* that only the human possesses. The background knowledge is often completely missing from the context of the Web page and thus our computers do not even stand a fair chance by working on the basis of the web page alone. In the case of the second query, an important piece of knowledge that the computer doesn't possess is the common knowledge that there is a city named Paris and there is a famous person named Paris Hilton (who is also different from the Hilton in Paris).

Answering the third query requires the kind of extensive background knowledge about musical styles, genres etc. that shop assistants and experts in music possess. This kind of knowledge is well beyond the information that is in the database of a typical music store. The third case is also interesting because there is also lacking background knowledge about the user. There has to be a way of providing this knowledge to the search engine in a way that it understands it.

The fourth query is noteworthy because it highlights the problem of aggregating information. The factual knowledge about particular products can be more or less extracted from the content of web pages, but if not, shop owners could be asked to provide it. It is unrealistic to expect, however, that all shops on the Web would agree to one unified product catalog (a listing of product types, properties, models etc) and provide information according to that schema. But if each shop provides information using its own classification we need additional knowledge in order to merge data from different catalogs. For example, we need to know that "mp3 players" and "mp3 capable mobile phones" both fit the category of digital music players, that "capacity" and "memory" are the same things and that 500 dollars is the equivalent of (about) 400 Euros.

1.1.3 The semantic solution (Introduction to Semantic web)

The idea of the Semantic Web is to apply advanced knowledge technologies in order to fill the knowledge gap between human and machine. This means providing knowledge in forms that computers can readily process and reason with. This knowledge can either be information that is already described in the content of the Web pages but difficult to extract or additional background knowledge that can help to answer queries in some way.

In the case of the first query the situation can be greatly improved by providing personal information in a semantic format. An existing solution is to attach a semantic profile to personal web pages that describe the same information that appears in the text of the web page but in a machine processable format. The Friend-of-a-Friend (FOAF) project provides a widely accepted vocabulary for such descriptions. FOAF profiles listing attributes such as the name, address, interests of the user can be linked to the web page or even encoded in the text of the page. As we will see several profiles may also exist on the Web describing the same person.

As all profiles are readable and comparable by machines, all knowledge about a person can be combined automatically.

For example, Frank van Harmelen has such a profile attached to his homepage on the Web. This allows a search engine to determine that the page in question is about a person with specific attributes. (Thus pages about persons and villages would not be confused.) Assuming that all other van Harmelens on the Web would provide similar information, the confusion among them could also be easily avoided. In particular, the search engine could alert us to the ambiguity of our question and ask for some extra information about the person we are looking for. The discussion with the search engine would be very different:

Q: *Who is Frank van Harmelen?*
A: *Your question is ambiguous: there is a great deal of information about a Frank van Harmelen who is a professor at the Vrije Universiteit. However, there are other persons named Harmelen and also a village in the municipality of Woerden. Which one did you mean?*

Similarly, the solution in the second case is to attach metadata to the images in question. For example, the online photo sharing site Flickr allows to annotate images using geographic coordinates. After uploading some photos users can add keywords to describe their images (e.g. “Paris, Eiffel-tower”) and drag and drop the images on a geographic map to indicate the location where the photo was taken. In the background the system computes the latitude and longitude of the place where the user pointed and attaches this information to the image. Searching of photos of Paris becomes a breeze: we can look up Paris on the map and see what other photos have been put there by other users. Although in this case the system is not even aware that Paris is a city, minimal additional information about photos (the geo-coordinates) enables a kind of visualization that makes the searching task much easier. And if over time the system notes that most images with the keyword “Paris” fall in a specific geographic area on the map, it can even conclude that Paris is a place on the map (see Figure 1.1.3).

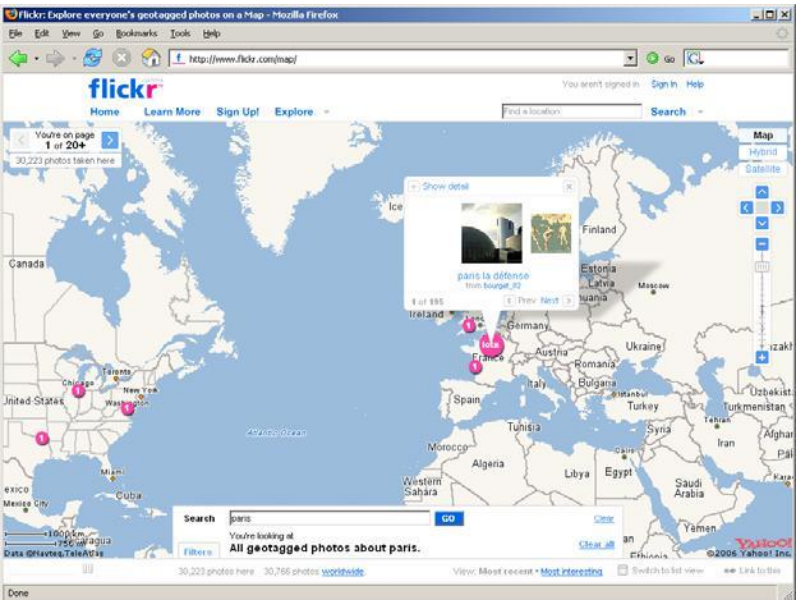


Figure 1.1.3. Searching for the keyword Paris using the geographic search of Flickr.

The FOAF-ing the Music project combines this idea with the retrieval of information about music releases and upcoming events related to our favorite artists. A particularly interesting feature of this system is that it can reuse information about our musical tastes from other sources such as a personal profile on one's homepage or an online playlist of our recently played music. In order to track the fast-paced world of music, the system tracks newsfeeds updated by data providers on a daily basis.

Our fourth problem, the aggregation of product catalogs can also be directly addressed using semantic technology. As we have seen the problem in this case is the difficulty of maintaining a unified catalog in a way that does not require an exclusive commitment from the providers of product information. (In practice, information providers often have their own product databases with a proprietary classification system.) Further, we would like to keep the catalogue open to data providers adding new, emerging categories of products and their descriptions (e.g. *mp3 players* as a subclass of music players with specific attributes such as capacity, size, color etc.)

The Semantic Web solution is to create a minimal, shared, top-level schema in one of the ontology languages defined for the Semantic Web. The advantage com-pared to publishing a monolithic product catalogue in XML (such as RosettaNet) is that semantic languages have been designed for extensibility in the distributed environment of the Web. This means that new characteristics and entire subcategories of products can be independently introduced by vendors. These vendor-specific extensions will be understood to the extent that they are described in terms of the existing elements of the shared schema. Further, mappings between entire schemas or parts of a schema can also be expressed using Semantic Web languages. For example, map-pings between the product classifications of independent vendors allow the products classified by one classification to be retrieved according to an alternative classification. The mappings themselves can be developed and maintained by any of the two parties or even by an independent, third party.

1.2 Development of the Semantic Web

1.2.1 Research, development and standardization

As the Semantic Web is a relatively new, dynamic field of investigation, it is difficult to precisely delineate the boundaries of this network. The complete list of individuals in this community consists of 608 researchers mostly from academia (79%) and to a lesser degree from industry (21%). Geographically, the community covers much of the United States, Europe, with some activity in Japan and Australia (see Figure 1.2.1). The core technology of the Semantic Web, logic-based languages for knowledge representation and reasoning has been developed in the research field of Artificial Intelligence

Figure 1.2.1. Semantic Web researchers and their network visualized according to geography.

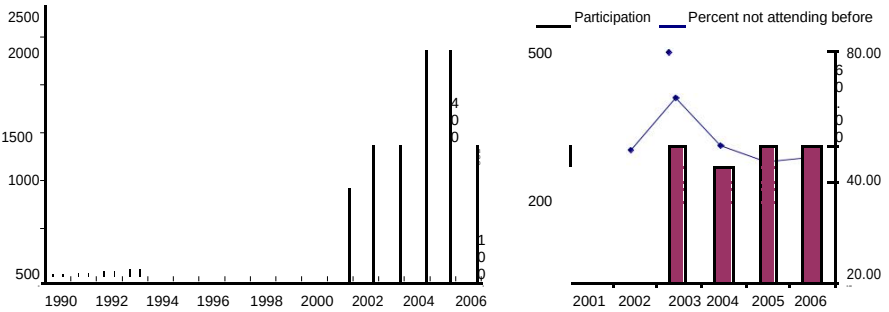


Figure 1.2.1.1. Semantic Web related publications per year (1990-2006) and participation at the yearly international Semantic Web events (2001-2006).

The World Wide Web Consortium still plays a key role in standardization where the interoperability of tools necessitates mediation between various developer and user communities, as in the case of the development of a standard query language and protocol to access ontology stores across the Web. Further, realizing the importance of adoption the W3C is active in promoting Semantic Web technology.

1.2.2 Technology adoption

The Semantic Web was originally conceptualized as an extension of the current Web, i.e. as the application of metadata for describing Web content. In this vision, the content that is already on the Web (text, but also multimedia) would be enriched in a collaborative effort by the users of the Web.

While the research effort behind the Semantic Web is immense and growing dynamically, Semantic Web technology has yet to see mainstream use on the Web and in the enterprise. In the following, we will illustrate the growth of the Semantic Web by tracing its popularity on the Web. Although this method does not allow us to survey the use of Semantic Web technology in enterprise or governmental settings, we believe that internal applications of Semantic Web technology ("Semantic Webs" within organizations) are likely lagging behind due to the natural caution with which industry and the government treat any new technology.

To follow the popularity of Semantic Web related concepts and Semantic Web standards on the Web; we have executed a set of temporal queries using the search engine Altavista. The queries contained single terms plus a disambiguation term where it was necessary. Each query measured the number of documents with the given term(s) at the given point in time.

Figure 1.2.2 shows the number of documents with the terms *basketball*, *Computer Science*, and *XML*. We have divided all counts with the number of documents with the word *web* to account for the general growth of the Web. The flat curve for the term basketball validates this strategy: we could have expected the popularity of basketball to be roughly stable over this time period. Computer Science takes less and less share of the Web as the Web shifts from scientific use to everyday use. The share of XML,

a popular pre-semantic web technology seems to grow and stabilize as it becomes a regular part of the toolkit of Web developers.

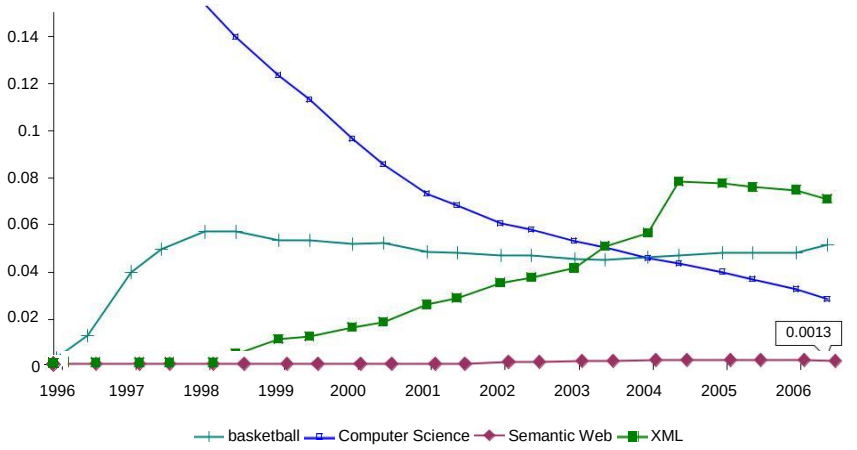


Figure 1.2.2. Number of webpages with the terms *basketball*, *Computer Science*, and *XML* over time and as a fraction of the number of pages with the term *web*.

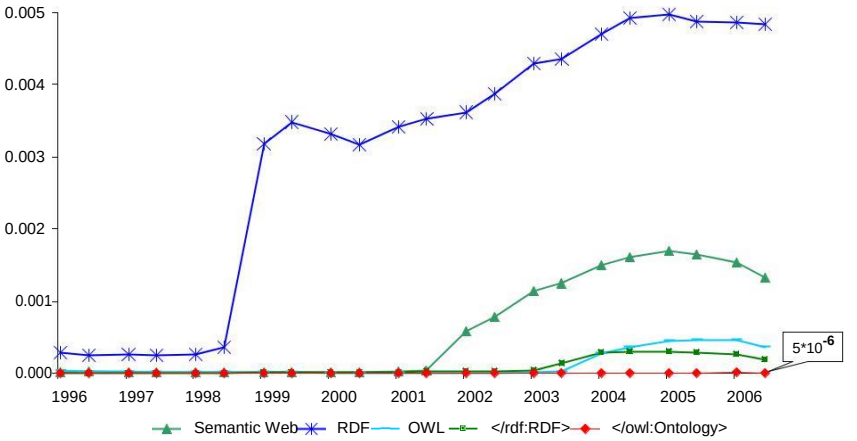
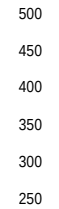


Figure 1.2.2.1. Number of webpages with the terms *RDF*, *OWL* and the number of ontologies (Semantic Web Documents) in RDF or OWL over time. Again, the number is relative to the number of pages with the term *web*.



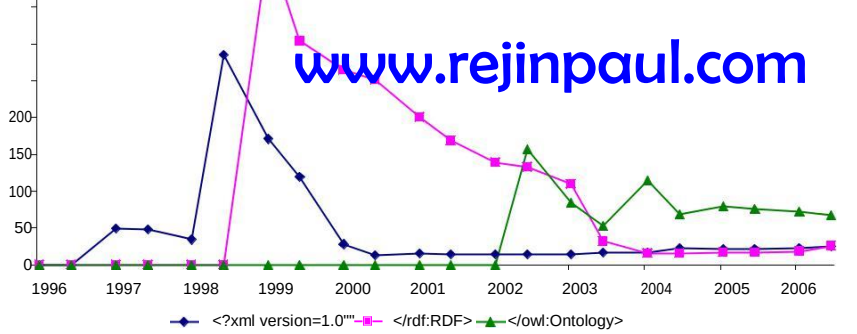


Figure 1.2.2.2. The hype cycle of Semantic Web related technologies as shown by the number of web pages about a given technology relative to its usage.

The hype cycle is defined by Gartner as follows:

The first phase of a Hype Cycle is the "technology trigger" or breakthrough, product launch or other event that generates significant press and interest. In the next phase, Technologies enter the "trough of disillusionment" because they fail to meet expectations and quickly become unfashionable. some businesses continue through the "slope of enlightenment" and experiment to understand the benefits and practical application of the technology. A technology reaches the "plateau of productivity" as the benefits of it become widely demonstrated and accepted.

1.3 The emergence of the social web

The resulting increase in our capacity to obtain information and social support online can be quantified. The survey confirms that not only networks are maintained and extended online, but they are also successfully activated for dealing with major life situations such as getting support in case of a major illness, looking for jobs, informing about major investments etc.

The first wave of socialization on the Web was due to the appearance of *blogs*, *wikis* and other forms of web-based communication and collaboration. Blogs and wikis attracted mass popularity from around 2003 (see Figure 1.3). Even more importantly, despite that weblogs have been first assessed as purely personal publishing, nowadays the blogosphere is widely recognized as a densely interconnected social network through which news, ideas and influences travel rapidly as bloggers reference and reflect on each other's postings.

Although the example of Wikipedia, the online encyclopedia is outstanding, wikis large and small are used by groups of various sizes as an effective knowledge management tool for keeping records, describing best practices or jointly developing ideas.

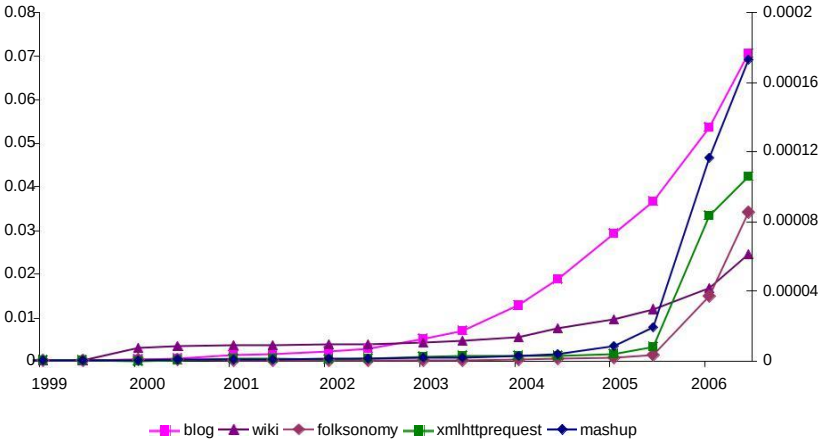


Figure 1.3. Development of the social web. The fraction of webpages with the terms *blogs*, *wiki* over time is measured on the left vertical axis. The fraction of webpages with the terms *folksonomy*, *XmlHttpRequest* and *mashup* is measured on the right hand vertical axis.

The first *online social networks* (also referred to as social networking services) entered the field at the same time as blogging and wikis started to take off. Although these sites feature much of the same content that appears on personal web pages, they provide a central point of access and bring structure in the process of personal information sharing and online socialization. Following registration, these sites allow users to post a profile with basic information, to invite others to register and to link to the profiles of their friends. The system also makes it possible to visualize and browse the resulting network in order to discover friends in common, friends thought to be lost or potential new friendships based on shared interests.

Explicit user profiles make it possible for these systems to introduce rating mechanism whereby either the users or their contributions are ranked according to usefulness or trustworthiness. Ratings are explicit forms of social capital that regulate exchanges in online communities in much the same way that reputation moderates exchanges in the real world.

1.3.1 Web 2.0 + Semantic Web = Web 3.0?

Web 2.0 is often contrasted to the Semantic Web, which is a more conscious and carefully orchestrated effort on the side of the W3C to trigger a new stage of developments using semantic technologies. In practice the ideas of Web 2.0 and the Se-mantic Web are not exclusive alternatives: while Web 2.0 mostly effects how users interact with the Web, while the Semantic Web opens new technological opportunities for web developers in combining data and services from different sources. In the following, we point out some of the opportunities that arise by the combination of ideas from these two developments.

Firstly, we note that the basic lesson of Web 2.0 is that *users are willing to provide content as well as metadata*. This may take the form articles and facts organized in tables

and categories in Wikipedia, photos organized in sets and according to tags in Flickr or structured information embedded into homepages and blog postings using *microformats*. These latter are mini-vocabularies for encoding metadata of all kinds in HTML pages, for example information about the author or a blog item.)

Secondly, due to the extensive collaborations online many applications have *access to significantly more metadata about the users*. Information about the choices, preferences, tastes and social networks of users means that the new breed of applications are able to build on a much richer user profiles. Clearly, semantic technology can help in matching users with similar interests as well as matching users with avail-able content.

Lastly, in terms of technology what the Semantic Web can offer to the Web 2.0 community is a standard infrastructure for the building creative combinations of data and services. Standard formats for exchanging data and schema information, support for data integration, along with standard query languages and protocols for querying remote data sources provide a platform for the easy development of mashups. San Francisco based startup MetaWeb is developing FreeBase, a kind of "data commons" that allows users to share, interlink, and jointly edit ontologies and structured data through a Web-based interface. A public API allows developers to build applications using the combined knowledge of FreeBase.

1.4. Social Network Analysis

Social Network Analysis (SNA) is the study of social relations among a set of actors. The key difference between network analysis and other approaches to social science is the focus on relationships between actors rather than the attributes of individual ac-tors.

Network analysis provides a vocabulary for describing social structures, provides formal models that capture the common properties of all (social) networks and a set of methods applicable to the analysis of networks in general.

The methods of data collection in network analysis are aimed at collecting relational data in a reliable manner. Data collection is typically carried out using standard questionnaires and observation techniques that aim to ensure the correctness and completeness of network data. Often records of social interaction (publication databases, meeting notes, newspaper articles, documents and databases of different sorts) are used to build a model of social networks. We return to the particular use of electronic data (data from the Internet and the Web)

1.4.1 Development of Social Network Analysis

A *sociogram* is a visual representation of social networks as a set of nodes connected by directed links. The nodes represented individuals while the edges stood for personal relations. However, similar representations can be used to depict a set of relationships between any kind of social unit such as groups, organizations, nations etc. While 2D and 3D visual modelling is still an important technique of network analysis, the sociogram is honored mostly for opening the way to a formal treatment of network analysis based on graph theory.

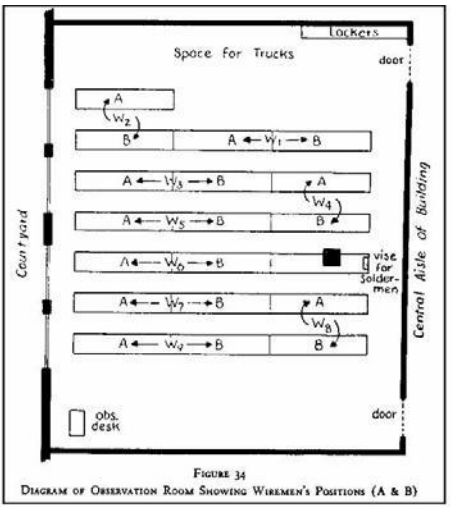


FIGURE 34
DIAGRAM OF OBSERVATION ROOM SHOWING WIREMEN'S POSITIONS (A & B)

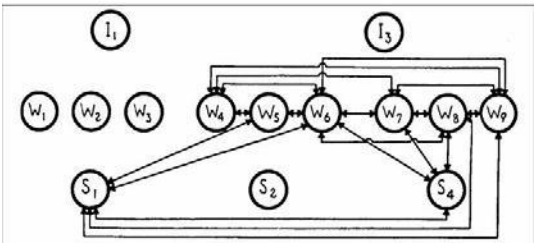


FIGURE 40
PARTICIPATION IN CONTROVERSIES ABOUT WINDOWS

Figure 1.4.1 Illustrations from an early social network study at the Hawthorne works of Western Electric in Chicago. The upper part shows the location of the workers in the wiring room, while the lower part is a network image of fights about the windows between workers (W), solderers (S) and inspectors (I).

The vocabulary, models and methods of network analysis also expand continuously through applications that require handling ever more complex data sets. An example of this process is the advances in dealing with longitudinal data. New probabilistic models are capable of modelling the evolution of social networks and answering questions regarding the dynamics of communities.

First, advances in information technology brought a wealth of electronic data and significantly increased analytical power. Second, the methods of SNA are increasingly applied to networks other than social networks such as the hyperlink structure on the Web or the electric grid.

1.5. Key concepts and measures in network analysis

Social Network Analysis has developed a set of concepts and methods specific to the analysis of social networks. We will proceed from the global structure of networks toward the measurement of *ego-networks* (personal networks), i.e. from the macro level to the micro level of network analysis.

1.5.1 The global structure of networks

As discussed above, a (social) network can be represented as a graph $G = (V, E)$ where V denotes the finite set of vertices and E denotes a finite set of edges such that $E \subseteq V \times V$. Recall that each graph can be associated with its characteristic

matrix $M := (m_{i,j})_{n \quad n}$ where $n = |V|$, $m_{i,j} = \begin{cases} 1 & (v_i, v_j) \in E \\ 0 & otherwise \end{cases}$. Some network analysis methods are easier to understand when we conceptualize graphs as matrices (see Figure 2.2). Note that the matrix is symmetrical in case the edges are undirected. We will talk of a valued graph when we are also given a real valued weight function $w(e)$ defined on the set of edges, i.e. $w(e) := E \times R$. In case of a valued graph, the

matrix is naturally defined as $m_{i,j} = \begin{cases} w(e) & (v_i, v_j) \in E \\ 0 & otherwise \end{cases}$

Loops are not excluded in the above definition, although they rarely in occur in practical social network data sets. (In other words, the main diagonal of the matrix is usually empty.) Typically, we also assume that the network is connected, i.e. there is a single *(weak) component* in the graph. Otherwise we choose only one of the components for analysis.

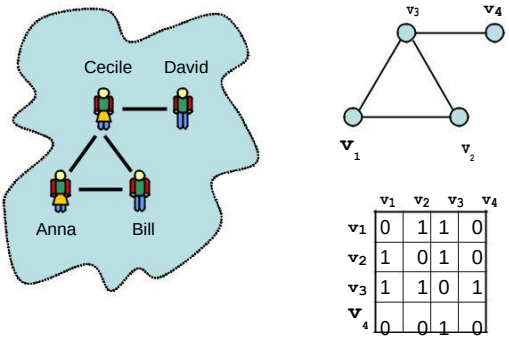


Figure 1.5.1.1. Most network analysis methods work on an abstract, graph based representation of real world networks.

Further, understanding the global structure of networks can lead us to discover commonly occurring patterns of relationships or typical network positions

Formally, what Milgram estimated is the size of the average shortest path of the network, which is also called *characteristic path length*. An open (simple) path in a graph is a sequence of vertices $v_{i0}, v_{i2}, \ldots, v_{in}$ such that $\forall j = 0 \ldots n - 1 (v_{ij}, v_{ij+1}) \in E$ and $\forall j, k = v_{ij} = v_{ik}$, in other words every vertex is connected to the next vertex and no vertex is repeated on the path.[7] The shortest path between two vertices v_s and v_t is a path that begins at the vertex v_s and ends in the vertex v_t and contains the least possible number of vertices. The shortest path between two vertices is also called a *geodesic*. The longest geodesic in the graph is called the

diameter of the graph: this is the maximum number of steps that is required between any two nodes. The average shortest path is the average of the length of the geodesics between all pairs of vertices in the graph. (This value is not possible to calculate if the graph is not (strongly) connected, i.e. in case there exists a pair of vertices with no path between them.)

A practical impact of Milgram's finding is that we can exclude certain kind of structures as possible models for social networks. The two dimensional lattice model shown in Figure 1.5.1.2, for example, does not have the small world property: for a network of size n the characteristic path length is $2/3 * \sqrt{n}$, which is still too large a number to fit the empirical finding.

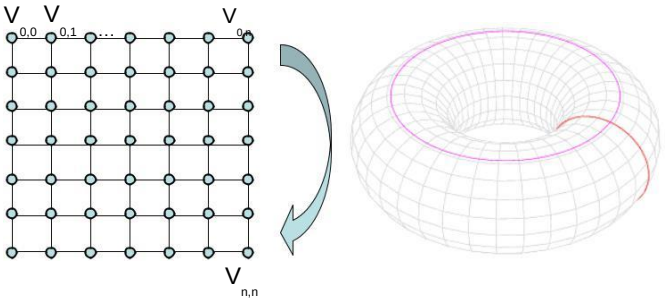


Figure 1.5.1.2. The 2D lattice model of networks (left). By connecting the nodes on the opposite borders of the lattice we get a toroidal lattice (right).

Another simple model could be the tree graph shown in Figure 1.5.1.3. However, a tree is unrealistic because it shows no *clustering*: we all know from practice that our friends are likely to know each other as well because we tend to socialize in groups. (If not for other reasons than other friends know each other because we introduced them to each other.) Clustering for a single vertex can be measured by the actual number of the edges between the neighbors of a vertex divided by the possible number of edges between the neighbors. When taken the average over all vertices we get to the measure known as *clustering coefficient*. The clustering coefficient of a tree is zero, which is easy to see if we consider that there are no triangles of edges (*triads*) in the graph. In a tree, it would never be the case that our friends are friends with each other.

The lattice and the tree also have the rather unappealing characteristic that every node has the same number of connections. We know from our everyday walks in life that some of us have much larger social circles than others. The *random graph* model proposed by the Hungarian mathematicians Erd os˝ and R´enyi offers an alternative. A random graph can be generated by taking a set of vertices with no edges connecting them. Subsequently, edges are added by picking pairs of nodes with equal probability. This way we create a graph where each pair of vertices will be connected with an equal probability. (This probability is a parameter of the process.)

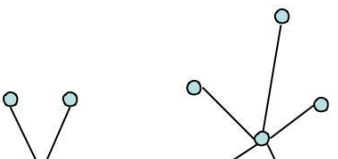


Figure 1.5.1.3 A tree is a connected graph where there are no loops and paths leading from a vertex to itself.

Due to limitations of space if not for other reasons we are unlikely to make friends completely in random from anywhere in the world.

1.5.2 The macro-structure of social networks

Based on the findings about the global characteristics of social networks we now have a good impression about what they might look like. In particular, the image that emerges is one of dense clusters or social groups sparsely connected to each other by a few ties as shown in Figure 1.5.2.1. (These weak ties have a special significance as we will see in the following Section.) For example, this is the image that appears if we investigate the co-authorship networks of a scientific community.

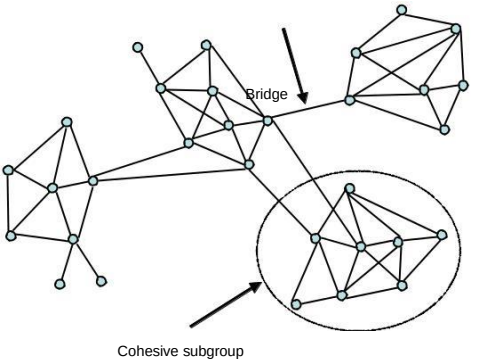


Figure 1.5.2.1. Most real world networks show a structure where densely connected subgroups are linked together by relatively few bridges

Network visualizations based on topographic or physical principles can be helpful in understanding the group structure of social networks and pinpoint hubs that naturally tend to gravitate toward the center of the visualization. Unfortunately, computer generated network visualizations rarely show the kind of image seen in Figure 1.5.2.1. Displays based on multi-dimensional scaling, for example, attempt to optimize the visualization in a way that distances on the paper correlate with the distances between the nodes in the graph. However, with as few as four nodes it is easy to construct an example where there is no optimal solution to the placement problem. In general, the more dense the graph and the fewer the dimensions of the visualization the more likely

the graph will degenerate into a meaningless "spaghetti bowl" tangle of nodes and edges.

Several definitions are based on the observations that subgroups are densely connected and their members are close to each other in the graph. For example, a *clique* in a graph is maximal complete subgraph of three or more nodes. As complete sub-graphs are very rare, the definition of a clique is typically relaxed by allowing some missing connections. For example, a *k-plex* is a maximal subgraph in which each node is adjacent to no fewer than $g_S - k$ nodes in the subgraph, where g_S is the number of nodes in the subgraph. The larger we set the parameter k, the larger the k-plexes that we will find. Other definitions constrain subgroups by putting limits on the maximum path length between the members.

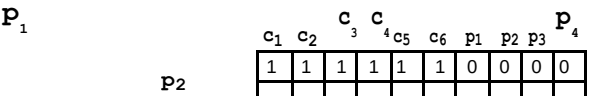
Yet another way of defining cohesiveness is to compare the density of ties within the group to the density of ties between members of the subgroup and the outside. The lambda-set analysis method we will use in our work is based on the definition of edge connectivity. Denoted with the symbol $\lambda(i, j)$, the edge connectivity of two vertices v_i and v_j is the minimum number of lines that need to be removed from a graph in order to leave no path between the two vertices. A lambda-set is then defined as a set of nodes where any pair of nodes from the set has a larger edge connectivity than any pair of nodes where one node is from within the set and the other node is from outside the set. Unlike the above mentioned k-plexes, lambda-sets also have the nice property that they are not overlapping.

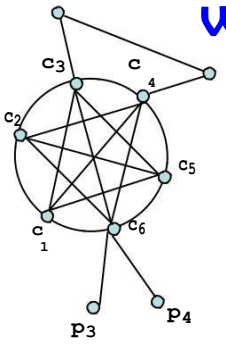
The edge-betweenness clustering method of Mark Newman takes a different approach Instead of focusing on the density of subgroups, this algorithm tar-gets the ties that connect them. The ties that are in between groups can be spotted by calculating their *betweenness*. The betweenness of an edge is calculated by taking the set of all shortest paths in the graph and looking at what fraction of them contains the given edge. An edge between clusters has a much higher betweenness than edges inside clusters because all shortest paths between nodes in the different clusters have to go through the given edge. By progressively removing the edges with the highest betweenness the graph falls apart in distinct clusters of nodes.

Clustering a graph into subgroups allows us to visualize the connectivity at a group level. In some cases we already have an idea of what this macro-structure might look like. A typical pattern that often emerges in social studies is that of a *Core-Periphery (C/P) structure*. A C/P structure is one where nodes can be divided in two distinct subgroups: nodes in the core are densely connected with each other and the nodes on the periphery, while peripheral nodes are not connected with each other, only nodes in the core (see Figure 1.5.2.2). The matrix form of a core periphery

.

block models (structural patterns) work by dividing the set of nodes in a way that the error the between the actual image and the "perfect" image is minimal. The result of the optimization is a classification of the nodes as core or periphery and a measure of the error of the solution.





1	1	1	1	1	1	0	0	0	0
1	1	1	1	1	1	1	0	0	0
1	1	1	1	1	1	0	1	0	0
1	1	1	1	1	1	0	0	0	0
1	1	1	1	1	1	0	0	1	1
0	0	1	0	0	0	0	1	0	0
0	0	0	1	0	0	1	0	0	0
0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	1	0	0	0	0

Figure 1.5.2.2. A Core-Periphery structure that would be perfect without the edge between nodes p_1 and p_2.

We will also encounter situations in our work where it is some additional information that allows us to group our nodes into categories. For example, in the case of a scientific community we might have data on the interests or affiliations of re-searchers. When we have such attribute data available we already have a division of our network into clusters based on shared interests or affiliations. These clusters are overlapping depending on whether a single person is allowed to have multiple interests or affiliations.

As it is fairly common to have attribute information on subjects besides the relational data, the study of *affiliation networks* is an important topic in network analysis. Affiliation networks contain information about the relationships between two sets of nodes: a set of subjects and a set of affiliations. An affiliation network can be formally represented as a *bipartite graph*, also known as a *two-mode network*. In general, an n-partite graph or n-mode network is a graph $G = V, E$ where there exists a partitioning $V = \bigcup_{i=1}^{n} V_i$ such that $\bigcap_{i=1}^{n} V_i = 0$ and $(V_i \times V_i) \cap E = 0$. In other words, the set of n vertices is divided into disjoint sets and there are no edges between vertices belonging to the same set.

There are relative few methods of analysis that are specific to affiliation networks; when dealing with affiliation networks they are typically transformed directly to a regular, one-mode network. This transformation considers the overlaps between the affiliations as a measure of tie strength between the actors (see Figure 1.5.2.3). For example, we can generate a network among scientists by looking at how many interests they have in common. We would place an edge between two researchers if they have interests in common and would weigh the edge according to the number of shared interests. The analysis of such a one-mode network would follow as usual.

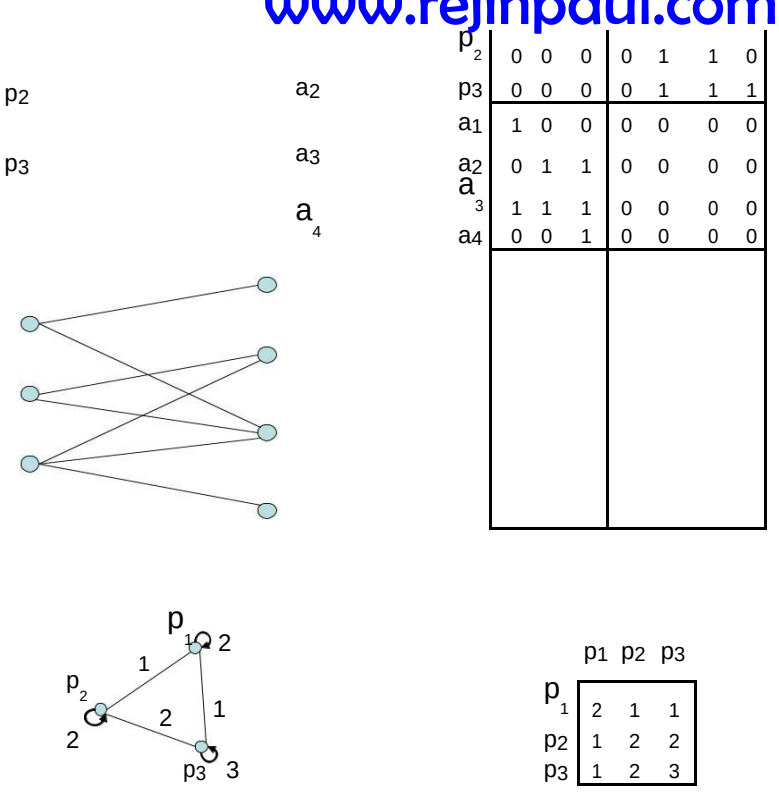


Figure 1.5.2.3 A two-mode network of persons and affiliation (shown as a graph and a matrix above) can be folded into a regular social network by considering the overlaps in affiliations (graph and matrix below).

Interestingly, we can also use this technique to map the relationship between the affiliations themselves. In this case, we create a network of the affiliations as the nodes and draw edges between the nodes based on the number of individuals who share that affiliation. This technique is commonly used, for example, to map *interlocking directorates*, overlaps in the board membership of companies. In such a setting the analysis starts with data on persons and their positions on the boards of various companies. Then a network of companies is drawn based on potential interactions through shared members on their boards. The website TheyRule shows the potential conflicts of interests in the American business, political and media elite by charting the connectivity of individuals and institutions using this technique (see Figure 1.5.3.1.

1.5.3 Personal networks

In many cultures the term *networking* gained a negative connotation as a reference to nepotism, the unfair access to advantages through “friends of friends”.

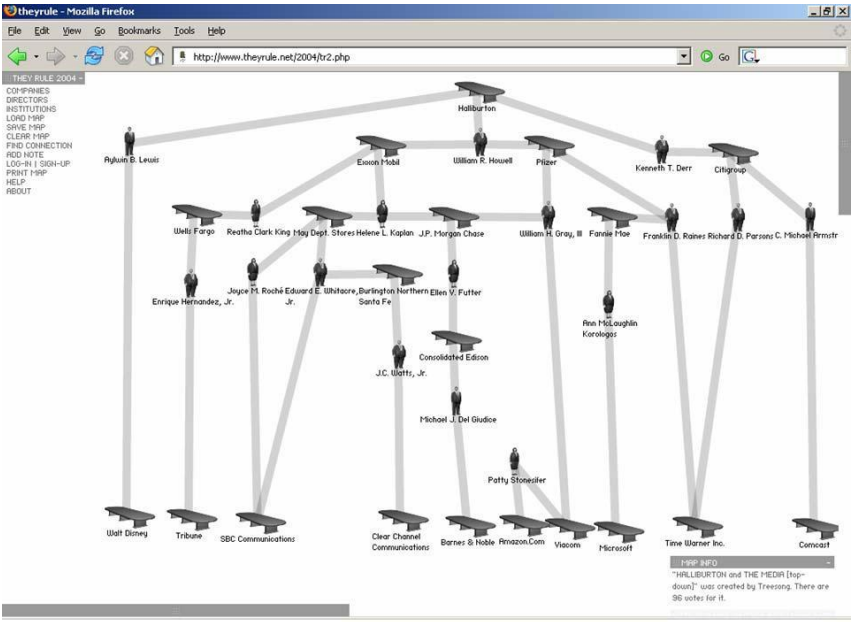


Figure 1.5.3.1 The site *theyrule.net* visualizes the interconnectedness of the American political, business and media elite by showing connections between executive boards of various institutions and their board members. Connections between individuals can be inferred by the boards in which they co-participate.

A person with such access to key players will be increasingly sought after by other individuals. Conversely, the absence of such relations means the lack of access to the intangible resources "flowing" through social networks and to those tangible resources controlled by certain individuals or groups.

In the following we summarize different dimensions of *social capital*
Which they call *intellectual capital*. In particular, they suggest that various aspects of social capital enable the creation of intellectual capital by allowing exchanges to take place that lead to the combination of knowledge (either incrementally or radically) possessed by individuals.

They identify three main dimensions: the structural, relational and cognitive dimensions of social cap-ital. Common to these dimensions is that they constitute some aspect of the social structure and that they facilitate actions of individuals.

The **structural dimension** of social capital refers to patterns of relationships or positions that provide benefits in terms of accessing large, important parts of the network. Common to structural measures of social capital is that they put a single node (the ego) in the center and provide a measure for the node based on his connectivity to other nodes (the alters).

A simple, but effective measure is the *degree centrality* of the node. Degree centrality equals the graph theoretic measure of degree, i.e. the number of (incoming, outgoing or all) links of a node.

A second, more intuitive measure of centrality is *closeness centrality*, which is obtained by calculating the average (geodesic) distance of a node to all other nodes in the network.

Two other measures of power and influence through networks are related to the similar advantages of *broker positions* and *weak ties*.

The measure of *betweenness centrality* identifies broker positions by looking at the extent to which other parties have to go through a given actor to conduct their dealings. Consequently, betweenness is defined as the proportion of paths among the geodesics between all pairs of nodes that pass through a given actor. (Betweenness centrality is measured in a similar way as edgebetweenness, but it is a measure of a node, not of an edge.) As with closeness centrality it is often desirable to compute betweenness in a fixed neighborhood of the ego.

The more complex measure of Ronald Burt is related to the idea of *structural holes*. A structural hole occurs in the space that exists between closely clustered communities.

Further, there is evidence that these are extreme views: overly dense networks can lead to *overembeddedness*. On the other hand, structural embeddedness is necessarily present as through transitivity our friends are likely to develop ties of their own over time.

The aspect of tie strength is an example of the **relational dimension** of social capital, which concerns the kind of personal relationships that people have developed with each other through a history of interaction.

Less attention is devoted in the literature of social capital to the last, **cognitive dimension** of the Nahapiet-Goshal framework. The cognitive dimension of social capital refers to those resources providing shared representations, interpretations and systems of meaning. In particular, cognitive ties are based on the existence of shared languages, signs and narratives, which facilitate the exchange of knowledge. However, excessive cognitive similarity (associated with strong ties) is likely to lead to cognitive over embeddedness.

case	var_1	var_2	...	var_n	var_O
Anna	0.92	0.23	...	0.37	3.2
Bill	1.73	...			...
Cecile	...				
David					

$$\lambda_1 var_1 + \cdots + \lambda_n var_n + c = var_O + \varepsilon$$

Figure 1.5.3. In a cases-by-variables analysis we fit a linear equation (below) to the model described by the table containing the values of the variables for each case.

1.6. Electronic sources for network analysis

1.6.1 Electronic discussion networks

One of the foremost studies to illustrate the versatility of electronic data is a series of works from the Information Dynamics Labs of Hewlett-Packard.

It includes email, group communication and local search. Group communication and collective decision taking in various settings are traditionally studied using much more limited written information such as transcripts and records of attendance and voting, see e.g. As in the case with emails Gloor uses the headers of messages to automatically recreate the discussion networks of the working group. The main technical contribution of Gloor is a dynamic visualization of the discussion network that allows to quickly identify the moments when key discussions take place that activate the entire group and not just a few select members. Gloor also performs a comparative study across the various groups based on the structures that emerge over time.

Although it has not been part of this work, it would be even possible to extend such studies with an analysis of the role of networks in the decision making process as voting records that are also available in electronic formats. Further, by applying emotion mining techniques from AI to the contents of the email messages one could recover agreements and disagreements among committee members. Marking up the data set manually with this kind of information is almost impossible: a single working group produces over ten thousand emails during the course of its work.

1.6.2 Blogs and online communities

Content analysis has also been the most commonly used tool in the computer-aided analysis of blogs (web logs), primarily with the intention of trend analysis for the purposes of marketing. While blogs are often considered as “personal publishing” or a “digital diary”, bloggers themselves know that blogs are much more than that: modern blogging tools allow to easily comment and react to the comments of other bloggers, resulting in webs of communication among bloggers. These discussion networks also lead to the establishment of dynamic communities, which often manifest themselves through syndicated blogs (aggregated blogs that collect posts from a set of authors blogging on similar topics), blog rolls (lists of discussion partners on a personal blog) and even result in real world meetings such as the Blog Walk series of meetings. Figure 1.6.2 shows some of the features of blogs that have been used in various studies to establish the networks of bloggers.

Figure 1.6.2. Features of blogs that can be used for social network extraction. Note also that —unlike web pages in general— blog entries are timestamped, which allows to study network dynamics, e.g. the spread of information in online communities.

Blogs make a particularly appealing research target due to the availability of structured electronic data in the form of RSS (Rich Site Summary) feeds. RSS feeds contain the text of the blog posts as well as valuable metadata such as the timestamp of posts, which is the basis of dynamic analysis. For example, Kumar et al. and Gruhl et al. study information diffusion in blogs based on this information. The early work of Efimova and Anjewierden also stands out in that they were among the first to study blogs from a communication perspective. Adar and Adamic offer a visualization of such communication in blogs.

The 2004 US election campaign represented a turning point in blog research as it has been the first major electoral contest where blogs have been exploited as a method of building networks among individual activists and supporters (see for example). Blog analysis has suddenly shed its image as relevant only to marketers interested in understanding product choices of young demographics; following this campaign there has been explosion in research on the capacity of web logs for creating and maintaining stable, long distance social networks of different kinds. Since 2004, blog networks have been the object of study for a number of papers in the blog research track of the yearly Sunbelt social networks conference.

1.7 Web-based networks

The content of Web pages is the most inexhaustible source of information for social network analysis. This content is not only vast, diverse and free to access but also in many cases more up to date than any specialized database. On the downside, the quality of information varies significantly and reusing it for network analysis poses significant technical challenges. Further, while web content is freely accessible in principle, in practice web mining requires efficient search that at the moment only commercial search engines provide.

There are two features of web pages that are considered as the basis of extracting social relations: links and co-occurrences (see Figure1.7.1). The linking structure of the Web is considered as proxy for real world relationships as links are chosen by the author of the page and connect to other information sources that are considered authoritative and relevant enough to be mentioned. The biggest drawback of this approach is that such direct links between personal pages are very sparse: due to the increasing size of the Web searching has taken over browsing as the primary mode of navigation on the Web. As a result, most individuals put little effort in creating

new links and updating link targets or have given up linking to other personal pages altogether.

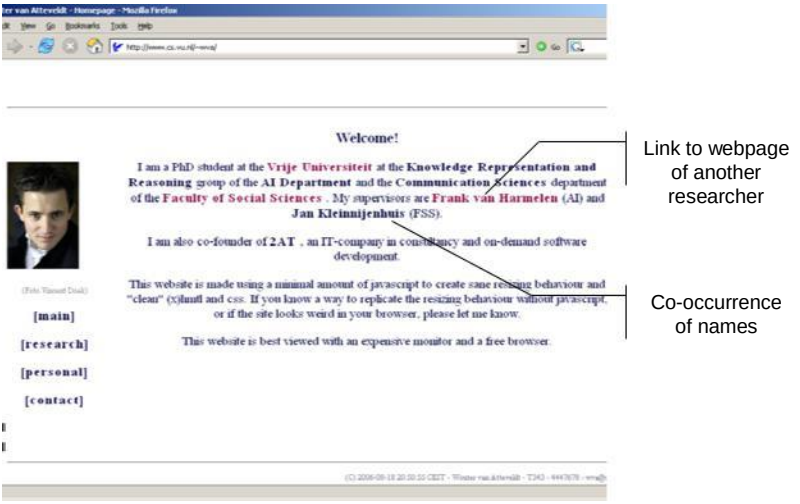


Figure 1.7.1 Features in web pages that can be used for social network extraction.

The data for this analysis comes from bibliographic records, project databases and hyperlink networks. The connections for the latter are collected by crawling the websites of the institutions involved. In principle it could be possible to extract more fine-grained networks from the homepages of the individual researchers. However, links between homepages are too sparse to be analyzed on their own and automating this task would also require solving what is known as the home page search problem: locating the homepage of individuals given their name and description.

Co-occurrences of names in web pages can also be taken as evidence of relationships and are a more frequent phenomenon. On the other hand, extracting relationships based on co-occurrence of the names of individuals or institutions requires web mining as names are typically embedded in the natural text of web pages. The techniques employed here are statistical methods possibly combined with an analysis of the contents of web pages.

Tie strength was calculated by dividing the number of co-occurrences with the number of pages returned for the two names individually (see Figure 1.7.2). Also known as the *Jaccard-coefficient*, this is basically the ratio of the sizes of two sets: the intersection of the sets of pages and their union. The resulting value of tie strength is a number between zero (no co-occurrences) and one (no separate mentioning, only

co-occurrences). If this number has exceeded a certain fixed threshold it was taken as evidence for the existence of a tie.

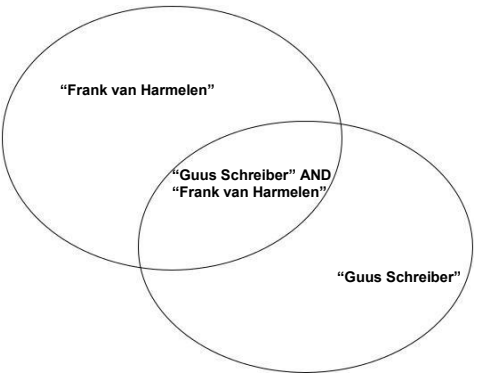


Figure 1.7.2. The Jaccard-coefficient is the ratio of the intersection and the union of two sets. In the case of co-occurrence analysis the two sets contain the pages where the individual names occur. The intersection is formed by the pages where both names appear.

The expertise of individuals was extracted by looking for capitalized phrases that appeared in documents returned by the search engine that were not proper names. The network in the system has grown two ways. Firstly, the documents from the Web were searched for new names using *proper name extraction*, a fairly reliable NLP technique. These names were then used to extract new names, a process that was repeated two or three times. (Note that this is similar to the *snowballing technique* of network analysis where the network under investigation is growing through new names generated by participants in the study.) Second, users of the system were also allowed to register themselves.

We also experiment with different measures of co-occurrence. A disadvantage of the Jaccard-coefficient is that it penalizes ties between an individual whose name often occurs on the Web and less popular individuals (see Figure 1.7.3). In the science domain this makes it hard to detect, for example, the ties between famous professors and their PhD students. In this case while the name of the professor is likely to occur on a large percentage of the pages of where the name of the PhD student occurs but not vice versa. For this reason we use an asymmetric variant of the coefficient. In particular, we divide the number of pages for the individual with the number of pages for both names and take it as evidence of a directed tie if this number reaches a certain threshold. We experiment with choosing an appropriate value for this threshold and the threshold for tie strength.

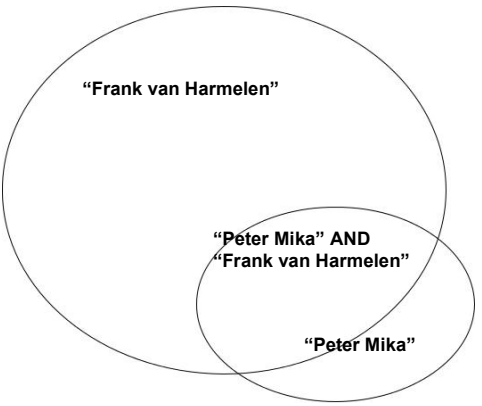


Figure 1.7.3. The Jaccard-coefficient does not show a correlation in cases where there is a significant difference in the sizes of the two sets such as in the case of a student and a professor.

Second, we associate researchers with topics in a slightly different way. In our study of the Semantic Web community, the task is to associate scientists with re-search topics that have been collected manually from the proceedings of ISWC conference series. The system calculates the strength of association between the names of a given person and a certain topic. This strength is determined by taking the number of the pages where the name of an interest and the name of a person co-occur divided by the total number of pages about the person. We assign the expertise to an individual if this value is at least one standard deviation higher than the mean of the values obtained for the same concept.

Kautz already notes that the biggest technical challenge in social network mining is the disambiguation of person names. Person names exhibit the same problems of polysemy and synonymy that we have seen in the general case of web search. Queries for researchers who commonly use different variations of their name (e.g. *Jim Hendler* vs. *James Hendler*) or whose names contain international characters (e.g. *Jer´omeˆ Euzenat*) may return only a partial set of all relevant documents known to the search engine. Queries for persons with common names such as *Martin Frank* or *Li Ding* return pages about all persons with the same name. Another problem is that the coverage of the Web can be much skewed: for example, *George Bush* the president is over-represented compared to *George Bush* the beer brewer.

There have been several approaches to deal with name ambiguity. Bekkerman and McCallum deal with this problem by using limited background knowledge

We also experiment with a second method based on the concept of *average precision*. When computing the weight of a directed link between two persons we consider an ordered list of pages for the first person and a set of pages for the second (the relevant set) as shown in Figure 1.7.4. In practice, we ask the search engine for the top N pages for both persons but in the case of the second person the order is irrelevant for the computation. Let's define $rel(n)$ as the relevance at position n, where $rel(n)$ is 1 if the document at position n is the relevant set and zero otherwise ($1 \le n \le N$). Let $P(n)$ denote the precision at position n (also known as $p@n$):

$$P(n) = \frac{\sum_{r=1}^{n} rel(r)}{n}$$

Average precision is defined as the average of the precision at all relevant positions:

$$P_{ave} = \frac{\sum_{r=1}^{N} (P(r) * rel(r))}{N}$$

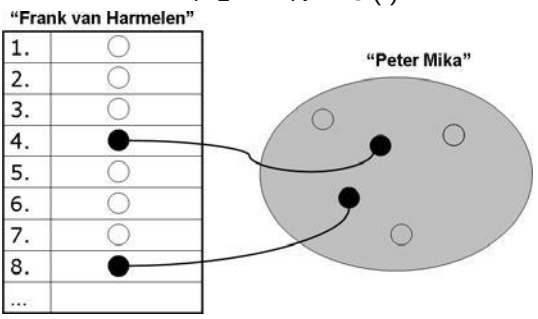


Figure 1.7.4. The average precision method considers also the position of the pages related to a second person in the list of results for the first person.

1.8. Applications of SNA

1.8.1. Flink: the social networks of the Semantic Web community

Flink has been the first system that exploits semantic technologies for the purposes of network analysis based on heterogeneous knowledge.

The information sources are largely the natural byproducts of the daily work of a community: HTML pages on the Web about people and events, emails and publications. From these sources Flink extracts knowledge about the social networks of the community and consolidates what is learned using a common semantic representation, namely the FOAF ontology.

The *raison d'etre˘* of Flink can be summarized in three points. First, Flink is a demonstration of the latest Semantic Web technology. In this respect, Flink is interesting to all those who are planning to develop systems using Semantic Web technology for similar or different purposes. Second, Flink is intended as a portal for anyone who is

interested to learn about the work of the Semantic Web community, as represented by the profiles, emails, publications and statistics. Hopefully Flink will also contribute to bootstrapping the nascent FOAF-web by allowing the export of the knowledge in FOAF format. This can be taken by the researchers as a starting point in setting up their own profiles, thereby contributing to the portal as well. Lastly, but perhaps most importantly, the data collected by Flink is used for the purposes of social network analysis, in particular learning about the nature of power and innovativeness in scientific communities.

The features of Flink

Flink takes a network perspective on the Semantic Web community, which means that the navigation of the website is organized around the social network of researchers. Once the user has selected a starting point for the navigation, the system returns a summary page of the selected researcher, which includes profile information as well as links to other researchers that the given person might know. The immediate neighborhood of the social network (the ego-network of the researcher) is also visualized in a graphical form (see Figure 1.8.1.1).

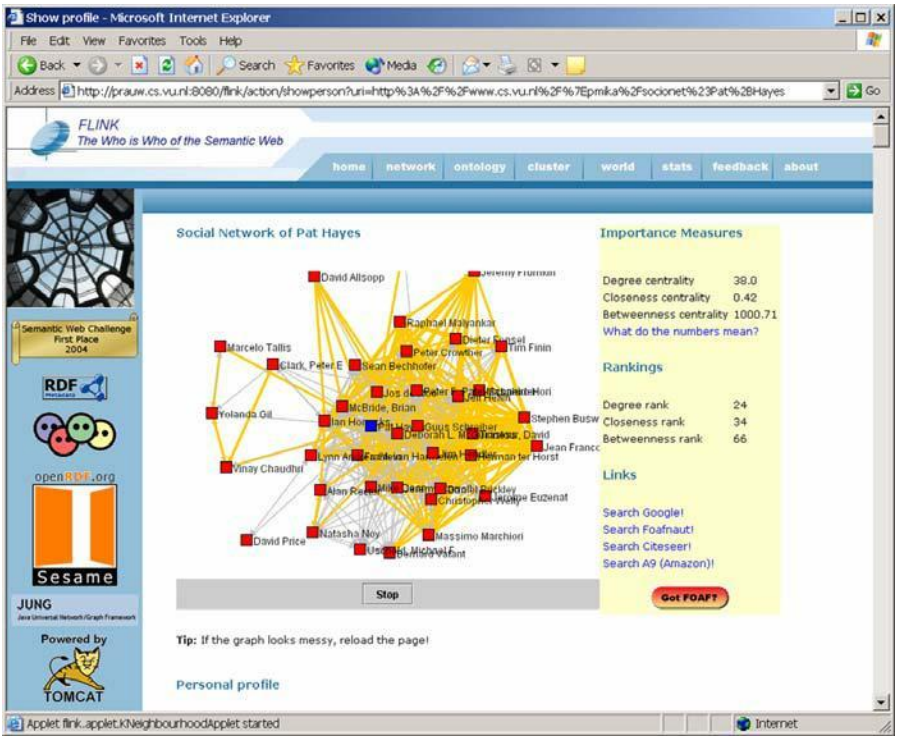


Figure 1.8.1.1. The profile of a researcher in Flink. Individual statistics (rankings) are shown on the right.

The profile information and the social network is based on the analysis of web-pages, emails, publications and self-created profiles. (See the following Section for the technical details.) The displayed information includes the name, email, home-page, image, affiliation and geographic location of the researcher, as well as his interests, participation at Semantic Web related conferences, emails sent to public mailing lists and publications written on the topic of the Semantic Web. The full text of emails and publications can be accessed by following external links. At the time of writing, the system contained information about 7269 publications authored by members of the community and 10178 messages sent via Semantic Web-related mailing lists.

The navigation from a profile can also proceed by clicking on the names of co-authors, addressees or others listed as known by this researcher. In this case, a separate page shows a summary of the relationship between the two researchers, in particular the evidence that the system has collected about the existence of this relationship. This includes the weight of the link, the physical distance, friends, interests and depictions in common as well as emails sent between the researchers and publications written together. The information about the interests of researchers is also used to generate a lightweight ontology of the Semantic Web community. The concepts of this ontology are research topics, while the associations between the topics are based on the number of researchers who have an interest in the given pair of topics (see Figure 1.8.1.2.

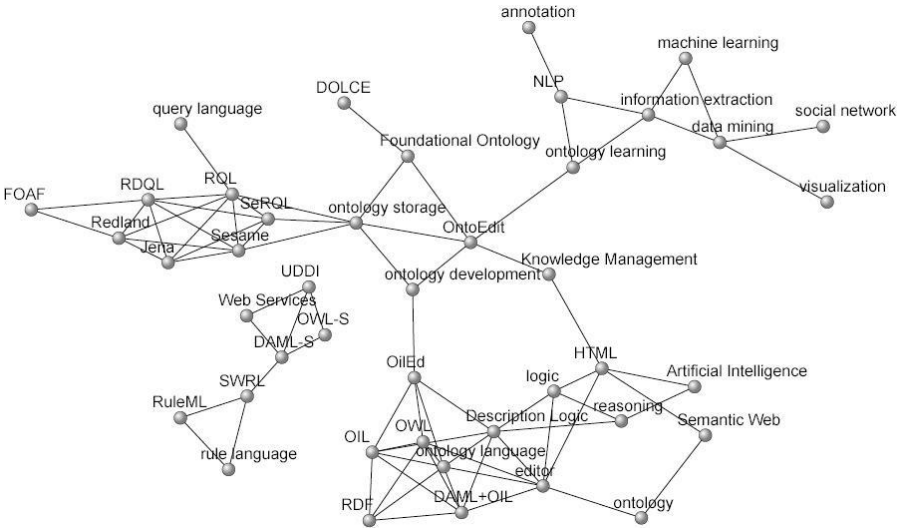


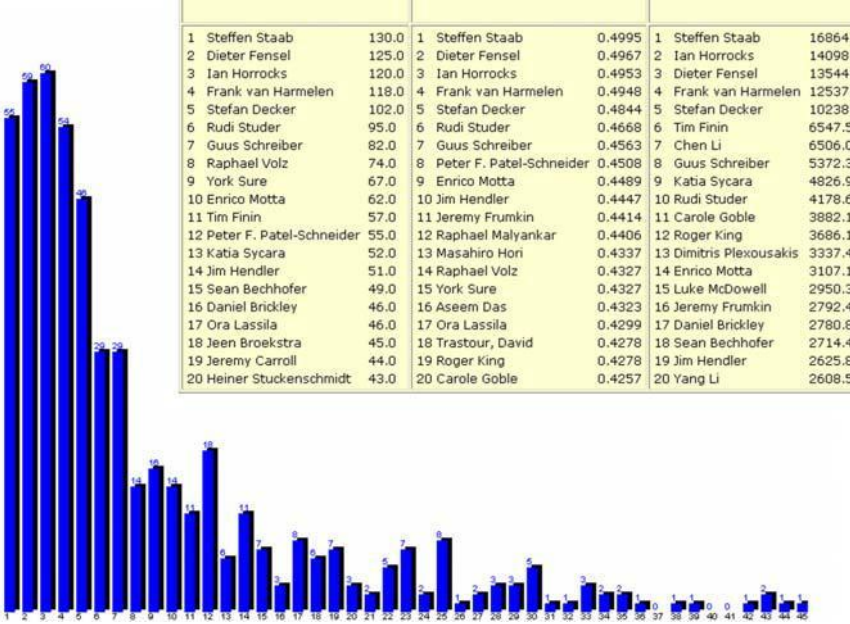
Figure 1.8.1.2. A visualization of the associations between research topics within the Semantic Web community.

System design

The architecture of Flink can be divided in three layers concerned with metadata acquisition, storage and presentation, respectively.

Information collection

This layer of the system concerns the acquisition of metadata. Flink uses four different types of knowledge sources: HTML pages from the web, FOAF profiles from the Semantic Web, public collections of emails and bibliographic data. Information



Indegree centrality		Closeness Centrality		Betweenness Centrality	
1 Steffen Staab	130.0	1 Steffen Staab	0.4995	1 Steffen Staab	16864.4403
2 Dieter Fensel	125.0	2 Dieter Fensel	0.4967	2 Ian Horrocks	14098.6831
3 Ian Horrocks	120.0	3 Ian Horrocks	0.4953	3 Dieter Fensel	13544.063
4 Frank van Harmelen	118.0	4 Frank van Harmelen	0.4948	4 Frank van Harmelen	12537.5246
5 Stefan Decker	102.0	5 Stefan Decker	0.4844	5 Stefan Decker	10238.5444
6 Rudi Studer	95.0	6 Rudi Studer	0.4668	6 Tim Finin	6547.5342
7 Guus Schreiber	82.0	7 Guus Schreiber	0.4563	7 Chen Li	6506.0598
8 Raphael Volz	74.0	8 Peter F. Patel-Schneider	0.4508	8 Guus Schreiber	5372.3001
9 York Sure	67.0	9 Enrico Motta	0.4489	9 Katia Sycara	4826.9771
10 Enrico Motta	62.0	10 Jim Hendler	0.4447	10 Rudi Studer	4178.6547
11 Tim Finin	57.0	11 Jeremy Frumkin	0.4414	11 Carole Goble	3882.1959
12 Peter F. Patel-Schneider	55.0	12 Raphael Malyankar	0.4406	12 Roger King	3686.1388
13 Katia Sycara	52.0	13 Masahiro Hori	0.4337	13 Dimitris Plexousakis	3337.4544
14 Jim Hendler	51.0	14 Raphael Volz	0.4327	14 Enrico Motta	3107.1791
15 Sean Bechhofer	49.0	15 York Sure	0.4327	15 Luke McDowell	2950.3047
16 Daniel Brickley	46.0	16 Aseem Das	0.4323	16 Jeremy Frumkin	2792.4088
17 Ora Lassila	46.0	17 Ora Lassila	0.4299	17 Daniel Brickley	2780.8619
18 Jeen Broekstra	45.0	18 Trastour, David	0.4278	18 Sean Bechhofer	2714.4584
19 Jeremy Carroll	44.0	19 Roger King	0.4278	19 Jim Hendler	2625.8347
20 Heiner Stuckenschmidt	43.0	20 Carole Goble	0.4257	20 Yang Li	2608.5416

Figure 1.8.1.3. A visualization of the degree distribution of the network. The inset shows the rankings of researchers according to various network measures.

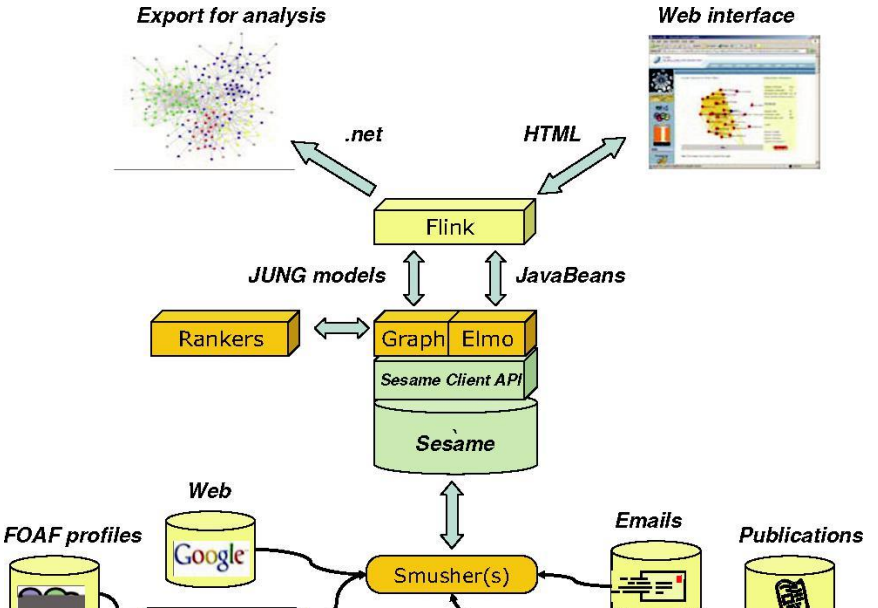


Figure 1.8.1.4. The architecture of Flink.

from the different sources is collected in different ways but all the knowledge that is learned is represented according to the same ontology (see the following Section). This ontology includes FOAF and minimal extensions required to represent additional information.

The web mining component also performs the additional task of finding topic interests, i.e. associating researchers with certain areas of research. The network ties, the interest associations and other metadata are represented in RDF using terms from the FOAF vocabulary such as *foaf* :*knows* for relationships and *foaf* :*topic interest* for research interests.

Information from emails is processed in two steps. The first step requires that the emails are downloaded from a POP3 or IMAP store and the relevant header information is captured in an RDF format, where FOAF is used for representing information about senders and receivers of emails, in particular their name (as appears in the header) and email address. (There is no common ontology for representing emails, so the rest of the header information is represented using a custom ontology.) The second step is smushing: matching the Person instances found in the email collection with the instances describing our target community.

Lastly, bibliographic information is collected in a single step by querying Google Scholar with the names of individuals (plus the disambiguation term). From the results we learn the title and URL of publications as well as the year of publication and the number of citations where available.

Publication metadata is represented using the "Semantic Web Research Community" (SWRC) ontology The SWRC ontology maps the types and fields of the BibTeX bibliographic description format in the most straightforward manner to RDF classes and properties. For example, the BibTeX item type InBook has the equivalent class of *swrc*:*InBook* in the ontology, with common properties such as *swrc*:*year* for the year of publication.

An alternative source of bibliographic information (used in previous versions of the system) is the Bibster peer-to-peer network, from which metadata can be exported directly in the SWRC ontology format.

Storage and aggregation

This is the middle layer of our system with the primary role of storing and aggregating metadata. In our case ontology mapping is a straightforward task: the schemas used are small, stable, lightweight web ontologies (SWRC and FOAF). Their mapping cause little problem: such mappings are static and can be manually inserted into the knowledge base. An example of such a mapping is the subclass relationship between the *swrc*:*Person* and *foaf* :*Person* classes or the subproperty relationship between *swrc*:*name* and *foaf* :*name*. Note that at the moment we throw away all the data we find through crawling that is not in one of the two ontologies. Incorporating knowledge in unknown schemas would require automated ontology mapping.

The aggregated collection of RDF data is stored in a Sesame server. Note that since the model is a compatible extension of FOAF, from this point the knowledge can be further processed by any FOAF-compatible tool, e.g the FOAF explorer. Another example is the generic component we implemented for finding the geographical locations (latitude, longitude coordinates) of place names found in the FOAF profiles. This component invokes the ESRI Place Finder Sample Web Service, which provides geographic locations of over three-million place names worldwide.

From a scalability perspective, the Sesame server offers very high performance in storing data on the scale of millions of triples, especially using native repositories or in memory storage. Speed of upload is particularly important for the RDF crawler, which itself has a very high throughput. Unfortunately, the speed of upload drops significantly when custom rules need to be evaluated.

Besides aggregation, we also use reasoning to enrich the data. For example, we infer *foaf* :*knows* relations between the senders and recipients of emails and the co-authors of publications.

Lastly, at this stage we pre-compute and store the kind of network statistics displayed in the interface.

User interface

The user interface of Flink is a pure Java web application based on the Model-View-Controller (MVC) paradigm. The key idea behind the MVC pattern is a separation of concerns among the components responsible for the data (the model), the application logic (controller) and the web interface (view). The Apache Struts Framework used by Flink helps programmers in writing web applications that respect the MVC pattern by providing abstract application components and logic for the pattern. The role of the programmer is to extend this skeletal application with domain and task specific objects.

The model objects of Flink are Elmo beans representing persons, publications, emails etc. When requests reach the controller, all the beans that are necessary to generate the page are retrieved from the store and passed on to the view layer. The GraphUtil utility is used again to read the social network from the repository, which is also handed over to the visualization. (Much like the Elmo beans the network is also kept in memory to improve performance.)

In the view layer, servlets, JavaServer Pages (JSP) and the Java Standard Tag Library (JSTL) are used to generate a front-end that hides much of the code from the designer of the front-end

In the current interface, Java applets are also used on parts of the site for interactive visualization of social networks. These applets communicate with a servlet that retrieves the part of the network to be visualized from the repository and sends back a serialized form to the applet, which then computes the layout. The user can pan and zoom the image as required.

1.8.2. Open academia: distributed, semantic-based publication management

Information about scientific publications is often maintained by individual researchers. Reference management software such as EndNote and BibTeX help researchers to maintain a personal collection of bibliographic references. Most researchers and research groups also have to maintain a web page about publications for interested peers from

other institutes. Typically, personal reference management and the maintenance of web pages is a separate effort: the author of a new publication adds the reference to his own collection, updates his web page and possible that of his research group. From then on it is waiting for other researchers to discover the newly added publication.

The openacademia system removes the unnecessary duplication of effort in-volved in maintaining personal references and webpages. It also solves the problem of creating joined publication lists for webpages at the group or institutional level. At the same time it gives a new way of instantly notifying interested peers of new works instead of waiting for them to visit the web page of the researcher or the institute.

Openacademia is a distributed system on its own. A public openacademia website is available on the Web for general use, i.e. anyone can submit his own publications to this service. Openacademia can also be installed at research groups locally in order to collect and manage the shared publication metadata of the group.

The features of openacademia

The most immediate service of openacademia is the possibility to generate an HTML representation of one's personal collection of publications and publish it on the Web. This requires filling out a single form on the openacademia website, which generates the code (one line of JavaScript!) that needs to be inserted into the body of the home-page. The code inserts the publication list in the page dynamically and thus there is no need to update the page separately if the underlying collection changes (see Figure 1.8.2.1). The appearance of the publication list can be customized by choosing from a variety of stylesheets.

More interestingly, one can also generate an RSS feed from the collection. Adding such an RSS feed to a homepage allows visitors to subscribe to the publication list using any RSS news reader. Whenever a new publication is added, the subscribers of the feed will be notified of this change through their reader (*information push*).

A number of generic tools are available for reading and aggregating RSS information, including browser extensions, online aggregators, news clients and desktop readers for a variety of platforms. Mozilla Firefox also natively supports RSS feeds as the basis for creating dynamic bookmark folders (see Figure 1.8.2.2). These folders refresh their contents from an RSS feed whenever the user opens them.

The RSS feeds of openacademia are RDF-based and can also be consumed by any RDF aware software such as Piggy Bank browser extension. Piggy Bank al-lows users to collect RDF statements linked to Web pages while browsing through the Web and to save them for later use.

Research groups can install their own openacademia server. Members of the re-search group can submit their publications by creating a FOAF profile pointing to the location of their publication collection. What the system provides is the possibility to create a unified group publication list and post it to a website similarly to personal lists. (Needless to say, groups can have RSS feeds as well.)

Figure 1.8.2.1. Dynamically generated publication list inserted to a web page using remote syndication.

There is also an AJAX based interface for browsing and searching the publication collection (see Figure 1.8.2.3). This interface offers a number of visualizations. For example, the important keywords in the titles of publication matching the current query can be viewed as a *tagcloud*, where the size of the tags shows the importance of the keyword. It is also possible to browse the co-authorship networks of researchers us-ing the same interactive applet used by Flink. Another interactive visualization shows publication along a timeline that can be scrolled using the mouse (see Figure 1.8.2.4).

Keywords or tags can be added to publications using the features of BibTeX or EndNote. The system also extracts keywords automatically from titles of publications. Lastly, openacademia connects to blog search engines in order to import blog comments about publications.

System design

The architecture of openacademia follows the same design as Flink: in the middle of the architecture is an RDF repository that is filled with a variety of information sources and queried by a number of services (see Figure 1.8.2.5).

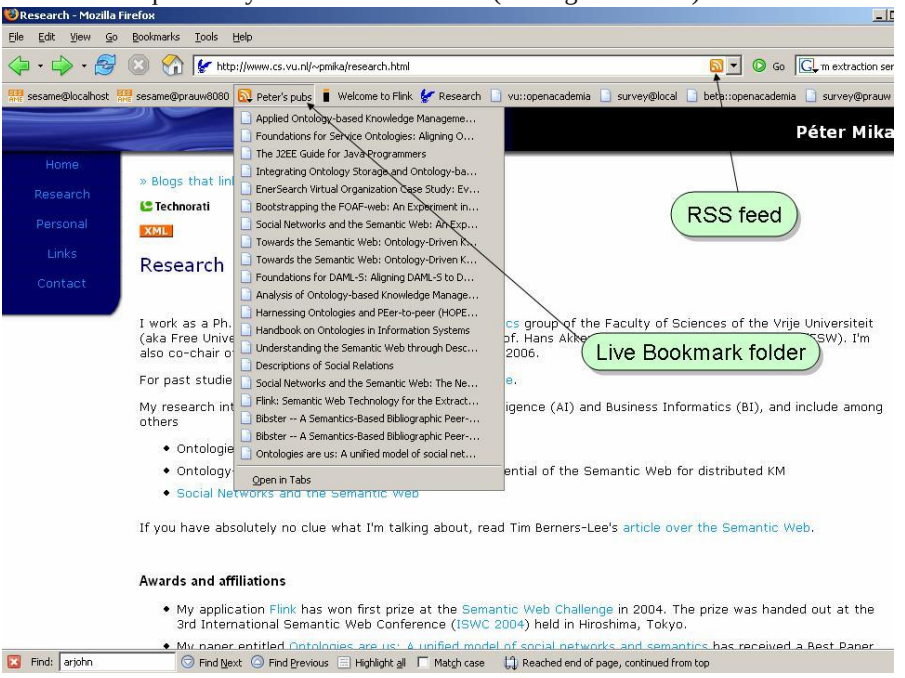


Figure 1.8.2.2. Modern browsers such as Mozilla Firefox have built in RSS feeders that allow to subscribe to RSS feeds attached to webpages. Using the Live Bookmark feature of Firefox it is also possible to save a publication list as a bookmark folder that automatically refreshes itself.

The difference lies in the dynamics of the two systems. Flink is filled with data every two or three months in a semi-automated fashion. Openacademia repositories refresh their content every day automatically.

In case a publication feed is generated from a single BibTeX or EndNote file the entire process of filling and querying the repository is carried out on the fly. In this case we use an in-memory repository that is discarded after the answer has been served.

Information collection

For obtaining metadata about publications, we rely on the BibTeX and EndNote for-mats commonly in use in academia worldwide. We ask authors to include a BibTeX file with their own publications on a publicly accessible part of their website. For most authors in the sciences this does not require additional work, as they typically maintain their own web space and have such a file at hand. Further, researchers fill out a form to create a basic FOAF profile, which contains at least their name and the location of their references. Needless to say, this step can be skipped in case the researcher already has a FOAF profile.

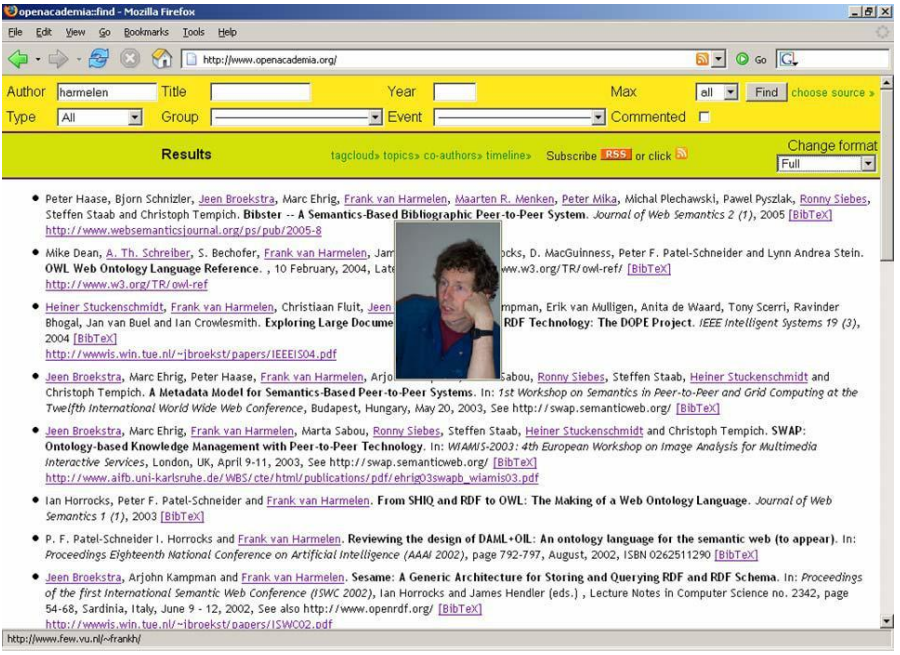


Figure 1.8.2.3. The AJAX-based query interface of openacademia builds queries and displays the results. If the underlying data source includes FOAF data even the researchers' photos are shown (with links to their homepage).

We use the Elmo crawler to collect such profiles. As mentioned before, the crawler can be restricted to a domain, which is useful for limiting the data collection to the domain of an institute. The BibTeX files are translated to RDF using the BibTeX-2-RDF service, which creates instance data for the "Semantic Web Research Community" (SWRC) ontology. A simple extension of the SWRC ontology was necessary to preserve the sequence of authors of publications. To this end we defined the *swrc-ext*:*authorList* and *swrc-ext*:*editorList* properties, which have *rdf*:*Seq* as range, comprising an ordered list of authors.

At the Vrije Universiteit, Amsterdam we have also implemented a service that dynamically queries the local LDAP database and represents the contents as FOAF profiles. Most importantly, the LDAP database contains information about group membership, but it also contains real names and email addresses. We do not reveal the email addresses of employees, but use a hash of the email address as identifier. By relying on the LDAP database for this information we delegate the task of maintaining user data to the existing infrastructure in the department i.e. the user account administration.

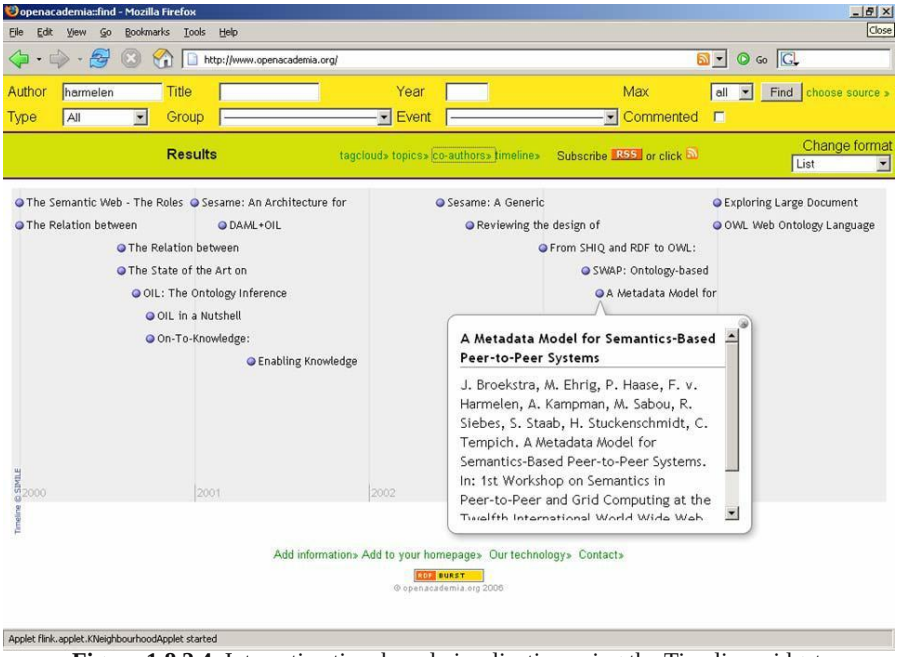


Figure 1.8.2.4. Interactive time based visualization using the Timeline widget.

Storage and aggregation

As our system is designed to reuse information in a distributed, web-based environment, we have to deal with the arising semantic heterogeneity of our information sources. Heterogeneity effects both the schema and instance levels.

Similarly to Flink, the schemas used are stable, lightweight web ontologies (SWRC and FOAF) and their mapping causes no problems. Heterogeneity on the instance level arises from using different identifiers in the sources for denoting the same real world objects. This certainly effects FOAF data collected from the Web (where typically each personal profile also contains partial descriptions of the friends

of the individual), but also publication information, as the same author or publication may be referenced in a number of BibTeX sources.

We use the Elmo smusher framework to match *foaf* :*Person* instances based on name and inverse-functional properties. Publications are matched on a combination of properties. In the current system, we look for an exact match of the date of the publication and a tight fuzzy match of the title. Matching publications based on authors is among the future work.

The instance matches that we find are recorded in the RDF store using the *owl*:*sameAs* property.

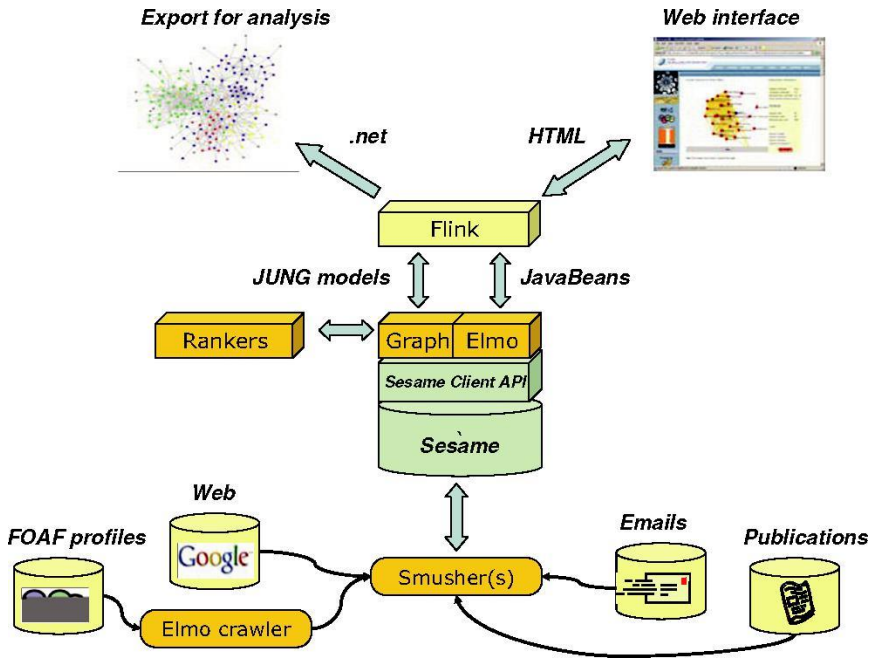
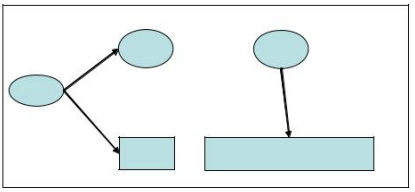


Figure 1.8.2.5. The architecture of openacademia.

Presentation

After all information has been merged, the triple store can be queried to produce publications lists according to a variety of criteria, including personal, group or publication facets. The online interfaces help users to build such queries against the publication repository. The queries are processed by another web-based component, the publication web service.

In order to appreciate the power of a semantics-based approach, it is illustrative to look at an example query. The query in Figure 1.8.2.6, formulated in the SeRQL language, returns all publications authored by the members of the Artificial Intelligence department in 2005. This department is uniquely identified by its homepage. (The *foaf* :*homepage* property is inverse functional according to the FOAF vocabulary, i.e. a certain URL uniquely identifies a given group.)



```
SELECT DISTINCT pub
 FROM {group} foaf:homepage
 {<http://www.cs.vu.nl/ai/>}; foaf:member {person},
 {pub} swrc:year {year}; swrc:author {person}
 WHERE year="2004"
 USING NAMESPACE
   foaf = <http://xmlns.com/foaf/0.1/>,
   swrc = <http://swrc.ontoware.org/ontology#>
```

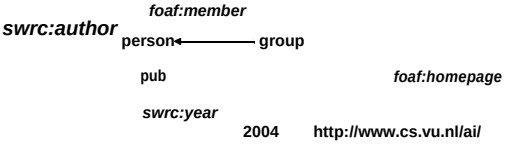


Figure 1.8.2.6. Sample SeRQL query and its graphical representation.

Note first that the successful resolution of this query relies on the schema and instance matching described in the previous section. The query answering requires publication data as well as information on department membership and personal data from either the LDAP database or the self-maintained profile of the researcher.

The publications that match this query necessarily come from different sources, i.e. from the collections of personal publications of individual researchers. Identifying matching publications is the task of the publication smusher introduced before, which is thus also essential to correctly answering this query.

The publish service takes a query like the one shown above, the location of the repository, the properties of the resulting RSS channel and optional style instructions as parameters. In a single step, it queries the repository, performs the post-processing and generates an RSS channel with the publications matching the query.

We attach publication metadata to RSS items us-ing the *burst*:*publication* property. This is the only property defined by the BuRST specification.

The presentation service can also add XSL stylesheet information to the RSS feed, which allows to generate different HTML layouts (tables, short citation lists or

longer descriptions with metadata). The HTML output can be viewed with any XSLT capable browser and it can be tailored even further by adding a custom CSS stylesheet (for changing colors, font styles etc.)

Stylesheets are also used to generate the XML input of the Timeline widget. One can even reproduce a BibTeX or EndNote representation of the publication feed by applying the appropriate stylesheets.

```
<rdf:RDF
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
   xmlns:swrc="http://swrc.ontoware.org/ontology/ontoware#"
   xmlns:foaf="http://xmlns.com/foaf/0.1/"
   xmlns:dc="http://purl.org/dc/elements/1.1/"
   xmlns:burst="http://xmlns.com/burst/0.1/">

 <rss:channel rdf:about="http://www.cs.vu.nl/~pmika/burst.rdf">
  <rss:title>Peter Mika's publications</rss:title>
  <rss:link>http://www.cs.vu.nl/~pmika/research/pub.rdf</rss:link>
  <rss:description>
    Semantic Web related publications authored by Peter Mika.
   </rss:description>
  <rss:items>
    <rdf:Seq>
       <rdf:li rdf:resource="http://www.cs.vu.nl/~pmika/burst#1" />
       <rdf:li rdf:resource="http://www.cs.vu.nl/~pmika/burst#2" />
    </rdf:Seq>
  </rss:items>
  <rdfs:seeAlso rdf:resource="http://www.cs.vu.nl/~mcaklein/pub.rdf"
 /> </rss:channel>

 <rss:item rdf:about="http://www.cs.vu.nl/~pmika/burst#1">
   <rss:title>Bootstrapping the FOAF-Web: An Experiment
            in Social Network Mining
   </rss:title>
   <rss:link>http://www.w3.org/2001/sw/Europe/events/foaf-
   galway/papers/fp/bootstrapping_the_foaf_web/</rss:link>
   <rss:description>
      Bootstrapping is a problem that affects all applications of the
      Semantic Web, including the network of interlinked Friend-of-a-Friend
      (FOAF) profiles known as the FOAF-web. In this paper we introduce a
      hybrid system ...
   </rss:description>
   <dc:subject>Semantic Web</dc:subject>
   <burst:publication>
      <swrc:InProceedings>
         <swrc:title>Bootstrapping the FOAF-Web: An Experiment in Social Network Mining</swrc:title>
         <swrc:author>

           <foaf:Person rdf:ID="PeterMika">
              <foaf:name>Peter Mika</foaf:name>
              <foaf:mbox_sha1sum>ffe33bbe8be2a2123f0adb793e61a6d84ae9a739</foaf:mbox_sha1sum>
              <rdfs:seeAlso rdf:resource="http://www.cs.vu.nl/~pmika/foaf.rdf" />
           </foaf:Person>
         </swrc:author>
         <swrc:year>2004</swrc:year>
      </swrc:InProceedings>
   </burst:publication>
 </rss:item>
 ...
```

Figure 1.8.2.7. Example of a BuRST channel.

UNIT 2 MODELLING, AGGREGATING AND KNOWLEDGE REPRESENTATION

2.1 Ontologies and their role in the Semantic Web

2.1.1 Ontology-based Knowledge Representation

The idea of the Semantic Web is to extend unstructured information with machine processable descriptions of the meaning (semantics) of information and to provide missing background knowledge where required.

The key challenge of the Semantic Web is to ensure a shared interpretation of information. Related information sources should use the same concepts to reference the same real world entities or at least there should be a way to determine if two sources refer to the same entities, but possibly using different vocabularies. Ontologies and ontology languages are the key enabling technology in this respect. An ontology, by its most cited definition in AI, is a shared, formal conceptualization of a domain, i.e. a description of concepts and their relationships. Ontologies are domain models with two special characteristics, which lead to the notion of shared meaning or semantics:

1. Ontologies are expressed in formal languages with a well-defined semantics.
2. Ontologies build upon a shared understanding within a *community*. This understanding represents an agreement among members of the community over the concepts and relationships that are present in a domain and their usage.

RDF and OWL are the languages most commonly used on the Semantic Web, and in fact when using the term ontology many practitioners refer to domain models described in one of these two languages. The second point reminds as that there is no such thing as a “personal ontology”. For example, the schema of a database or a UML class diagram that we have created for the design of our own application is not an ontology. It is a conceptual model of a domain, but it is not shared: there is no commitment toward this schema from anyone else but us.

The simplest structures are glossaries or controlled vocabularies, in essence an agreement on the meaning of a set of terms. An example would be a controlled vocabulary used inside a support center for describing incidents reported. Such a controlled vocabulary facilitates the communication among the helpdesk and the technical staff of a support center as it enables a uniform description of the reported problems.

Semantic networks are essentially graphs that show also how terms are related to each other. Thesauri are richer structures in that they describe a hierarchy between concepts and typically also allow to describe related terms and aliases. Thesauri are also the simplest structures where logic-based reasoning can be applied: the broader-narrower

relationships of these hierarchies are transitive, in that an item that belongs to a narrower category also belongs to its direct parent and all of its ancestors.

In the context of the Semantic Web research, it is often assumed that an ontology at least contains a hierarchy between the concepts (subclass relationships).

The term *lightweight ontology* is typically applied to ontologies that make a distinction between classes, instances and properties, but contain minimal descriptions of them. On the other hand, *heavyweight ontologies* allow to describe more precisely how classes are composed of other classes, and provide a richer set of constructs to constrain how properties can be applied to classes.

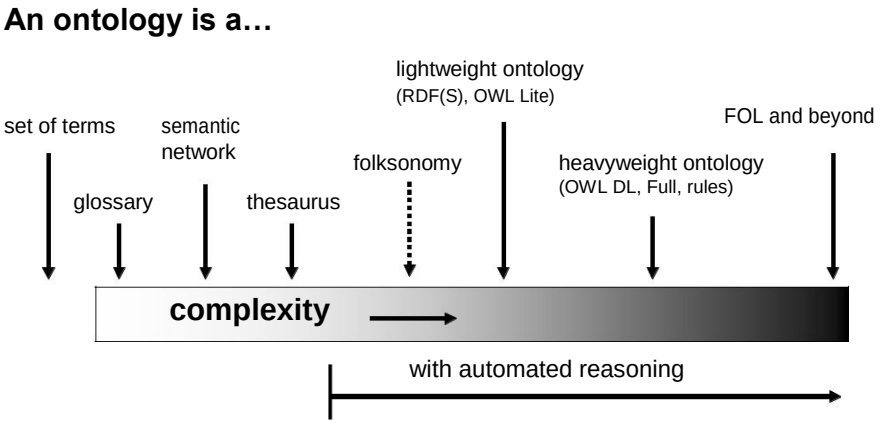


Figure 2.1. Ontologies can be organized according to complexity (informally, the level of semantics).

In practice, the most common Web ontologies are all lightweight ontologies due to the need of serving the needs of many applications with divergent goals. Large, heavyweight ontologies are more commonly found in targeted expert systems used in focused domains with a tradition of formalized processes and vocabularies such as the area of life sciences and engineering.

2.2 Ontology languages for the Semantic Web

2.2.1 The Resource Description Framework (RDF) and RDF Schema

The Resource Description Framework (RDF) was originally created to describe resources on the World Wide Web, hence the name. In reality, RDF is domain-independent and can be used to model both real world objects and information resources. RDF itself is a very primitive modelling language, but it is the basis of more complex languages such as OWL.

There are two kinds of primitives in RDF: resources and literals. The definition of a resource is intentionally vague; in general everything is modelled as a resource that can be identified and described. Resources are either identified by a URI or left blank. URIs are identifiers with a special syntax. Blank resources (*blank nodes*) are the existential quantifiers of the language: they are resources with an identity, but

whose identifier is not known or irrelevant. Literals are strings with optional language and datatype identifiers.

Expressions are formed by making statements (*triples*) of the form (subject, predicate and object). The subject of a statement must be a resource (blank or with a URI), the predicate must be a URI and the object can be either kind of resource or a literal. Literals are thus only allowed at the end of a statement.

RDF is very easy to understand in practice.

RDF document on the Web defining some of the terms that we are using in the example.

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#label> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix example: <http://www.example.org/> .

example:Rembrandt rdf:type foaf:Person .
example:Saskia  rdf:type foaf:Person .
example:Rembrandt foaf:name "Rembrandt" .
example:Rembrandt foaf:mbox <mailto:rembrandt@example.org> .
example:Rembrandt foaf:knows example:Saskia .
example:Saskia  foaf:name "Saskia" .
```

Figure 2.2.2.1. A set of triples describing two persons represented in the Turtle language.

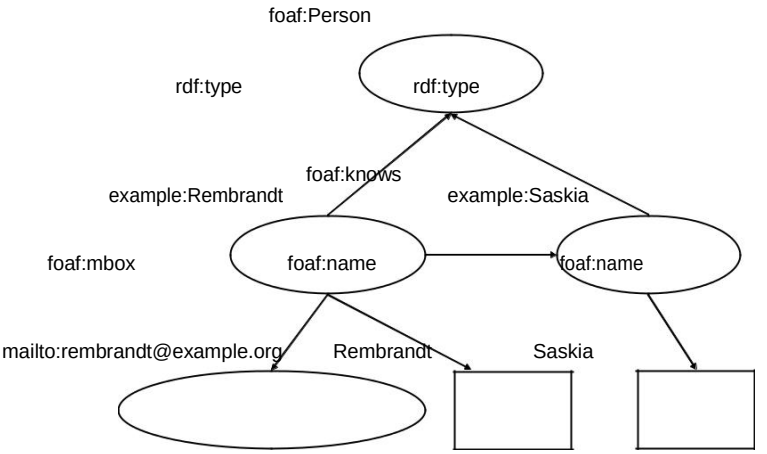


Figure 2.2.1.2. A graph visualization of the RDF document shown in Figure 2.2.1.1.

The RDF language provides the basic term to assign a type to a resource (*rdf :type*) and to declare a resource as a property. It also provides features to describe collections of instances and to make statements about statements (*reification*). The descriptive power of RDF is minimal that in practice it is always used in combination with RDF Schema. RDF Schema is a simple extension of RDF

defining a modelling vocabulary with notions of classes and subclasses. Classes and properties can be connected by specifying the domain and range of properties.

Note that the division of terms between the RDF and RDF Schema namespaces is very unnatural: for example, *rdf* :*type* property appears in the RDF namespace, even though there are no classes in RDF. On the other hand, the *rdfs*:*Literal* class is in the RDF(S) namespace even though literals are a feature of RDF. For this reason and for the limited use of RDF on its own, most people refer to an RDF Schema ontology when talking about an 'RDF ontology' or 'RDF data'. In the following, we will also use the term RDF in this sense and apply the term RDF Schema when we specifically intend to speak about the RDF Schema vocabulary.

Figure 2.2.1.4 shows some of the statements from the FOAF ontology. The first group of statements describe the class Person. The type of the resource is specified as *owl*:*Class*, we give a label, name a superclass and state that this class is disjoint from the class of documents. We then describe the *foaf* :*knows* and *foaf* :*name* properties we have

Basic constructs	*rdfs*:*domain*
	rdfs:*range*
	rdfs:*Resource*
	rdfs:*Literal*
	rdfs:*Datatype*
	rdfs:*Class*
	rdfs:*subClassOf*
	rdfs:*subPropertyOf*
Collections	*rdfs*:*member*
	rdfs:*Container*
	rdfs:*ContainerMembershipProperty*
Documentation & reference	*rdfs*:*comment*
	rdfs:*seeAlso*
	rdfs:*isDefinedBy*
	rdfs:*label*

Table 2.2.1.3. The RDF Schema vocabulary.

used above, specifying their type label, domain, range and a superproperty in the case of *foaf* :*name*.

```
@prefix foaf: <http://xmlns.com/foaf/0.1/>.

foaf:Person rdf:type owl:Class . foaf:Person
rdfs:label "Person" . foaf:Person
rdfs:subClassOf foaf:Agent foaf:Person
owl:disjointWith foaf:Document .

foaf:knows rdf:type owl:ObjectProperty .
foaf:knows rdfs:label "knows" .
foaf:knows rdfs:domain foaf:Person .
foaf:knows rdfs:range foaf:Person .

foaf:name rdf:type owl:DatatypeProperty .
```

```
foaf:name rdfs:label "name" .
foaf:name rdfs:subPropertyOf rdfs:label .
foaf:name rdfs:domain owl:Thing .
foaf:name rdfs:range rdfs:Literal
```

Figure 2.2.1.4. Some statements from the FOAF ontology about the terms used in the previous example.

RDF and the notion of semantics

In the above we have been largely concerned with the syntax of the language: the kind of symbols (resources and literals) we have and the way to form statements from them. Further, we introduced some special resources that are part of the RDF and RDF Schema languages (e.g. *rdfs*:*subClassOf*) and gave their meaning in a colloquial style.

The meaning of RDF(S) constructs is anchored in a model-theoretic semantics, one of the most common ways to provide semantics. Using model-theoretic semantics meaning is defined by establishing a mapping from one model to a meta-model where the truth of propositions is already uniquely determined. In the context of RDF such a mapping is called an *interpretation*.

Thus an interpretation can be thought of as a mapping between symbols and the objects or relations they intended to describe (see Figure 4.6).

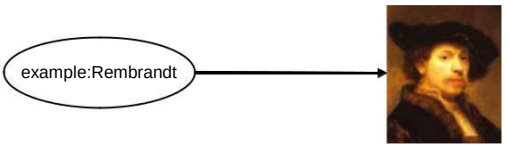
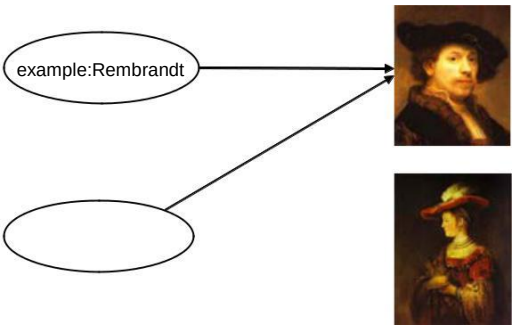


Figure 2.2.1.5. An interpretation is a mapping between the terms of an ontology and an interpre-tation domain.

The constructs of RDF are used to put constraints on possible interpretations and to exclude thereby some of the unintended interpretations. For example, we only want to allow interpretations where symbols for the two persons are mapped to different objects. In other words, we want to exclude interpretations where the two instances are mapped to the same object. In order to achieve this we can specify that the instance Rembrandt is *owl*:*differentFrom* the second instance Saskia. This excludes the interpretation shown in Figure 2.2.1.6.



example:Saskia

Figure 2.2.1.6. An unintended model where symbols for different persons are mapped to the same object. We can exclude this interpretation by adding a statement to the ontology claiming that the two individuals are different from each other. (*owl*:*differentFrom*)

The model theory of RDF(S) provides some axiomatic triples that are true in every RDF(S) interpretation (for example, *rdf* :*Property rdf* :*type rdfs*:*Class*) and de-fines the constraints that language elements put on possible interpretations of RDF(S) models. In practice, these semantic conditions can be expressed as a set of rules: for example, the semantics of *rdfs*:*subPropertyOf* is given by the following rules:

(*aaa*, *rdfs*:*subPropertyOf* , *bbb*) ∧ (*bbb*, *rdfs*:*subPropertyOf* , *ccc*)
→ (*aaa*, *rdfs*:*subPropertyOf* , *ccc*)
(*aaa*, *rdfs*:*subPropertyOf* , *bbb*) → (*aaa*, *rdf* :*type*, *rdf* :*Property*)
(*aaa*, *rdfs*:*subPropertyOf* , *bbb*) → (*bbb*, *rdf* :*type*, *rdf* :*Property*)
(*xxx*, *aaa*, *yyy*) ∧ (*aaa*, *rdfs*:*subPropertyOf* , *bbb*) → (*xxx*, *bbb*, *yyy*)

The most noteworthy feature of RDF semantics is that the interpretation of the language is based on an open world assumption and is kept monotonic. An open world assumption means that based on a single document we cannot assume that we have a complete description of the world or even the resources explicitly described therein. Monotonicity means additional knowledge added to an RDF knowledge base cannot make previous inferences invalid. For example, if we specify that the range of the foaf:knows property is Person and then state that Rembrandt knows an instance of another class such as Pluto the dog (or even a literal value) we do not cause a (logical) contradiction; it is assumed that there could exist a statement defining that some other class (e.g. Dog) is also in the range of *foaf* :*knows*.

A consequence of monotonicity is that it makes no sense of talking about RDF validation: RDF Schema statements about an RDF resource only add additional in-formation about the resource and cannot invalidate or contradict statements previously inferred about that resource. In fact, RDF Schema semantics is specified as a set of inference rules; in the example, these rules would allow to infer that the re-source provided as the object of the *foaf* :*knows* property is (also) a Person besides possibly being other things (a Dog).

2.2.2.SPARQL: querying RDF sources across the Web

World Wide Web Consortium is working on creating a recommendation called SPARQL, establishing a standard query language and a protocol for interacting with RDF

sources. The query language captures the common set of features of the existing RDF query languages. The protocol prescribes the way a software application would query a remote ontology store across the Internet, i.e. the way to submit a SPARQL query to a server and the expected format of the results (and eventual error messages). The SPARQL specifications are expected to significantly increase the interoperability of Semantic Web applications. Also, the content of non-RDF databases may also be exposed using a SPARQL inter-face, opening up their content to the Semantic Web world. (The process of wrapping a relational database in a SPARQL interface can be partially automated.

2.3 The Web Ontology Language (OWL)

The Web Ontology Language (OWL) was designed to add the constructs of Description Logics (DL) to RDF, significantly extending the expressiveness of RDF Schema both in characterizing classes and properties. Description Logics are a set of Knowledge Representation languages with formal semantics based on their mapping to First Order Logic (FOL). Description Logics have been extensively studied since the 1980s including studies on the tradeoffs between the expressivity of the chosen language and the efficiency of reasoning. OWL has been designed in a way that it maps to a well-known Description Logic with tractable reasoning algorithms.

The Web Ontology Language is in fact a set of three languages with increasing expressiveness: OWL Lite, OWL DL and OWL Full. These languages are extensions of each other ($OWL_{Lite} \subseteq OWL_{DL} \subseteq OWL_{Full}$) both syntactically and seman-tically. For example, every OWL Lite document is a valid OWL DL document and has the same semantics when considered as an OWL DL document, e.g. it leads to the same logical conclusions. The vocabularies of these languages extend each other and languages further up in the hierarchy only relax the constraints on the use of the vocabulary. Although it is generally believed that languages of the OWL family would be an extension of RDF(S) in the same sense, this is only true for OWL Full, the most expressive of the family ($RDF(S) \subseteq OWL_{Full}$).

The middle language, OWL DL was the original target of standardization and it is a direct mapping to an expressive Description Logic. This has the advantage that OWL DL documents can be directly consumed by most DL reasoners to perform inference and consistency checking. The constructs of OWL DL are also familiar, although some of the semantics can be surprising mostly due to the open world assumption [RDH+04]. (Table 2.2.1.3 shows the OWL DL vocabulary, which is the same as the vocabulary of OWL Full.) Description Logics do not allow much of the representation flexibility introduced above (e.g. treating classes as instances or defining classes of properties) and therefore not all RDF documents are valid OWL DL documents and even the usage of OWL terms is limited.

For example, in OWL DL it is not allowed to extend constructs of the language, i.e. the concepts in the RDF, RDF Schema and OWL namespaces. In the case of the notion of a Class, OWL also introduces a separate *owl*:*Class* concept as a subclass

of *rdfs*:*Class* in order to clearly distinguish its more limited notion of a class. Similarly, OWL introduces the disjoint classes of object properties and datatype properties. The first refers to properties that take resources as values (such as *foaf* :*knows*) and the latter is for properties ranging on literals such as *foaf* :*name*.

OWL Full is a "limitless" OWL DL: every RDF ontology is also a valid OWL Full ontology and has the same semantics when considered as an OWL Full document. However, OWL Full is undecidable, which means that in the worst case OWL Full reasoners will run infinitely. OWL Lite is a lightweight sub-language of OWL DL, which maps to a less expressive but even more efficient DL language. OWL Lite has the same limitations on the use of RDF as OWL DL and does not contain some of the terms of OWL DL.

In summary, RDF documents are not necessarily valid OWL Lite or OWL DL ontologies despite the common conviction (see also Figure 4.11). In fact, "down-grading" a typical RDF or OWL Full ontology to OWL DL is a tedious engineering task. It typically includes many simple steps such as declaring whether properties are object properties or datatype properties and importing the external ontologies used in the document, which is mandatory in OWL but not in RDF. However, the process often involves more fundamental modelling decisions when it comes to finding al-ternate representations.

Most existing web ontologies make little use of OWL due to their limited needs, but also because general rule-based knowledge cannot be expressed in OWL. The additional expressivity of OWL, however, is required for modelling complex do-mains such as medicine or engineering, especially in supporting classification tasks where we need to determine the place of a class in the class hierarchy based on its description.

2.4. Modelling and aggregating social network data

Firstly, as we will demonstrate, maintaining the semantics of social network data is crucial for aggregating social network information, especially in heterogeneous environments where the individual sources of data are under diverse control.

Secondly, semantical representations can facilitate the exchange and reuse of case study data in the academic field of Social Network Analysis. With the current state-of-the art in network analysis, however, the network data collected in various studies is stored and published either in data formats not primarily intended for network analysis

While the representation of social individuals is relatively straightforward, the representation of social relations is a much more challenging problem. Our main contribution is thus a first step towards an ontology of social relations.

While reasoning with social relations is still future work, we discuss the aggregation of social individuals, which can be well automated based on the current representations and their formal semantics.

2.4.1 State-of-the-art in network data representation

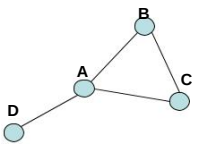
The most common kind of social network data can be modelled by a graph where the nodes represent individuals and the edges represent binary social relationships.

A number of different, proprietary formats exist for serializing such graphs and attribute data in machine-processable electronic documents. The most commonly encountered formats are those used by the popular network analysis packages Pajek and UCINET. These are text-based formats which have been designed in a way so that they can be easily edited using simple text editors. Figure 2.4.1.1 shows a simple example of these formats.

Unfortunately, the two formats are incompatible. (UCINET has the ability to read and write the .net format of Pajek, but not vice versa.) Further, researchers in the social sciences often represent their data initially using Microsoft Excel spreadsheets, which can be exported in the simple CSV (Comma Separated Values) format. As this format is not specific to graph structures (it is merely a way to export a table of data), additional constraints need to be put on the content before such a file can processed by graph packages. To further complicate matters for the researcher, visualization software packages also have their own proprietary formats such as the dot format used by the open source GraphViz package developed at AT&T Research.

The GraphML format represents an advancement over the previously mentioned formats in terms of both interoperability and extensibility. GraphML originates from the information visualization community where a shared format greatly increases the usability of new visualization methods. GraphML is therefore based on XML with a schema defined in XML Schema. This has the ad-vantage that GraphML files can be edited, stored, queried, transformed etc. using generic XML tools.

Common to all these generic graph representations is that they focus on the graph structure, which is the primary input to network analysis and visualization. Attribute



*Vertices 4
1 ”A”
2 ”B”
3 ”C”
4 ”D”
*Edges
1 1
1 2
1 3
1 4
2 3

dl
n = 4
labels embedded
format = edgelist
data:
A B
A C
A D
B C

```
<?xml version="1.0" encoding="UTF-8"?>
<graphml xmlns="http://graphml.graphdrawing.org/xmlns">
  <graph id="G" edgedefault="undirected">
    <node id="a"/>
    <node id="b"/>
    <node id="c"/>
    <node id="d"/>
    <edge source="a" target="b"/>
    <edge source="a" target="c"/>
    <edge source="a" target="d"/>
    <edge source="b" target="c"/>
```

```
  </graph>
</graphml>
```

Figure 2.4.1.1. A simple graph (upper left) described in Pajek .NET, UCINET DL and GraphML formats.

data when entered electronic form is typically stored separately from network data in Excel sheets, databases or SPSS tables.

However, none of these formats support the aggregation and reuse of electronic data, which is our primary concern. To motivate our case for data aggregation, consider the typical scenario in Figure 5.2 where we would like to implement a network study by reusing a number of data sources describing the same set of individuals and their relationships (for example, email archives and publication databases holding information about researchers). One of the common reasons to use multiple data sources is to perform *triangulation* i.e. to use a variety of data sources and/or methods of analysis to verify the same conclusion. Often the data sources to be used contain complementary information: for example, in *multi-layer studies* such

as the one performed by Besselaar et al. a multiplex network is studied using data sources that contain evidence about different kinds of relationships in a community. This allows looking at, for example, how these networks differ and how relationships of one type might affect the building of relationships of another type.

In both cases we need to be able to recognize matching instances in the different data sources and merge the records before we can proceed with the analysis. In fact, the graph representations discussed above strip social network data of exactly those characteristics that one needs to consider when aggregating data or sharing it for reuse: namely, these graph formats reduce social individuals and their relationships to nodes and edges, which is the only information required for the purposes of analyzing single networks.

What we need to support aggregation and reuse is a representation that allows capturing and comparing the identity of instances and relationships. Maintaining the identity of individuals and relationships is also crucial for preserving our data sets in a way that best enables their reuse and secondary analysis of data.

Solving these problems requires a very different kind of representation from the graph based formats shown above. Our proposed solution is a rich, semantic-based representation of the primary objects in social networks data, namely social individuals and their relationships. A semantic-based representation will allow us to wield the power of ontology languages and tools in aggregating data sets through domain-specific knowledge about identity, i.e. what it requires for two instances to be considered the same. As we will see, a semantic-based format has the additional advantage that at the same time we can easily enrich our data set with specific domain knowledge such as the relatively simple fact that if two people send emails to each other, they know each other, or that a certain kind of relationship implies (or refutes) the existence of another kind of relationship.

The two key problems in aggregating social network data are the identification and disambiguation of social individuals and the aggregation of information about social relationships.

2.5 Ontological representation of social individuals

The Friend-of-a-Friend (FOAF) ontology that we use in our work is an OWL-based format for representing personal information and an individual's social net-work. FOAF greatly surpasses graph description languages in expressivity by using the powerful OWL vocabulary to characterize individuals. More importantly, how-ever, using FOAF one can rely on the extendibility of the RDF/OWL representation framework for enhancing the basic ontology with domain-specific knowledge about identity.

The classes and properties in the current version of the FOAF vocabulary are shown in Figure 2.5.2.

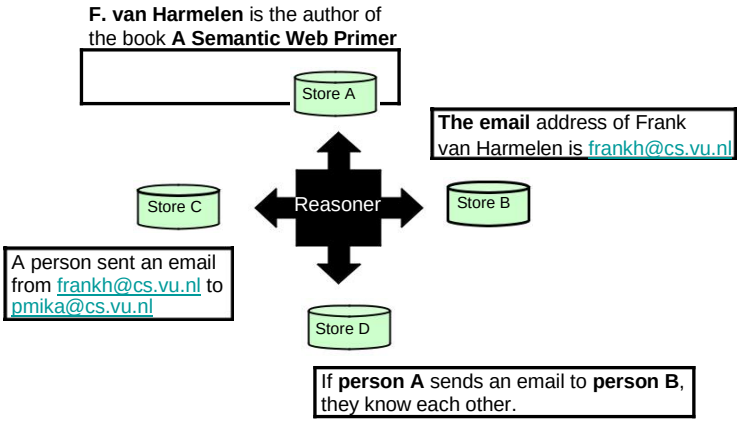


Figure 2.5.1. Example of a case of identity reasoning. Based on a semantic representation, a reasoner would be able to conclude, for example, that Peter Mika knows the author of the book *A Semantic Web Primer*

In terms of deployment, two studies give an insight of how much the individual terms of the FOAF are used on the Web. Both studies note that the majority of FOAF profiles on the Web are auto-generated by community sites such as LiveJournal, Opera Communities and Tribe.net. As FOAF profiles are scattered across the Web it is difficult to estimate their number, but even the number of manually maintained profiles is likely to be in the tens of thousands.

FOAF started as an experimentation with Semantic Web technology. The idea of FOAF was to provide a machine processable format for representing the kind of information that made the original Web successful, namely the kind of personal

information described in homepages of individuals. Thus FOAF has a vocabulary for describing personal attribute information typically found on homepages such as name and email address of the individual, projects, interests, links to work and school homepage etc. FOAF profiles, however, can also contain a description of the individual's friends using the same vocabulary that is used to describe the individual himself. Even more importantly, FOAF profiles can be linked together to form networks of web-based profiles.

FOAF profiles are created and controlled by the individual user and shared in a distributed fashion. FOAF profiles are typically posted on the personal website of the user and linked from the user's homepage with the HTML META tag. The distributed nature of FOAF networks means that FOAF requires a mechanism to link individual profiles and thus allow the discovery of related profiles. For this purpose, FOAF uses the *rdfs*:*seeAlso* mechanism described above. Related profiles can be thus discovered by crawling the FOAF network along these links. This is the way the so-called scutters (RDF crawlers) operate. Note that FOAF profiles generated from user profiles at online communities only contain links to members of the same website. For example, the FOAF representation of a LiveJournal profile only links to other LiveJournal members. This means that such sites are 'black holes' in terms of crawling: there are only links leading to them, but no links leading out of them.

When making a list of friends how would one find out their URIs?

The answer is to identify persons using their characteristic properties such as their email address, homepage etc. When describing a friend one would create a blank node with enough detail to uniquely identify that particular person. Recall that blank nodes in RDF are unidentified resources which can be thought of as the existential quantifier of the language. For example, the statements in the example of Figure 4.3 can be read as *There exists a Person named Peter with email ... who knows a person named Dirk.*

With the introduction of OWL it has become possible to describe these identifying properties of the *foaf* :*Person* class as inverse-functional properties (IFP). IFPs are properties whose value uniquely identifies a single object. In other words, if two resources have the same value for an inverse-functional property than they must denote the same object. For example, email address is an inverse-functional as every email address belongs to a single person. Name, however is not inverse-functional: the same name can belong to multiple persons. (Nevertheless, the probability of name clashes might be negligibly small depending on the size of the community we consider).

An advantage of FOAF in terms of sharing FOAF data is the relative stability of the ontology. The number of FOAF users means that the maintainers of the ontology are obliged to keep the vocabulary and its semantics stable. Interestingly, contingency requires that even inconsistencies in the naming of terms are left uncorrected: while most terms follow the Java convention, in some cases an underscore is used to separate words (mbox sha1sum, family name). When the meaning of terms does change, the evolution is toward generalization so as not to break existing applica-tions. For example, various sub-properties of the *foaf* :*knows* (*foaf* :*knowsWell* and *foaf* :*friend*) term have been removed due to their overlap in meaning. The descrip-tion of the *foaf* :*schoolHomepage* has been extended to sanction its use for describing university homepages.

FOAF Basics
Agent
Person
name
nick
title
homepage
mbox
mbox_sha1sum
img
depiction (depicts)
surname
family_name
givenname
firstName

Personal Information
weblog
knows
interest
currentProject
pastProject
plan
based_near
workplaceHomepage
workInfoHomepage
schoolHomepage
topic_interest
publications
geekcode
myersBriggs
dnaChecksum

Online Accounts / IM
OnlineAccount
OnlineChatAccount
OnlineEcommerceAccount
OnlineGamingAccount
holdsAccount
accountServiceHomepage
accountName
icqChatID
msnChatID
aimChatID
jabberID
yahooChatID

Projects and Groups
Project
Organization
Group
member
membershipClass
fundedBy
theme

Documents and Images
Document
Image
PersonalProfileDocument
topic (page)
primaryTopic
tipjar
sha1
made (maker)
thumbnail
logo

Table 2.5.2 Classes and properties of the FOAF ontology.

2.6 Ontological representation of social relationships

Ontological representations of social networks such as FOAF need to be extended with a framework for modelling and characterizing social relationships for two principle reasons: (1) to support the automated integration of social information on a semantical basis and (2) to capture established concepts in Social Network Analysis.

we list below some of the most commonly discussed characteristics of social relationships.

- Sign: (*valence*) A relationship can represent both positive and negative attitudes such as like or hate. The positive or negative charge of relationships is important on its own for the study of balance within social networks, which is the subject of balance theory.
- Strength: The notion of tie strength was first introduced by Granovetter in his groundbreaking work on the benefits of weak ties. Tie strength itself is a complex construct of several characteristics of social relations. In her survey, Hite lists the following aspects of tie strength discussed in the literature: Affect/philos/passions , Frequency/frequent contact , Reciprocity, Trust/enforceable trust , Complementarity, Accommodation/adaptation,

Indebtedness/imbalance, Collaboration, Transaction investments, Strong history, Fungible skills, Expectations, Social capital, Bounded solidarity, Lower opportunistic behavior, Density, Maximize relationship over org., Fine-grained information transfer, Problem solving, Duration, Multiplexity, Diffusion, Facilitation, Personal involvement, Low formality (few contracts), Connectedness.

- Provenance: A social relationship may be viewed differently by the individual participants of the relationship, sometimes even to the degree that the tie is unreciprocated, i.e. perceived by only one member of the dyad. Similarly, outsiders may provide different accounts of the relationship, which is a well-known bias in SNA.
- Relationship history: Social relationships come into existence by some event (in the most generic, philosophical sense) involving two individuals. (Such an event may not require personal contact (e.g. online friendships), but it has to involve social interaction.[16] From this event, social relationships begin a lifecycle of their own during which the characteristics of the relationship may change through interaction or the lack of (see e.g. Hite and Hesterly.
- Relationship roles: A social relationship may have a number of social roles associated with it, which we call relationship roles. For example, in a student/professor relationship within a university setting there is one individual playing the role of professor, while another individual is playing the role of a student. Both the relationship and the roles may be limited in their interpretation and use to a certain social context (see below). Social roles, social contexts and their formalization are discussed in Masolo et al.

In the case of Web-based social networking services we also see a variety of ways of capturing relationship types and other attributes of relationships. For example, Orkut allows to describe the strength of friendship relations on a 5-point scale from “haven’t met” to “best friend”, while other sites may choose other scales or terms.

In summary, a rich ontological characterization of social relationships is needed for the aggregation of social network information that comes from multiple sources and possibly different contexts, which is the typical scenario of the Web but is also the case for network data aggregation in the social sciences. We propose a representation of relationships on the basis of patterns (descriptions) that can be mapped to or extracted from observations about the environment. In other words, we consider relationships as higher-order concepts that capture a set of constraints on the interaction of the individuals.

2.6.1 Conceptual model

The importance of social relationships alone suggests that they should be treated as first-class citizens. Alternatively, social relations could be represented as n-ary predicates; however, n-ary relations are not supported directly by the RDF/OWL languages. There are several alternatives to n-ary relations in RD-F/OWL described in a document created by the Semantic Web Best Practices and Deployment Working Group of the W3C.

In all cases dealing with n-ary relations we employ the technique that is known as *reification*: we represent the relation as a class, whose instances are concrete relations of that type. One may recall that RDF itself has a reified representation of statements: the *rdf* :*Statement* object represents the class of statements. This class has three properties that correspond to the components of a statement, namely *rdf* :*subject*, *rdf* :*predicate*, *rdf* :*object*. These properties are used to link the statement instance to the resources involved in the statement.

Thus the trivial way to enrich the representation of relations in FOAF is to use the reification feature of the RDF language. We propose two alternative RDF(S) representation of relationships, both using the reification mechanisms of RDF(S) to reify the original triple asserting the existence of the relationship (for example, a *foaf* :*knows* statement). In other words relationships become subclasses of the *rdf* :*Statement* class (see Figures 2.6.1.1 and 2.6.1.2). Common also to both representations is that the new Relationship class is related to a general Parameter class by the hasPa-rameter relationship. Relationship types such as Friendship are subclasses of the Re-lationship class, while their parameters (such as strength or frequency) are subtypes of the Parameter class. Note that the hasParameter metaproperty cannot be defined in OWL DL (its domain is *rdf* :*Statement* while its range is *owl*:*Class* or some subclass of it).

The two alternatives differ in the representation of parameters. The first scheme borrows from the design of OWL-S for representing service parameters, as used in the specification of the profile of a Web Service. Here, parameters are related by the valued-by metaproperty to their range (*owl*:*Thing* or a datatype, de-pending on whether the parameter takes objects or datatypes as values). For example in an application Strength may be a subclass of Parameter valued-by integers. The disadvantage of this solution is that specifying values requires two statements or the introduction of a constructed property (the necessary axiom is not expressible in OWL).

The second alternative differs in that the “native” method of RDF is used for representing parameters: the generic Parameter class is defined as a subclass of

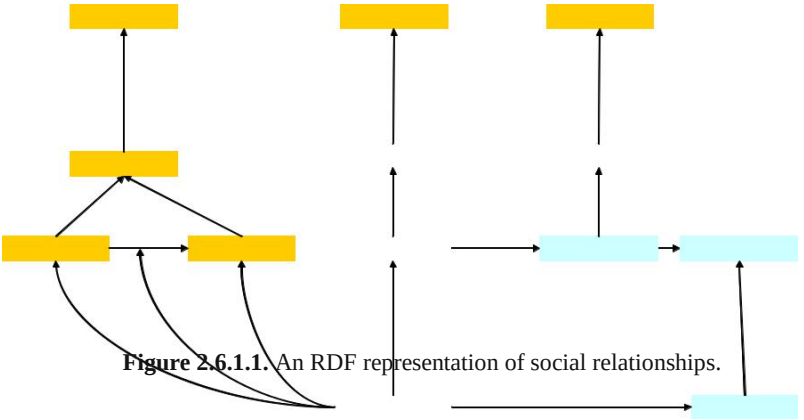


Figure 2.6.1.1. An RDF representation of social relationships.

rdf :*Property*. This model has the advantage that it becomes more natural to represent parameter values and restrictions on them. The disadvantage is that this solution is

not compliant with OWL DL: declaring properties ranging on properties and creating subclasses *rdf* :*Property* are not allowed in this species of OWL.

The advantage of describing relations using statement reification is that this form of reification is directly supported by RDF and can be efficiently stored and queried in quad-based triple stores.

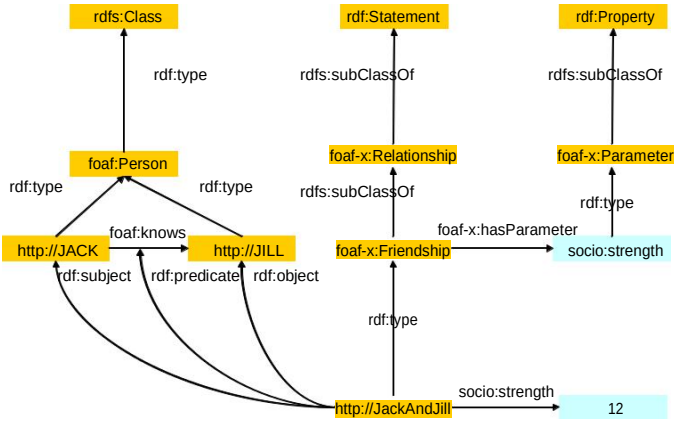
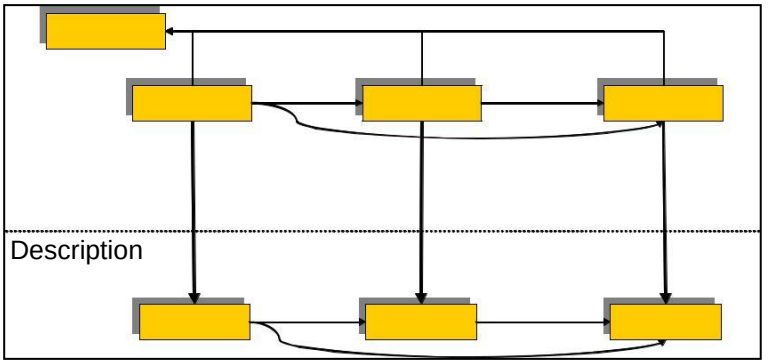
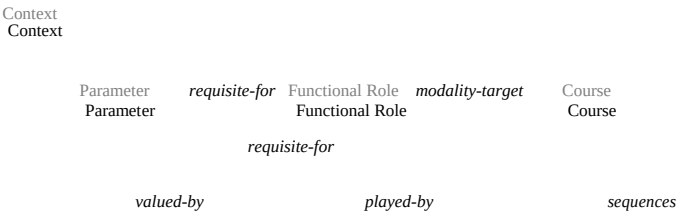


Figure 2.6.1.2. An alternative RDF representation of social relationships.

Further, there are two related characteristics of social relations that we would like to capture in a more advanced representation. The first characteristic is the context dependence of social relations and the second is the separation between an observable situation and its interpretation as a social relation between two individuals.

A solution proposed is the Descriptions and Situations ontology design pattern that provides a model of context and allows to clearly delineating these two layers of representation (Figure 2.6.1.3).





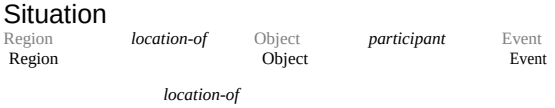


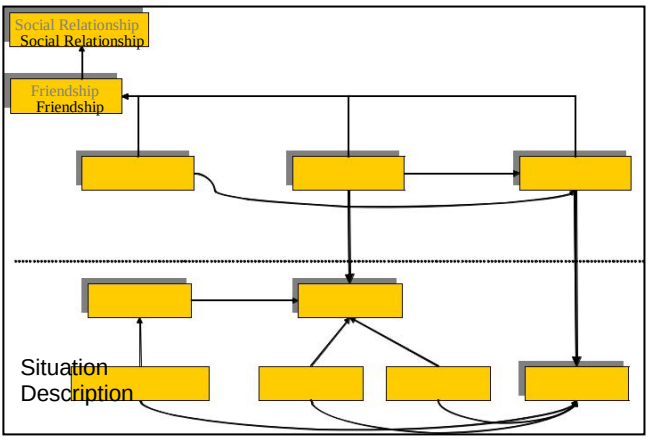
Figure 2.6.1.3. The Descriptions and Situations ontology design pattern.

D&S is a generic pattern for modelling non-physical objects whose intended meaning results from statements, i.e. it emerges in combination with other entities. For example, a norm, a plan, or a social role is usually represented as a set of statements and not as a concept. On the other hand, non-physical objects may change and be manipulated similar to physical entities, and are often treated as first-order objects. That means that an ontology should account for such objects by modelling the context or frame of reference on which they depend.

D&S is an ontology-design pattern in the sense that it is used as a template for creating domain ontologies in complex areas. D & S has been successfully applied in a wide range of real-life ontology engineering projects from representing Service Level Agreements (SLAs) to the descriptions of Web Services

As depicted in the Figure, the notion of Context in D & S is composed of a set of Parameters, Functional Roles and Courses of Events. Axioms enforce that each descriptive category acts as a selector on a certain basic category of DOLCE: Parameters are valued-by Regions, Functional Roles are played-by Objects (endurants) and Courses of Events sequence Events (perdurants). D & S captures the intuition that multiple overlapping (or alternative) con-texts may match the same world or model, and that such contexts can have systematic relations among their elements.

In particular, we model a Social Relationship as a subclass of Context and particular social relationships such as Friendship a subclass of this generic concept. As contexts, Social Relationships can have a number of Parameters, Roles and single Course as components.



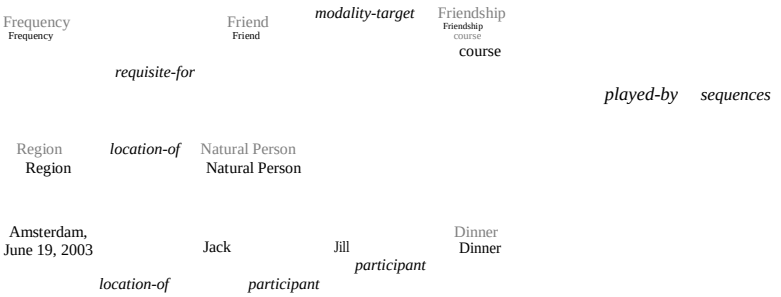


Figure 2.6.1.4. An instantiation of the Descriptions and Situations pattern for social relationships.

A typical Role is the Relationship Role, a subclass of the Social Role concept. An example of a relationship role is (the trivial) Friend role in a friendship relation, the Student and Professor roles in a student/advisor relationship and the Uncle/Nephew roles of kinship. Relationship Roles are restricted to be played by Natural Persons. Figure 2.6.1.4 shows the representation of the friendship relation (some property instances are not shown for visualization purposes). Friendship in general is a social relationship with a single role called Friend, played by actual persons such as Jack and Jill. Friendship also has a typical course; an event such as a dinner where both Jack and Jill have participated may be related to this course, which would indicate at least that it has a significance to the development of a friendship between Jack and Jill. (Friendship is difficult to capture more precisely in this respect in that there is hardly a typical course for a friendship.

2.7 Aggregating and reasoning with social network data

Supposing that we have started out with some data sets in traditional formats (relational databases, Excel sheets, XML files etc.) our first step is to convert them into an RDF-based syntax, which allows to store the data in an ontology store and manipulate it with ontology-based tools. In this process we need to assign identifiers to resources and re-represent our data in terms of a shared ontology such as FOAF.

In case our data sets come from external sources it is often more natural to preserve their original schema. For example, in case of converting data from a relational database or Excel sheet it is natural to preserve the schema of the database or spread-sheet as represented by the table definitions or table headings. We can then apply *ontology mapping* to unify our data on the schema level by mapping classes (types) and properties from different schemas to a shared ontology such as FOAF. In effect, ontology mapping allows us to treat the data set as if it had a single shared schema. Note that research on automated methods for ontology mapping is an active research area within the Semantic Web community. It is not our primary concern as the number of ontologies involved and their size does not make it necessary to automate ontology mapping in our typical case.

The task of aggregation, however, is not complete yet: we need to find identical resources across the data sets. This is a twostep process. First, it requires capturing the domain-specific knowledge of when to consider two instances to be the same. As we will see the FOAF ontology it also prescribes ways to infer the equal-ity of two instances, for example based on their email address. However, beyond these properties it is likely that we need to introduce some domain-specific criteria based

on domain-specific properties. In order to do this, we need to consider the general meaning of equality in RDF/OWL. In practice, only a limited part of the knowledge regarding instance equality can be captured in RDF or OWL. For example, determining equality is often considered as applying a threshold value to a similarity measure, which is a weighted combination of the similarity of certain properties. (The threshold is determined by experimentation or through machine learning.) In this and many other practical cases that involve computation we need a procedural representation of knowledge.

Once we determined the rules or procedures that determine equality in our domain, we need to carry out the actual instance unification or *smushing*. Unlike much of the related work on instance unification, we consider smushing as a reasoning task, where we iteratively execute the rules or procedures that determine equality until no more equivalent instances can be found. The advantage of this approach (compared to a one-step computation of a similarity measure) is that we can take into account the learned equalities in subsequent rounds of reasoning.

2.7.1 Representing identity

One of the main advantages of RDF over other representation formalisms such as UML is the possibility to uniquely identify resources (instances, classes, as well as properties). The primary mechanism for this is the assignment of URIs to resources. Every resource, except blank nodes is identified by a URI.

However, in practice it is often the case that there are a number of candidates for identifiers. For example, in the publishing world a number of identifier schemes are in use. Standard schemes such as ISBN and ISSN numbers for books and periodicals as well as DOIs (Digital Object Identifiers) are widely adopted, while each publisher and online repository also maintains its own identifier scheme. Further, publications that are accessible online can be represented by their URL. All of these identifiers are unique (an identifier identifies a single resource), but mostly not single (a resource may be identified by multiple identifiers). This is the case with URLs (the same publication may be posted at multiple URLs), but it happens even with DOIs whose registration is centralized that the same publication is assigned two or more identifiers.

Multiple identifiers can be represented in RDF in two separate ways. First, one can introduce a separate resource and use the identifiers as URIs for these resources. Once separate resources are introduced for the same object, the equality of these resources can be expressed using the *owl*:*sameAs* property (see the following section). The other alternative is to choose one of the identifiers and use it as a URI.

Note that in all cases resource identifiers need to conform to the URI specification and good practice. Many modern identifier schemes such as DOIs have been designed to conform to the URI specification. Other identifier schemes can be recoded as URIs with the new info: prefix (protocol). It is also a common practice to create URIs within a web domain owned or controlled by the creator of the metadata description. For example, if the registered domain name of a public organization is `http://www.example.org` then a resource with the identifier 123 could be represented as `http://www.example.org/id/123`. This satisfies the guidelines for good URIs, in particular that good URIs should be unique and stable. The first criterion means that good URIs should be unambiguous or at least should be chosen such that it is unlikely that someone else would use the same URI for something different. This is important because resources with the same URI but different intended meanings are likely to result in

inconsistencies (a *URI clash*). The second criterium is also important because there is no way to rename resources (to reassign URIs). Once a URI changes the only solution is to introduce a new resource and assert its equivalence with the old resource. However, in large scale systems such as the web there is no way to notify remote systems of the new identifier and they are likely to continue referencing the resource by its old identifier. Thus it is bad idea for example to encode unstable property values in identifiers, where publications are assigned identifiers based on e.g. the journal, the volume and the page number. If there turns out to be a data entry mistake in one of the values, the identifier becomes obsolete and is potentially ambiguous.

The use of http:// URIs has been somewhat complicated by the long outstanding issue of whether such URIs could be used for only web resources or also for abstract concepts (which has been the practice), and if yes, what should web servers respond to HTTP requests related to such URIs.The resolution of this issue is that the response code of the web server should indicate whether the URI denotes a web resource (HTML page, image about a person, publication etc.) or some abstract concept (such as a person, a publication etc.) The response code used by web servers to indicate success should be used only for web resources.

A practical consequence is that for abstract concepts one should not choose URIs that are recognized by a server, e.g. the location of an existing HTML page, as this would be a case of URI clash. While the decision also opens the road to look up metadata about abstracts concepts using the HTTP protocol there are currently very few web servers configured to support this.

2.7.2 On the notion of equality

RDF and OWL (starting with OWL Lite) allow us to identify resources and to represent their (in) equality using the *owl*:*sameAs* and *owl*:*differentFrom* properties. However, these are only the basic building blocks for defining equality. In particular, the meaning of equality with respect to the objects of the domain depends on the domain itself and the goals of the modelling. For example, it is part of the domain knowledge what it takes for two persons or publications to be considered the same, i.e. what are the characteristics that we take into account. Further, depending on the level of modelling we may distinguish, for example, individual persons or instead consider groups of persons (e.g. based on roles) and consider each individual within the group to be equivalent (role equivalence).

The Leibniz-law is formalized using the logical formula given in Formula 2.7.2.1. The converse of the Leibniz-law is called Indiscernibility of Identicals and written as Formula 2.7.2.2. The two laws taken together (often also called Leibniz-law) are the basis of the definition of equality in a number of systems.

$$\forall P : P(x) \leftrightarrow P(y) \rightarrow x = y \qquad (2.7.2.1)$$

$$\forall P : P(x) \leftrightarrow P(y) \leftarrow x = y \qquad (2.7.2.2)$$

The reflexive, symmetric and transitive properties of equality follow from these definitions. Notice that both formulas are second-degree due to the quantification on properties. This quantification is also interesting because it provides the Leibniz-law different interpretations in open and closed worlds. Namely, in an open world the number of properties is unknown and thus the Leibniz-law is not useful in practice: we can never conclude that two resources are equal since we can never be certain whether there is

another property out there that could allow us to distinguish them. In a closed world we can possibly iterate over all properties to check if two resources are equal; if we can distinguish them by some property we can assume they are equal.

In practice, a closed world assumption can be equally undesirable as an open one. In most cases we have almost complete information about our resources, but still we may not want two resources to be considered identical just because of a lack of information. For example, if we have two resources and we only know that they are the same gender, we may not want to assume they are identical (which would be the consequence in a closed world).

The semantics of OWL is built on an open world assumption, which means that the Leibniz-law cannot be used to infer identity, not even if we reduce the property space even further. However, we can still infer the equality of instances by necessity (see the following Section).

On the other hand, the semantics of *owl*:*sameAs* conforms to Formula 2.7.2.2. Namely, *owl*:*sameAs* restricts the interpretation of the theory to those models where the two symbols denote the same object and thus they must be indiscernible in the sense that they are interchangeable in statements:

$$(s_1 , owl{:}sameAs, s_2) \wedge (s_1, p, o) \rightarrow (s_2 , p, o)$$
$$(p_1, owl{:}sameAs, p_2) \wedge (s, p_1, o) \rightarrow (s, p_2, o)$$
$$(o_1 , owl{:}sameAs, o_2) \wedge (s, p, o_1) \rightarrow (s, p, o_2) \qquad (2.7.2.3)$$

The reflexive, symmetric and transitive properties of sameAs also follow:

$$\forall s : (s, owl{:}sameAs, s)$$
$$(s_1, owl{:}sameAs, s_2) \rightarrow (s_2, owl{:}sameAs, s_1)$$
$$, owl{:}sameAs, s_2) \wedge (s_2, owl{:}sameAs, s_3) \rightarrow (s_1 , owl{:}sameAs,$$
$$(s_1 s_3) \qquad (2.7.2.4)$$

Note that it is not inconsistent to have resources that are *owl*:*sameAs* but have different stated properties, e.g. Formula 2.7.2.5 is not an inconsistent ontology. The explanation lies again in the open world assumption: we can assume that the missing statements (that s_1 has the *foaf* :*name* Paul and s_2 has the *foaf* :*name* John) exist somewhere. In a closed world this ontology would be inconsistent.

$$(s_1 , owl{:}sameAs, s_2)$$
$$(s_1 , foaf{:}name, \text{"John"})$$
$$(s_2 , foaf{:}name, \text{"Paul"}) \qquad (2.7.2.5)$$

2.7.3 Determining equality

In our case, we are interested in capturing knowledge about the identity of resources that can lead to conclude the (in) equality of resources.

In OWL there are a limited set of constructs that can lead to (in) equality statements. Functional and inverse functional properties (IFPs) and maximum cardinality restrictions in general can lead to conclude that two symbols must denote the same re-source when otherwise the cardinality restriction could not be fulfilled. For example, the *foaf* :*mbox* property denoting the email address of a person is inverse-functional as a mailbox can only belong to a single person. As another example, consider a hypothetical *ex*:*hasParent* property, which has a maximum cardinality of two. If we state that a single person has three parents (which we are allowed to state) an OWL reasoner should conclude that at least two of them has to be the same. Once inferred, the equality of instances can be stated by using the *owl*:*sameAs* property. There are more ways to conclude that two symbols do not denote the same object, which is ex-pressed by the *owl*:*differentFrom* property. For example, instances of disjoint classes must be different from each other.

Often, the knowledge that we need to capture cannot be expressed in OWL, only in more expressive rule languages. A typical example is the case of complex keys: for example, we may want to equate two persons only if both their names and their birth dates match. This is not possible to express in OWL but can be captured in Horn logic, a typical target for rule languages for the Semantic Web.

However, there are even situations when the expressivity of rule languages is not sufficient either. In our early work on one of the first ontology-based Knowledge Management systems we have already noted that several elementary reasoning steps could not be expressed in declarative, logic based formalisms. Functions for simple data type manipulation such as the concatenation of literals is not part of either DL or rule languages. An example of this is the matching of person names.

In our applications, person names are matched by separating the first and the last name in the first step. We apply a simple heuristic for determining the cutting point. (It is not possible to completely automate the separation of first names and last names in a language-independent way.) Then the last names are compared in a case unsensitive manner. If they match, the first names are compared using fuzzy string matching. In both cases we have to deal with abbreviations, e.g. to match *F.v. Harmelen* against *Frank van Harmelen*.

This algorithm is easy to describe in a programming language or more powerful transformation languages such as XSLT, which contains the necessary datatypes, data manipulation functions and the three basic constructs of procedural programming (the sequence operator, the if statement and the for loop). Rule languages in a Semantic Web context, however, lack this expressive power.

The solution is to combine declarative and procedural reasoning. Similar to procedural attachments, the main idea is to compute certain relationships with components/services that conform to a particular interface. Reasoning is a combination of calling these services and executing a regular Description Logic or rule-based reasoner.

2.7.4 Reasoning with instance equality

Reasoning is the inference of new statements (facts) that *necessarily* follow from the set of known statements. Every fact we add to our knowledge base has a meaning in the sense that it represents some constraint on the mapping between the symbols of our representation and some domain of interpretation. In other words, every piece of additional knowledge excludes some unintended interpretation of our knowledge base.

In most sufficiently complex languages, an infinite number of new statements could be inferred from any non-trivial knowledge base. In practice, however, we are not interested in all statements that might be deduced from a knowledge base, but only those that are relevant to some task. In other words, we would like to complete our knowledge base to contain all the important knowledge relevant to that task.

There are a number of choices one has to consider when implementing reasoning, which mostly concern trade-offs between efficiency and scalability.

Description Logic versus rule-based reasoners

Our task is instance equality reasoning, i.e. the inference of *owl*:*sameAs* and *owl*:*differentFrom* statements. In general, OWL (DL) reasoners are of limited use in performing this kind of reasoning. As mentioned in the previous Chapter, our representation needs are also slightly different from the OWL language: we use a limited set of OWL constructs (corresponding to the unofficial OWL Tiny variant), while some of our knowledge can only be captured by rules.

Nevertheless, we can use OWL reasoners to reason with the part of the knowledge that can be expressed in OWL. In particular, OWL DL was designed with the idea that the language should be as expressive as possible with the limitation that it should be also *decidable* and that it should be possible to create *efficient* reasoners for it.

In many practical cases OWL is more expressive than necessary. Description Logic reasoners are designed to support primary tasks of classification and the consistency checking of ontologies. Other reasoning tasks are usually reformulated as consistency checking. In the case of equality reasoning this means that if we want to check whether two instances are necessarily the same, we add the opposite to the ontology and check for inconsistency.

A better alternative is to consider rule-based reasoning and this is the method we explored in our work in combination with procedural attachments. The semantics of RDF(S) can be completely expressed using a set of inference rules. Further, a significant part of the OWL semantics can be captured using rules and this contains the part that is relevant to our task. Horst shows a partial rule-based axiomatization of OWL. These rules are implemented among others by the OWLIM reasoner, but can also be used with any sufficiently expressive rule-based reasoner such as the query language or the custom-rule based reasoner of Sesame.

Forward versus backward chaining

Rules can be used either in a forward-chaining or backward-chaining manner with different trade-offs. Forward chaining means that all consequences of the rules are computed to obtain what is called a *complete materialization* of the knowledge base. Typically this is done by repeatedly checking the prerequisites of the rules and adding their conclusions until no new statements can be inferred. The advantage of this method is that queries are fast to execute since all true statements are readily avail-able. (The reasoning is performed before queries are answered, typically right after adding some new knowledge.) The disadvantage is that storing the complete materialization often takes exorbitant amounts of space as even the simplest RDF(S) ontologies result in a large amount of inferred statements (typically several times the size of the original repository), where most of the inferred statements are not required for the task at hand. Also, the knowledge base needs to be updated if a statement is removed, since in that case all other statements that have been inferred from the

removed statements also need to be removed (if they cannot be inferred in some other way).

With backward-chaining, rules are executed "backwards" and on demand, i.e. when a query needs to be answered. While with forward chaining we check the pre-requisites of rules and add all their conclusions, here we are given a conclusion and check whether it is explicitly stated or whether it could inferred from some rule, either because the prerequisites of the rule are explicitly stated to be true or again because they too could be inferred from some other rule(s). The drawback of backward-chaining is longer query execution times.

The advantage of a rule-based axiomatization is that the expressivity of the reasoning can be fine-tuned by removing rules that would only infer knowledge that is irrelevant to our reasoning task.

However, some of the rules required cannot be expressed declaratively and thus we implemented our own identity reasoner in the Java language. The reasoning is performed using a combination of forward and backward chaining in order to balance efficiency and scalability. The rules that are used to infer the equality of instances are executed iteratively in a forward chaining manner by querying the repository for the premise(s) and adding the consequents of the rules.

For example, consider the following SeRQL query which asks for the email address of the instance Peter:

```
SELECT email FROM
{example:Peter} foaf:mbox
{email} USING NAMESPACE
foaf = <http://xmlns.com/foaf/0.1/>,
example = <http://www.example.org/>
```

This query is rewritten in a way to also return all email addresses of other instances that are *owl*:*sameAs* this instance:

```
SELECT email FROM
{example:Peter} owl:sameAs
{other}, {other} foaf:mbox {email}
USING NAMESPACE
foaf = <http://xmlns.com/foaf/0.1/>,
example = <http://www.example.org/>
```

Note that this will also return the email attached to the instance *example*:*Peter*, because of the reflexivity of the *owl*:*sameAs* property. At this point we already added for all instances that they are sameAs themselves. This query rewriting is implemented in the Elmo API.

The timing of reasoning and the method of representation

There are three basic variations on what point the identity reasoning is performed. Also, there is the related choice of whether to represent equivalence using the *owl*:*sameAs* relationship or to directly merge the descriptions of the equivalent instances.

In the first variation, smushing is carried out while adding data into a repository. This is the method chosen by Ingenta when implementing a large-scale publication

metadata repository: when a new instance is added to the repository, it is checked whether an equivalent instance already exists in the repository.

The second variation is when the reasoning is performed after the repository has been filled with data containing potential duplicates. This is the choice we take in the Flink system. In this case again there is also a choice between representing the results by merging identifiers or by *owl*:*sameAs* relationships. However, the benefits of the first are not as obvious, because removing statements is an expensive operation with most triple stores.

Lastly, reasoning can be performed at query time. This is often the only choice such as when querying several repositories and aggregating data dynamically in an AJAX interface such as the with the openacademia application. In this case the solution is to query each repository for instances that match the query, perform the duplicate detection on the combined set and present only the purged list to the end-user.

2.7.5 Evaluating smushing

Smushing can be considered as either a retrieval problem or a clustering problem. In terms of retrieval, we try to achieve a maximum precision and recall of the theoretically correct set of mappings. Note that unlike in ontology mapping where there is typically a one-to-one mapping between elements of two ontologies, in the in-stance unification problem a single resource may be mapped to a number of other resources. Thus we can also conceptualize this task as clustering and evaluate our clusters against the ideal clustering.

Instance unification or smushing can be considered as an optimization task where we try to optimize an information retrieval or clustering-based measure.

2.8. Advanced representations

As we have seen, some of the knowledge we wanted to express could not be captured in OWL DL. Some of the missing features that do not significantly change the complexity of the language such as the conspicuously absent *owl*:*ReflexiveProperty* are likely to appear in future revisions of OWL. However, OWL will remain limited by the original complexity concerns, most of which do not apply to practical scenarios. More expressive power in representation is expected from the Rule Interchange Format (RIF) currently under development by the W3C. Although its development is primarily driven by requirements from use cases, RIF will most likely include first-order logic (FOL).

Additional expressive power can be brought to bear by using logics that go beyond predicate calculus. For example, *temporal logic* extends assertions with a temporal dimension: using temporal logic we can express that a statement holds true at a certain time or for a time interval. Temporal logic is required to formalize ontology versioning, which is also called change management. With respect to our information integration task, changes may affect either the properties of the instances or the description of the classes, including the rules of equivalence, i.e. what it means for two instances of a class to be equivalent. In most practical cases, however, the conceptualization of classes in the ontology should be more stable than the data and the rules of equivalence should not include dynamic properties. Never theless, it is sometimes still desirable to match on such properties. For example, when merging person descriptions we might consider the color of a person's hair. Such a

property can change its value over time, which means we have to consider the time when the assertions were true n.

Extending logic with *probabilities* is also a natural step in representing our problem more accurately. As we have already discussed, the matching of properties often result in probabilistic statements. For example, when we match person names using a string matching algorithm we get a measure of the similarity of the two strings. This can be taken on its own or combined with similarity measures for other properties to form a probability for two persons to be the same. When combining similarities we can use weights to express the relevance of certain properties. Such weights can be determined ad hoc or can be learned from training data using machine learning. The resulting probability could be represented in a probabilistic logic. In our work we apply a threshold to such probabilities in order to return to binary logic where two instances are either the same or not.

UNIT 3 EXTRACTION AND MINING COMMUNITIES IN WEB SOCIAL NETWORKS

3.1 Extracting Evolution of Web Communities from a Series of Web Archives

Recent advances in storage technology make it possible to store a series of large Web archives. It is now an exciting challenge for us to observe evolution of the Web. A web community is a set of web pages created by individuals or associations with a common interest on a topic. So far, various link analysis techniques have been developed to extract web communities. The web community slightly differs from a community of people, for example, a web community may include competing companies.

Figure 3.1: Typical graph of hubs and authorities

We describe the global behavior of web community evolution. For describing evolution behaviors, we introduce several evolution metrics for communities that measure the degree of evolution, such as growth rate, novelty, and stability.

By using such metrics, users can extract detailed evolution of target communities. To examine their feasibility, we developed a system for extracting communities based on the evolution, which is composed of two parts. The first component extracts whole communities and their relevance from each web archive. It is based on our previous work of *web community chart* that is a graph of communities, in which related communities are connected by weighted edges. The main advantage of our web community chart is existence of relevance between communities. We can navigate through related communities, and locate evolution around a particular community. The second component is a web community evolution viewer for browsing how communities evolved through three years. It provides various ways for locating the evolution of communities such as emerged and growing. With some examples, we demonstrate that we can easily locate interestingly evolving communities by combining evolution metrics and relevance.

3.1.1 Web Community Chart

The web community chart is a graph that consists of web communities as nodes, and weighted edges between related communities. The weight of each edge represents the relevance of communities at both ends. In this section, we first explain intuition for underlying techniques, then briefly de-scribe the algorithm for building the chart.

3.1.1.1 Intuition for Underlying Techniques

Our algorithm builds the web community chart from a given set of seed pages. The main idea is applying a HITS based related page algorithm (RPA) to each seed, and then investigate how each seed derives other seeds as related pages. RPA first builds a subgraph of the Web around the seed, and extracts authorities and hubs in the graph using HITS. Then authorities are returned as related pages.

To identify web communities and to deduce their relation-ships, we put focus on relationships between a seed page and related pages derived by Companion–. Figure 3.1.1 depicts an example of these derivation relationships. In this example, we use five seed pages, IBM, TOSHIBA, SONY, and two fan pages of SONY PC. In Figure 3.1.2, the graph (A) shows how each seed is pointed to by hub pages, and the directed graph (B) shows how each seed derives each other as related pages.

From the graph (B) in Figure 2, we can see that symmetric derivation is a strong relationship, and asymmetric derivation is a weak relationship.

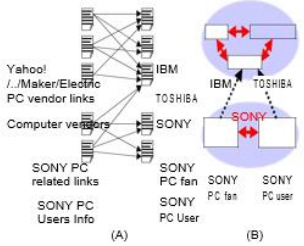


Figure 3.1.2: An example of derivation reationships

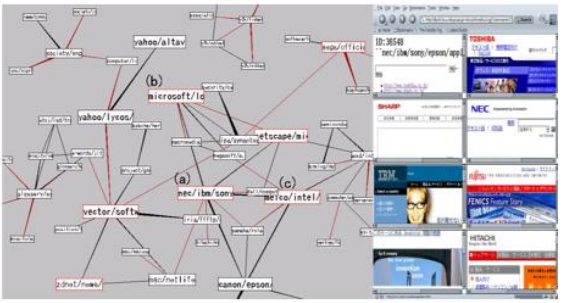


Figure 3.1.3: A part of the web community chart

Under these observations, we define that a community is a set of pages densely connected by symmetric derivation relation-ships, and two communities are related if there is an asymmetric derivation relationship between members of them.

3.1.1.2Algorithm for Building Charts

The first step is selecting a seed set from a web archive. As seeds, we select web pages that have in links from *I N* or more different servers. Here, *I N* is a parameter to determine the seed set. Only the number of different servers is counted, and intra-server links are not considered, because links from the same server are often made by the same author.

The second step is to build a directed graph that shows how each seed derives other seeds by Companion–. Nodes represent seeds in the seed set. Each directed edge, from a node *s* to another node *t*, represents the fact that *s* derives *t* as one of the related pages by Companion–. We create directed edges between nodes by applying Companion– to each seed, so that an edge from a node *s* to another node *t* exists when *s* derives *t* as one of the top *N* authorities, where *N* is a parameter. We call this graph the authority derivation graph (ADG) in the following.

The third step is to extract a symmetric derivation graph (SDG) from ADG, and to extract web communities from SDG. In this step, we put focus on the symmetric derivation relationship, in which two nodes derive each other by Companion–. SDG includes nodes in the seed set, and an edge from *s* to *t* exists when *s* and *t* point to each other in ADG. Then we extract densely connected subgraphs in SDG as *cores* of web communities, and form cores into communities by adding remaining nodes to these cores. Although cliques in SDG seem to be a good definition of cores, we have found many communities sparser than cliques in SDG. Therefore, we use a heuristic definition of the core. We use a node triangle as a unit of extraction, and a core is defined as a set of triangles that share edges in SDG. Note that a core becomes a 1-connected subgraph of SDG, and may be a complete graph with four or more nodes. In the following, we explain the detailed extraction process. After finishing this process, every connected node in SDG becomes a member of a community. Note that communities become disjunctive sets of nodes.

A: Extract all triangles of nodes from SDG. Then make every core that is a subgraph consists of triangles sharing edges. When two cores share some nodes, we temporarily isolate them, and pass them to the next step.

B: Add each remaining node in SDG to a neighboring core, if the node has edges connected to the core. When there are multiple candidates, select one core taking into account of directed edges in ADG. That is, to select a core that has the most incoming edges from the node in ADG. Each core then becomes a community.

C: There remain connected components that do not form triangles, such as lines of nodes. We also extract such components as communities.

Finally, we construct a web community chart that can be used to navigate from a community to other related communities. The chart is a directed graph that includes communities as nodes, and directed edges between related communities. Each edge has a weight that represents the strength of relationships. We create a directed edge from a community c to another community d with a weight w, when there exists w directed edges in ADG from nodes in c to nodes in d. In the following, we use the simplified weight that is the sum of weights of directed edges between c and d ignoring directions. This is because the semantics of the direction is not yet clear. We call this simplified weight as the *relevance* between communities at both ends.

3.1.1.3. An Example of a Web Community Chart

Figure 3.1.3 shows a part of the web community chart built from our web archive in 2002, using our chart browser. Communities including keyword" Computer" are displayed in Figure 3.1.1.3. Each box with a label rep-resents a web community, and edges represent relationships between communities. The size of each node is determined by the number of URLs and edges connected to the node. The thickness of each edge represents the relevance value of edges. Labels on communities are automatically attached by selecting frequent keywords from anchor texts that point to URLs in the community. When the user clicks one of a community, its pages are displayed with a web browser as shown in Figure 3.1.1.3.

Using this chart, the user can overview and navigate com-munities related to computers. The community (a) includes major computer companies in Japan, such as SONY and NEC, and its contents are shown in the web browser. Around the community (a), there are communities of related companies. The community (b) includes Japanese pages of soft-ware makers, such as Microsoft and Lotus. The community includes computer device and peripheral companies such as Intel and Adaptec.

3.2.Evolution Of Web Communities

This section explains how web communities evolve, and what kinds of metrics can measure degree of the evolution, such as growth rate and novelty. We first explain the details of changes of web communities, and then introduce evolution metrics that can be used for finding patterns of evolution. Here we summarize the notations used in this section.

t_1 , t_2, ..., t_n : Time when each archive crawled. Currently, we use a month as the unit time.

W (t_k): The Web archive at time t_k .

C(t_k): The web community chart at time t_k .

c(t_k), d(t_k), e(t_k), ...: Communities in C(t_k).

3.2.1 Types of Changes

We observe the evolution of communities from a series of web archives by (1) building web community charts (C(t_1), C(t_2), ..., C(t_n)) from all web archives, and (2) investigating differences between neighboring charts.

There are two ways to see the evolution of communities, backward and forward. For simplicity, here we explain back-ward examination. That is, first we select a web community

chart $C(t_k)$, and see how communities had been evolved until time t_k by comparing $C(t_{k-1})$ and $C(t_k)$. We can do the same thing for forward examination by comparing $C(t_k)$ and $C(t_{k+1})$. Here we show how communities change from $C(t_{k-1})$ to $C(t_k)$, such as growing and shrinking.

Emerge: A community $c(t_k)$ emerges in $C(t_k)$, when $c(t_k)$ shares no URLs with any community in $C(t_{k-1})$. Note that not all URLs in $c(t_k)$ are newly appeared in $W(t_k)$. Some URLs in $c(t_k)$ may be included in $W(t_{k-1})$, and do not have enough connectivity to form a community.

Dissolve: A community $c(t_{k-1})$ in $C(t_{k-1})$ has dissolved, when $c(t_{k-1})$ shares no URLs with any community in $C(t_k)$. Note that not all URLs in $c(t_{k-1})$ disappeared from $W(t_{k-1})$. Some URLs in $c(t_{k-1})$ may still be included in $W(t_k)$ losing connectivity to any community. Grow and shrink: When $c(t_{k-1})$ in $C(t_{k-1})$ shares URLs with only $c(t_k)$ in $C(t_k)$, and vice versa, only two changes can occur to $c(t_{k-1})$. The community grows when new URLs appear in $c(t_k)$, and shrinks when URLs disappeared from $c(t_{k-1})$. When the number of appeared URLs is greater than the number of disappeared URLs, we consider that it grows. In the reverse case, we consider that it shrinks.

Split: A community $c(t_{k-1})$ may splits into smaller communities. In this case, $c(t_{k-1})$ shares URLs with multiple communities in $C(t_k)$. A split is caused by disconnections of URLs in SDG. Split communities may grow and shrink. They may also merge (see the next item) with other communities.

Merge: When multiple communities ($c(t_{k-1})$, $d(t_{k-1})$, ...) share URLs with a single community $e(t_k)$, these com-munities are merged into $e(t_k)$ by connections of their URLs in SDG. Merged communities may grow and shrink. They may also split before merging.

3.2.2 Evolution Metrics

Our evolution metrics measure how a particular community $c(t_k)$ has evolved. For example, we can know how much $c(t_k)$ has grown, and how many URLs newly appeared in $c(t_k)$. Our metrics can be used for finding various patterns of evolution described in Section 3.1. To measure changes of $c(t_k)$, we need to identify the community at time t_{k-1} corresponding to $c(t_k)$. We define this *corresponding community*, $c(t_{k-1})$, as the community that shares the most URLs with $c(t_k)$. If there were multiple communities that share the same number of URLs, we select a community that has the largest number of URLs.

We can reversely identify the community at time t_k corresponding to $c(t_{k-1})$. When this corresponding community is just $c(t_k)$, we call the pair ($c(t_{k-1})$, $c(t_k)$) a *main line*. Otherwise, the pair is called a *branch line*. A main line can be extended to a sequence by tracking such symmetrically corresponding communities over time. A community in a main line is considered to keep its identity, and can be used for a good starting point for finding changes around its topic.

The metrics are defined by differences between $c(t_k)$ and its corresponding community $c(t_{k-1})$. To define metrics, we use following attributes representing how many URLs the focused community obtains or loses.

$N(c(t_k))$: the number of URLs in the $c(t_k)$.

$N_{sh}(c(t_{k-1}), c(t_k))$: the number of URLs shared by $c(t_{k-1})$ and $c(t_k)$.

$N_{dis}(c(t_{k-1}))$: the number of disappeared URLs from $c(t_{k-1})$ that exist in $c(t_{k-1})$ but do not exist in any community in $C(t_k)$.

$N_{sp}(c(t_{k-1}), c(t_k))$: the number of URLs split from $c(t_{k-1})$ to communities at t_k other than $c(t_k)$.

$N_{ap}(c(t_k))$: the number of newly appeared URLs in $c(t_k)$ that exist in $c(t_k)$ but do not exist in any community in $C(t_{k-1})$.

$N_{mg}(c(t_{k-1}), c(t_k))$: the number of URLs merged into $c(t_k)$ from communities at t_{k-1} other than $c(t_{k-1})$.

Then our evolution metrics are defined as follows.

The *growth rate*, $R_{grow}(c(t_{k-1}), c(t_k))$, represents the in-crease of URLs per unit time. It allows us to find most growing or shrinking communities. The growth rate is de-fined as follows. Note that when $c(t_{k-1})$ does not exist, we use zero as $N(c(t_{k-1}))$.

$$R_{grow}(c(t_{k-1}), c(t_k)) = \frac{N(c(t_k)) - N(c(t_{k-1}))}{t_k - t_{k-1}} \quad (1)$$

stability, $R_{stability}(c(t_{k-1}), c(t))$, represents the amount of disappeared, appeared, merged and split URLs per unit time. When there is no change of URLs, the stability be-comes zero. Note that $c(t_k)$ may not be stable even if the growth rate of $c(t_k)$ is zero, because $c(t_k)$ may lose and obtain the same number of URLs. A stable community on a topic is the best starting point for finding interesting changes around the topic. The stability is defined as follows.

$$R_{stability}(c(t_{k-1}), c(t_k)) =$$

$$\frac{N(c(t_{k-1})) + N(c(t_k)) - 2N_{sh}(c(t_{k-1}), c(t_k))}{t_k - t_{k-1}} \quad (2)$$

The *novelty*, $R_{novelty}(c(t_{k-1}), c(t_k))$, represents the number of newly appeared URLs per unit time. When the novelty becomes high, we can say that $c(t_k)$ has grown mainly by newly appeared URLs. We can find most emerged communities at time t_k by sorting communities by the novelty. The following is the definition of the novelty.

$$R_{novelty}(c(t_{k-1}), c(t_k)) = \frac{N_{ap}(c(t_k))}{t_k - t_{k-1}} \quad (3)$$

The *disappearance rate*, $R_{disappear}(c(t_{k-1}), c(t_k))$, is the number of disappeared URLs from $c(t_{k-1})$ per unit time. Higher disappear rate means that the community has lost URLs mainly by disappearance. The disappear rate is defined as

$$R_{disappear}(c(t_{k-1}), c(t_k)) = \frac{N_{dis}(c(t_{k-1}))}{t_k - t_{k-1}} \quad (4)$$

The *merge rate*, $R_{merge}(c(t_{k-1}), c(t_k))$, is the number of absorbed URLs from other communities by merging per unit time. Higher merge rate means that the community has obtained URLs mainly by merging. The merge rate is defined as follows.

$$R_{merge}(c(t_{k-1}), c(t_k)) = \frac{N_{mg}(c(t_{k-1}), c(t_k))}{t_k - t_{k-1}} \quad (5)$$

The *split rate*, $R_{split}(c(t_{k-1}, c(t_k))$, is the number of split URLs from $c(t_{k-1})$ per unit time. When the split rate is low, we can know that $c(t_k)$ is larger than other split com-munities. Otherwise, $c(t_k)$ is smaller than other split com-munities. The split rate is defined as follows.

$$R_{split}(c(t_{k-1}), c(t_k)) = \frac{N_{sp}(c(t_{k-1}), c(t_k))}{t_k - t_{k-1}} \quad (6)$$

By combining these metrics, we can represent some complex evolution patterns. For example, a community has stably grown when its growth rate is positive, and its disappearance and split rates are low. Similar evolution patterns can be defined for shrinkage.

Longer range metrics (more than one unit time) can be calculated for main lines. For example, the novelty metrics of a main line ($c(t_i)$, $c(t_{i+1})$, ..., $c(t_j)$) is calculated as

Year	Period	Crawled pages	Total URLs	Links
1999	Jul. to Aug.	17M	34M	120M
2000	Jun. to Aug.	17M	32M	112M
2001	Early Oct.	40M	76M	331M
2002	Early Feb.	45M	84M	375M

Table 1: Details of our web archives

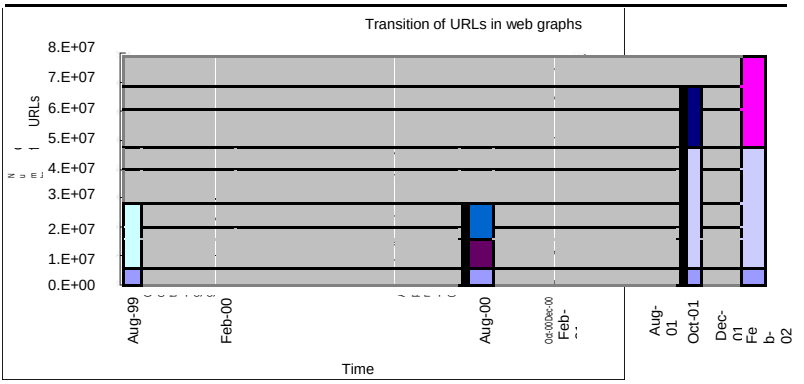


Figure 3.2.2.1: Transition of URLs in web graphs

follows.

$$Rnovelty(c(ti),(c(tj)) = \frac{P_k^{j} {}_{=i} \; Nap\,(c(tk))}{tj - ti} \quad (7)$$

Other metrics can be calculated similarly.

3.3. Analysis Of Web Archives And Evolution Of Web Communities

3.3.1 Web Archives and Graphs

For experiments, we used four web archives of Japanese web pages (in .jp domain) crawled in 1999, 2000, 2001, and 2002 (See Table 1). We used the same web crawler in 1999 and 2000, and collected about 17 million pages in each year. In 2001, the number of pages became more than twice of the 2000 archive, since we improved the crawling rate. Our crawlers collected pages in the breadth-first order.

From each archive, we built a web graph with URLs and links by extracting anchors from all pages in the archive. Our graph included not only URLs inside the archive, but also URLs outside pointed to by inside URLs. As a result, the graph included URLs outside .jp domain, such as .com and .edu. Table 1 also shows the number of links and the total URLs. For efficient link analysis, each web graph was stored in a main-memory database that provided out-links and in-links of a given URL. Its implementation was similar to the connectivity server [2].

We implemented the whole system on Sun Enterprise Server 6500 with 8 CPU and 4GB memory. Building our connectivity database of 2002 took about one day.

By comparing these graphs, we found that the Web was extremely dynamic. More than half of the URLs disappeared or changed its location in one year. We first exam-ined how many URLs in our web graphs were changed over time, by counting the number of URLs shared between these graphs. Figure 4 depicts the transition of URLs in our web graphs. Each bar shows the number of URLs, and is separated into blocks according to life spans of URLs[1]. Blocks with the same set of URLs have the same color, and lines are drawn between these blocks, so that you can see when these URLs appeared and how long they survived. As shown in Figure 3.2.2.1, about 60% of URLs disappeared from both 1999 and 2000 graphs. From 2001 to 2002, about 30% of URLs disappeared in four months. The number of URLs surviving through four archives was only about 5 million. Cho reported that more than 70% of pages survived more than one month in their four month observation. This result is close to our result from 2001 to 2002 (30% disappearance in four months). Although it is not easy to estimate the rate for one year from , we can say that our results does not deviate so much. We also examined the indegree distributions of our web graph. Indegree of a URL stands for the number of URLs pointing to the URL. Previous work, such as , shows indegree distributions fit to a *power law*, in which the probability of the positive integer value i is proportional to $1/i^k$ for a small positive number k. The reported number of k is 2.1. Indegree distributions of our graphs also exhibit a power law. The exponent of the power law is around 2.2 for all graphs. It means that our graphs are slightly sparser than ones of previous work.

Year	Period	Seeds	Communities
1999	Jul. to Aug.	657K	79K
2000	Jun. to Aug.	737K	88K
2001	Early Oct.	1404K	156K
2002	Early Feb.	1511K	170K

Table 2: The number of seeds and communities

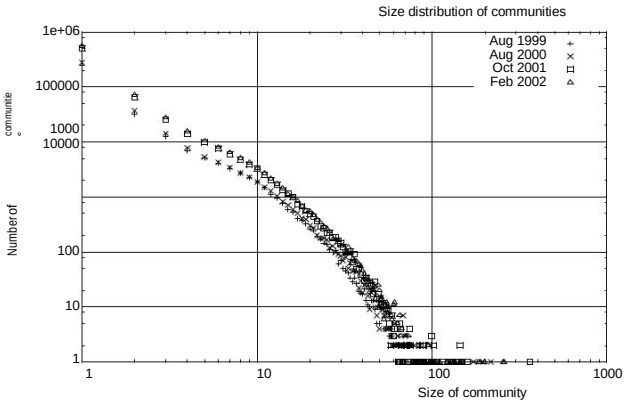


Figure 3.3.1.1: Size distribution of communities

3.3.2 Evolution of Web Community Charts

In this section, we describe the global behavior of community evolution. From the above four web graphs, we built four community charts using the technique. To compare web community charts in the same condition, we fixed values of parameters, I N and N , for the chart building algorithm in Section 2.2. We used the value R In Figure 3.2.2.1, we ignored a

few million URLs that appeared intermittently in our web graphs, e.g. URLs that appeared only in 1999 and 2001

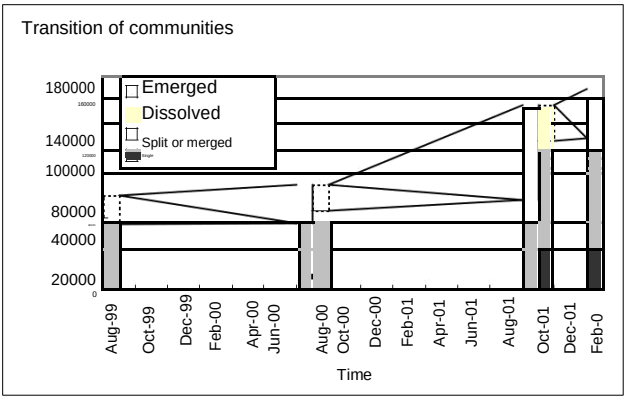


Figure 3.3.2.1 : Transition of communities

	t_{k-1} : Aug. 99 t_k : Aug. 00	Aug. 00 Oct. 01	Oct. 01 Feb. 02
# Branch lines at t_{k-1}	28,467	32,490	41,501
# Main lines	26,723	34,396	83,771
# Branch lines at t_k	29,722	41,752	44,305

Table 3: Number of main lines in split or merged communities

3.3.2.1 Size distribution

The size distribution of communities also follows the power law and its exponent did not change so much over time. Figure 3.3.1.1 shows log-log plots of the communities size distributions for all charts. All four curves roughly fit to a power law distribution with exponent 2.9 to 3.0. This result is similar to distributions of connected components in the Web graph.

3.3.2.2 Types of Changes

Although the size distribution of communities is stable, the structure of communities changes dynamically. Figure 3.3.2.1 shows how many communities are involved in each type of change from t_{k-1} to t_k . Each bar represents the number of communities in charts at the time. Bars at 2000 and 2001 are split vertically, since they have the previous and the next charts to be compared. Each block represents the number of communities involved in a particular change. Dotted blocks represent dissolved communities from t_{k-1}, and white blocks represent emerged communities. Gray blocks represent com-munities that are involved in split or merge. Finally, black blocks represent single communities that are not involved in these changes, but may grow or shrink.

We can see that the structure of our chart changes mainly by split and merge, in which more than half of communities are involved. The number of single communities is small (10% in 1999, 14% in 2000, and 25% in 2001). Since the seed sets of the charts are stable parts in our archives, the number of communities dissolved from each chart is rather small. About 24% to 30% of communities are dissolved in one year from 1999 to 2001, while 20% of communities are dissolved in four months from 2001.

Changes by split and merge are complicated, since split communities may be merged with other communities in the next time. However, it is not totally chaotic. We can see rather stable communities by extracting main lines. Table 3 shows the number of main lines and branch lines in each intervals. About half of survived (not dissolved) communities in 1999 and 2000 are included in main lines for one year, and about 66% of survived communities in 2001 are included in main lines for four months. Note that the main lines include single communities. We can use those main lines as a good starting point for finding changes around the topic

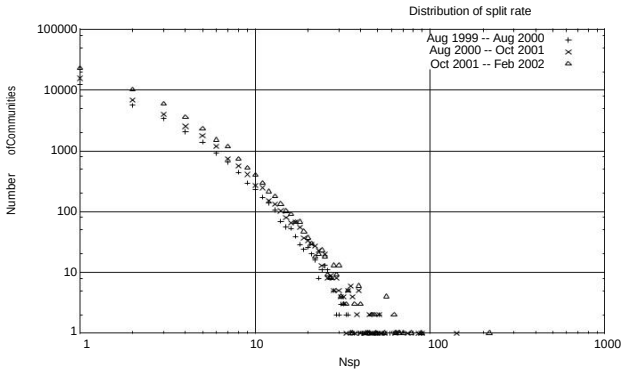


Figure 3.3.2.1: Distribution of split rate

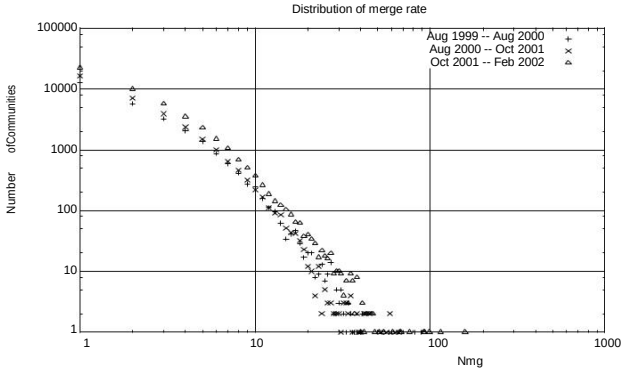


Figure 3.3.2.2: Distribution of merge rate

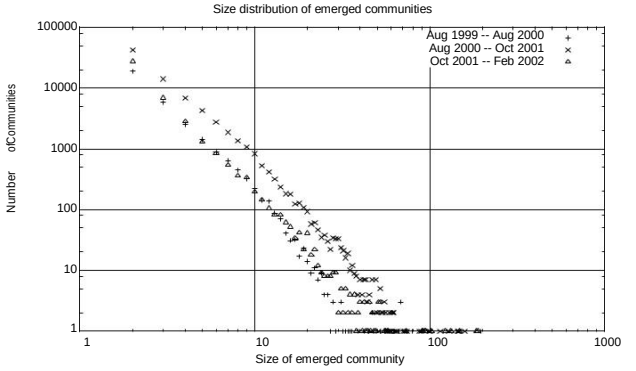


Figure 3.3.2.3: Size distribution of emerged communities

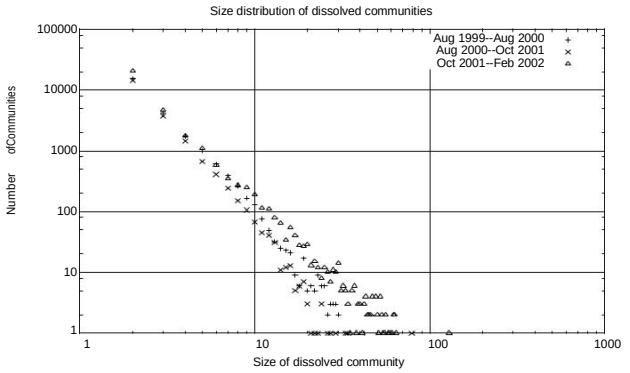


Figure 3.3.2.4: Size distribution of dissolved communi-ties

3.3.2.3 Split and Merged Communities

We first show the distributions of the split rate and merge rate in Figure 3.3.2.1 and 3.3.2.2. We plot the number of split or merged communities as a function of the number of split and merged URLs (N_{sp} and N_{mg}) in the log-log scale. Both distributions roughly follow the power law, and show that split or merge rate is small in most cases. Their shapes and scales are also similar. That is, when communities at t_{k-1} split with a split rate, almost the same number of communities are merged at t_k with the same rate as the split rate. This symmetry is part of the reason why the size distribution of communities does not change so much.

3.3.2.4Emerged and Dissolved Communities

The size distributions of emerged and dissolved communities also follow the power law, and contribute to preserve the size distribution of communities. Figure 9 and 10 show these distributions for all periods in the log-log scale. In most cases, the exponent of the power law is greater than 3.2, while the exponent of the whole chart is around 3.0. This means that small communities are easy to emerge and dissolve.

3.3.2.5Growth Rate

Finally, we examine the distribution of the growth rate. Figure 3.3.2.5 shows the number of communities as a function of the growth rate. We use a log scale on the y-axis. For simplicity, we only plot the growth rate of main lines in this graph. The growth rate is small for most of communities, and the graph has clear y-axis symmetry. This is also part of the reason why the size distribution of communities is preserved over time.

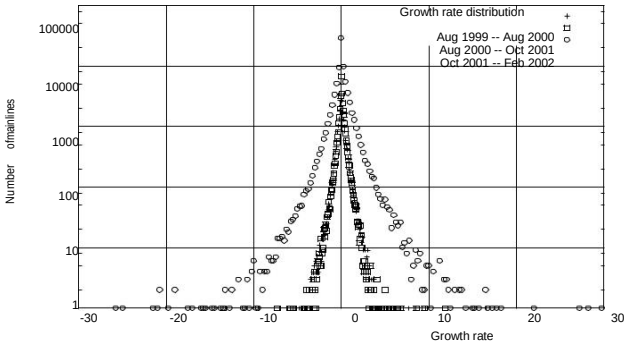


Figure 3.3.2.5: Distribution of growth rate

3.2. Detecting Communities in Social Networks

3.2.1 Introduction

Relations of real-world entities are often represented as networks, such as social networks connected with friendships or co-authorships. In many cases, real social networks contain denser parts and sparser parts. Denser sub networks correspond to groups of people that are closely connected with each other. Such denser sub-networks are called "communities" in this chapter. Detecting communities from given social networks are practically important for the following reasons:

Communities can be used for information recommendation because members of the communities often have similar tastes and preferences. Membership of detected communities will be the basis of collaborative filtering.

Communities will help us understand the structures of given social networks. Communities are regarded as components of given social networks, and they will clarify the functions and properties of the networks.

Communities will play important roles when we visualize large-scale social networks. Relations of the communities clarify the processes of information sharing and information diffusions, and they may give us some insights for the growth the networks in the future.

In general, detecting communities is not an easy task. There are many definitions of communities and most of them are computationally expensive. For example, detecting communities by optimizing modularity is NP-hard.

3.3 Definition of Community

The word "community" intuitively means a sub network whose edges connecting inside of it (intra community edges) are denser than the edges connecting outside of it (intercommunity edges). Definitions of community can be classified into the following three categories.

3.3.1 Local definitions

The attention is focused on the vertices of the sub network under investigation and on its immediate neighborhood. Local definitions of community can be further divided into self-referring ones and comparative ones. The former considers the sub network alone, and the latter compares mutual connections of the vertices of the sub network with the connections with external neighbors.

The examples of self-referring definitions are clique (a maximal sub networks where each vertex is adjacent to all the others), n-clique (a maximal sub network such that the distance of each pair of vertices is not larger than n), and k-plex (a maximal sub network such that each vertex is adjacent to all the others except at most k of them).

The examples of comparative definitions are LS set (a sub network where each vertex has more neighbors inside than outside of the sub network), and weak community (the total degrees of the vertices inside the community exceeds the number of edges lying between the community and the rest of the network).

3.3.2 Global definitions

Global definitions of community characterize a sub network with respect to the net-work as a whole. These definitions usually starts from a null model, in another words, a network which matches the original network in some of its topological features, but which does not display community structure. Then, the linking properties of sub networks of the initial network are compared with those of the corresponding sub networks in the null model. If there is a wide difference between them, the sub networks are regarded as communities. The simplest way to design a null model is to introduce randomness in the distribution of edges among vertices. The most popular null model is that proposed by Newman and Girvan . It consists of a randomized version of the original network, where edges are rewired at random, under the constraint that each vertex keeps its degree. This null model is the basic concept behind the definition of modularity, a function which evaluates the goodness of partitions of a network into communities.

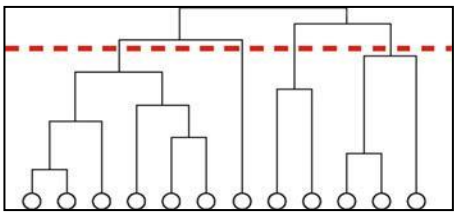


Fig. 3.3.1 Dendrogram

3.3.3 Definitions Based on Vertex Similarity

Definitions of the last category is based on an assumption that communities are groups of vertices which are similar to each other. Some quantitative criterion is employed to evaluate the similarity between each pair of vertices. Similarity

measures are at the basis of the method of hierarchical clustering. Hierarchical clustering is a way to find several layers of communities that are composed of vertices similar to each other. Repetitive merges of similar vertices based on some quantitative similarity measures will generate a structure shown in Fig. 3.3.1. This structure is called dendrogram, and highly similar vertices are connected in the lower part of the dendrogram. Subtrees obtained by cutting the dendrogram with horizontal line correspond to communities. Communities of different granurality will be obtained by changing the position of the horizontal line.

3.4 Evaluating Communities

In general, there are many ways of partitioning given network into communities. It is necessary to establish which partition exihibit a real community structure. Therefore, we need a quality function for evaluating how good a partition is. The most popular quality function is the modularity of Newman and Girivan . It can be written in several ways.

$$Q = \frac{1}{2m} \sum_{ij} \left(A_{ij} - \frac{k_i k_j}{2m} \right) \iota(C_i, C_j) \tag{3.4.1}$$

where the sum runs over all pairs of vertices, A is the adjacency matrix, k_i is the degree of vertexi i and m is the total number of edges of the network. The element of A_{ij} of the adjacency matrix is 1 if vertices i and j are connected, otherwise it is 0. The ι-function yields 1 if vertices i and j are in the same community, 0 otherwise.

Modularity can be rewritten as follows.

$$Q = \sum_{s=1}^{n_m} \left[\frac{l_s}{m} - \left(\frac{d_s}{2m} \right)^2 \right] \tag{3.4.2}$$

where n_m is the number of communities, l_s is the total number of edges joining vertices of community s, and d_s is the sum of the degrees of the vertices of s. In the above formula, the first term of each summand is the fraction of edges of the network inside the community, whereas the second term represents the expected fraction of edges that would be there if the network were a random network with the same degree for each vertex. Therefore, the comparison between real and expected edges is expressed by the corresponding summand of the above formula. Figure 3.4.1 illustrates the meaning of modularity.

The latter formula implicitly shows the definition of a community: a sub network is a community if the number of edges inside it is larger than the expected number in modularity's null model. If this is the case, the vertices of the sub network are more tightly connected than expected. Large positive values of Q are expected to indicate good partitions. The modularity of the whole network, taken as a single community, is zero. Modularity is always smaller than one, and it can be negative as well. Modularity has been employed as quality function in many algorithms. In addition, modularity optimization is a popular method for community detection.

Modularity values cannot be compared for different networks. Another thing we have to remember is that modularity optimization may fail to identify communities smaller than a scale which depends on the total size of the network and on the degree of interconnected-ness of the communities, which is called resolution limit.

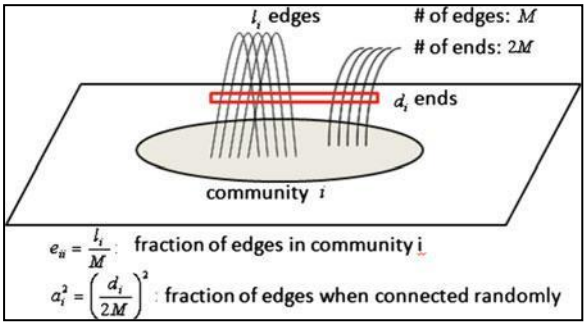


Fig. 3.4.1 Modularity

3.5 Methods for Community Detection

There are naive methods for dividing given networks into sub networks, such as graph partitioning, hierarchical clustering, and k-means clustering. However, these methods needs to provide the numbers of clusters or their size in advance. It is desirable to devise methods that have abilities of extracting a complete information about the community structure of networks. The methods for detecting communities are roughly classified into the following categories: (1) divisive algorithms, (2) modularity optimization, (3) spectral algorithms, and (4) other algorithms.

3.5.1 Divisive Algorithms

A simple way to identify communities in a network is to detect the edges that connect vertices of different communities and remove them, so that the communities get disconnected from each other.

The most popular algorithm is that proposed by Girvan and Newman. In this algorithm, edges are selected according to the values of measures of edge centrality, estimating the importance of edges according to some property on the network. The steps of the algorithm are as follows: (1) Computation of the centrality of all edges, Removal of edge with largest centrality, (3) Recalculation of centralities on the running network, and (4) Iteration of the cycle from step (2).

Girvan and Newman focuses on the concept of edge betweenness. Edge betweenness is the number of shortest paths between all vertex pairs that run along the edge. It is intuitive that intercommunity edges have a large value of the edge betweenness, because many shortest paths connecting vertices of different communities will pass thorough them. In the example of Fig. 3.5.1, two communities will be obtained by removing the central edge, whose edge betweenness is the largest.

3.5.2 Modularity Optimization

As mentioned above, modularity is a quality function for evaluating partitions. Therefore, the partition corresponding to its maximum value on a given network

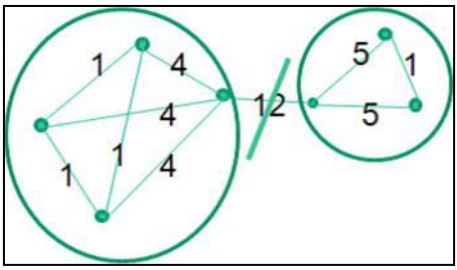


Fig. 3.5.1 Detecting communities based on edge betweenness

should be the best one. This is the main idea for modularity optimization. An exhaustive optimization of Q is impossible since there are huge number of ways to partiton a network. It has been proved that modularity optimization is an NP-hard problem. However, there are currently several algorithms that are able to find fairly good approximations of the modularity maximum in a reasonable time. One of the famous algorithms for modularity optimization is CNM algorithm proposed by Clauset et al. Another examples of the algorithms are greedy algorithms and simulated annealing.

3.5.3 Spectral Algorithms

Spectral algorithms are to cut given network into pieces so that the number of edges to be cut will be minimized. One of the basic algorithm is spectral graph bipartitioning. As a concrete procedure, Laplacian matrix L of given network is used. The Laplacian matrix L of a network is an n n symmetric matrix, with one row and column for each vertex. Laplacian matrix is defined as L D D A, where A is the adjacency matrix and D is the diagonal degree matrix with Di i D k A*ik*. All eigenvalues of L are real and non-negative, and L has a full set of n real and orthogonal eigenvectors. In order to minimize the above cut, vertices are partitioned based on the signs of the eigenvector that corresponds to the second smallest eigen-value of L. In general, community detection based on repetitive bipartitioning is relatively fast. Newman proposes a method for maximizing modularity based on the spectral algorithm.

3.5.4 Other Algorithms

There are many other algorithms for detecting communities, such as the methods focusing on random walk, and the ones searching for overlapping cliques. Danon compares the computational costs and their accuracies of major community detection methods.

3.5.5. Methodologies of Network Community Mining

In view of the basic strategies adopted, most of the existing community mining algorithms can be classified into two main categories: *optimization based algorithms* and *heuristic based algorithms* (Fig. 3.5.5). The former solves an NCMP by transforming it into an optimization problem and trying to find an optimal solution with respect to a pre-defined objective function, such as various cut criteria adopted by spectral methods the evaluation function introduced by the Kernighan– Lin algorithm, the network modularity employed in several algorithms and others. On the contrary, heuristic algorithms do not explicitly state optimization objectives, and they solve an NCMP based on certain intuitive assumptions or heuristic rules. For example, the heuristic rule used in the maximum flow community (MFC) algorithm is based on the assumption that "flows" through inter-community links should be larger than those of intra-community links. Similarly, the heuristic rule employed by the GN algorithm is that the "edge betweenness" of inter-community links should be larger than that of intra-community links. Others such as the Wu-Huberman algorithm, the HITS algorithm, the CPM, and the FEC have adopted different assumptions.

3.5.5.1 Optimization Based Algorithms

Spectral methods and local search based methods are two representatives of optimization based algorithms for solving NCMPs.

Spectral methods optimize certain predefined *cut* criteria by using the quadratic optimization technique. In graph theory, the *cut* of a bipartition of a network is defined as the number of inter-group links. An optimal bipartition of a network is the one with the minimum cut. Yet, in most cases, the minimum cut criterion

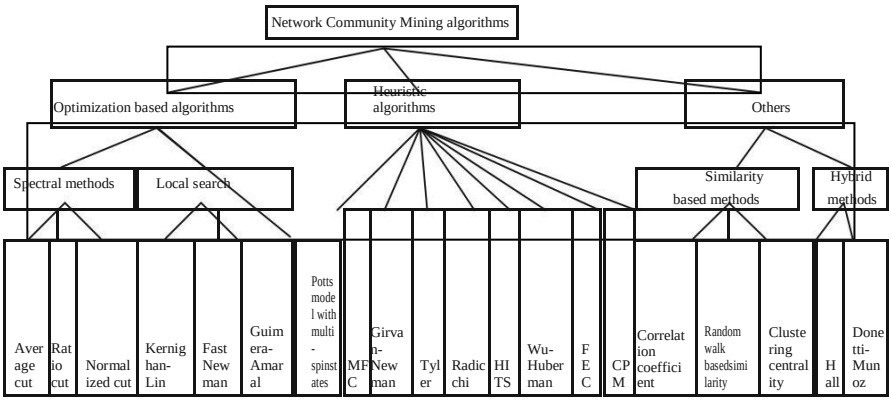


Fig. 3.5.5 The classification chart for some of the existing community mining algorithms

will lead to bias partitions. In order to avoid this problem, other criteria, such as *average cut*, *ratio cut*, *normalized cut*, and their variants, have been proposed to compute the density, instead of the number, of inter-group links. Unfortunately, the problems of finding different optimal cuts have been proven to be NP-complete. Based on matrix theory, the spectral methods try to find an approximately optimal cut by transforming the problem into a constraint quadratic optimization problem represented as $min.X^T MX/=.X^T X/$, where X denotes the indicator vector of a bipartition, and M is a symmetric positive semi-definite matrix. In the case of minimizing the average cut, M corresponds to the Laplacian matrix of a given graph. In the case of minimizing the ratio cut, normalized cut, or others, M corresponds to a variant of the Laplacian matrix. Thus, an approximately optimal solution of the constraint quadratic optimization problem can be obtained by means of calculating the second smallest eigenvector of M.

The Kernighan–Lin algorithm, the fast Newman algorithm, and the Guimera–Amaral algorithm are three popular local search based optimization methods for solving NCMPs. The Kernighan–Lin algorithm (or KL for short) aims to minimize an evaluation function defined as the difference of the numbers of intra-community links and inter-community links. Starting from an initial partition of a network, in each iteration, KL moves or swaps nodes between communities in order to decrease the evaluation function. This iterative process stops when the evaluation function re-mains unchanged. KL runs moderately fast with the time complexity of $O(n^2)$. During the local search process, KL only accepts better neighbor solutions and re-jects all worse ones, and thus it often finds a local, rather than a global, optimal solution. The principal restriction of KL is that it needs to have some prior knowledge, such as the number, as well as the average size of, communities in order to generate an initial partition. KL is also sensitive to initial partitions; that is, a bad one could result in a slow convergence and hence a poor solution.

Newman has presented a faster algorithm (or FN for short) for detecting community structures with the time complexity of O .m n/. In essence, FN is also a local search based optimization method. Starting from an initial state in which each community only contains a single node, FN repeatedly joins communities together in pairs by choosing the best merge, until only one community is left. In this bottom-up way, the *dendrogram* of community structure is constructed. In order to choose the best merge in each iteration, a new metric, *modularity*, is proposed to quantitatively measure how well-formed a community structure is. The modularity of a given network in terms of a Q-function is defined as follows:

$$Q = \sum_i (e_{ii} - a_i^2) \qquad (3.5.5.1)$$

where e_{ij} denotes the fraction of all weighted links in networks that connect the nodes in community i to the those in community j , and $a_i = \sum_j e_{ij}$. It is expected that better partitions of a given network will be those with larger Q-values.

The algorithm proposed by Guimera and Amaral (or GA for short) also tries to find a partition of a network with the maximum modularity.

3.5.5.2 Heuristic Methods

The maximum flow community (MFC) algorithm, the Girvan–Newman algorithm (GN) and its improvements, the Hyperlink Induced Topic Search algorithm (HITS), the Wu–Huberman algorithm (WH) and the Clique percolation method (CPM) are typical heuristic based algorithms for solving NCMPs.

The MFC algorithm was proposed by Flake et al. based on the Max Flow-Min Cut theorem in graph theory. The basic assumption behind is that the maximum flow through a given network is decided only by the capacity of network "bottle-necks," which is the capacity of the Min-Cut sets, and the sparse inter-community links can be regarded as the "bottlenecks" in the flow within the network. There-fore, the inter-community links can be discovered by calculating the Min-Cut sets. By iteratively removing "bottleneck" links, involved communities will be gradually separated from each other. Based on MFC, Flake and his colleagues proposed a method for discovering Web communities and verified an interesting hypothesis through their experiments, that is, self-organized and hyperlink-based Web community are highly topic-related. So far, the best time for calculating the Min-Cut sets of a graph is O .m nl og.n^2 = m// , which decides the time of each bipartition happened in MFC. But in practice, MFC runs fast because the Min-Cut computation is restricted within a fairly small area around some pre-selected Web pages rather than the global Web.

Similarly, the GN algorithm detects all communities by recursively breaking inter-community links. But, different from MFC, the heuristic rule introduced in GN is that the inter-community links are those with the maximum "edge between-ness," which is defined as the number of geodesic paths running through a given link. The GN algorithm is a hierarchical method and can produce a dendrogram of community structure in a top-down fashion. Its time complexity is O .m^2 n/, which makes it not suitable for large-scale networks. In order to speed up the basic GN algorithm, several improvements have been proposed.

Tyler et al. introduced a statistical technique into the basic GN algorithm. Instead of computing the exact edge betweenness of all links, they used the Monte Carlo method to estimate an approximate edge betweenness value for a selected link set. Inevitably, an improvement in speed is gained at the price of a reduction in accuracy.

Because calculating edge betweenness is time-consuming, Radicchi et al. defined a new metric, called *link clustering coefficient*, to replace it. The assumption behind their method is that inter-community links are unlikely to belong to a short loop, such as triangles and squares; otherwise, links in the same loop would likely be across communities, and thus such links would inevitably increase the density of inter-community links. Using this heuristic rule, they define the link clustering coefficient as the number of triangles or squares in which a link is involved. In each iterative step, links with the minimum link clustering coefficient will be cut off. The average time complexity for computing the link clustering coefficient of all links is O .m^3 = n^2 /, lower than that for computing edge betweenness, which is O .m n/. Thus, their method is on average faster than GN with the time complexity of O .m^4= n^2 /.

Similar to MFC, HITS presented by Kleinberg aims to discover hyperlink-based Web communities. The basic assumption behind HITS is that there exist authorities and hubs on the Web, and authorities are often pointed to by hubs that preferentially point to authorities. Based on the mutually reinforcing relationship between

authorities and hubs, they developed an iterative method for inferring such authority-hub communities from the Web by computing the principal eigenvectors of two special matrices in terms of the adjacency matrix of the Web. A search engine based on HITS can return the most topic-related pages to users.

In the WH algorithm proposed by Wu and Huberman, a network is modeled as an electrical circuit by allocating one unit resistor on each link. Then, it selects two nodes from two distinct communities as the positive and negative poles, respectively. The assumption of the WH algorithm is that the resistance within communities will be much less than that between communities because the intra-community links are much denser than inter-community links, and thus voltage difference of distinct communities should be more significant. Based on this heuristic, the WH algorithm can separate the group with a high voltage and the group with a low voltage from a network, by finding two maximum gaps in the node sequence sorted by their respective voltage values. Then, it determines the final division by considering the co-occurrence of nodes in such separated groups. As authors argued, their method is very fast with a linear time in terms of the size of a network. However, the WH algorithm depends heavily on its prior knowledge, which is hard to obtain beforehand. For instance, it needs to identify two "poles" belonging to different communities. Also, it needs the approximate size of each community in order to find multiple communities.

Palla and his colleagues presented CPM to discover an overlapping community structure. They assumed that a network community is made of "adjacent" k cliques, which share at least k 1 nodes with each other. Each clique uniquely be-longs to one community, but cliques within different communities may share nodes. Using the above heuristic information, CPM is able to find the overlaps of communities. For a given K, CPM first locates all k cliques (k K) from a given network, and then build a clique-clique overlap matrix to find out communities in terms of different k. But in practice, for an unknown network, it is not easy to decide what value of k will result in a more reasonable community structure.

3.5.5.3 Other Methods

Besides the above-discussed two main categories, there exist some other algorithms for solving NCMPs. For example, we can cluster a network through a bottom-up approach by repetitively joining pairs of current groups based on their similarities, such as correlation coefficients and random walk similarities, which are defined in terms of their linkage relation. Also, we can first transform an NCMP into a clustering problem in a vector space by allocating a k-dimensional coordinate to each node, and then cluster such spatial points using any typical spatial clustering algorithm, such as k-mean. Back in 1970, Hall had proposed an efficient way for transforming a network into a group of one-dimensional points by using a weighted quadratic equation in order that the nodes with dense links will be put close together, and those with sparse links will be put far away from each other. Recently, Donetti and Munoz proposed a method for solving NCMPs based on quite a similar idea. Their method maps a network into a k-dimensional vector space using the smallest eigenvectors of the Laplacian matrix before clustering spatial points.

3.6 Tools for Detecting Communities

Several tools have been developed for detecting communities. These are roughly classified into the following two categories: detecting communities from large-scale networks, and interactively analyzing communities from small networks.

3.6.1 Tools for Large-Scale Networks

3.6.2 Tools for Interactive Analysis

There are many tools for interactively visualizing and analyzing small networks.

JUNG
Pajek
igraph
SONIVIS
Commetrix
Network Workbench
visone
CFinder

As an example, community detection of igraph is shown below. Igraph is a free software for network analysis, and it can be used as the libraries of C programing language, and as the combination with R, a programming language for statistical analysis. Figure 3.6.2.1 is the screenshot of the combination RCigraph. Details of R are available at http://cran.r-project.org/, and details of igraph are available at http://igraph.sourceforge.net/.

As a typical example in the field of social network analysis, Zachary's karate club network (Fig. 3.6.2.2) is used. This network shows the relations of certain Karate club members in a university. While Zachary was observing the relations of the members, a club administrator and a teacher were opposed to each other and the club was splitted into two factions. Figure 3.6.2.2 shows the vertices that belong to each faction. Reproducing the factions from given social network is often employed as a testbed of community detection methods. Karate club network represented in GML

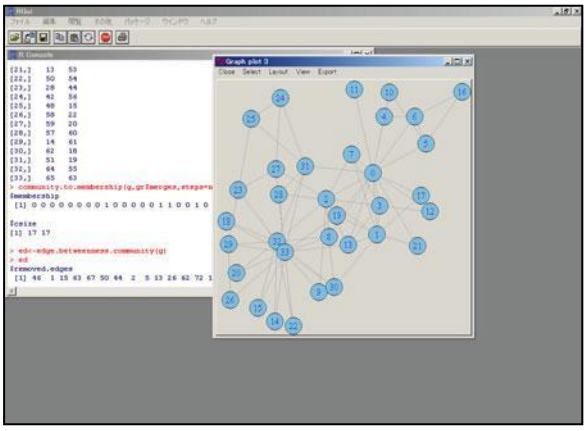


Fig. 3.6.2.1 Screenshot of RCigraph

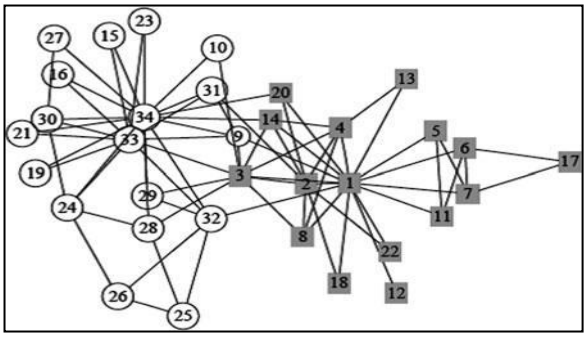


Fig. 3.6.2.2 Karate club network

format can be downloaded from Mark Newman's Web site. Graph modelling language (GML) is one of the formats for representing networks. The following shows the processes of community detection of RCigraph. Explanation of each command are shown after #.

Figure 3.6.2.5 shows the result of community detection by fastgreedy, which maximize modularity in a greedy approach. The command read.graph loads network data, and the command tkplot perform visualization (Fig. 3.6.2.3). Initial positions of vertices are assigned randomly, and users can select major algorithms for visualization, such as Kamada–Kawai and Fruchterman–Reingold. Positions of vertices can be adjusted manually.

The command fastgreedy.community is for maximizing modularity greedily, and its results are stored in variable gr. The variable gr is composed of gr$modularity (a list of modularity values in the process of maximization) and gr$merge (vertex IDs

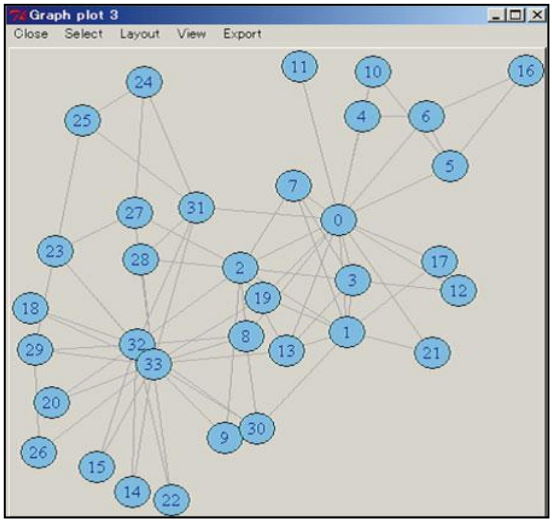


Fig. 3.6.2.3 Visualization of karate club network by igraph

that are merged at each step). The command communty.to.membership generates m$membership (membership of each vertex) and m$csize (size of each community). This example shows that the first eight vertices of lower numbers belong to community 0 and the sizes of the two communities are both 17. After storing the membershop m$membership to V(g)$color, the second tkplot command specifies a visualization algorithm and colors of vertices in order to perform visualization. Vertices of each community are colored as shown in Fig. 3.6.2.4.

As you can see from the figures, the only difference between the communities obtaind by fastgreedy and actual factions in Fig. 3.6.2.2 is the tenth vertex in Fig. 3..6.2.2 (the ninth vertex in Fig. 3.6.2.4). Please keep in mind that numbers starts from 1 in R, while from 0 in igraph.

Other methods for detecting communities are also available in igraph. Figure 3.6.2.6 shows community detection based on edge betweenness.

The command edge.betweenness.community detect communities by repetitively removing edges of high edge betweenness. The contents of the obtained vari-able ed is different from previous fastgreedy.community: ed$removed.edges (list of removed edges), ed$edge.betw eenness (list of edge betweenness), ed$merges (list of vertices that are merged in a dendrogram), and ed$bridges (list of bridging edges). Edge betweenness divides given network in an top-down manner, while ed$merges shows list of vertices in a reversed order (just like bottom-up manner). You can see that communities detected based on edge betweenness are differnet from the actual factions in Fig. 3.6.2.2.

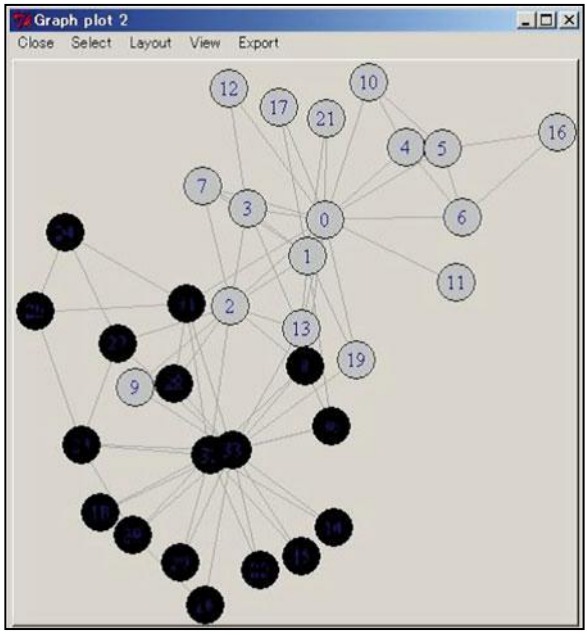


Fig. 3.6.2.4 Community detection by fastgreedy algorithm

```
> library("igraph")                          #load igraph library
> setwd("R")                                 #change directory
> dir()                                      #show files
[1] "karate.gml"
> g<-read.graph("karate.gml",format="gml") #load network data
> tkplot(g)                                  #visualize network
[1] 1
> gr<-fastgreedy.community(g)                #modularity optimization
> gr$modularity                              #list of modularity values
 -0.04980276 -0.03763971 -0.01397107 -0.00147929  0.01775148  0.04906312
...
[31]   0.37598619  0.38067061  0.37179487  0.00000000
> gr$merge                                   #list of merged vertices
      [,1]  [,2]
 [1,]    5    16
...
[33,]   65   63
> m<-community.to.membership(g,gr$merges,steps=nrow(gr$merges)-1)
                                             #store membership and
                                             size #show the contents

     0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 1 0 0 1 0 1 0 1 1 1 1 1 1 1 1 1 1 1
1 $csize
 17 17
> V(g)$color<-m$membership                   #store membership
 V(g)$color
     0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 1 0 0 1 0 1 0 1 1 1 1 1 1 1 1 1 1 1
 tkplot(g,layout=layout.kamada.kawai,vertex.color=V(g)$color)
                                             #visualize network
 2
>
```

Fig. 3.6.2.5 Community detection by fastgreedy in R C igraph

```
> ed<-edge.betweenness.community(g)          #detect communities
> ed                                         #show result
$removed.edges                               #list of removed edges
  [1] 46  1 15 63 67 50 44  2  5 13 26 62 72 17  7  8  6 18 76 60 73 36 39 16
...
 [73] 57 70 48 58 74 77
$edge.betweenness                            #list of edge betweenness
  71.392857  66.895177  77.317399  82.002906 123.232917 100.205556
...
  1.500000   3.000000   1.000000   1.000000   2.000000   1.000000
$merges                                      #list of merged vertices
       [,1]  [,2]
  [1,]   32    33
...
 [33,]   65   64
$bridges                                     #list of bridges
     78 77 75 74 71 70 68 67 66 65 63 61 58 55 53 50 49 47 45 43 41 39 37 35
     33 31 29 27 25 24 18 14 11
 community.to.membership(g,ed$merges,steps=nrow(ed$merges)-1)
                                             #membership of two communities
$membership
     1 1 0 1 1 1 1 1 0 0 1 1 1 1 0 0 1 1 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0
0 $csize
 19 15
> community.to.membership(g,ed$merges,steps=nrow(ed$merges)-2)
                                             #membership of three communities
$membership
     0 0 1 0 0 0 0 0 1 2 0 0 0 0 1 1 0 0 1 0 1 0 1 1 1 1 1 1 1 1 1 1 1 1
1 $csize
 15 18  1
> community.to.membership(g,ed$merges,steps=nrow(ed$merges)-3) #membership of
                                             four communities
$membership
```

```
     1 1 0 1 2 2 2 1 0 3 2 1 1 1 0 0 2 1 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0
0 $csize
 18 10  5  1
>
```

Fig. 3.6.2.6 Community detection based on edge betweenness in RCigraph

The command edge.betweenness.community generates a dendrogram. Communities of different granularity can be obtained from the dendrogram: if two communities of size 19 and 15 are divided further, communities of size 18, 1, 10, and 5 are obtained.

Many other methods for community detection are installed in RCigraph.

label.propagation.community community detection by label propergation
leading.eigenvector.community community detection by spectral partitioning
spinglass.community community detection based on statistical mechanics
walktrap.community community detection by random walk

3.7 Applications of Community Mining Algorithms

3.7.1 Network Reduction

Network reduction is an important step in analyzing social networks. In this section, we will illustrate how to reduce a complex network into a dendrogram, by means of community mining. The bibliography contains 360 papers written by 314 authors. Its corresponding network is a bipartite graph, in which each node denotes either one author or one paper, and link .i; j / represents author i publishing a paper j , as shown in Fig. 3.7.1.1.

Figure 3.7.1.2 provides the community structure as detected using a community mining algorithm, called ICS, in which 147 communities are uncovered from the bibliography network. As one would have expected, each community contains some papers and their corresponding coauthors.

Most of the detected communities are self-connected components. The component A is the biggest one containing 13 communities, 158 papers, and 86 authors. Figure 3.7.1.3 shows the network indicating the collaborations among these 86 coau-thors, in which link .i; j / with weight *w* denotes authors i and j have coauthored *w* papers. Then, ICS is again applied to the coauthor network and totally 14 communi-ties are uncovered, as shown in Fig. 3.7.1.4a, in which different gray degrees indicate different communities.

Moreover, the clustered coauthor network can be reduced into a much smaller one by condensing each community as one node, as shown in Fig. 3.7.1.4b.

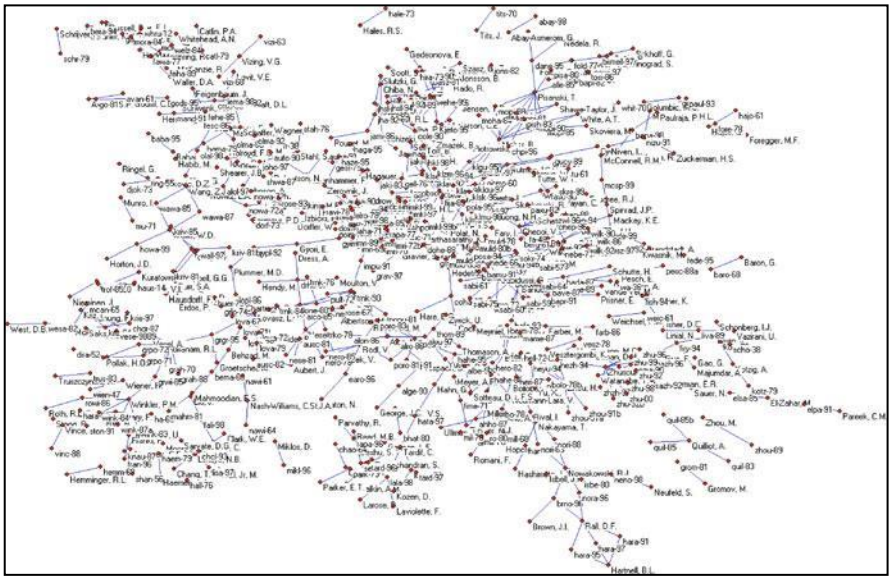


Fig. 3.7.1.1 The bibliography network for the book entitled "graph products: structure and recognition"

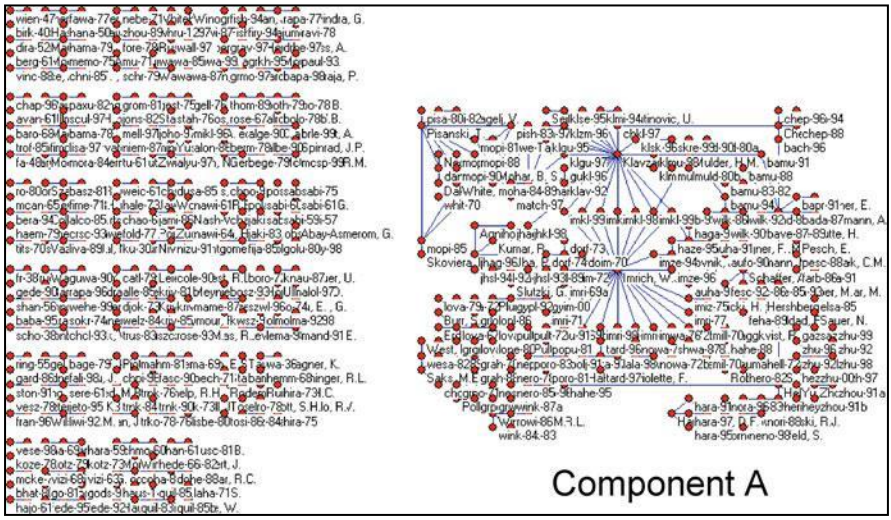


Fig. 3.7.1.2 The community structure of the bibliography network as detected using ICS

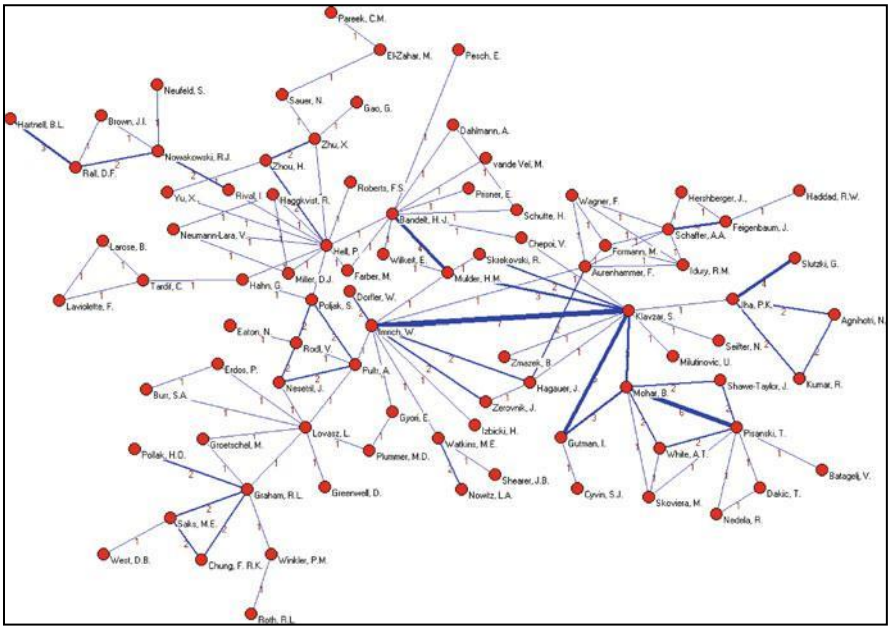


Fig. 3.7.1.3 The coauthor network corresponding to the biggest component

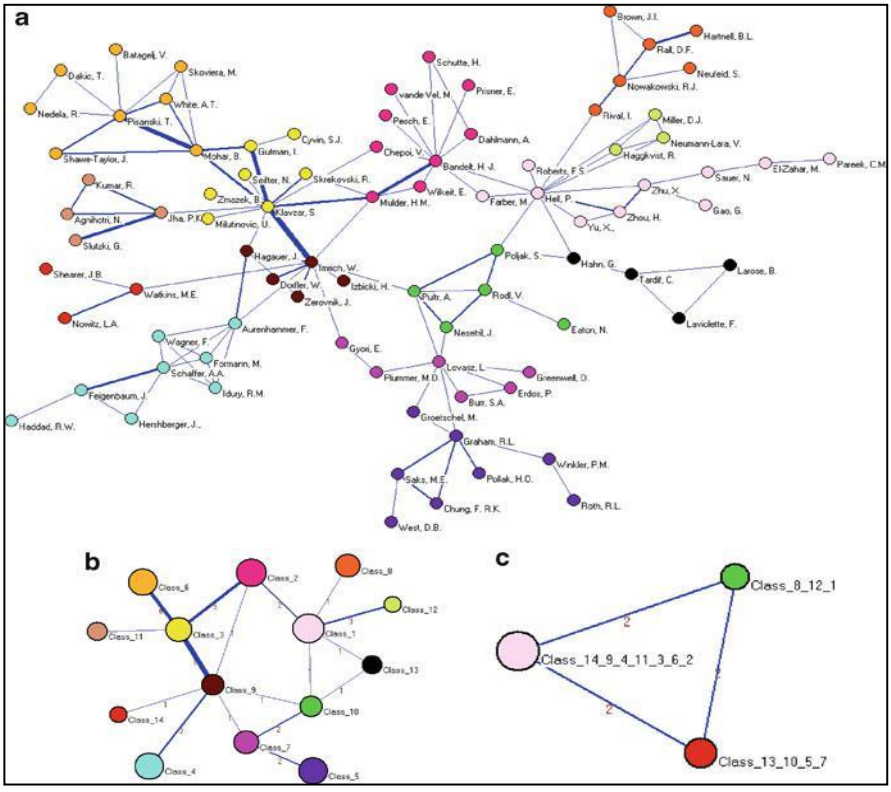


Fig. 3.7.1.4 The reduction of a coauthor network . (**a**) The community structure of the network. (**b**) The condensed network. (**c**) The top-level condensed network

3.7.2 Discovering Scientific Collaboration Groups from Social Networks

In this section, we will show how community mining techniques can be applied to the analysis of scientific collaborations among researcher. Flink is a social network that describes the scientific collaborations among 681 semantic Web researchers. The network was constructed based on semantic Web technologies and all related semantic information was automatically extracted from "Web-accessible information sources", such as "Web pages, FOAF profiles, email lists, and publication archives". Directed links between nodes, as shown in Fig. 3.7.2.2a, denote "know relationships," indicating scientific activities happening between researchers. The weights on the links measure the degrees of collaboration. From the perspective of social network analysis, one may be especially interested in such questions as: (1) among all researchers, which ones would more likely to collaborate with each other?

what are the main reasons that bind them together? Such questions can be answered by means of applying the above-mentioned community mining techniques.

Figure 3.7.2.2a shows the network structure of Flink, while Fig. 3.7.2.2b presents the output of the NCMA community algorithm proposed, where each dot de-notes a non-zero entry in the adjacency matrix and the gray bar at the righthand side encodes the hierarchical community structure in which different shades denote different hierarchical levels. Three biggest groups in the first level can be detected, which, respectively, contain 51, 294, and 336 members, as separated by the solid lines in the matrix. The first group is almost separated from the entire network, and its 49 members constitute the outside cycle of Fig. 16.8a. In total, 93 communities are detected and the average size of a community is 7.32, as shown in Fig. 16.8c. The self-organized communities would provide the answer to the first question. By referring to them, one can know the specific collaboration activities among these researchers. Approximately, we can observe a power-law phenomenon; most com-munities have a small size, while a small number of communities contain quite a large number of members.

After manually checking the profiles of members within different communities, an interesting fact has been confirmed. That is, most of communities are organized according to the locations or the research interests of their respective members. As an example, we can look into details of the largest community in which Steffen Staab is in the center, as shown in Fig. 3.7.2.2d. In this community, 22 of 39 come from Germany, 21 come from the same city, Karlsruhe, and 12 out of such 21 come from the same university, the University of Karlsruhe, where Steffen Staab works. Moreover, the community is research topic related; most of its members are interested in the topic related to ontology learning. These results might offer the clue to answer the second question, i.e., researchers in adjacent locations and with common interests prefer to intensively collaborate with each other.

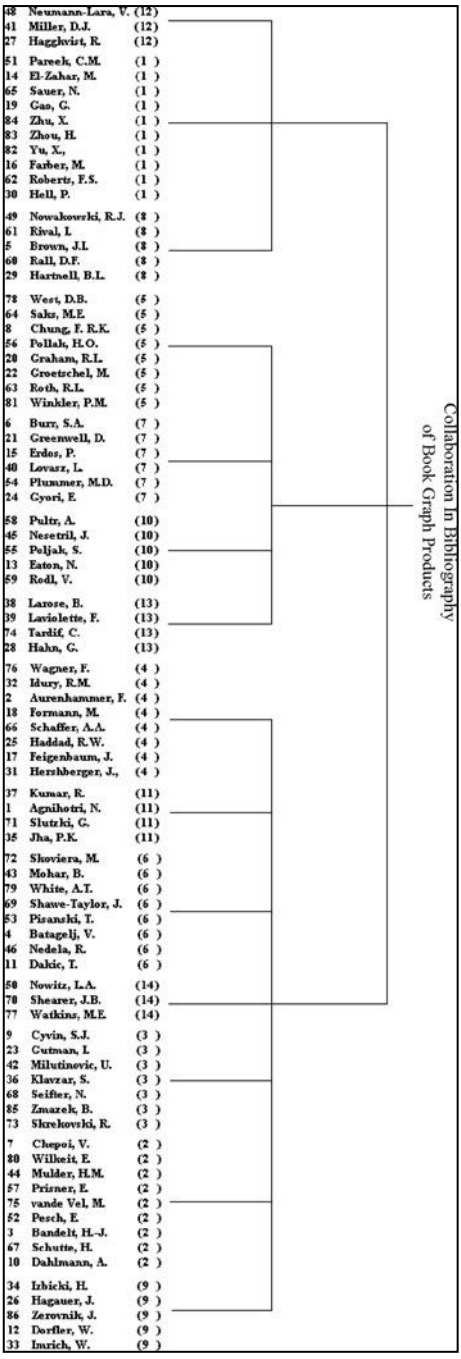


Fig. 3.7.2.1 The dendrogram of the coauthor network as shown in Fig. 3.7.1.5

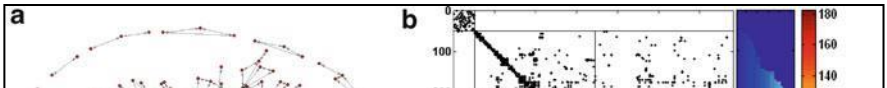


Fig. 3.7.2.2 Mining a scientific collaboration network . (**a**) The network of Flink. (**b**) The matrix with a hierarchical structure. (**c**) The statistics of detected communities. (**d**) An illustrated community

3.7.3 Mining Communities from Distributed and Dynamic Networks

Most of the existing methods for addressing NCMPs are centralized, in the sense that they require complete information about networks to be processed. Moreover, they assume the structures of networks will not dynamically change. However, in the real world, many applications involve distributed and dynamically-evolving net-works, in which resources and controls are not only decentralized but also updated frequently. In such a case, we need find a way to solve a more challenging NCMP; that is to adaptively mine hidden communities from distributed and dynamic networks. One promising solution is based on an Autonomy-Oriented Computing (AOC) approach, in which a group of self-organizing agents are utilized. The agents will rely only on their locally acquired information about networks. In what follows, we will highlight its main ideas through an example.

Intelligent Portable Digital Assistants (or iPDAs for short) that people carry around can form a distributed network, in which their users communicate with each other through calls or messages. One useful function of iPDAs would be to find and recommend new friends with common interests, or potential partners in research or business, to the users. The way to implement it will be through the following steps: (1) based on an iPDA user's communication traces, selecting individuals who have frequently contacted or been contacted with the user during a certain period of time; (2) taking the selected individuals as the input to an AOC-based algorithm, which will in turn automatically compute other members within the same community of this user in a decentralized way, by exchanging a limited number of messages with related iPDAs; (3) ranking and

recommending new persons, who might not be included the current acquaintance book, the user. In such a way, people can periodically receive recommendations about friends or partners from their iPDAs.

3.8. Decentralized Online Social Networks

3.8.1 Introduction

A decentralized online social network is an online social network implemented on a distributed information management platform, such as a network of trusted servers or a peer-to-peer systems. In contrast to centralized OSNs where the vendor bears all the cost in providing the services, a distributed or peer-to-peer OSN offers a cost-effective alternative. In fact, a P2P approach helps lower the cost of the provider drastically, e.g., the case of Skype. Other benefits of a DOSN include better control of user privacy and the enhancement of innovative development. By decentralizing OSNs, the concept of a service provider is changed, mas there is no single provider but a set of peers that take on a share of the tasks needed to run the system. This has several consequences: in terms of privacy (no central data collection, reduced economic incentive for advertisement) and operation, nor any central entity that decides or changes the terms of service whimsically. Moving from a centralized web service to a decentralized system also means that different modes of operations become possible: using one's own storage or cloud storage, delay-tolerant social networks, and local treatment of local content, to name some of them.

DOSNs combine *social* and *decentralized* elements. The path to a system with these properties vary: a DOSN can be achieved by adding one of these two properties to an existing system and thus transforming it, as by taking centralized OSNs and decentralizing them or by adding a social component to current decentralized applications that do not have a social component yet. It can also be achieved by adding both properties at the same time: decentralizing and adding a social component to current centralized applications.

3.8.2 Scope of the Chapter

Generic mechanisms for distributed systems, such as cryptography for privacy, key management, storage distribution, numbers and location of replicas, incentives for cooperation and resource-sharing, topology, p2p substrate and administration tasks, trust needed for the program, etc. may be developed either within the context of DOSNs from the start or be adapted from other contexts.

3.8.3 Challenges for DOSN

Decentralizing the existing functionality of online social networks requires finding ways for distributed storage of data, updates propagation and versioning, a topology and protocol that enables search and addressing, i.e., a mechanism to find friends in the topology, robustness against churn, openness for third-party applications, and means for content revocation (by encryption and/or time), dealing with heterogeneity of user resources, demands, online behavior, etc.

Storage. Where should content be stored? Should they be stored exclusively at nodes run by friends, or be encrypted and stored at random nodes, or should the nodes be chosen using some other heuristics such as in a DHT or based on uptime

history? As in file-sharing, there will be several answers to this question. The requirement for redundancy to provide availability of data depends to a large extent on the duration and distribution of time peers are online. These activity patterns are also influenced by the geographic distribution of the peers and shifted by time zones. The distribution of interested and authorized peers and the desired probability of availability are to be traded off with storage requirements, especially if the system should allow for storing of media files and not only links to websites where such media files can be found.

Updates. How can we deal with updates, e.g., status updates of friends? In peer collaboration systems, updates, e.g., of a workplace, are sent to a small group of peers via a decentralized synchronization mechanism. In P2P social networks, with distributed storage and replication – and a potential need for scalability, the requirements change. P2P publish/subscribe mechanisms are a possibility, but their security in terms of access control will have to be developed further. Unlike a traditional peer-to-peer environment, where many peers are involved, each of the sub-networks will be much smaller (though larger than typical collaborative groups), making it relatively simpler to realize quorum systems and deal with updates.

Topology. Should nodes be connected according to their social connection? This would cluster friends in the overlay network, which would facilitate updates. As a downside, given the possibility of a relatively small set of friends, this would limit the availability and robustness of data access. How can we build a decentralized, p2p topology suitable for social networks? In pure file-sharing networks, the topology does not depend on whether the peers know each other and nodes exchange content with any other nodes in the network. At the other end of the spectrum, existing examples of decentralized social networks (in the widest sense) are mostly platforms for collaboration or media sharing and they tend to consist of collaborative groups that are relatively closed circles, e.g., using a “ring of trust” or dark nets. In contrast, online social networking services have overlapping circles.

Search, Addressing. Related to the topic of updates above, how can users find their friends from the real social network in the P2P virtualization thereof, and conversely, how can they discover new friends by virtue of common interests. Over multiple sessions, peers may change their physical address. In a typical file sharing network, this is not an issue. One just needs to find some peer with the content it is looking for.
However friends and trust links of a social network are essential, and so it is crucial to both be able to find back friends even if they have changed their physical ad-dress, and also authenticate the identity. Traditionally, peer identity is tied with an IP address which clearly is not sufficient. However, handling peer identity in a self-contained manner in a P2P system is also feasible. It may also be difficult to maintain a complete ring like in traditional structured over-lays as an index structure, if the network is based on only social links. Recent advances in realizing distributed indexing with a ring less overlay potentially holds the key to this issue. These mechanisms composed together potentially can help maintain social network links under churn. Another search issue is how can users find out about information available concerning their interests? In social networks, tagging or folksonomies is the basic mechanism to annotate content. Recently, there have also been advances made in enabling decentralized tagging, which paves another step towards realizing social networks on top of a P2P infrastructure. Note that there is a trade-off between privacy protection and search capabilities.

Openness to New Applications. One of the most alluring features of current online social networks is that they are open to third-party applications, which enables a constant change of what a social networking service provides to the users. There is a core functionality for maintaining social ties, such as profile information, connection to friends, status updates, internal messaging, posting on each other's sites, events notification. In addition, third-party applications provide more and unpredictable ways of contacting users, finding out about other users' interests, forming groups and group identities, etc. This openness to extensions potentially provides great benefits for the users. The price for these benefits is the risk that comes with opening the service to untrusted third parties, extending the privacy problem from the single service provider to all application providers. In a decentralized environment, if some users choose to enable a third-party application, their choice should not affect other users or even users connected directly to them. How to draw this boundary is an open and challenging question.

Security. Keeping control over their data with the user implies the need for security support, so the classical requirements for security (confidentiality, access control, integrity, authentication and non-repudiation) apply, albeit modified for the context of decentralized social networks. The main questions for user control are in the domain of access control, e.g., how can we ensure that only authorized friends can access content. For distributed storage with other peers that the user not necessarily wants to access data, the content has to be encrypted, as done for example for file backup or anonymous peer-to-peer file-sharing. To manage access to encrypted data, key distribution and maintenance have to be handled such that the social network group can access data but be flexible enough to handle churn in terms of going offline and coming back, additions and removal to the user's social network. Group membership research has dealt with questions of key management and renewal and how to give access to new members of a group by issuing new keys in rounds. Likewise dark nets also share a key within the group. Such existing mechanisms are however grossly inadequate to meet the finer granularity of access control needs for social network features.

Even in the simplest scenario, where all members are allowed full access, if one wishes to realize control on membership itself, then sharing a secret key is not enough. Any member who already has the shared key can pass it on to new members. Therefore, keys and identities need to be combined for access control, but without access to a file system, mechanisms like access control lists are not feasible. In many online networks (for example Yahoo! Groups), a smaller subset of members own and moderate the membership of a group.

There is on top of that the need for a finer granularity of access control, determining who can read, write or modify and delete each shared object, and how to enforce such access control in a decentralized setting, while still guaranteeing non-repudiation as well as preventing impersonation and replay. Achieving such finer granularity of access control in a decentralized manner is, we believe one of the hardest security challenges, and the biggest hurdle in realizing a P2P infrastructure for social networking applications.

Other security issues like prevention of DDoS and Sybil attacks, enforcing cooperation and preventing free-riding or content pollution, and establishing trust are also of course long-standing issues in the community, but since they have been in the spotlight for years now, we do not highlight these here. That of course does not

mean that these are trivial, or even practically solved. However, in the social network context, some of these issues may actually get simpler to deal with.

For peer identities, one can take advantage of opportunistic networks and peer authentication by in-person contact, when friends meet in real life and exchange keys over their phones. For bootstrapping authentication, a central authority (trusted third party) seems hard to avoid.

Robustness. Against misbehavior: In a centralized system, one can turn to the provider in case of user misbehavior, there is usually a process defined for dealing with such complaints. In a decentralized system, there is no authority that can ban users for misbehavior or remove content. Robustness against free-riding: With-out the monetary incentive offered by advertising, other incentives have to come in to make users shoulder the responsibilities for keeping up the infrastructure, providing storage and ensuring availability by staying online. Robustness and Trust: Once access to content is granted, it is difficult to revoke that right. When a user allows a friend to see a message, the friend can store the message and keep access to it even after a change of key. Trust has to be at least equal to assigned access rights, due to this difficulty.

Limited Peers. To take advantage of the decentralized nature of social networks, a mapping of physical social network to virtual and vice versa enables extensions to offering access via web browsers by phone applications and direct exchange of data in physical proximity. A major impedance to widespread adoption of a decentralized system for OSNs will be users' reluctance to install yet another software. Consequently, it will be essential to allow for two classes of users, a core network of users who run the decentralized infrastructure software as well as a web service front-end, and the other, who are essentially clients accessing this service. This of course throws open Pandora's box with lots of questions, including technological feasibility as well as game theoretic issues like incentives and fairness in such a two tier system. Another immediate benefit however of allowing such two-tier system is that users can then participate in the social network with resource constrained (e.g., mobile) devices, which they may use as an auxiliary, even when they contribute resources to the core of the system with their primary device.

Locality. Using direct exchange between devices, real-life social networks can be used to support the decentralized social networking application. In addition to such opportunistic networks between users, a distributed architecture also enables us to take advantage of geographic proximity and its correlation with local interests. For example, most access routers for home Internet access now come with USB slots where storage can be added or they already have unused storage on the device itself. These routers are typically always on and thus would provide some stability for availability of data of local interest. This local interest can arise from the locality of events but also from the locality of typical real-life social networks of friends and neighbors. How to best harness this locality remains to be seen.

3.8.4. General Purpose DOSNs

To give the readers a better overview of existing approaches towards the decentralization of online social network services, we propose a reference architecture of a general-purpose DOSN platform is given in Fig. 3.8.4.1. The reference architecture consists of

six layers and provides an architectural abstraction of variety of current related approaches to decentralized social networking in the research literature. The lower layer of this architecture is the physical communication network, which can be the Internet or a (mobile) ad hoc network (in case we consider a mobile online social network). The distributed or P2P overlay management provides core functionalities to manage resources in the supporting infrastructure of the system, which can be a distributed network of trusted servers or a P2P overlay. Specifically, this layer provides higher layers the capabilities of looking up resources, routing messages, and retrieving information reliably and effectively among nodes in the overlay.

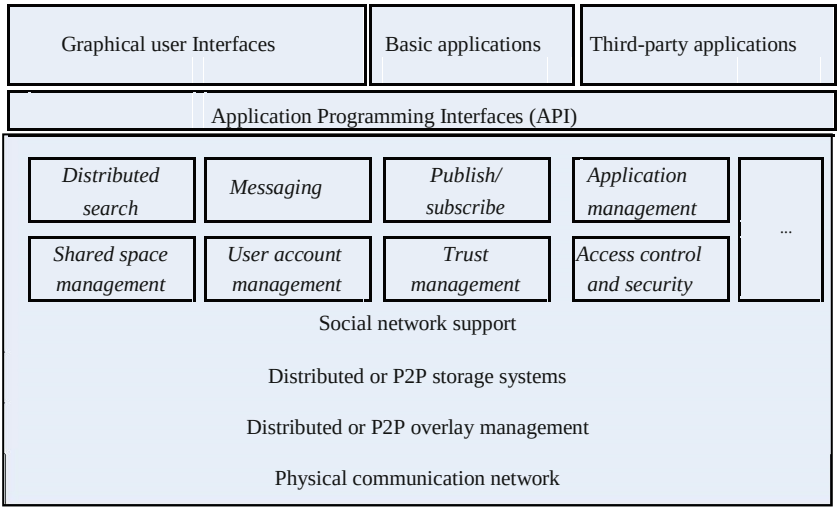


Fig. 3.8.4.1 The general architecture of a distributed online social networ

On top of this overlay is the decentralized data management layer, which implements functionalities of a distributed or peer-to-peer information system to query, insert, and update various persistent objects to the systems.

The social networking layer implements all basic functionalities and features that are provided by contemporary centralized social networking services. Among these functionalities the most important ones are given in Fig. 3.8.4.1, namely the capability to search the system (Distributed search) for relevant information, the management of users and shared space (User account and share space management), the management of security and access control issues (Trust management, Access control and security), the coordination and management of social applications developed by third parties (Application management).

It is expected that the social networking layer exposes and implements an application programming interface (API) to support the development of new applications by freelance developers and other third-parties, as well as to enable the customization of the social network service to suit various preferences of the user. To enable better interoperability with available social network services, e.g., better portability of applications across OSN providers, this API should conform to existing API standards, e.g., OpenSocial.

The top layer of the architecture includes the user interface to the system and various applications built on top of the development platform provided by the

DOSN. The DOSN user is expected to provide the user the necessary transparency to use the DOSN as any other centralized OSN. Applications can be either implemented by the DOSN provider or developed by third-parties, and can be installed or removed from the system according to user's preferences.

3.8.5. Proposed DOSN Approaches

Safebook: Safebook , an approach to provide a decentralized general purpose OSN, follows the main objective of protecting its users' privacy.

It considers adverse or erroneous behavior of a centralized service provider, possible adversaries which are misusing the functions of the social networking service, as well as external adversaries that could eavesdrop or modify data on the networking layer.

The main goals of offering the full set of services that centralized general purpose SNS usually implement, and of assuring the three security objectives of privacy, integrity, and availability, it is based on two simple assumptions: that decentralization and cooperation between friends will facilitate the implementation of a secure, and privacy preserving OSN. Considering the centralized storage to be a potential risk, safebook chooses a distributed implementation architecture. The social links between friends, family members, and acquaintances, which are represented as the core intrinsic knowledge of an OSN, are leveraged for multiple purposes. Requests from a user and for a user's profile are anonymized hop-by-hop on recursive routing paths traversing links of the OSN. Additionally, since friends are assumed to cooperate, they are leveraged to increase the availability of profiles.

Safebook consists of three major different components, the TIS, matryoshkas, and a peer-to-peer location substrate (see Fig. 3.8.7.1).

All users have to acquire certificates from a *Trusted Identity Service (TIS)* upon joining the OSN, which they are able to do by invitation, only. The TIS is a stateless, offline service. It does not store any information, but simply implements a cryptographic function to issue keys, identifiers, and certificates based on the identity of the requesting user. It currently is run at the institutions participating in the Safebook development (Institut Eur´ecom and TU Darmstadt), but may be further distributed to other trusted third parties. The TIS represents a powerful entity in safebook, yet, it does not cause any threats, since it is only involved in the identification and certification of users, but does neither store nor retrieves or accesses any data of the users.

Safebook discerns between identified participation in the OSN, and anonymous participation to provide services to other users. In order to protect the identified participation, safebook introduces *matryoshkas*, which are specialized overlays encompassing each user. All contacts of a user represent the innermost shell of a user's matryoshka. Each user selected for a matryoshka in turn selects one of his contacts to be part of the next matryoshka shell, unless a predefined number of shells is reached. Current evaluation indicates that three to four shells represent a good trade-off between the resilience towards statistical identification attacks against the core, vs. the efficiency and performance of the system. The matryoshkas are used to anonymize requests, to hide the existence of the user in the center. Safebook implements the original routing structure and distance metric as Kademlia, simply switching for iterative to recursive routing and stochastically including access

through contacts to protect the identity of the requesting user. The identifier used for the peer-to-peer substrate additionally is de-rived from the certificates, and hence determined by the TIS, which prevents denial of service attacks on the peer-to-peer overlay.

Safebook initially has been analyzed in formal models and large scale simulations. A first functional prototype to date is tested by the developers, which predict a first version for the public to be available by the end of 2010.

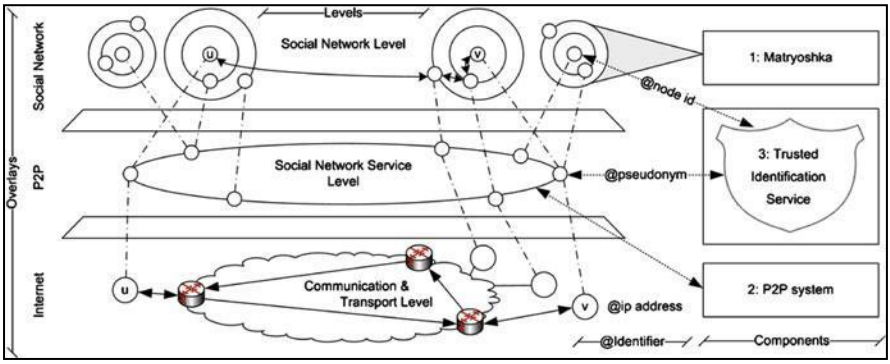


Fig. 3.8.7.1 Main design components of safebook

FOAF The framework enables users to export their FOAF profiles, store them on dedicated trusted servers. Users query and man-age these profiles through open Web-based protocols such as WebDAV or SPARQL/Update. A clear advantage of this approach is its compatibility to current social networking platforms and its entirely Web-based nature. The use of a set of trusted servers for storing user's data, however, raises some other security issues that necessitates further considerations. We believe this approach has a high potential of being adopted, as it is supported by the W3C consortiums.

LifeSocial Considering the immense increase of users that online social networks have experienced in the recent past, and which are expected for the future, too, LifeSocial primarily aims at keeping social networking services scalable by distributing the load to their users' resources.

The main functional components of OSNs are data storage and interaction. Both are classic domains of peer-to-peer systems, and have very successfully been implemented as file sharing and instant messaging (jabber), or telephony (Skype) applications. With current social network providers being central entities, who have to provide the entire resources for storing the data uploaded to the OSN and for making it accessible, they are soon to become a bottle neck when the number of users increases further.

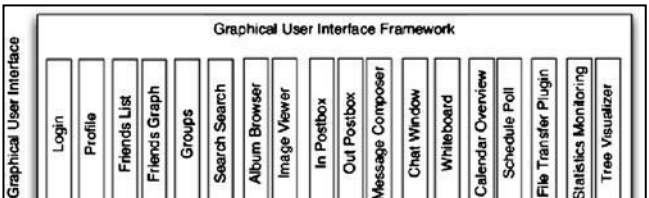


Fig. 3.8.7.2 LifeSocial plugin architecture

LifeSocial hence proposes to distribute the service provision in a peer-to-peer fashion in order to mitigate the resource problem and to balance the service load to the resources at the users' devices.

LifeSocial is designed with the main premise to leverage on existing and proven components and to create a modular plugin-architecture to assure extensibility. It consequently is assembled using FreePastry, a structured peer-to-peer overlay for data storage, and PAST to achieve reliable replication of the data. LifeSocial implements its own access control scheme on top of these components. Some plug-ins are mandatory to implement a general purpose OSN: The whole system demands plugins for profile management, friend management and group management and photo albums as a minimum set. Additional plugins, such as a whiteboard and a chat sys-tem have been proposed.

The different components of LifeSocial, and the overall system, have been eval-uated in simulation studies. It additionally is one of the few systems of which a prototype exists at the time of this writing (see Fig. 3.8.7.2). The prototype consists of all mandatory plugins, as well as basic white-board and chat functions. It has been presented in different conferences and exhibitions and the system currently is tested between the groups of its developers.

PeerSoN PeerSoN aims at keeping the features of OSNs but overcoming two limitations: privacy issues and the requirement of Internet connectivity for all

transactions. To address the privacy problem, it uses encryption and access control coupled with a peer-to-peer approach to replace the centralized authority of classical OSNs. These two measures prevent privacy violation attempts by users, providers, or advertisers. Extending the decentralized approach, PeerSoN also enables direct exchange of data between devices to allow for opportunistic, delay-tolerant net-working. This includes making use of ubiquitous storage to enable local services.

The main properties of PeerSoN are encryption, decentralization, and direct data exchange. In a nutshell, encryption provides privacy for the users, and decentralization based on the use of a P2P infrastructure provides independence from OSN providers. Decentralization makes it easier to integrate direct data exchange between users' devices into the system. Direct exchange allows users to use the sys-tem without constant Internet connectivity, leveraging real-life social networking and locality.

The current PeerSoN implementation replicates the following features of OSNs. In the category of social links, users (peers) can become friends and thus establish a social link between each other. Digital personal spaces are provided in that users can maintain their own profile and a wall, a space for items posted by themselves or their friends. Communications between users are directly peer-to-peer when both are online, and the implementation supports asynchronous messaging when this is not the case.

PeerSoN uses a DHT as a lookup system and then lets peers connect directly; all data is encrypted and keys for accessing an object are encrypted for the exclusive use of authorized users and stored in a separate file associated with a particular object, such as a user's profile. The prototype implementation has been tested on PlanetLab and uses OpenDHT for the lookup service. Using OpenDHT facilitated the PeerSoN deployment on PlanetLab but will be replaced for the next iteration of the implementation.

Likir Likir is a Kademlia-based DHT that is aimed to protect the overlay against attacks common to these systems, by embedding a strong identity notion at overlay level. This increases the reliability of the overlay and offers many identity-based services that well suit social applications. Likir targets at the main goals of avoiding a centralized storage of personal information, offering reliability, and integrating the identities of users deep into the system.

The main motivation of Likir is to avoid a central data repository, for both the reason to avoid a single point of failure and aggregation of user data. The ratio-nale is to use a reliable, identity-based DHT layer, to provide the main services. The applications built on top of this substrate consequently are relieved from identity management, as it is already provided by the underlying DHT (see Fig. 3.8.7.3). Likir is designed to achieve full confidentiality of data as well as anonymity of the users. It additionally provides access control in a granularity to applications: "authorized disclosure" ensures that malicious applications installed at some individual's device is unable to access data disclosed to other applications.

Likir builds its core properties on introducing cryptography in a plain DHT-based approach. All nodes are furnished with an identifier in the form of an OpenId by a certification service. Subsequent communication events consequently are encrypted and authenticated by both communicating parties. A supplemental access control scheme is integrated that requires all service providing nodes to check grants that are appended to each request before returning any data.

In addition to an analytical study on the bandwidth and computational overhead consumed by Likir, the authors have run large scale emulations on PlanetLab. A

prototype of the Likir middleware has been published and is available for download, including a simple chat application for the purpose of demonstrations.

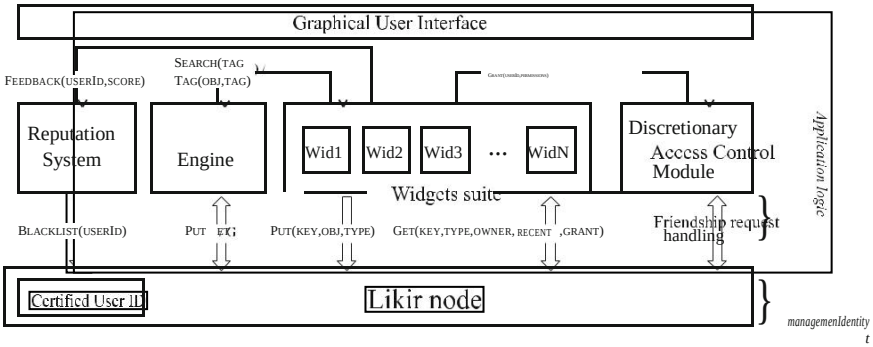


Fig. 3.8.7.3 Likir architecture

3.8.8. Social Distributed Systems

Besides the initiatives to emulate specifically online social networks, as described in the previous section, there is also a growing trend to develop *social distributed systems* (SocDS) which provide equivalent functionalities to non-social (and some-times, also centralized) systems. Such initiatives try to leverage and translate the social trust into properties leading to system reliability. For example, distributed hash tables realized using exclusively social links can provide robustness against Sybil attack. Likewise, P2P storage systems relying on social trust to do data backup provide resilience against free-riding address limitations of computational trust models and stand-alone algorithmic solutions. Such social distributed systems can be used for carrying out any task the non-social counterparts are typically used for, but additionally, and naturally, SocDS are ideal to be used as substrates and building blocks for distributed/decentralized online social networks.

Essentially, one can argue that with the use of SocDS, we have a two way symbiotic design. Distributed/peer-to-peer infrastructure for OSNs provide desir-able properties and addressing the shortcomings and constraints of traditional OSNs which use centralized resources provided by OSN service providers. On the other hand leveraging on social networks to design robust distributed systems provide new systems design opportunities which are beyond the scope of traditional algorithmic solutions. Such distributed systems can be used for different applications, but are of-course naturally suitable to form the underlying substrates for decentralized online social networks.

3.8.8.1. Social DHT: SocialCircle

Structured overlays, e.g., Distributed Hash Tables (DHTs) provide essential in-dexing and resource discovering in distributed information systems. Typically, structured overlays are based on enhanced rings, meshes, hypercubes, etc., lever-aging on the topological properties of such geometric structures. The ring topology is arguably the simplest and most popular structure used in various overlays. In a ring based overlay network like Chord nodes are assigned to distinct points over a

circular key-space, and the ring invariant is said to hold if each node correctly knows the currently online node which succeeds it (and the one which precedes it) in the ring. The ring is both a blessing and a curse.

On the one hand, an intact ring is sufficient to guarantee correct routing.

Virtual ring routing (VRR) is a DHT style overlay layer approach used to de-fine the underlying network's routing mechanism. It is implemented directly on top of the link layer and provides both traditional point-to-point network routing and DHT routing to the node responsible for a hashed key, without either flood-ing the network or using location dependent addresses. While traditional DHTs take for granted point-to-point communication between any pair of participating nodes, VRR extends the idea, using only link layer connectivity. Essentially this means that the VRR scheme relaxes the traditional DHT assumption of a completely

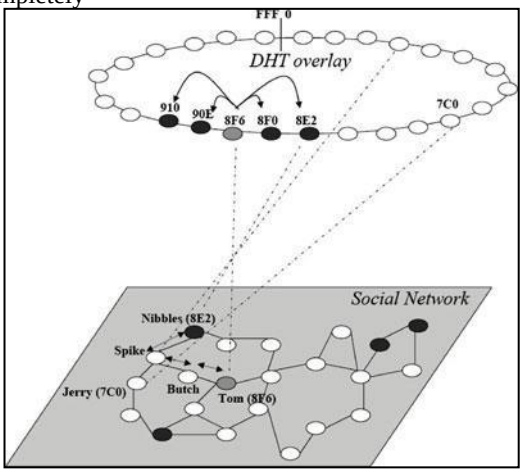


Fig. 3.8.8.1 Sybil attack resistant *SocialCircle* DHT exploiting social connections. This example of DHT over Tom & Jerry's social graph is adapted from the virtual ring for routing in ad-hoc networks

connected underlying graph. Each node in VRR has a unique address and loca-tion independent fixed identifier, organized in a virtual ring, emulating Chord style network. Each node keeps a list of r=2 closest clockwise and counter-clockwise neighbors for the node on the virtual ring. Such a set of neighbors is called the node's virtual neighbor set (*vset*).

Typically, members in a node's *vset* won't be directly accessible to it through the link layer. Thus each node also maintains a second set called the physical neighbor set (*pset*), comprising nodes physically reachable to it through the link layer. In SocialCircle, this idea in exploited by replacing VRR's *pset* with the set of friends a node has - its social set *sset*.

Thus, instead of exploiting the physical layer connectivity as VRR does, Social-Circle builds the overlay over the *social plane* exploiting people's social connec-tions. The lower plane shows the social graph, while the upper plane shows the SocialCircle DHT.

Finally, each peer maintains a routing table, which comprises of routes to its *vset* neighbors using its *sset*. These routes can be established and maintained using different strategies typically inspired by mobile ad-hoc routing protocols. The DHT abstraction ensures efficiency and certainty of routing to the appropriate target.

Thus, in the example from Fig. 3.8.8.1, *Tom* with logical identifier 8F 6 on the So-cialCircle has 8F 0, 8E 2, 90E and 910 in its *vset*. *Spike* has *Jerry*, *Nibbles* and *Butch* in its *sset* since they are his direct social connections.

Tom needs to maintain routes to all its *vset* nodes, and thus, for 8E 2, he will have a route through his *sset* entry *Butch*, who will route through his *sset* entry *Spike*.

So when *Tom* needs to route a message to 7C 0, then it will try to forward the message closest to the target on the SocialCircle, which happens to be 8E 2. While the message is being routed to 8E 2 following the *sset* nodes at each peer, *Spike* will observe that the ultimate destination is 7C 0, for which it may already have a route passing through it, and will thus forward the message to *Jerry*, instead of sending it to the intermediate destination *Nibbles*. *Jerry* processes the routing request, and forwards it to the final destination *Quacker*, who happens to have the identifier 7C 0 on SocialCircle.

VRR works in an opportunistic manner where the route is forwarded along the virtual ring, but discovers shortcuts, so that the search is still efficient. SocialCircle preserves the same benefits by routing over the social links. Each hop on the social link involves IP level routing, which may need several hops, just like any logical overlay hop of traditional DHTs.

3.8.8.2. Storage/Back-up

A p2p storage (or back-up) system uses the storage space of its participants to increase the availability or the survivability of the data. Coupled with mechanisms for content sharing and privacy, a distributed storage system is a building block for a DOSN. For instance, it can ensure that, when a user is off-line, her profile is available through the replicas.

Many distributed storage systems have the option of sharing stored objects among a group of users. OceanStore is one of the first distributed storage infrastructures, focusing on storing objects. As all the objects are encrypted, read sharing (i.e., many agents accessing an object) is accomplished through sharing of the encryption key. OceanStore also permits multiple agents to modify the same object through Access Control Lists (ACL), which are OceanStore object themselves (a write on an object is authenticated by a trusted server against an ACL associated with the object).

Wuala is an online storage system that combines distributed and centralized elements. Data is stored both on users' machines and on Wuala's servers; the algorithm associating data with particular machines is most probably centralized. Wuala can act as a crude DOSN as it enables its users to share files in a group of "friends"; however other OSN features, such as profile information, are missing.

BlockParty and FriendStore are p2p backup system in which data is stored only on designated peers corresponding to real-world "friendships" be-tween users. In fact, the "friendship" relation acts as a proxy for a simple rep-utation system: it verifies user identities and provides off-system discouragement for free-riding. Moreover, argues that friendship-based system will have less permanent node departure; and thus the data survivability will depend on (rare) hardware failures rather than permanent departures frequent in usual p2p systems. Friendship-based systems can be thus more forgiving to transient errors instead of assuming a permanent departure and, consequently, large-scale data transmis-sions, longer delays can be used.

The main disadvantage of friendship-based p2p backup systems is that, in general, systems that constrain choice of replicating nodes have lower. In a friendship-based system, achieved data survivability strongly depends on survivability of machines of

friends and friendships are fostered rather based on the person's character, and not her computer's up-time, which is the important factor for the system's reliability.

3.8.9. Delay-Tolerant DOSN

Current online social networking services require the user to be connected to the Internet for every interaction, not only for real-time information but also for older information such as data posted by the user or her friends in the past. Since online social networks are part of the so-called Web 2.0, they run on dedicated web servers.

All information in the online social network is thus stored on logically central servers, even though they may be replicated or cached in different geographic regions using content distribution services. Due to such centralization, there is no distinction between information of global or exclusively local relevance.

We propose to implement online social network functionality in a distributed, delay-tolerant way. Intermittent Internet connectivity can be used to connect with the wider user community, while users can exchange data among each other in direct physical proximity during offline times. The need for constant Internet connectivity, which can be costly, is thus eliminated. When information is of local relevance only, it need not be transferred to a central server that is potentially far away. These need-less long-distance transfers can be replaced by local storage. In addition, it becomes easier to take locality into account logically when keeping local information also local physically.

While portable user devices, such as phones, laptops and personal digital assis-tants (PDAs) can be used to exchange data directly, also fixed devices can contribute resources. Schioeberg proposed to use storage on home routers, such as ADSL modems with WLAN capabilities to support peer-to-peer social networks. Many home routers now have unused storage or can at least be extended by USB sticks or external hard drives. Fixed devices that typically are switched on irrespective of user activity not only contribute resources but also increase availability and robustness of a system for delay-tolerant social networks.

Such delay-tolerant, local social networks allow us to build on other propos-als and new opportunities. For example, Antoniadis et al. proposed to use local wireless networks to enhance communities such as neighborhoods in towns. Collec-tively, users would build wireless neighborhood networks by pooling their resources to support the creation and operation of the underlying communication network. They envision user participation and cooperation at several layers, physic, access, network, and application layers. They argue that *the design of communities suit-able for this environment will encourage users to participate, enable trustworthy network creation, and provide a social layer, which can be exploited in order to design cross-layer incentive mechanisms that will further encourage users to share their resources and cooperate at lower layers.* The goal is to bridge the gap between online and offline communities.

The way we envision delay-tolerant social networks can be a vehicle to such fos-tering of communities. Beyond the features of current social networks that allow users to keep in touch and up-to-date with the friends they already have and increasingly, the new ones they found thanks to the service itself, delay-tolerant

social networks would allow users to benefit from locality. They could find others who live nearby and have similar interests, find or start events in the neighborhood, organize or collaborate for creative of political collective action, found local marketplaces of ideas, goods, or services, edit local information repositories or wikis, to name just a few possibilities.

Local social networks could also be established to never connect to a wider collection of networks but form islands of social networks, effectively making censorship or data mining prohibitively difficult.

The possibilities of use of delay-tolerant social networks are of course not limited to the examples given above, once the technology is available, users may come up with novel and original applications, as has been the case with online social networks or indeed the advent of the Internet and the World-Wide Web itself. Delay-tolerant social networks can thus be seen as enablers for applications or uses not yet foreseen.

3.9. Multi-Relational Characterization of Dynamic Social Network Communities

3.9.1 Introduction

Motivation: human community as meaning-making eco-system. Being influenced by this theory, we believe that semantics is an *emergent arti-fact of human activity that evolves over time*. Human activity is mostly social, and the social networks of human are conceivable loci for the construction of meaning. Hence, it is crucial to identify real human networks as *communities of people inter-acting with each other through meaningful social activities, and producing stable associations between concepts and artifacts in a coherent manner*.

Motivating applications. The discovery of human communities is not only philosophically interesting, but also has practical implications. As new concepts emerge and evolve around real human networks, community discovery can result in new knowledge and provoke advancements in information search and decision-making. Example applications include:

Context-sensitive information search and recommendation
Content organization, tracking and monitoring
Behavioral prediction

Data characteristics and challenges. Large volumes of social media data are being generated from various social media platforms including blogs, FaceBook, Twitter, Digg, Flickr. The key characteristics of online social media data include:

Voluminous
Dynamic
Context-rich

The large-scale, fine-grained, rich online interaction records pose new challenges on community discovery:

Lack of well-defined attributes

Limitation in network centric analysis

Scalability requirement

Problem overview. we have focused on the following research problems:

How to identify meaningful interpersonal relationship from online social actions?

How to identify sustained evolving communities from dynamic networks?

How to identify communities with rich interaction context?

Our approach. Our work concerns approach to the three problems (see Fig. 3.9.1.1 for illustrating summarization):

Mutual awareness: We propose mutual awareness (Fig. 3.9.1.11a), a bi-directional relationship indicating how well a pair of bloggers is aware of each other, as fundamental property of a community.

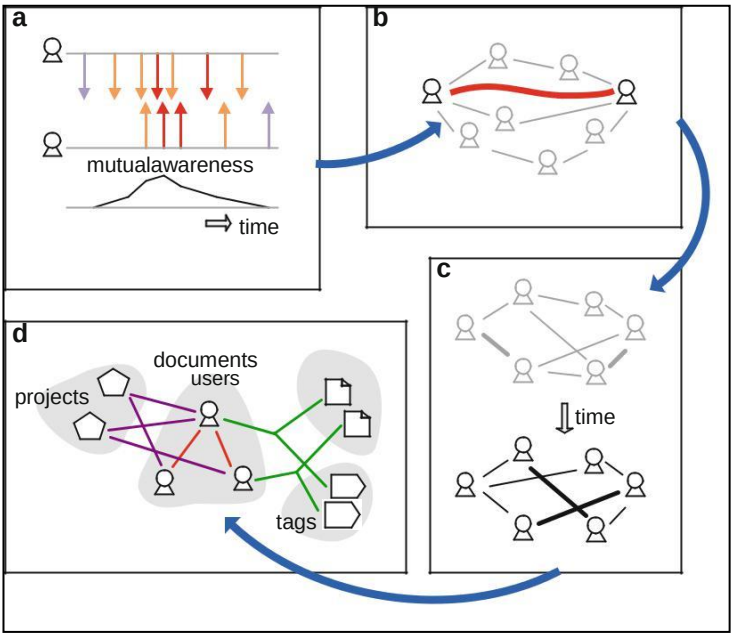


Fig. 3.9.1.1 Our work concerns multiple aspects on community analysis: (**a**) Mutual awareness – a bi-directional relationship indicating how well a pair of bloggers is aware of each other, as fundamental property of a community. (**b**) Mutual awareness expansion – a random walk based distance measure which estimates the probability that two bloggers are aware of each other on the network. (**c**) FacetNet – for analyzing communities and their evolutions in a unified process. (**d**) MetaFac – the first graph-based multi-tensor factorization framework for analyzing the dynamics of heterogeneous social networks

FacetNet: We introduce the FacetNet framework to analyze communities and their evolutions in a unified process. In our framework (Fig. 3.9.1.1c), the community structure at a given timestep is determined both by the observed networked data and by the prior distribution given by historic community structures. Al-gorithmically, we propose the first probabilistic generative model for analyzing communities and their evolution. The experimental results suggest that our tech-nique is scalable and is able to extract meaningful communities based on social media context. (e.g., dramatic change in a short time is unlikely).

MetaFac: We propose MetaFac, the first graph-based tensor factorization framework for analyzing the dynamics of heterogeneous social networks (Fig. 3.9.1.1d). In this framework, we introduce metagraph, a novel relational hypergraph repre-sentation for modeling multi-relational and multi-dimensional social data. Then we propose an efficient multi-relational factorization algorithm for latent com-munity extraction on a given metagraph. Extensive experiments on large-scale real-world social media data and from the enterprise data suggest that our tech-nique is able to extract meaningful communities that are adaptive to social media context.

Organization

3.9.2. Actions, Networking and Community Formation

A "community of web pages" due to the links of *relevance* is thus different from a "community of bloggers" formed due to the links of interactions. The analysis of blog network also deviates from traditional social network analysis because the *social meaning* of the blog network is not as well-defined as in traditional social networks (e.g. links represent friendship). Hence, community discovery in the blogosphere requires a new analytical frame-work grounded in the unique characteristics of the blog media.

3.9.2.1 Mutual Awareness and Community Discovery

The notion of *virtual community*, or online community, has been discussed extensively in prior research. Rheingold defined virtual communities to be "social aggregations that emerge from the Net when enough people carry on those public discussions long enough, with sufficient human feeling, to form webs of personal re-lationship in cyberspace." Jones considered four characteristics as the necessary conditions for the formation of a virtual community: interactivity, communicators, virtual common-public-place where the computer-mediated communication takes place, and sustained membership. These conditions echo Garfinkel's observation on the necessity of mutually observable actions. The same idea that interac-tivity forms a social reality has also been discussed by Dourish. According to Dourish, interaction involves presence and awareness. In what Dourish called an *action community*, members share the common sense un-derstandings through the reciprocal actions. The common aspect of the prior work is the emphasis on the significance of action and interaction in online communities. However, little work has studied the counter perspective – how to discover communities due to actions.

We introduce *mutual awareness* that is fundamental to blog community formation. By mutual awareness of action we mean that individual blogger actions must lead to bloggers becoming aware of each other's presence. The idea is in the light of Locale theory that discusses how social organization of activity is supported in different spaces. While the domains of activity must provide means for the commu-nity members to act, the space must also accord members' presence and facilitate mutual awareness.

3.9.2.2 Extracting Communities Based on Mutual Awareness Structure

Computable Definition for Mutual Awareness

Let us examine the actions of individual bloggers – how bloggers read and communicate ideas with other bloggers. The bloggers can act in the blogosphere, in several ways: surf/read, create entries (containing entry-to-entry links, entry to blog/web, or no link), comment or change blogroll. Some actions (e.g. surf/read) may be hidden, while others may be observable.

How a specific blogger action leads to mutual awareness may depend on (a) if the action is mutually observed, and (b) the importance of the action for the blogger who performs the action. Note that some blogger actions are not observable by other bloggers. For example, let us consider two hypothetical bloggers, Mary and John. Let us assume that Mary creates an entry with a hyperlink that points to John's blog. In this case John would be unaware of Mary's entry. On the other hand, if Mary leaves a comment on John's entry, then John is immediately aware of her presence. If Mary mostly leaves comments on other bloggers, and the importance of a comment for Mary is low while many bloggers are aware of Mary, she may not feel that she in engaged in dialogue with them. The assessment of mutual awareness is the first step toward the discovery of blog communities.

We thus characterize mutual awareness as follows (see Fig. 3.9.2.2.1 for an illustration): mutual awareness between two bloggers is affected by the type of action, the number of actions for each type, and when the action occurred. It depends on sustained actions – it increases if there are follow-up actions that lead to mutual awareness and decreases if actions are not sustained over time.

We represent the set of bloggers in the blogspace as a weighted directed graph G D .V; E /, where each node v 2 V represents a blogger, each edge between any pair of nodes u and v represents an action performed by u with respect to v. The weight on each edge f .u; v/ indicates the mutual awareness between two bloggers and v. The corresponding matrix **M** with each entry $\mathbf{M}_{uv}$ D f .u; v/ is called mutual awareness matrix and is defined as follows.

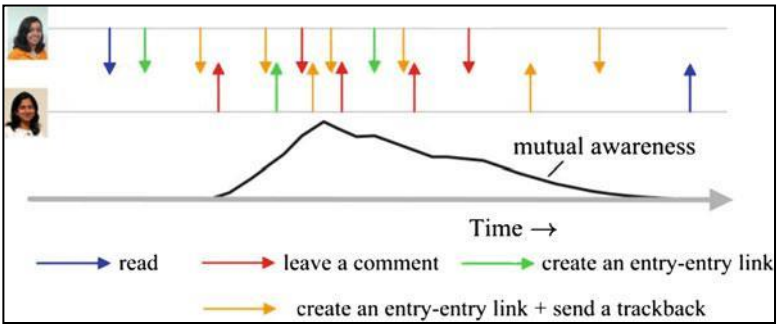


Fig. 3.9.2.2.1 Mutual awareness between two bloggers is affected by the type of actions, the number of such actions, and when such actions occur. The arrow direction indicates the source and the destination blogger on whom the action is performed. A mutual awareness curve is plotted to show the action impacts

Mutual Awareness Expansion

We extend the idea of mutual awareness to community extraction. Mutual awareness quantifies the relationship between two bloggers. To extract a set of bloggers having high mutual awareness, we hypothesize how mutual awareness expands in a blog network:

Transitivity: One could become aware of a member without direct interaction since he or she can observe his or her direct peers interacting with other people.

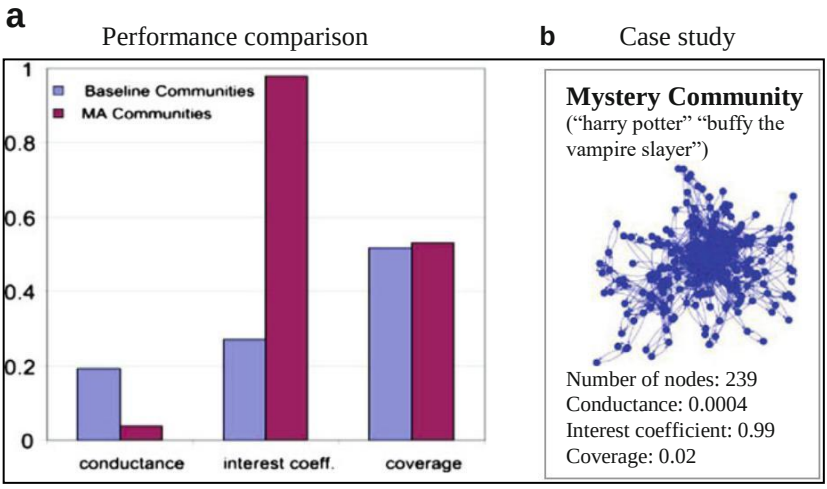


Fig. 3.9.2.2.2 (**a**) Performance comparison between the Baseline communities and the MA communi-ties in terms of metrics conductance, interest coefficient, and coverage. (**b**) An example subgroup cohesive topic about mystery novels, extracted using MA matrix

Thus awareness is transitive. (The transitivity property in social network has been first examined in Travers and Milgram's well-known small world experiment, which motivates our proposed algorithm.)

Reciprocity: Such transitive awareness must be reciprocal. If expansion of aware-ness is only one directional, one might not feel belonging to the community

Frequency: The amount of observed interaction must be sufficient for members to feel connected to each other

We characterize such *mutual awareness expansion* process by a random walk model. The probability that two bloggers are aware of each other on the entire network is quantified using the random walk expected length between two nodes corresponding to the bloggers. We refer to this expected length as *symmetric social distance*. It is computed as follows: Given a direct graph G D .V; E / and the mutual awareness matrix

W associated with G, the random walk on G is defined to be the Markov chain with state space V and the transition matrix **P** D **D** ^{1}**W**, where **D** is a diagonal matrix with element d_{ii} D $\sum_j w_{ij}$. A random walker at a node i on G will follow the transition probability p_{ij} D P_{ij} to visit the next node j . Note that by construction

w_{ii} D $\sum_j w_{ij}$ (i.e. p_{ii} D 1=2) for i ¤ j .

Let $u!v$ denote the one-way social distance from node *u* number of steps to reach node *v* from node *u*. We define $u!v$ *awareness* property:

Application: Query-Sensitive Community Extraction

We apply the community extraction algorithm to the extraction of *query-sensitive communities*, i.e. blog communities that have a strong content related theme with respect to a given query. We summarize the idea as follows:

Step 1: Given a query topic Q, extract query-sensitive graph G_Q to represent interactions relevant to the topic.

Step 2: Given G_Q, extract communities

In the first step, we construct a weighted action matrix with respect to a query Q. Q contains query keywords that represent the given topics, e.g. “Katrina”, “london bomb”, etc. The weight of an interaction is determined by the relevance score of the blog content involved in the interaction. Because the query keywords in Q are relatively short, in order to further incorporate relevant blogs in G_Q, we compute the “query relevancy” by employing a web-based similarity function.

Figure 3.9.2.2.3 shows an example from our experiment. In this case, we extract communities with respect to the query keyword is “katrina”, which is about a natural disaster caused by the hurricane Katrina in August 2005. In order to understand the “meaningfulness” of the extracted communities, we employ a heuristic method to examine the relationship between the topic and the communities over time: We connect those communities extracted from different time snapshots based on an interaction similarity measure.

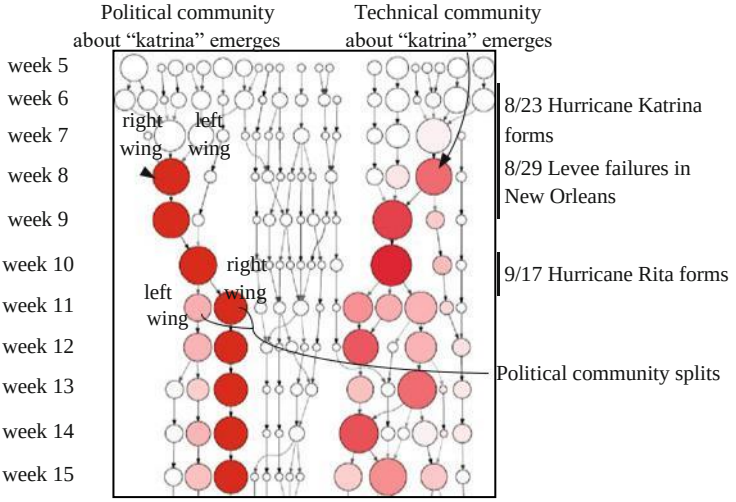


Fig. 3.9.2.2.3 Communities with respect to the query "katrina." The node size reflects the number of community members and the reddish shade of node is proportional to the query relevancy for the keyword "katrina"

In Fig. 3.9.2.2.3 each node represents a detected community where the communities detected during the same week are aligned horizontally and the communities at different time snapshots are connected by arrows. The grayscale of an arrow is proportional to the interaction similarity between the two communities. The node size reflects the number of community members and the reddish shade of node is proportional to the query relevancy for the keyword "katrina". More saturated node represents more relevant community.

3.9.3. Analyzing Communities and Evolutions in Dynamic Network

3.9.3.1 Sustained Membership, Evolution and Community Discovery

Sustained membership is the key to discovery time-evolving communities. We introduce the FacetNet framework to extract sustained and evolving communities from dynamic social networks.

3.9.3.2 Extracting Sustained Evolving Communities

Extracting Communities and Evolutions

Community Membership

Community Evolution

3.9.3.3. Extracting Communities from Rich-Context Social Networks

It uses Metagraph factorization

Application: Context-Sensitive Prediction in Enterprise

UNIT 4 PREDICTING HUMAN BEHAVIOR AND PRIVACY ISSUES

4.1. Understanding and Predicting Human Behavior for Social Communities

4.1.1 Introduction

However, despite all the technological revolutions, for the end user (Humans) it is the perceived Quality of Experience (QoE) that counts, where QoE is a consequence of a user's internal state (e.g., predispositions, expectations, needs, motivation, mood), the characteristics of the designed system (e.g., usability, functionality, relevance) and the context (or the environment) within which the interaction occurs (e.g., social setting, meaningfulness of the activity).

4.2 User Data Management, Inference and Distribution

Current service creation trends in telecommunications and web worlds are show-ing the convergence towards a Future Internet of user-centric services. In fact, some works already provide user-oriented creation/execution environments, but these are usually tied to specific scopes and still lack on the capability to adapt to the heterogeneity of devices, technologies and the specificity of each individual user. Based on these limitations, the research identifies flexibility and personalization as the foundation for users' satisfaction, where the demand for different types of awareness needs to be present across the entire value of chain of a service.

In order to apply user information across a range of services and devices, there is a need for standardization of user related data and the architecture that enables their interoperability. These efforts have been seen at both fixed and mobile worlds and are usually taken under the European Telecommunications Standards Institute (ETSI), the Third Generation Partnership Project (3GPP), Open Mobile Alliance (OMA), among others. Considering data requirements from a wide range of facilities and from different standardization or-ganizations, the concept of Common Profile Storage (CPS) is defined by 3GPP in as a framework for streamlining service-independent user data and storing it un-der a single logical structure in order to avoid duplications and data inconsistency. Being a logically centralized data storage, it can be mapped to physically distributed configurations and should allow data to be accessed in a standard format. Indeed, several approaches have been proposed to guarantee a certain interoperability degree and can be grouped into three main classes: the syntactic, semantic and modeling approaches. The work proposes a combination of them to enable interoper-ability of user profile data management for a Future Internet.

Independently from the technology, all systems should allow user related data to be queried, subscribed or syndicated and ideally through web service interfaces. However, standardization, interoperability, flexibility and management are not the only challenges. To improve the degree of services personalization it is important to generate new information from the existing one. In this sense, social networks, user modeling and reality mining techniques can be empowered to study patterns

and predict future behaviors. Consequently, all the adjacent data necessary to perform such operations must be managed within the scope of a user/human profile. Nevertheless, due to the sensitiveness of the information we are referring to, it is important to efficiently control the way this information is stored, accessed and distributed, preserving user's privacy, security and trust.

With the aim of inferring user's needs, desires or intentions, several research initiatives from different fields (e.g., eHealth, Marketing, Telecoms) are starting to become a reality. Despite the different methodologies and approaches, the user requirements and the technologies involved to address the problems are usually the same. They commonly involve social network analysis, context-awareness and data mining. The basic motivation is the demand to exploit knowledge from various amounts of data collected, pertaining to social behavior of users in online environments.

Real world situations usually have to be derived from a complex set of features. Thus, context or behavior aware systems have to capture a set of features from het-erogeneous and distributed sources and process them to derive the overall situation. Therefore, recent approaches are intended to be comprehensive, i.e., comprise all components and processing steps necessary to capture a complex situation, starting with the access and management of sensing devices, up to the recognition of a com-plex situation based on multiple reasoning steps and schemes. To handle complex situations, the concept of decomposition is applied to the situation into a hierarchy of sub-situations. These sub-situations can be handled autonomously with respect to sensing and reasoning. In this way, the handling of complex situations can be simplified by decomposition. Another similar perspective is called layered rea-soning, where the first stage involves feature extraction and grouping (i.e., resulting in low-level context), the second event, state and activity recognition (i.e., originat-ing mid-level context), while the last stage is dedicated to prediction and inference of new knowledg. In what concerns social networks, research usually focuses on quantifying or qualifying the relationship between peers, where algorithms such as centrality and prestige can be used to calculate the proximity, influence or importance of a node in a network, while clustering and classification can be applied to similarity computation, respectively. In addition, when user related data is associated with time and space dimensions, by empowering data mining tech-niques it is possible to find hidden patterns that can be used in any of the previously identified stages of reasoning.

4.3 Enabling New Human Experiences

4.3.1. The Technologies

4.3.1.1 Social Networks

Humans in all cultures at all times form complex social networks; the term social network here means ongoing relations among people that matter to those engaged in the group, either for specific reasons or for more general expressions of mutual solidarity. Likewise, social networks among individuals who may not be related can be validated and maintained by agreement on objectives, social values, or even by choice of entertainment. Usually, network members tend to trust and rely on each other, and to provide information that other members might find use-ful and reliable. Social networks

are trusted because of shared experiences and the perception of shared values and shared needs.

4.3.1.2 Reality Mining

To overcome the discrepancy between online and "offline" networks, reality mining techniques can be empowered to approximate both worlds, proving awareness about people actual behavior. It typically analyzes sensor data (from mobiles, video cameras, satellites, etc) to extract subtle patterns that help to predict and understand future human behavior. These predictive patterns begin with biological "honest signals," human behaviors that evolved from ancient primate signaling mechanisms, and which are major factors in human decision making.

4.3.1.3 Context-Awareness

In today's services, the sought to deal with linking changes in the environment with computer systems is becoming increasingly important, allowing computers to both sense and react based on their environment. Additionally, devices may have information about the circumstances under which they are able to operate and based on rules, or an intelligent stimulus, react accordingly. By assessing and analyzing visions and predictions on computing, devices, infrastructures and human interaction, it becomes apparent that:

a. context is available, meaningful, and carries rich information in such environments, that users' expectations and user experience is directly related to context, acquiring, representing, providing, and using context becomes a crucial enabling technology for the vision of disappearing computers in everyday environments.

4.3.2 Architectural Framework and Methodology

In order to enable human behavior understanding and prediction, there are several independent but complementary steps that can be grouped into three different categories: Data Management, New Knowledge Generation and Service Exposure and Control. Figure 4.3.2.1 depicts these relationships as well as the sequence of activities involved.

4.3.2.1 Data Management

This activity usually starts with data acquisition. This process involves gathering information from different information systems. In our experiments we included user preferences, social networks, devices, policies, profiling algorithms, external

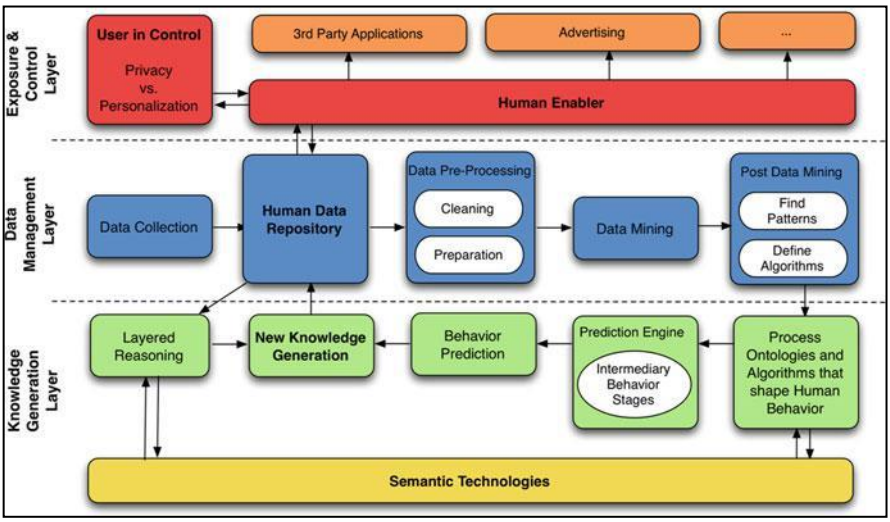


Fig. 4.3.2.1 Human Behavior Understanding and Prediction process

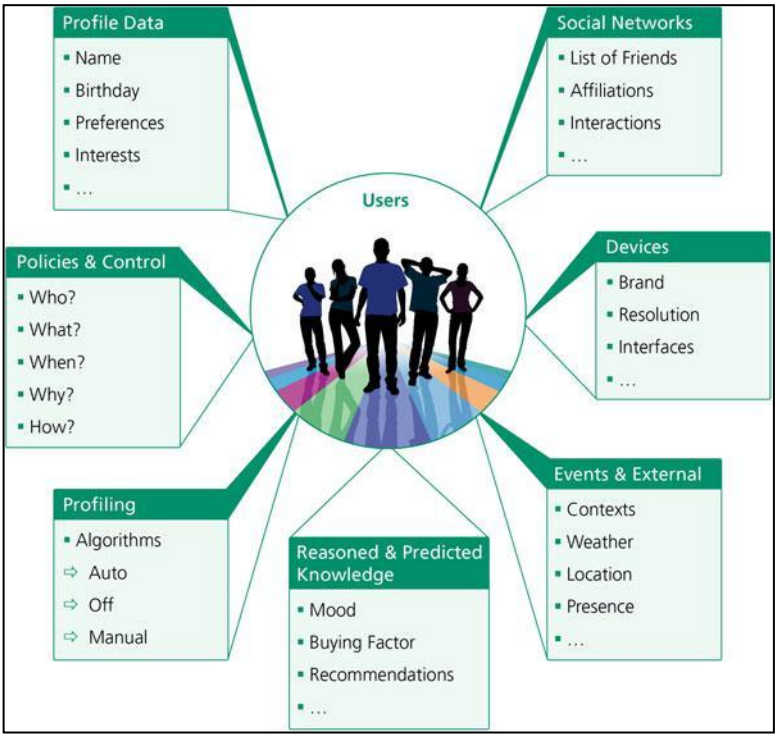


Fig. 4.3.2.2 Example of information to be stored in the Human Data Repository

contexts, as well as reasoned and predicted knowledge. Figure 4.3.2.2 exemplifies the type of information that can be stored in the Human Data Repository, a set of

properties build within a generic structure that allow services of the future to use user related information, while respecting their privacy, needs and concerns.

Due to real systems limitations, data is usually not captured without errors, therefore it is necessary to pre-process it in advance (before mining) and otherwise it would not be possible to correlate information correctly. Once this is done, data is mined by using two different approaches: the first, uses know statistical algorithms to help pattern recognition and consequent algorithmic modeling, the second uses the op-posite approach, where specific algorithms are designed to identify patterns in the data (this requires previous modeling). Combining both, allows us to address the specifics of our applications, and at the same time, automatically detect new rele-vant correlations that might occur after a few iterations.

4.3.2.2 Knowledge Generation

New information inference is based on user related data, which we call context and can be separated into three different categories: real-time, historical data and

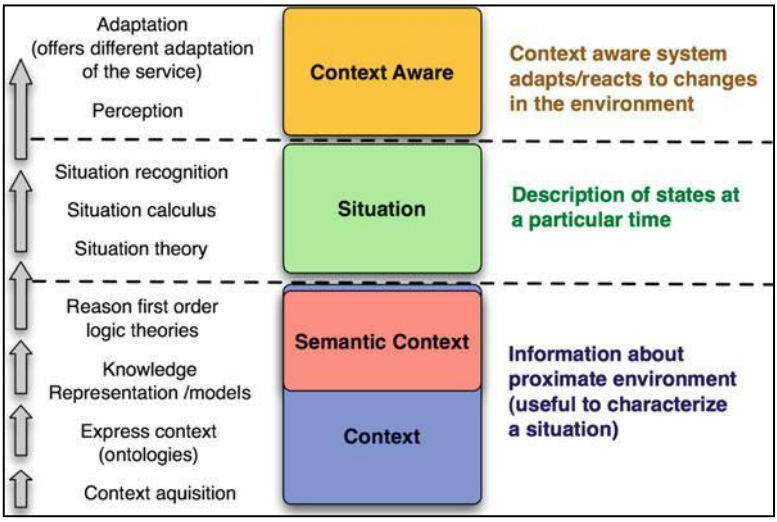


Fig. 4.3.2.2.1 Context layering model

reasoned context. Nevertheless, only real-time information is considered as context in the real meaning of this term. There are several layers of abstraction in a context-aware system and any context-aware middleware or architecture must therefore be capable of building representations and models of these abstractions. However, these high-level abstractions can only be made from lower level context, which requires some form of context management function. In our case, this is performed at the Human Data Repository.

The main context management features are context acquisition, context aggregation & fusion, context dissemination, discovery and lookup. In order to manipulate context information, it must be represented in some form that is compatible with the models that will be used in the reasoning and situation recognition processes. These models could be object oriented, ontological, rule based, logic based, based on semantic graphs or fuzzy logic sets. Furthermore, the representation must lend itself to the reasoning and

inference techniques to be used, such as classification, filtering, aggregation, feature extraction, taxonomies, data mining, clustering, pattern recognition and prediction. Reasoning mechanisms allow high-level context to be deduced or situations to be recognized and often, the output of one process can be used as an input to another. Moreover, reasoning is also used to check the consistency of context and context models.

4.3.2.3 Service Exposure and Control

The third layer is divided into two main capabilities. The first is user-centric and relates to the ability of the user to stay in control of the whole scenario, enabling it to specify when, what, why, who, where and how the data is or can be accessed. This opens the doors for opportunistic communications, as user context is disclosed according to contextual privacy policies and settings, enabling systems and devices to sense how, where and why information and content are being accessed and respond accordingly. Furthermore, through the Human Enabler, users are able to influence the way their behavior is predicted, by controlling how there are being profiled. In an extreme situation, they can build their own profiling algorithms. In fact, people will wish to manage their identities in different ways, sometimes opting for full disclosure, at other times disclosing it only in an anonymous way to preserve their privacy. This is essential for establish-ing and managing trust and for safeguarding privacy, as well as for designing and implementing business security models and policies. The second set of features is associated with the capacity of exposing this information (both raw data and inferred one) to third party service providers, through well-defined web service interfaces. Once again, always considering the users privacy restrictions. Besides exposing user related information, the human enabler allows data to be subscribed, syndicated or updated on request.

4.3.3 Innovations

The analysis of the first results indicated the following key findings:

- It is possible to infer user behavior based on user preferences, social networks and context-aware systems, with the help of reality/data mining techniques.
- Proximity and Similarity are great weight indicators for inferring influence and can be computed or calculated analytically.
- Both online and offline social networks have influence over a person's behavior. User perceived QoE is improved as the methodology delivers personalization, contextualization, interactivity, adaptation and privacy.
- Users are willing to participate in their own profiling experience and the results are positive.

Applying these techniques into different fields of computer social sciences may have significant applicability in different parts of the value chain. Here are some examples:

- Infer and suggest missing information in users profile according to his/her peers contextual information.
- Understand how a specific user can be influenced by another user or community and vice versa.

Understand how similar two users are, even if they do not have friends in common.

Infer strengths of relationships by analyzing interactions within multimedia content available on social networks.

Improve visualization of social relationships according to a set of know or inferred parameters.

Improve users perceived Quality of Experience by focusing on aspects such as personalization, contextualization, interactivity, adaptation and privacy.

Enable users to participate in their own profiling experience.

Leverage user behavior predictions to be accessible to third party providers while concerning user privacy, preferences, desires and intentions.

Improve recommendation systems with predictions that correspond to what users want, need or desire. In other words, it is a personalized and automated electronic word of mouth that reasons contextualized information for a specific user or set of users.

4.4. Privacy in Online Social Networks

4.4.1 Managing Trust in Online Social Networks

4.4.1.1 Introduction

Users of the online social networks may share their experiences and opinions within the networks about an item which may be a product or service. The user faces the problem of evaluating trust in a service or service provider before making a choice. Recommendations may be received through a chain of friends' network, so the problem for the user is to be able to evaluate various types of trust opinions and recommendations. This opinion or recommendation has a great in-fluence to choose to use or enjoy the item by the other user of the community. Collaborative filtering system is the most popular method in recommender system. The task in collaborative filtering is to predict the utility of items to a particular user based on a database of user rates from a sample or population of other users. Because of the different taste of different people, they rate differently according to their subjective taste. If two people rate a set of items similarly, they share similar tastes. In the recommender system, this information is used to recommend items that one participant likes, to other persons in the same cluster.

Online trust and reputation systems are emerging as important decision support tools for selecting online services and for assessing the risk of accessing them.

Trust models based on subjective logic are directly compatible with Bayesian reputation systems because a bijective mapping exists between their respective trust and reputation representations. This provides a powerful basis for combining trust and reputation systems for assessing the quality of online services.

4.4.1.2 Online Social Networks

Professor J. A. Barnes has introduced the term "Social Network" in 1967 to describe the associations of people drawn together by family, work, hobby etc.; for emotional, instrumental, appraisal and information support. These networks may operate in many levels from family level to a level of nations and can play important roles in communications among people, organizations and even nations Online

social networks have broader and easier coverage of members worldwide to share information and resources.

The first online social networks were called UseNet Newsgroups (www.usenet.com) designed and built by Duke University graduate students Tom Truscott and Jim Ellis in 1979.

Based on number of registered user and monthly visit; Facebook is the largest and most popular online social network at this moment (www.insidefacebook.com). It had 350 million Monthly Active Users (MAU) at the beginning of January 2010. But it has been growing too fast around the world since then. Now, at the beginning of February 2010, the number increases to 373 million MAU across the world. As on 10 February 2010, roughly 23 million more people are using Facebook compared to 30 days ago, many in countries with big populations around the world.

Table 1 Brief Timeline of online social networking

1971	Ray Tomlinson invents email
1973	First group chat program
1975	First mailing list, called *MsgGroup*
	First computer conferencing system
1978	First Multi-User Dungeon (MUD) for multi-user gaming
1979	*USENET* newsgroups created
1984	Birth of the *Fido* network of Bulletin Board Systems (BBSes)
1985	Whole Earth Letronic Link (WELL) community begins
1988	Internet Relay Chat (IRC) invented
1991	Tim Berners-Lee posts "World-Wide Web: Executive Summary" to *USENET* Group.
	"Gopher", the first simple menu-driven client to Internet resources launches
1992	Berners-Lee creates his "What's New?" page, arguably the first blog
1993	Howard Rheingold publishes *The Virtual Community*
	Mosaic Web browser is released
1994	"Christ is coming" is the first spam on *USENET*
1995	Ward Cunningham launches the first wiki
	AltaVista, the first full Web search engine, launches
1996	ICQ: first peer-to-peer instant messaging appears
	January: 100,000 Web servers
1997	April: 1,000,000 Web servers
	Slashdot, the first blog to enable reader comments, goes online
	Jorn Barger coins the term "Weblog"
1998	Open Directory Project (DMOZ), later acquired by *Netscape*
1999	Peter Merholz coins the term "blog" as a contraction of "Weblog"
	LiveJournal and *Blogger* launch
	Kuro5hin, a blog where users vote for what goes to the front page, launches
2000	*HotOrNot.com* created with zero capital
2001	*Wikipedia*, an open collaborative wiki encyclopedia, goes live
	Movable type (leading blog software) initial beta release
2002	10,000,000th Web server goes live
	10,000,000th post on *Blogger*
	Friendster launches
2003	Venture capital investment in social network space exceeds $50 million
	LiveJournal and *Friendster* pass one million accounts
	Skype released
	LinkedIn, social network focused on business professionals, secures

	Series A financing of $4.7 million led by Sequoia Capital
2004	*Skype* hits ten million downloads
	Social Networking *Metalist* (SocialSoftware.BlogsInc.com) lists more than 200 different social networking systems
2005	*Skype* hits 100 million downloads
2006	*Google* acquires *YouTube*, video social network, for a stock transaction worth $1.65 billion

(continued)

Table 2 (continued)

2007	IBM launches enterprise social networking suite
	LinkedIn surpasses 10,000,000 members
	Germany social networking site *OpenBC/Xing* successful IPO
	Wikipedia exceeds 1,700,000 English articles
2008	The fastest growing sites were Twitter (664%), Tagged (421%), and Ning (303%) MySpace had 58.4 million unique visitors in December, Facebook had 55.2 million
	Facebook passed MySpace in time per person: 2 h, 7 min–1 h, 40 min
2009	*Facebook* hits 350 million registered users

Table 3 Top ten mostly visited social networks in Jan'09– based on MAU

Rank	Site	Monthly visit
1	facebook.com	1,191,373,339
2	myspace.com	810,153,536
3	twitter.com	54,218,731
4	flixster.com	53,389,974
5	linkedin.com	42,744,438
6	tagged.com	39,630,927
7	classmates.com	35,219,210
8	myyearbook.com	33,121,821
9	livejournal.com	25,221,354
10	imeem.com	22,993,608

Table 4 Country wise monthly growth of Facebook users- as on Feb 2010

Country	1/1/2010	1/2/2010	Change	Change (%)
U.S.	102,681,240	108,062,900	5,381,660	5
Indonesia	15,301,280	17,301,760	2,000,480	13
Turkey	16,961,140	18,556,840	1,595,700	9
U.K.	23,076,700	24,342,820	1,266,120	5
France	14,301,020	15,498,220	1,197,200	8
Mexico	6,671,560	7,624,120	952,560	14
Germany	5,796,940	6,674,740	877,800	15
India	5,658,080	6,342,800	684,720	12
Philippines	8,806,300	9,317,180	510,880	6
Brazil	2,373,520	2,869,920	496,400	21

4.4.2 Trust in Online Environment

Trust has become important topic of research in many fields including sociology, psychology, philosophy, economics, business, law and IT. It is not a new topic to discuss. In fact, it has been the topic of hundreds books and scholarly articles over a long period of time. Trust is a complex word with multiple dimensions. A vast literature on trust has grown in several area of research but it is relatively confusing and sometimes contradictory, because the term is being used with a variety of meaning.

Table 5 Top ten largest social networks in Feb'10 – based on registered users

No.	Network name	Web link	Reg user
1	Facebook	www.facebook.com	350,000,000
2	QZone (Chinese)	http://qzone.qq.com	200,000,000
3	MySpace	www.myspace.com	130,000,000
4	Windows Live Spaces	http://home.spaces.live.com	120,000,000
5	Habbo	www.habbo.com.au	117,000,000
6	Orkut	www.orkut.com	100,000,000
7	Friendster	www.friendster.com	90,000,000
8	Hi5	www.hi5.com	80,000,000
9	Flixster	www.flixster.com	63,000,000
10	Netlog	www.netlog.com	59,000,000

Though dozens of proposed definitions are available in the literature, a complete formal unambiguous definition of trust is rare. In many occasions, trust is used as a word or concept with no real definition. Hussain and Chang present an overview of the definitions of the terms of trust and reputation from the existing literature. They have shown that none of these definitions is fully capable to satisfy all of the context dependence, time dependence and the dynamic nature of trust. Trust is such a concept that crosses disciplines and also domains. The focus of definition differs on the basis of the goal and the scope of the projects. Two generalized definitions of trust defined by Jøsang et al. which they called reliability trust (the term "evaluation trust" is more widely used by the other researchers, therefore we use this term) and decision trust respectively will be used for this work. Evaluation trust can be interpreted as the reliability of something or somebody. It can be defined as the subjective probability by which an individual, A, expects that another individual, B, performs a given action on which its welfare depends. On the other hand, the decision trust captures broader concept of trust. It can be defined as the extent to which one party is willing to depend on something or somebody in a given situation with a feeling of relative security, even though negative consequences are possible.

4.4.3 Trust Models Based on Subjective Logic

Subjective logic is a type of probabilistic logic that explicitly takes uncertainty and belief ownership into account. Arguments in subjective logic are subjective opinions about states in a state space. A binomial opinion applies to a single proposition, and can be represented as a Beta distribution. A multinomial opinion applies to a collection of propositions, and can be represented as a Dirichlet distribution.

Subjective logic defines a trust metric called *opinion* denoted by !x D A .b; *u*; a/, which expresses the relying party A's belief over a state space X . Here E b, *u* belief masses over the states of X , and *u* represent uncertainty mass where E 2 [0, 1] and E C D E 2 and is used for computing the probability expectation value of a state x as E.x/ D b.x/ a.x/*u*, meaning that a determines how uncertainty contributes to E.x/. Binomial opinions are expressed as !x D *(b, d, u, a)* where d denotes disbelief in x. When the statement x for example says "*David is honest and reliable*", then the opinion can be interpreted as reliability trust in David. As an example, let us assume that Alice needs to get her car serviced, and that she asks Bob to recommend a good car mechanic. When Bob recommends David, Alice would like to get a second opinion, so she asks Claire for her opinion about David. This situation is illustrated in Fig. 4.4.3.1 below where the indexes on arrows indicate the order in which they are formed.

When trust and referrals are expressed as subjective opinions, each transitive trust path Alice ! Bob ! David, and Alice ! Claire ! David can be computed with the *transitivity operator*, where the idea is that the referrals from Bob and Claire are discounted as a function Alice's trust in Bob and Claire respectively. Finally the two

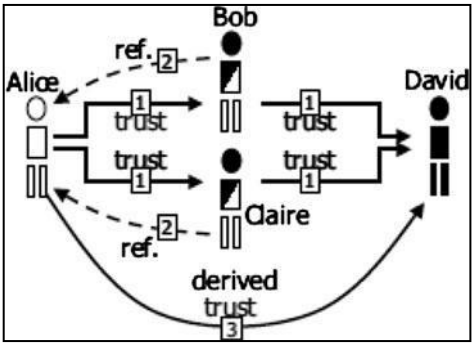


Fig. 22.1 Deriving trust from parallel transitive chains

paths can be combined using the cumulative or averaging fusion operator. These operators form part of *Subjective Logic*, and semantic constraints must be satisfied in order for the transitive trust derivation to be meaningful. Opinions can be uniquely

mapped to beta PDFs, and in this sense the fusion operator is equiv-alent to Bayesian updating. This model is thus both belief-based and Bayesian.

A trust relationship between A and B is denoted as [A:B]. The transitivity of two arcs is denoted as ":" and the fusion of two parallel paths is denoted as "}". The trust network of Fig. 4.4.3.1 can then be expressed as:

$$\text{ŒA; D D .ŒA; B W ŒB; D / } \} \text{ .ŒA; C W ŒC ; D /} \quad (4.4.3.1)$$

The corresponding transitivity operator for opinions denoted as "˝" and the corresponding fusion operator as "°". The mathematical expression for combining the opinions about the trust relationships of Fig. 4.4.3.1 is then:

$$!D^{A} \; D \; !B^{A} \text{ ˝ } !D^{B} \; ° \; !C^{A} \text{ ˝ } !D^{C} \quad (4.4.3.2)$$

Arbitrarily complex trust networks can be analysed with TNA-SL which consists of a network exploration method combined with trust analysis based on subjective logic. The method is based on simplifying complex trust networks into a directed series parallel graph (DSPG) before applying subjective logic calculus.

4.4.4 Trust Network Analysis

Trust networks consist of transitive trust relationships between people, organisazions and software agents connected through a medium for communication and interaction. By formalising trust relationships, e.g. as reputation scores or as subjective trust measures, trust between parties within a domain can be derived by analysing the trust paths linking the parties together. A method for trust network analysis using subjective logic (TNA-SL) has been described by Jøsang et al. TNA-SL takes directed trust edges between pairs as input, and can be used to derive a level of trust between arbitrary parties that are interconnected through the network. Even in case of no explicit trust paths between two parties exist; subjective logic allows a level of trust to be derived through the default vacuous opinions. TNA-SL therefore has a general applicability and is suitable for many types of trust networks. A potential limitation with the TNA-SL is that complex trust networks must be simplified to *series-parallel* networks in order for TNA-SL to produce consistent results. The simplification consisted of gradually removing the least certain trust paths until the whole network can be represented in a series-parallel form. As this process removes information it is intuitively sub-optimal.

4.4.4.1 Operators for Deriving Trust

Subjective logic is a belief calculus specifically developed for modeling trust relationships. In subjective logic, beliefs are represented on binary state spaces, where each of the two possible states can consist of sub-states. Belief functions on binary state spaces are called *subjective opinions* and are formally expressed in the form of an ordered tuple $!x^{A}$ D .b; d ; *u*; a/, where b, d , and *u* represent belief, disbelief and uncertainty respectively where b; d ; *u* 2 [0, 1] and b C d C *u* D 1.

The base rate parameter a 2 [0, 1] represents the base rate probability in the absence of evidence, and is used for computing an opinion's probability expectation value E !$_x^A$ D b C *au*, meaning that a determines how uncertainty shall contribute to E !$_x^A$. A subjective opinion is interpreted as an agent A's belief in the truth of statement x. Ownership of an opinion is represented as a superscript so that for example A's opinion about x is denoted as !$_x^A$. Subjective logic has a sound mathematical basis and is compatible with binary logic and traditional Bayesian analysis. Subjective logic defines a rich set of operators for combining subjective opinions in various ways. Some operators represent generalizations of binary logic and probability calculus, whereas others are unique to belief calculus because they depend on belief ownership.

Transitivity is used to compute trust along a chain of trust edges. Assume two agents A and B where A has referral trust in B , denoted by !$_B^A$, for the purpose of judging the functional or referral trustworthiness of C . In addition B has functional or referral trust in C , denoted by !$_C^B$. Agent A can then derive her trust in C by discounting B 's trust in C with A's trust in B , denoted by !$_C^{AWB}$. By using the symbol "˝" to designate this operator, we define

$$!_C^{A\,B\,W} \;D\; !_B^{A} \;\breve{}\; !_C^{B} \qquad \begin{array}{l} b_C^{AWB} \;D\; b_B^A\, b_C^B \\ d_C^{AWB} \quad b_{D_B}^A\, d_C^B \\ u_C^{A\,B\,W} \;D\; d_B^{A}\;C\; u_B^{A} \quad C\; b_B^{A\,B}\, u_C \\ a_C^{AWB} \;D\; a_C^B : \end{array} \tag{4.4.4.1}$$

The effect of discounting in a transitive chain is that uncertainty increases, not disbelief. Cumulative *Fusion* is equivalent to Bayesian updating in statistics. The cumulative fusion of two possibly conflicting opinions is an opinion that reflects both opinions in a fair and equal way. Let !$_C^A$ and !$_C^B$ be A's and B 's trust in C respectively. The opinion !$_C^{A\}B}$ is then called the fused trust between !$_C^A$ and !$_C^B$, denoting an imaginary agent [A; B]'s trust in C , as if she represented both A and

. By using the symbol "°" to designate this operator, we define

$$!_C^{A\}B} \;D\; !_B^{A} \;^{\circ}\; !_C^{B} \qquad \begin{array}{l} b_C^{A\}B} \\ d_C^{A\}B} \\ u_C^{A\}B} \\ a_C^{A\}B} \end{array}$$

$$\begin{array}{l} D\;\; b_C^A\, u_C^B \;C\; b_C^B\, u_C^A \quad u_C^A \;C\; u_C^B \quad u_C^A\, u_C^B \\ D\;\; d_C^A\, u_C^B \;C\; d_C^B\, u_C^A \quad u_C^A \;C\; u_C^B \quad u_C^A\, u_C^B \\ D\;\; u_C^A\, u_C^B u_C^A \;C\; u_C^B \quad u_C^A\, u_C^B \\ D\; a_C^A : \end{array} \tag{4.4.4.2}$$

where it is assumed that $a_C^A = a_C^B$. Limits can be computed for $u_C^A = u_C^B = 0$. The effect of the cumulative fusion operator is to amplify belief and disbelief and reduce uncertainty.

4.4.5. Trust Transitivity Analysis

Assume two agents A and B where A trusts B, and B believes that proposition x is true. Then by transitivity, agent A will also believe that proposition x is true. This assumes that B recommends x to A. In our approach, trust and belief are formally expressed as opinions. The transitive linking of these two opinions consists of discounting B's opinion about x by A's opinion about B, in order to derive A's opinion about x. This principle is illustrated in Fig. 4.4.5.1 below. The solid arrows represent initial direct trust, and the dotted arrow represents derived indirect trust.

Trust transitivity, as trust itself, is a human mental phenomenon, so there is no such thing as objective transitivity, and trust transitivity therefore lends itself to different interpretations. We have identified two main difficulties. The first is related to the effect of A disbelieving that B will give a good advice. The second difficulty relates to the effect of base rate trust in a transitive path.

4.4.5.1 Uncertainty Favoring Trust Transitivity

A's disbelief in the recommending agent B means that A thinks that B ignores the truth value of x. As a result A also ignores the truth value of x.

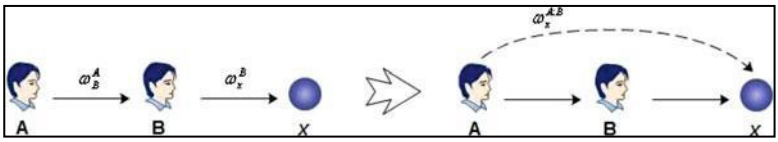


Fig. 4.4.5.1.1 Principle of trust transitivity

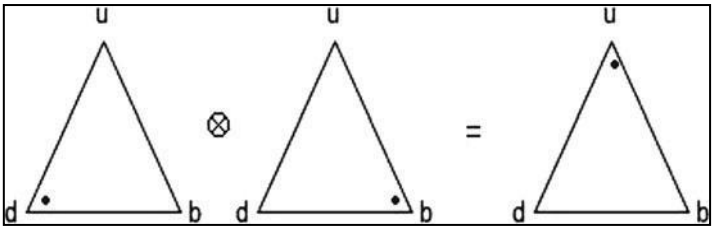


Fig. 4.4.5.1.2 Example of applying the discounting operator for independent opinions

4.4.5.2 Opposite Belief Favoring

A's disbelief in the recommending agent B means that A thinks that B consistently recommends the opposite of his real opinion about the truth value of x. As a result,

not only disbelieves in x to the degree that B recommends belief, but she also believes in x to the degree that B recommends disbelief in x, because the combination of two disbeliefs results in belief in this case

4.4.5.3 Base Rate Sensitive Transitivity

In the transitivity operators, a_B^A had no influence on the discounting of the recommended b_x^B ; d_x^B ; $u^B{}_x$ parameters. This can seem counterintuitive in many cases such as in the example described next.

Imagine a stranger coming to a town which is know for its citizens being honest. The stranger is looking for a car mechanic, and asks the first person he meets to direct him to a good car mechanic. The stranger receives the reply that there are two car mechanics in town, David and Eric, where David is cheap but does not always do quality work, and Eric might be a bit more expensive, but he always does a perfect job. Translated into the formalism of subjective logic, the stranger has no other info about the person he asks than the base rate that the citizens in the town are honest. The stranger is thus ignorant, but the expectation value of a good advice is still very high. Without taking a_B^A into account, the result of the definitions above would be that the stranger is completely ignorant about which of the mechanics is the best. An intuitive approach would then be to let the expectation value of the stranger's trust in the recommender be the discounting factor for the recommended b_x^B ; d_x^B parameters.

4.4.5.4 Mass Hysteria

One of the strengths of this work is in its analytical capabilities. As an example, consider how mass hysteria can be caused by people not being aware of dependence between opinions. Let's take for example; person A recommend an opinion about a particular statement x to a group of other persons. Without being aware of the fact that the opinion came from the same origin, these persons can recommend their opinions to each other

4.4.6 The Dirichlet Reputation System

Reputation systems collect ratings about users or service providers from members in a community. The reputation centre is then able to compute and publish reputation scores about those users and services. Figure 4.4.6.1 illustrates a reputation centre where the dotted arrow indicates ratings and the solid arrows indicate reputation scores about the users.

Multinomial Bayesian systems are based on computing reputation scores by statistical updating of Dirichlet Probability Density Functions (PDF), which therefore

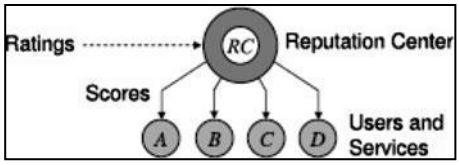


Fig. 4.4.6.1 Simple reputation system

are called Dirichlet reputation systems. A posteriori (i.e. the updated) reputa-tion score is computed by combining a priori (i.e. previous) reputation score with new ratings.

In Dirichlet reputation systems agents are allowed to rate others agents or services with any level from a set of predefined rating levels, and the reputation scores are not static but will gradually change with time as a function of the received ratings. Initially, each agent's reputation is defined by the base rate reputation which is the same for all agents. After ratings about a particular agent have been received, that agent's reputation will change accordingly.

Let there be k different discrete rating levels. This translates into having a state space of cardinality k for the Dirichlet distribution. Let the rating level be indexed by i . The aggregate ratings for a particular agent are stored as a cumulative vector, expressed

$$R \quad .R.L$$
$$\text{asW } E \text{ D } E \;_i /ji \text{ D } 1 : : : k/: \qquad (4.4.6.1.1)$$

This vector can be computed recursively and can take factors such as longevity and community base rate into account.

For visualisation of reputation scores, the most natural is to define the reputation score as a function of the probability expectation values of each rating level. Before any ratings about a particular agent y have been received, its reputation is defined by the common base rate aE. As ratings about a particular agent are collected, the aggregate ratings can be computed recursively (8, 9) and the derived reputation scores will change accordingly. Let E

$$E$$
$$S_{E_y w} \quad S_{E_y} .L/_i \text{ D } \frac{R \;.L/ \;C \;C\, a.L \;/}{C \qquad k \qquad R \;.L/} \Big|_j{}^i \text{ D } 1 : : : k \;: \qquad (4.4.6.1.2)$$
$$C \quad _{j\,D1} \; E_y \quad _j$$

R represent a target agent's aggregate ratings.

is the corresponding multinomial probability reputation score. The parameter C represents the non-informative prior weight where C D 2 is the value of choice, but larger value for the constant C can be chosen if a reduced influence of new evidence over the base rate is required.

The reputation score S can be interpreted like a multinomial probability measure as an indication of how a particular agent is expected to behave in future transactions. It can easily be verified that

$$\sum_{D1}^{k} S .L_{E\ i/D1:} \qquad (4.4.6.4.3)$$

While informative, the multinomial probability representation can require considerable space on a computer screen because multiple values must be visualised. A more compact form can be to express the reputation score as a single value in some predefined interval. This can be done by assigning a point value to each rating level L_i, and computing the normalised weighted point estimate score .

Assume e.g. k different rating levels with point values $v.L_i$ / evenly distributed in the range [0, 1] according to $v.L_i / D^{\frac{i-1}{k-1}}$. The point estimate reputation score of a reputation $\overset{E}{R}$ is then:

$$D \sum_{D1}^{k} {}_{v.L_i\ E} /S .L /:_i \qquad (4.4.6.4.4)$$

A point estimate in the range [0, 1] can be mapped to any range, such as 1–5 stars, a percentage or a probability.

Bootstrapping a reputation system to a stable and conservative state is important. In the framework described above, the base rate distribution aE will define initial default reputation for all agents. The base rate can for example be evenly distributed over all rating levels, or biased towards either negative or positive rating levels. This must be defined when setting up the reputation system in a specific market or community.

As an example we consider five discrete rating levels, and the following sequence of ratings:

Periods 1–10: L1 Mediocre
Periods 11–20: L2 Bad
Periods 21–30: L3 Average
Periods 31–40: L4 Good
Periods 41–50: L5 Excellent

The longevity factor is $\text{œ} D 0{:}9$, and the base rate is dynamic. The evolution of the scores of each level as well as the point estimate are illustrated in Fig. 4.4.6.2.

In Fig. 4.4.6.2 the multinomial reputation scores change abruptly between each sequence of ten periods. The point estimate first drops as the score for L1 increases during the first ten periods. After that the point estimate increases relatively smoothly during the subsequent 40 periods. Assuming a dynamic base rate and an indefinite series of L5 (Excellent) ratings, the point estimate will eventually converge to 1.

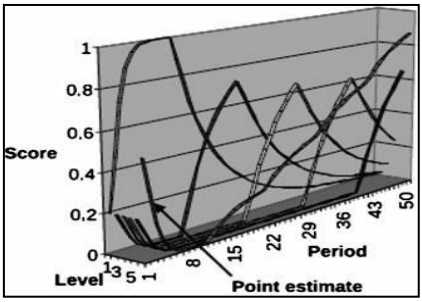


Fig. 4.4.6.2 Scores and point estimate during a sequence of varying ratings

4.5 Combining Trust and Reputation

A bijective mapping can be defined between multinomial reputation scores and opinions, which makes it possible to interpret these two mathematical representations as equivalent. The mapping can symbolically be expressed as:

$$! \$ \quad \text{Ł} \quad R \tag{4.5.1}$$

This equivalence which is presented with proof in (3) is expressed as:

Theorem . Equivalence Between Opinions and Reputations. *Let* ! D .b; *u*; a/ *be an opinion, and* R *be a reputation, both over the same state space X so that the base rate* a *also applies to the reputation. Then the following equivalence holds (4.5.1):*

For u ¤ 0*:*

$$b.x/ \quad D \frac{C \quad R.x_i/}{C \quad C} \quad u \quad D \frac{C \quad R.x_i/}{C \quad C} \quad \text{„} \quad R.x/ \quad D \frac{C \quad b.x/}{u} \quad u C \quad b.x/ \quad D 1 \tag{4.5.2}$$

For u D 0*:*

$$b.x/ \quad D \quad .x/ \quad u D 0 \quad \text{„} \quad R.x/ \quad D \quad .x/ \quad R.x/ \quad D \quad .x/ \quad 1 \quad m.x_i/ \quad D 1 \tag{4.5.3}$$

The case *u* D *0* reflects an infinite amount of aggregate ratings, in which case the parameter determines the relative proportion of infinite ratings among the rating levels. In case *u* D *0* and .x_i / D 1 for a particular rating level x_i , then R.x/ D 1 and all the other rating parameters are finite. In case .x_i / 1= k for all i D 1 : : : k, then all the rating parameters are equally infinite. As already indicated, the non-informative prior weight is normally set to C D *2*.

Multinomial aggregate ratings can be used to derive binomial trust in the form of an opinion. This is done by first converting the multinomial ratings to binomial ratings below, and then to apply Theorem.

Let the multinomial reputation model have k rating levels x_i I i D 1; : : : k, where R.x_i / represents the ratings on each level x_i , and let represent the point estimate reputation score. Let the binomial reputation model have positive and negative ratings r and s respectively. The derived converted binomial rating param-eters .r; s/ are given by:

$$r = \frac{\sum_{i=1}^{k} R_{E_y} \cdot x_i}{s} \qquad (4.5.4)$$

$$= \frac{\sum_{i=1}^{k} E_y \cdot x_i}{r}$$

With the equivalence of Theorem it is possible to analyse trust networks based on both trust relationships and reputation scores. Figure 4.4.7.1 illustrates a scenario where agent A needs to derive a measure of trust in agent F .

Agent B has reputation score E_B R_{RC} (arrow 1), and agent A has trust $!^A_{RC}$ in the Reputation Centre (arrow 2), so that A can derive a measure of trust in B (arrow 3). Agent B 's trust in F (arrow 4) can be recommended to A so that A can derive a measure of trust in F (arrow 5). Mathematically this can be expressed as:

$$!^A_F = !^A_{RC} \otimes E_B \otimes !^B_F \qquad (4.5.5)$$

The compatibility between Bayesian reputation systems and subjective logic makes this a very flexible framework for analysing trust in a network consisting of both reputation scores and private trust values.

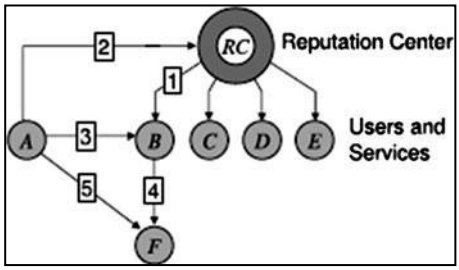


Fig. 4.5.5.1 Combining trust and reputation

4.6 Trust Derivation Based on Trust Comparisons

It is possible that different agents have different trust in the same entity, which intuitively could affect the mutual trust between the two agents. Figure 4.6.1 illustrates a scenario where A's trust $!^A_B$ (arrow 1) conflicts with B 's reputation score E_{RC} R_B (arrow 2). As a result A will derive a reduced trust value in the Reputation Centre (arrow 3). Assume that A needs to derive a trust value in E , then the reduced trust value must be taken into account when using *RC*'s reputation score for computing trust in E . The operator for deriving trust based on trust conflict produces a binomial opinion over the binary state space $\{x, \neg x\}$, where x is a proposition that can be interpreted as x: *"RC provides reliable reputation scores"*, and $\neg x$ is its complement. Binomial opinions have the special notation $!_x = (b, d, u, a)$ where d represents disbelief in proposition x.

What represents difference in trust values depends on the semantics of the state space. Assume that the state space consists of five rating levels, then Fig. 4.6.2a represents a case of polarised ratings, whereas Fig. 4.6.2b represents a case of average ratings. Interestingly they have the same point estimate of 0.5 when computed

We will define an operator which derives trust based on point estimates. Two agents having similar point estimates about the same agent or proposition should induce mutual trust, and dissimilar point estimates should induce mutual distrust.

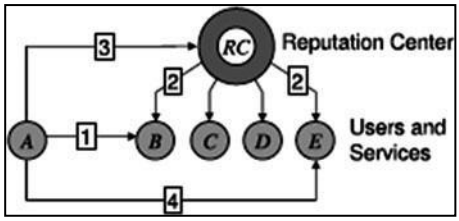


Fig. 4.6.1 Deriving trust from conflicting trust

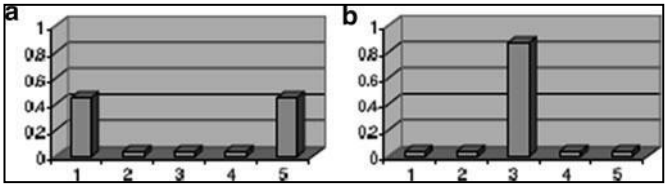


Fig. 4.6.2 Comparison of polarized and average reputation scores. (**a**) Reputation score from polarized ratings (**b**) Reputation score from avg ratings

Definition (Trust Derivation Based on Trust Comparison). Let $!^{A}$ and $!^{RC}$ be two opinions on the same state space B with a set rating levels. A's trust in RC based on the similarity between their opinions is defined as:

$$d^{A}_{RC} \quad \begin{matrix} D & E_B \\ R^A & \end{matrix} \quad \begin{matrix} E_B \\ R_{RC} \end{matrix}$$

$$!_{RC}^{A} \; D \; !_{B}^{A} \; \# \; !_{B}^{RC} \quad \text{where} \quad u_{RC}^{A} \; D \; Max\, u_{B}^{A} ; u_{B}^{RC}$$

$$b_{RC}^{A} \; D \; 1 b_{RC}^{A} \qquad u_{RC}^{A}$$

The interpretation of this operator is that disbelief in *RC* is proportional to the greatest difference in point estimates between the two opinions. Also, the uncertainty is equal to the greatest uncertainty of the two opinions.

With the trust comparison trust derivation operator, A is able to derive trust in *RC* (arrow 3). With the above described trust and reputation measures, A is able to derive trust in E expressed as:

$$!_{E}^{A} \; D \; !_{RC}^{A} \; " \; !_{E}^{RC} \qquad (22.17)$$

This provides a method for making trust derivation more robust against unreliable or deceptive reputation scores and trust recommendations.

UNIT 5 Visualization of Social Networks

5.1. Visualization of Social Networks

5.1.1 Introduction

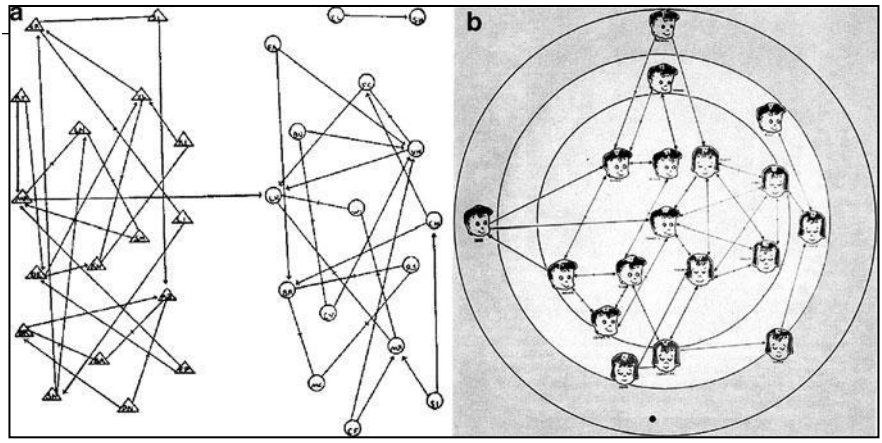


Fig. 5.1.1 Visualization of social network using hand-drawn images. (**a**) Friendship choices among fourth grade children. (**b**) The sociogram of a first grade class

Visualization is a powerful technique to facilitate exploring social relationships within social networks. Although many visualization techniques have been focused on the discussions, such as displaying fine graph layouts, coloring, and presenting clear node-edge relations, visualizing complex relations is still challenging to social network visualization.

Visualizing social networks thus plays an important role of accessing social networks and connecting people in a more effective and efficient way.

Today, many visualization applications have been also employed in different online social networks to help people manipulate their social relationships and access the abundant resources on the Internet.

5.1.2 Social Network Analysis

Social networks are built when actors belonging to different social groups are connected to each other. In early 1930s, Moreno brought in the concept of *sociograms* to represent social networks and started to explore the social relations in a formal study. The metrics:

5.1.2.1 Graph Theory

Many fundamental concepts and metrics in social network analysis are derived from graph theory, because graph theory formally represents social networks with

structural properties. In the following descriptions of terms, some fundamental definitions related to graph theory and social network analysis are depicted mainly referring to:

Node degree:

In graph theory, the degree of a node in a graph is the number of edges incident to the node. If there are loops in the graph, the degree of a node will be counted twice. Therefore, the maximum number of unique edges in a graph can be obtained when the loops are excluded. There are at most N .N 1/=2 edges for an undirected graph and at most N .N 1/ edges for a directed graph, where N is the number of nodes.
Node density:

The density of an undirected graph can be defined as .2 E /=N .N 1/, where E is the number of edges. On the other hand, the density of a directed graph can be defined as E=N .N 1/.
Path length:

The path length is the number of edges in the sequence that a walk follows. In a path, all nodes and edges appear only once in the sequence. Therefore, the path length can be defined as the distances between pairs of nodes in a network graph, and average path length is the average of these distances between all pairs of nodes.

Component size:

When the component size is concerned, a connected graph needs to be discovered first since the component size is counted by the number of connected nodes in a graph. A graph is connected if all pairs of nodes are reachable, and for each pair of two nodes, one of them is reachable from the other. On the other hand, if a graph is not connected, the graph can be partitioned into several connected subgraphs where each component size can be calculated by the number of connected nodes in each subgraph.

5.1.2.2 Centrality

One of the key applications in social networks is to identify the most important or central nodes in the network. The measure of centrality is thus used to give a rough indication of the social power of a node based on how well they connect the network. HITS and PageRank are two most famous representatives using centrality for ranking. HITS analyzes the important nodes based on calculating Authorities (in-degrees) and Hubs (out-degrees), and PageRank calculates node values based on out-degrees. In social network analysis, “Degree”, “Betweenness”, and “Closeness” centrality are three most popularly adopted methods to measure the centrality of a social network. In the descriptions below, we depict the distinction between the three popular individual centrality measures: degree centrality, betweenness centrality, and closeness centrality.

Degree centrality:

Degree centrality is defined as the number of edges incident upon a node, and thus it is usually the first way to calculate the nodes that are most potential to determine other nodes. For calculating degree centrality, the nodes that have direct connection to a large

number of nodes are considered. If the edges in a graph are directed, the in-degree centrality is differentiated from the out-degree centrality.

Betweenness centrality:

In addition to degree centrality, betweeness centrality is another key metrics for computing the extent to which a node lies between other nodes in the network. If a node is the only node that links two groups of nodes in the network, this node shall be seen as an important node for keeping the social network together. Therefore, betweenness centrality is to measure the connectivity of the neighbors of a node and to give a higher value for nodes which bridge clusters. Besides, this measure reflects the number of nodes which a node is connecting indirectly through the direct links.

Closeness centrality:

The measure of closeness centrality is to take into account how distant a node is to the other nodes in the network. Hence, closeness centrality is to measure the order of magnitude that a node is near all other nodes in a social network by calculating the mean shortest path for a node to all other nodes in the graph. As mentioned in, nodes that are ranked high with closeness centrality can be seen as the nodes that are more likely to act as information distributors in the social network.

5.1.2.3 Clustering

Many social networks contain subsets of nodes that are highly connected within the subset and have relatively few connections to nodes outside the subset. The nodes in such subsets are likely to share some attributes and form their own communities. Since the detection of these community structures is not trivial, how to efficiently and effectively discover such community structures is important. Therefore, the main measure described below is to help explore the grouping effects by clustering coefficient.

Clustering coefficient:

A clustering coefficient is to measure the degrees of nodes to decide which nodes in a graph tend to be clustered together. Thus, the clustering coefficient measure is to quantify how close its neighbors are to being a complete graph. As the nodes grouped in the real-world social network tend to have relatively high density of ties, the clustering coefficient is also utilized for small world analysis.

5.1.3 Visualization

Visualization plays a crucial role of linking the human vision and computer, helping identify patterns, and extracting insights from large amounts of information. For visualizing social networks, some visual representations are considered appropriate to present network structures, such as node-edge diagrams and matrix representations.

5.1.3.1 Node-Edge Diagrams

A node-edge diagram is an intuitive way to visualize social networks. With the node-edge visualization, many network analysis tasks, such as component size

calculation, centrality analysis, and pattern sketching, can be better presented in a more straightforward manner. Many node-edge layouts have been presented to place the nodes in the graph for users to clearly recognize the structure of the social network. However, different layouts have their own pros and cons to display the network graph depending on the size, complexity, and structure of the social network. For example, some layouts are suitable to display social networks in a moderate size, but they are not suitable for showing larger networks. Three kinds of layouts, namely, random layout, force-directed layout, and tree layout, are described to explain the node-edge diagrams.

Random Layout

A random layout is to put the nodes at random geometric locations in the graph, and thus it may not yield very clear visualization results, particularly when the number of nodes immensely increases, e.g. more than thousands of nodes. However, since a random layout algorithm can efficiently draw the social network graph in linear time, O .N /, sometimes it can be usable to visualize very large network graphs. Therefore, random graphs have been proposed as a possible model to take into account the structural characteristics of instances that appear in many practical applications. Figure 5.1.3.1a shows a random geographic layout.

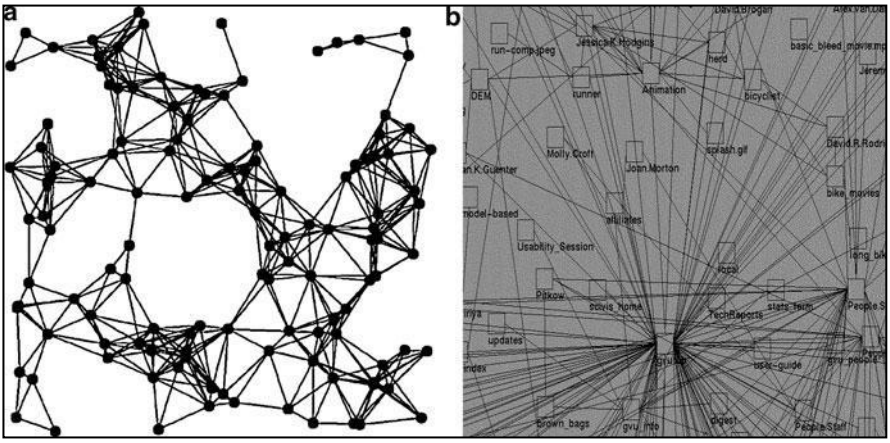


Fig.5.1.3.1 Node-link graph layouts for social networks. (**a**) A random geographic layout. (**b**) A force-based graph layout

Force-Directed Layout

A force-directed layout is also known as a spring layout, which simulates the graph as a virtual physical system. In a force-directed layout, the edges act as spring and the nodes act as repelling objects, just like the Hooke's law and the Coulomb's law. Hence, there exists gravitational attraction or magnetic repulsion between each node in the graph. Generally, an initial random layout will be yielded first, and then the force-directed algorithms will run iteratively to adjust the positions of nodes until all graph nodes and attractive forces between the adjacent nodes run to

convergence. Figure 5.1.3.1b displays a force-based graph layout for forming better layouts of the space.

Since a force-directed layout may take hundreds of iterations to obtain a stable layout, the running time is at least O .N log N / or O .E /, where N is the number of nodes and E is the number of edges. Compared with a random layout, the running cost of a force-directed layout is much higher than that of a random layout, especially when the number of nodes is large. It is therefore not suitable for graphs larger than hundreds of nodes.

Tree Layout

A basic tree layout is to choose a node as the root of tree, and the nodes connected to the root become children of the root node. Nodes that are at more levels away from the root become the grand-children of the root and so on. A tree layout can display a more structural layout than graph layouts by considering more contextual information. Because of the hierarchical nature of a tree layout, trees are more straightforward to grasp human eye than general graphs. Drawing a tree layout thus takes more constraints than drawing a general graph since tree structures are a special case of graphs. Meanwhile, more contextual information of a graph can be extracted to present a hierarchical layout and facilitate network analysis.

For a better visual presentation of domain specific information, more suitable variants of the tree layout were proposed, such as hyperbolic tree layout and a radial tree layout. These tree visualizations utilize the idea of focus C context to better the visualization effects with animation techniques and help users to ob-tain both global and local views of a social network in a 2D display. Figure 5.1.3.2a shows the visualization of a hyperbolic tree view, and Fig.5.1.3.2b shows a radial tree layout. In addition, an advanced visualization technique, called H3, further maps the hyperbolic tree into a 3D space with a fisheye effect to better utilize the visualization space. Figure 5.1.3.c displays the H3 visualization.

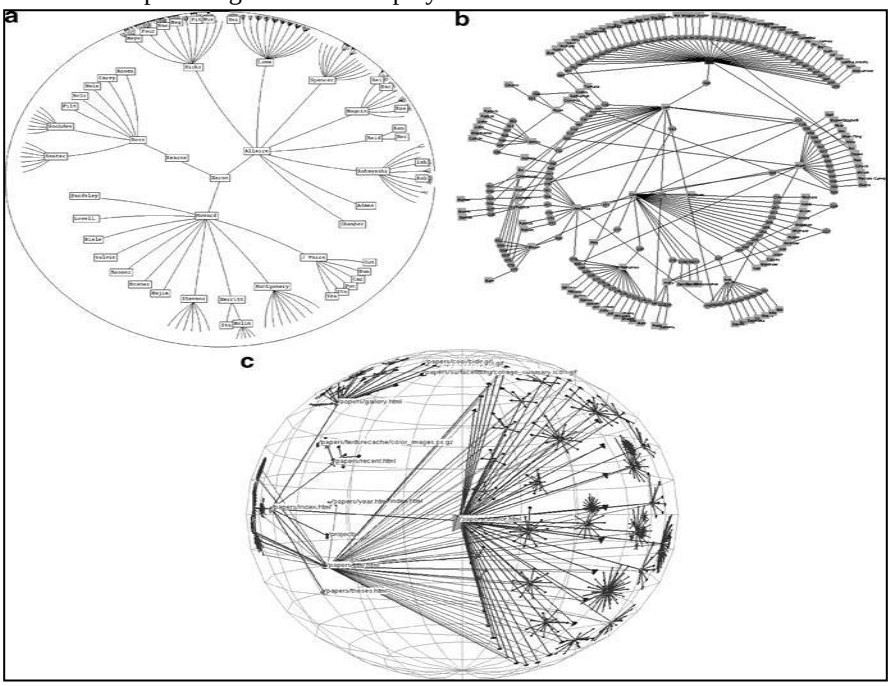


Fig. 5.1.3.2 Three kinds of tree layouts for social network visualization. (**a**) A hyperbolic tree view (**b**) A radial tree layout. (**c**) An H3 view

5.1.3.2 Matrix Representations

Since a social network graph consists of nodes connected with edges, it can be transformed into a simple Boolean matrix whose rows and columns represent the vertices of the graph. Moreover, the Boolean values in the matrix can be further replaced with valued attributes associated with the edges to provide more informative network visualizations. Since a matrix presentation can help minimize the occlusion problems caused by the node-edge diagram, the matrix-based representation of graphs offers an alternative to the traditional node-edge diagrams. With a matrix-based representation, clusters and associations among the nodes can also be better discovered when the number of nodes increases. Particularly, when the relationships are complex, a matrix-based representation can effectively outperform a node-edge diagram in readability since the high connectivity of a node-edge representation will easily diffuse the focus.

In 2006, an enhanced matrix-based representation, called MatrixExplorer, was developed to visualize social networks with a Dual-Representation. In MatrixExplorer can provide users with two synchronized representations of the same network: matrix and node-edge. When a social network is composed of highly interlaced edges, the matrix-based view can help users quickly recognize the associations between nodes. Figure 5.1.3.3 illustrates a matrix-base view of MatrixExplorer with an initial order on the left and a traveling salesman problem (TSP) order on the right. In Fig. 5.1.3.3, a reordered matrix can evidently help users find more clusters. A matrix-based visualization may not entirely replace a conventional node-edge diagram, yet it could complement the shortcomings of a node-edge diagram to better the social network visualization.

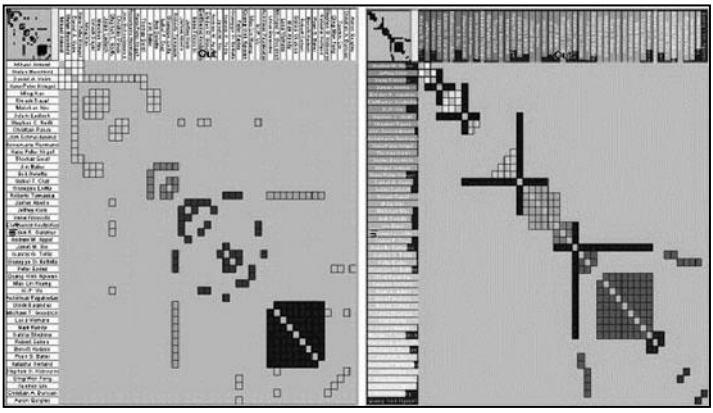


Fig. 5.1.3.3 MatrixExplorer: initial order (*left*) and TSP order (*right*)

5.4 Visualizing Online Social Networks

As the fast development of the Internet, many online social network services are created to connect social relationships among people. Versatile visualization skills are employed to facilitate analyzing online social networks, and more in-depth studies have been investigated to improve the presentation of online social networks. When the attributes of sociality are concerned, online social networks can be classified into different social communities. Therefore, visualizations of online social networks were developed according to the attributes of network sociality to present their network structure including Web communities, email groups, digital libraries, and Web 2.0 services.

5.4.1 Web Communities

After the six degrees of separation concept and Web services were becoming popular in late 1990s, different social network services were created on the Web to help people maintain their social relationships. In addition, many social network websites are developed with interactive visualization interfaces to facilitate people connecting their communities and maintaining social relationships.

In 2003, Club Nexus was established based on the friendship network data of Stanford students and allowed them to explicitly list their friends by their profiles. For example, students registered on Club Nexus were provided with the profiles of their year in school, major, residence, gender, personalities, hobbies and interests to facilitate interacting with their online social networks. The Club Nexus community thus provided very rich profiles for its users to allow for detailed social network analysis, including identifying activities and preferences that determine the formation of friendship.

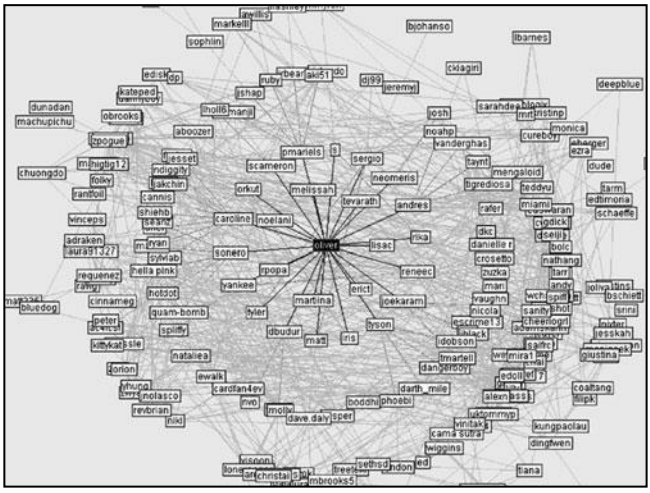


Fig. 5.4.1.1 Club Nexus: visualization of a Web community of Stanford students

In addition to listing actors with their profiles for social network analysis, a modern visualization of social networks, Vizster, was contributed with customized techniques to visualize social relationships and community structures in 2005. Vizster was developed based on node-edge network layouts for exploring connectivity in large graph

structures, supporting visual search and analysis, and automatically identifying and visualizing community structures. Figure 5.4.1.2a shows the visualization of social relationships, and Fig. 5.4.1.2b further demonstrates the linkage view of Vizster visualization. Form the visualization design of Vizster, many advanced search and interactive navigation techniques are developed on Vizster to facilitate the analysis of social networks, such as highlighting, panning, zooming, and distortion techniques. Moreover, visual community analysis is provided to help users construct and explore higher-level structures of their online communities. In Fig. 5.4.1.2c, the visualization of community structures is presented.

As the development of Semantic Web, a project called FOAF (Friend-of-a-friend) was proposed to visualize such human-centric social relationships based on Semantic Web social metadata. With XML/RDF format, the FOAF relations can be explicitly defined for further social network analysis and visualization. Figure 5.4.1.3 shows Semantic Web social networks with FOAF.

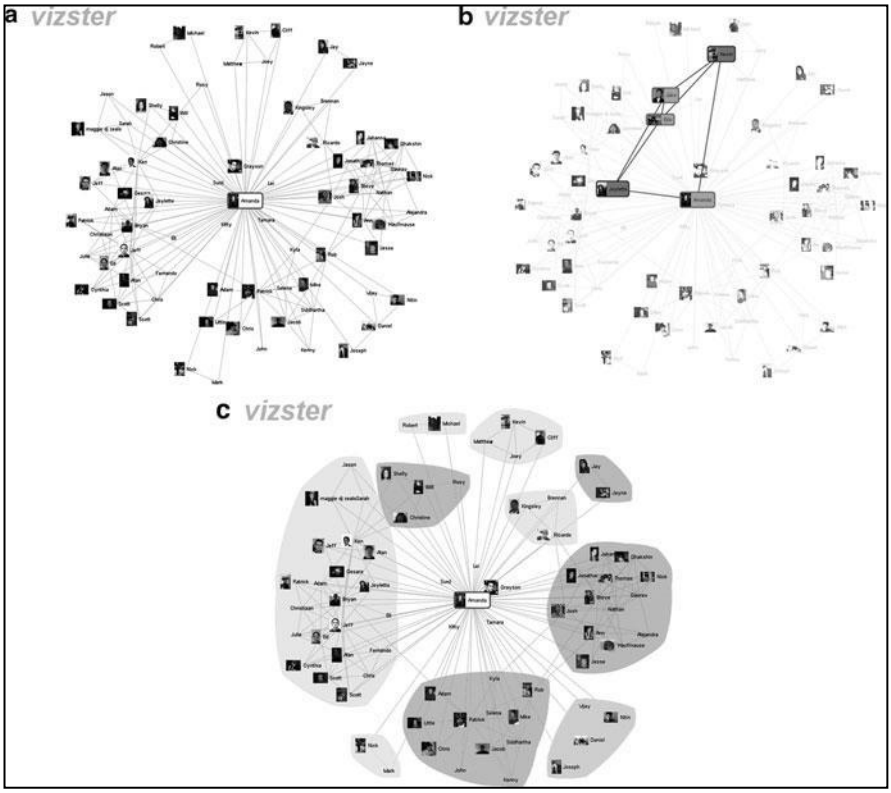


Fig. 5.4.1.2 Vizster: visualization of the Friendster service. (**a**) Visualization of social relations. (**b**) The linkage view of Vizster. (**c**) Visualization of community structures

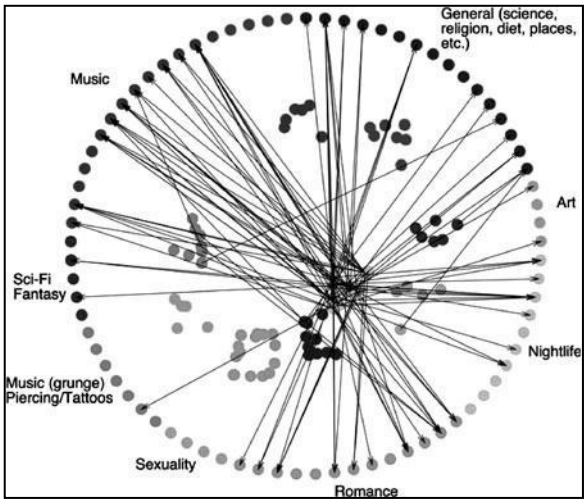


Fig. 5.4.1.3 FOAF: groups of actors with shared interests and social relations

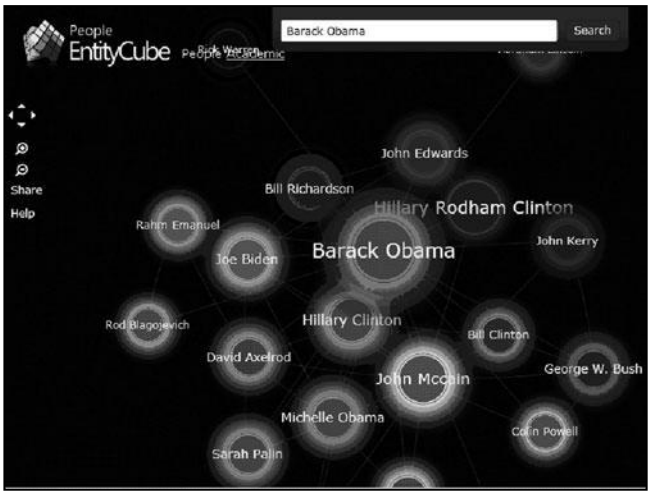


Fig. 5.4.1.4 People EntityCube: visualizing human social relationships

Recently, Microsoft Research Asia proposed a novel object-level search service, called EntityCube, to help people discover real-world entities, such as people, locations, and organizations, and explore their social relationships. In the EntityCube interface, Web entities are summarized from billions of Web pages with a modest presence. Therefore, users can intuitively browse and determine the social relationships among the returned entities. Figure 5.4.1.4 demonstrates People Entity-Cube with querying the American president "Barack Obama". From the visualized objects and relationships connected to Barrack Obama, users can immediately identify people with higher interactions, such as Hillary Clinton and John Mccain.

5.4.2 Email Groups

Email service is one of the most popular applications that people often use to connect each other and deliver messages in their daily lives. For analyzing the social structures of the daily email activities, visualization techniques are employed to explore different patterns. In 2004, Soylent was developed to study the social patterns and the temporal rhythms of daily email activities. Through the Soylent visualization, mutual interactions between different users and groups, and their everyday collaboration activities can be clearly displayed. Some recurrent patterns discovered in the social networks, such as the onion pattern, the nexus pattern, and the butterfly pattern, can thus suggest regular ways of understanding their interactions. For example, the butterfly pattern, named for its two large "wings" surrounding a single center, can be interpreted as a member of a design team also involving in a research team. Figure 5.4.2.1 shows the full network view of Soylent.

In addition to the network view of email visualization, different aspects of social networks can be visualized for analysis. In 2005, two visual metaphors, Social

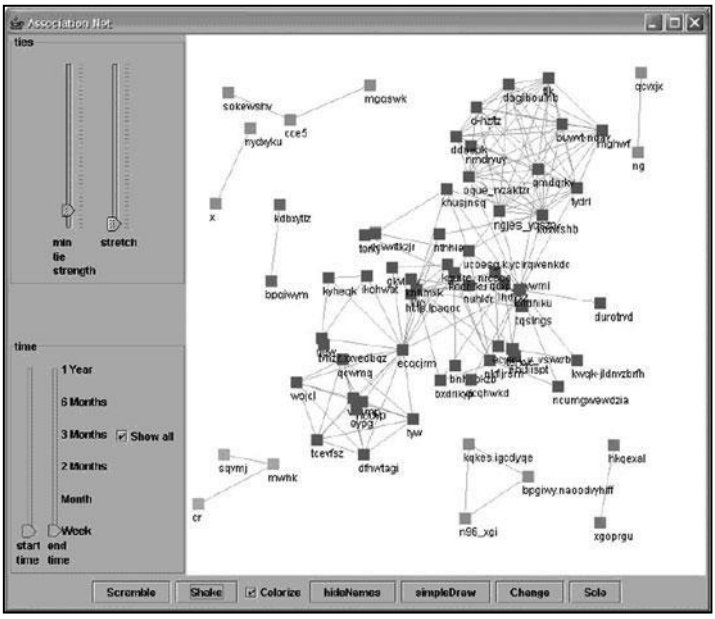


Fig. 5.4.2.1 Soylent: visualizing social relationships among email groups

Network Fragments (SNF) and PostHistory, were employed to visualize the major two dimensions of email activities: people and time. In contrast to the integrated visualization of Soylent, the social relationships derived from the TO and CC lists in email archives are highlighted in SNF, and the temporal rhythms of interactions between ego and individual contacts over time are illustrated with a calendar metaphor in PostHistory, respectively. Figure 5.4.2.2a displays a complex cluster of contacts in SNF, where different colors are utilized to indicate people from different contexts of ego's social life.

In Fig. 27.10b, PostHistory presents social network visualization with a calendar panel on the left and a contacts panel on the right. The email exchange activities with time progress can thus be visualized in an interactive calendar-like interface. More-over, both interfaces can work interoperably with each other to facilitate accessing email-based social network. For example, when users spot an interesting cluster of contacts in SNF, they can turn to PostHistory to locate the patterns of intensive email exchange that made those people's names coalesce into a single cluster.

In 2006, an improved calendar-like visualization interface, called The mail, was developed to help analyze email-based social networks with a chronological sequence and the corresponding topics. Through the analysis of email content, the social relations and mutual interactions between a user and her contacts can be clear presented in Themail. In advance, topic changes over time progress can be substantially traced with the chronological bar. Figure 5.4.2.2c displays the screenshot of Themail showing a user's email exchange with a friend during 18 months.

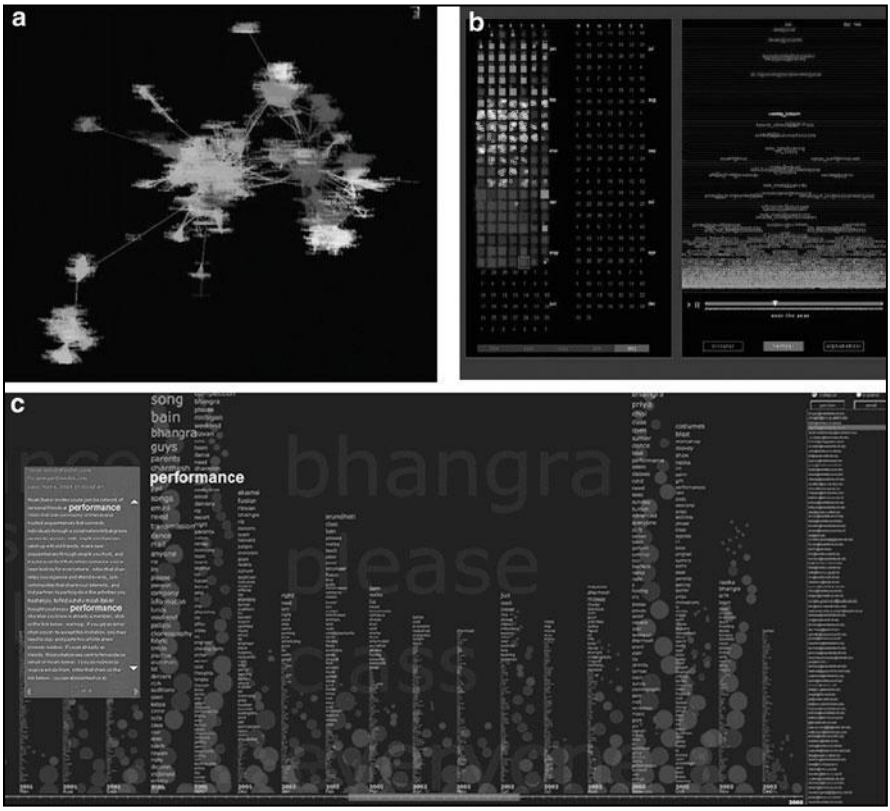


Fig. 5.4.2.2 Visualization of email groups and their social relations. (**a**) SNF: visualization of a complex cluster of contacts. (**b**) PostHistory: visualization interface with calendar and contacts. (**c**) Themail: visualization of email exchange with a friend during 18 months

5.4.3 Digital Libraries

With the speedy publishing of digital contents on the Web, social networks are fast shaped among these electronic publications, particularly among the academic publications. In digital libraries, social networks can be mainly analyzed from two aspects: authors and writings.

5.4.3.1 Co-Authorship Networks

On the aspect of authors, co-authorships can be mined from the existing publications and organize the co-authorship networks. With the visualization of co-authorships, some characteristics, such as clustering coefficient and average path length, can be hence analyzed in co-authorship networks. Figure 5.4.3.1a shows the visualization

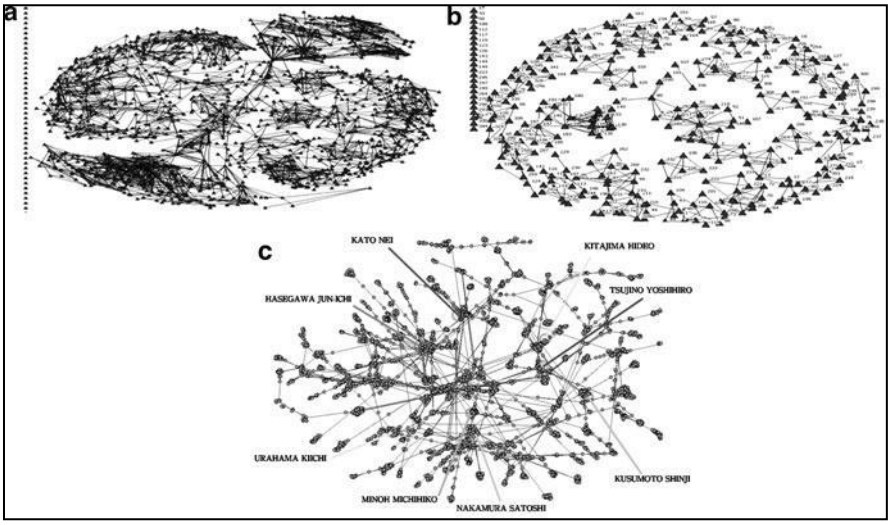


Fig. 5..4.3.1 Visualization of co-authorships in academic digital libraries. (**a**) The co-authorship network of JODL and D-lib community. (**b**) The collaboration network of CiteSeer. (**c**) Eight of the top co-authors from IEICE Transactions D (1993–2005)

of co-authorship networks in the joint community of JODL and D-lib, and Fig. 5.4.3.1b illustrates the collaboration network of CiteSeer, respectively. In 2007, a node-edge representation was also employed in visualizing the co-authorship network in a large Japanese academic society, IEICE, to show the similar characteristics of co-authorship networks in a specific domain. Figure 5.4.3.1c demonstrates eight of the top co-authors with about four thousand nodes from IEICE Transactions D during 1993 and 2005. Beside the co-authorship network, the collaboration network of journal editors can be visualized for network analysis. Figure 5.4.3.2 displays the editorial board network of top 56 journals in the field of digital libraries.

From the visualization of the co-authorship networks illustrated above, a small world graph can be drawn with the connections of authors from different places in the world. In 2005, social network analysis for co-authorship was in-depth studied in digital libraries. Figure 5.4.3.3a illustrates the concept of a small world with calculating the largest component in the Joint Conference on Digital Libraries (JCDL) co-authorship network. Other characteristics of social network analysis,

such as higher clustering coefficient and longer path length, also indicate that co-authors of one author are more likely to publish together in the JCDL community, and authors from different groups are not as well connected as those in other co-authorship communities. In addition to the node-edge representation, a matrix representation was used in the co-authorship network to help analyze different co-authorship patterns. With the matrix representation, the interlaced problem of the node-edge representation caused by a large amount of nodes and complex relations can be effectively improved. Figure 5.4.3.3b shows the visualization of co-authorship network with a matrix representation.

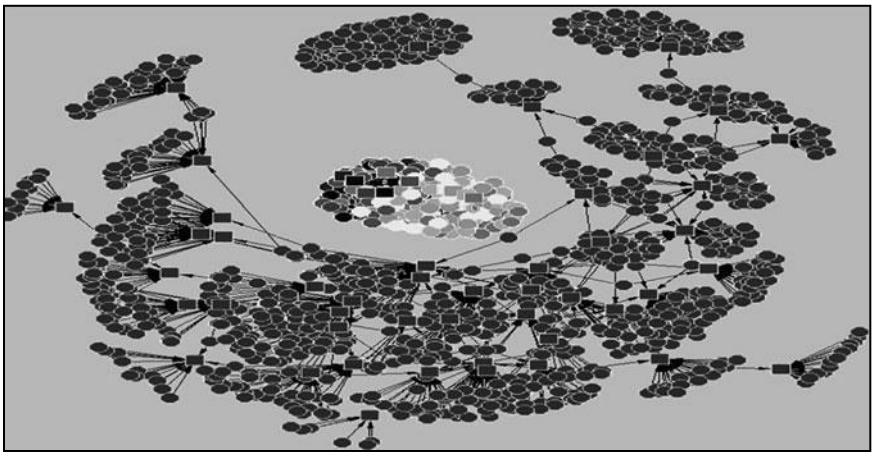


Fig. 5.4.3.2 Visualization of collaboration of journal editors

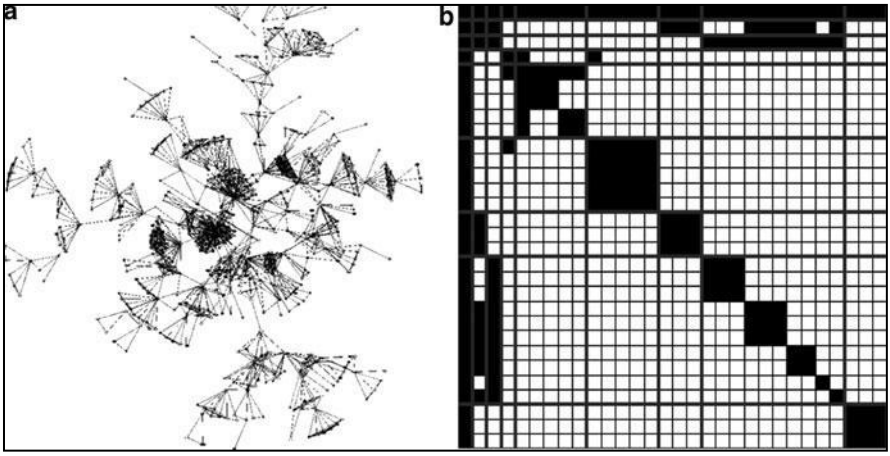


Fig. 5.4.3.3 Visual analysis of co-authorship in academic papers. (**a**) Visualization of co-authorship in JCDL. (**b**) Visualization of co-authorship with the Matrix representation

5.4.3.2 Co-Citation Relations

On contrary to the authorship view, social networks in digital libraries can be discovered from the citations and co-citations among writings themselves. Since references are a crucial part of a document for readers to obtain information source, co-citation social

networks can be formed through the continuously accumulated publications. With proper visualization of co-citation networks, documents with high impacts or similar citation patterns can be immediately identified, and the co-citation relationships can be intuitively observed as well. In 2006, a novel visualization tool, called CircleView, was proposed to visualize academic citation relations

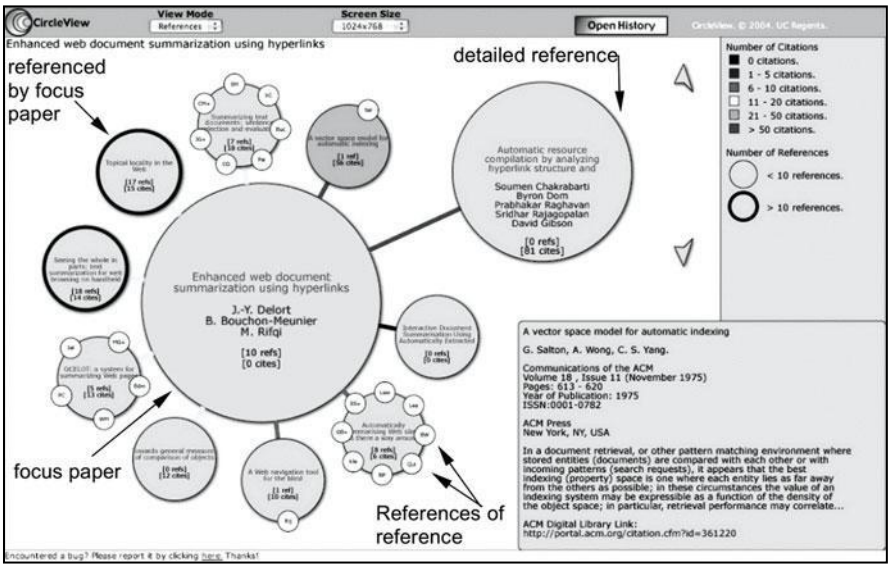


Fig. 5.4.3.4 CircleView: visualization of paper citation relations

with interactive design and highlighted color. In CircleView, the summarized contents of the focus papers, such as references, detail reference, and references of reference, are marked with different colors and highlighted circles. Figure 5.4.3.4 displays the CircleView user interface with scalable visualization and navigation of citation networks.

In 2007, an interactive visualization tool was developed to present large co-citation networks with latent visual cues and allows direct interaction with the visualized graphs. An interaction-based citation visualization system was proposed to visualize larger datasets with scalable functionalities and comple-mented the citation visualization of domain content analysis. Figure 5.4.3.5a displays the full view of network visualization with the interactive control panel in the right. In Fig. 5.4.3.5b, a closer look is presented by click at on of the cluster of Fig. 5.4.3.5a.

In co-citation networks, paper-reference matrices are generally transformed into reference-reference matrices to obtain co-citation relationships. Then these co-citation relationships can be visualized in different representations, typically as node-edge networks, to represent the intellectual structures of scientific domains. However, the reference-reference visualization will bring about tightly knit components and make visual analysis of the intellectual structure a challenging task. In 2009, an innovative visualization technique, called FP-tree, was developed to present co-citation network from a new perspective, namely, visualizing social networks based on a paper-reference matrix instead of using a reference-reference matrix.

The paper-reference matrix was transformed into an FP-tree visualization to analyze the intellectual structure of two domains: Information Visualization

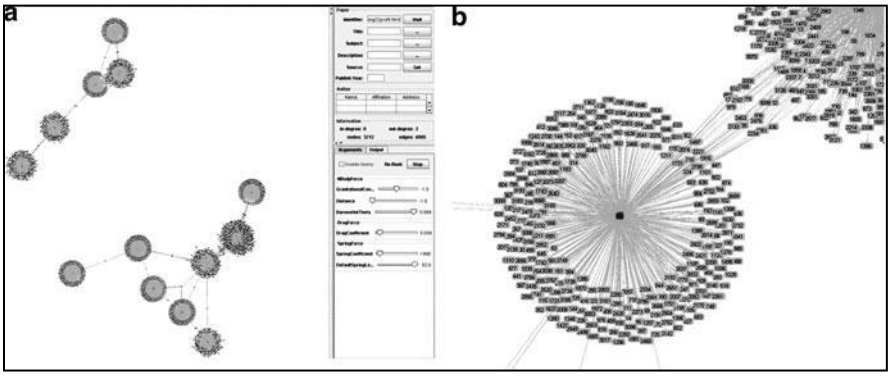


Fig. 5.4.3.5 Interactive visualization of co-citation networks by querying "knowledge management" in CiteSeer . (**a**) Two sub graphs emerged from the co-citation graph. (**b**) A closer look at one of the cluster of (**a**)

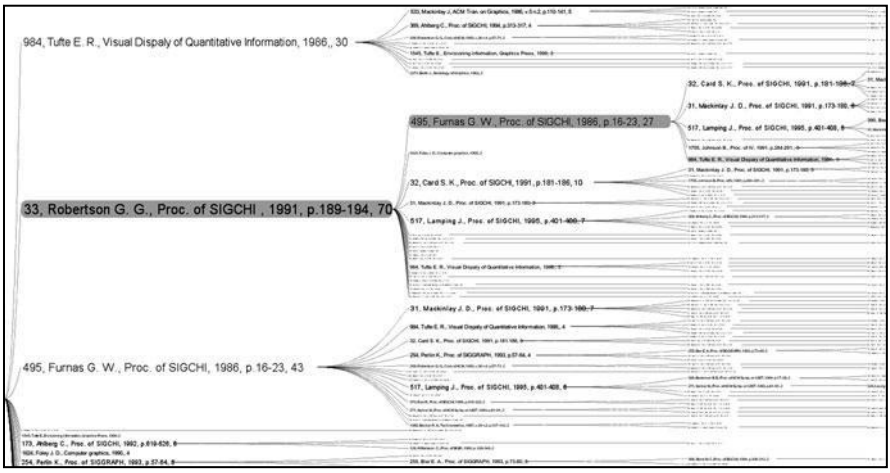


Fig. 5..4.3.6 The FP-tree visualization of the Infovis 2004 contest data

and Sloan Digital Sky Survey (SDSS). The two real-world case studies further reveal that the FP-tree visualization can retain the important information in the intellectual structure of a domain and facilitate domain analysts to reveal fine-grained sub-structure when tightly knit clusters occurred in co-citation networks. Although the FP-tree visualization is friendly to help users analyze the intellectual structure, it will also cause multiple distributions of the same reference and make the tree structure larger than the co-citation network in several magnitude levels. Figure 5.4.3.6 shows the FP-tree visualization of co-citation relationships among three key documents.

5.4.4 Web 2.0 Services

Since the concept of Web 2.0 was proposed in 2004, online social activities are becoming more prosperous than before. Many Web 2.0 applications are popularly accessed by users to connect their social networks, such as Twitter and Facebook. For example, Twitter provides users convenient functionalities to share the up-todate status with their followers. Since the following and the followed relationships can be quickly constructed, large social networks are created over the Twitter service, and many visualization applications are developed to analyze Twitter social networks. In 2007, a visualization interface was built to explore the community structure and properties of Twitter social networks. Figure 5.4.4.1a visualizes the Twitter social network of at least one-way acquaintanceship, and Fig. 5.4.4.1b displays the Twitter social network of mutual acquaintances. From analyzing the community structure and properties of the two graphs, users can intuitively recognize the features and differences of two social groups over Twitter.

Another example of Web 2.0 services that forms large social networks is Facebook. As Facebook was proposed at Harvard in 2004, its social community grows explosively from U.S. to the rest of the world. Until today, Facebook has over 400 million active users worldwide, and large and complex social networks come along with the explosive growth of Facebook communities. Therefore, many interactive visualization applications and tools are developed to assist users to access and analyze such complicated social networks. Nexus is a visualization application on Facebook communities to illustrate their large network graphs. With the Nexus visualization, user relationships can be properly displayed, and user names will pop

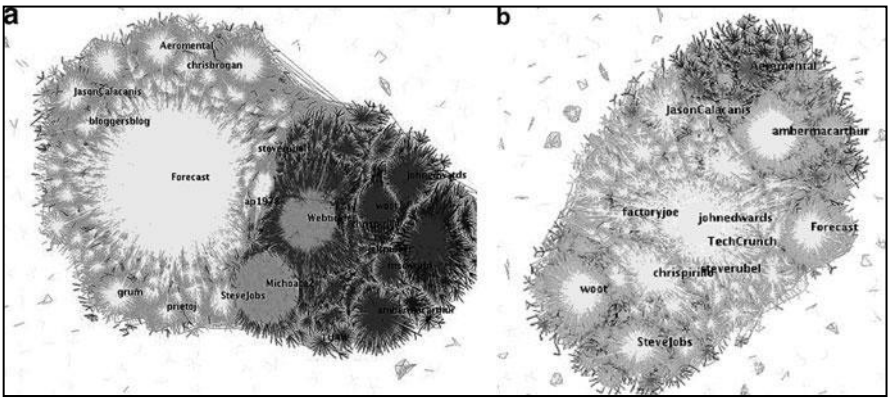


Fig. 5.4.4.1 Visual analysis of Twitter social network . (**a**) Twitter social network of at least one-way acquaintance. (**b**) Twitter social network of mutual acquaintances

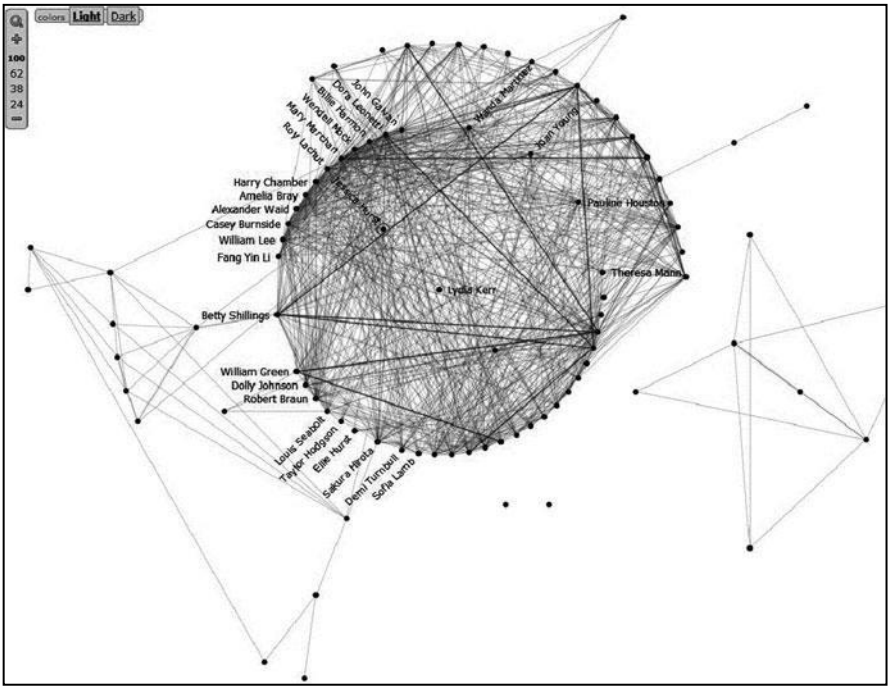


Fig. 5.4.4.2 Visualizing social relationships of Facebook with Nexus

up when mouse is moved over the nodes. However, when user relationships are complicated, the social relationships in the Nexus visualization may become difficult to be recognized. Figure 5.4.4.2 shows the screenshot of Nexus.

To improve the interlaced visualization problem caused by the node-edge representation, visual icons and user photos are further employed in the Facebook visualization. TouchGraph is a modern Facebook visualization that provides users with interactive functionalities to help access social networks. Figure 5.4.4.3 shows the TouchGraph visualization for Facebook. In Fig. 5.4.4.3, the user profiles, relations, and related photos are clearly displayed with the TouchGraph visualization interface, and thus, users can friendly interact with their social networks. However, the panorama of the community structure cannot be browse in a glance since Touch-Graph takes many clicks to obtain other attributes of the social network.

In 2010, an advanced interactive visualization interface, called IRNet, was proposed to further improve the shortcomings of Nexus and TouchGraph on visualizing Facebook communities. IRNet visualizes interpersonal relationships in social networks with focus C context techniques to present both local and global views. To present a clear visualization of social networks, IRNet employs a hyperbolic view instead of using a force-directed model used in Vizster, and thus IRNet

Fig. 5.4.4.3 Visualizing social relationships of Facebook with TouchGraph

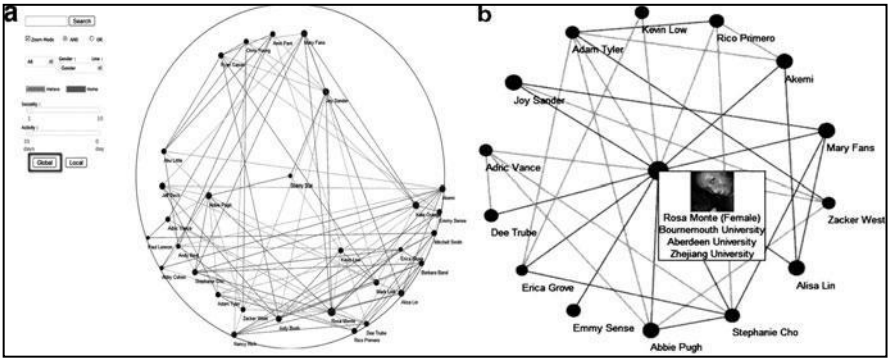


Fig. 5.4.4.4 The IRNet visualization for Facebook communities. (**a**) A hyperbolic view for exploring the global social networks. (**b**) A local view with focus zooming from a single user perspective

can effectively improve the identification of social relationships. Many interactive operations, such as focus change zooming, color highlighting, attribute filtering, searching, and photo showing, are also designed in IRNet to facilitate accessing personal social networks. In Fig. 5.54.4.a, a hyperbolic plane is displayed for browsing the global social network. Figure 5.4.4.4b shows the local view of the personal social relationships with a focus zooming view.

In addition to visualizations mentioned above, there are still versatile visualization applications developed to help people access social networks over Web 2.0 services.

5.5 Visualizing Social Networks with Matrix-Based Representations

5.5.1 Matrix or Node-Link Diagram?

Matrix and node-link diagrams have different properties making them suitable representations for different tasks and datasets.

The advantages of matrices:

Matrices provide powerful overview visualization since the time to create them is low and since they are always readable. They constitute a good representation to initiate an exploration.

Matrices do not suffer from node overlapping, if the task requires to always read the actors' labels, this representation is more appropriate.

Matrices do not suffer from link crossing each other; therefore they are a viable alternative for dense networks.

Matrices show all possible pairs of vertices, they can highlight the lack of connections and also the directedness of the connections. They are particularly appropriate for directed and dense networks.

The advantages of node-link diagrams:

These representations are familiar to a wide audience; they constitute a powerful communication tool. In contrast, matrices require training and help decoding their meaning for novice users.

For small or sparse networks, Ghoniem et al. proved that node-link diagrams were more effective than matrices.

For a similar level of details, the space used by matrices is larger than the space to display node-link diagrams. Therefore, for a compact representation, node-link diagrams are a better choice.

When the analysis requires to perform a number of path-related tasks (e.g. find the shortest path from John to Mary), node-link diagrams are more appropriate. Ghoniem et al. showed that such tasks were difficult to perform with matrices.

5.5.2 Matrix C Node-Link Diagrams

To combine advantages of both representations and to support the visual exploration of social networks, we designed MatrixExplorer (Fig. 5.5.2.1). To conceive this system, we observed and discussed with a small group of social scientists. We divided their analysis process in four main stages. For each, we describe how matrices and node-link diagrams can be combined to achieve the best of both worlds.

Initiate the exploration
Explore interactively and iteratively
Find a consensus in the data or validate an hypothesis
Present the findings

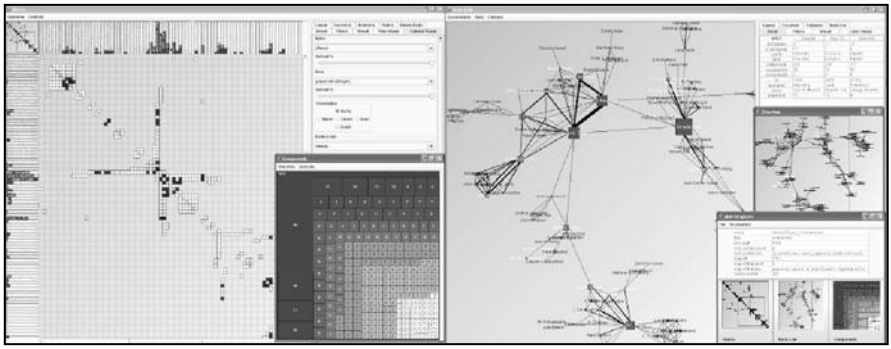


Fig. 5.5.2.1A screenshot of MatrixExplorer. This system combines matrices (*left large window*) and node-link diagrams (*right large window*). The smaller window on the *left* shows a treemap view of the macrostructure of the network (connected components). The windows in the *lower right* corner show miniature bird's eye views of the visualizations. Queries and textual data are shown in *top* windows

Initiate Exploration

The main advantage of matrices is to always provide a readable representation of a network even when it is very large. Associated to their low rendering time, these two properties make them suitable representations to initiate the exploration. To illustrate this idea, we study the following example.

Figure 5.5.2.2 shows a matrix and a node-link representation of a social network containing the email exchange of more than 450 persons during a year. Persons are nodes or rows/columns, email exchanges between two persons are represented by a link or a cell filled with black in the matrix. The node-link representation, using a traditional force-directed layout, makes it difficult to identify specific nodes or links. After studying this diagram, an analyst may retain that the network is very dense and form the hypothesis that almost everyone have been exchanging emails with each other. One may also identify a few nodes on the periphery, indicating that a few persons did not communicate with the rest.

Studying the matrix representation conveys far more information. Each black dot represents a connection between a row and a column (i.e. an email exchange between two persons); the gray background shows the lack of connection. From the matrix presented in Fig. 5.5.2.2, an analyst can quickly assess that the network represented is, in fact, not very dense. Indeed, there is a majority of gray in the matrix showing that many actors did not exchange email with each other.

Studying further the representation, the analyst can observe clusters of black dots in the matrix. These blocks are groups of persons, exchanging a lot of email with each other. Since this data shows the email communication of a large research organization, glancing at the names of the actors reveals that these groups are in fact research teams. In addition to the clusters, the analyst can observe a cross pattern: vertical and horizontal lines constituted of [illegible] with an approximate length

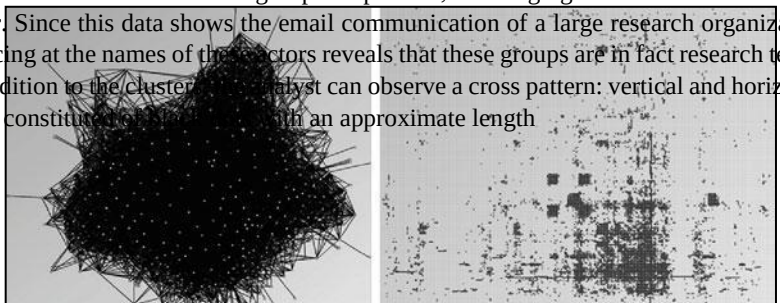


Fig. 5.5.2.2 Social network representing the email communication of more than 450 persons in a research institution over a year. The *left* image is a node-link diagram; the right image is a matrix (*black* shows connection, *grey* is no connection)

of half the matrix. Glancing at the names of the actors reveal that this patterns is associated with the administrative service, dealing with travels of the whole institutions and thus, communicating with many persons in the network.

This simple example shows that matrices have a strong potential to convey the overview of a network and initiate its exploration. We showed that, when correctly reordered, matrices highlight salient patterns of a network such as clusters or central actors. However, since they are far less familiar than node-link diagrams, some time is required to learn to decode and interpret these visual patterns.

Explore Interactively

After interviewing and observing multiple social network analysts, we realized that the exploration process itself is iterative and requires the creation of multiple visualizations. Interaction on these representations includes the configuration of the visualization (adjust its layout and its graphical attributes), the filtering as well as the grouping and possible aggregation of some of its elements.

Both the matrix and node-link representations support the analysis of the network at different levels of details. For instance, if an analyst is looking for an overview of the network to identify its main communities, the matrix is the best option. Then, when a more detailed analysis is required, to identify actors bridging two communities for example, node-link diagrams constitute a better alternative. With MatrixExplorer, we provide multiple views of the network and provide a number of tools to interactively manipulate matrix and node-link representations. Initially, the matrix and node-link representations are synchronized to combine their advantages and ease the identification of visual patterns. Selecting a row or column in the matrix highlights the corresponding node in the other representation.

In addition, visual variables such as size or color can be shared by both visualizations. Thus, it is possible to use matrices for some tasks and node-link diagrams for other. Selecting a visual pattern in the matrix and visualizing its equivalent in the node-link diagram also ease the understanding and learning of matrix represen-tations, making them accessible to less expert users.

To interactively manipulate matrix and node-link representations, we provide the following set of tools:

Interactive specification of visual attributes. The user controls the mapping data-visual encoding by entering values in a text field or selecting a value in a list.

Visual attributes of nodes, rows or columns such as label, color, transparency or size as well as attributes of links or cells such as thickness, color or texture may be associated to a data attribute.

Interactive layout and reordering. Users may directly move a node or a row/column in both representations to change its position or order.

Automatic layout and reordering techniques. Since laying out node-link diagrams or reordering large matrices by hand may be extremely tedious, we provide al-gorithms to automate layouts and reorderings. These techniques vary in their computation time and quality. As we mentioned earlier, it is difficult to identify the appropriate techniques *a priori*, thus we provide users with several.

Computer-assisted layout and reordering techniques. We developed tools to support reordering, allowing users to apply layout and reordering algorithms to specific subsets of the network.

Interactive filtering. This functionality allows filtering actors or connections according to a selection or by selecting a specific value of a data attribute from a list (such as age or sex for example). Using the principle of dynamic queries, the system provides dynamic feedback when the user modifies the parameters of the filter.

Interactive clustering. Once groups of actors are identified, we provide a simple mechanism to mark them and associate them to a visual attributed such as the color or shape of the nodes.

Overview C *Detail techniques to navigate in both representations*. To support navigation in large visual spaces, we propose two techniques providing focus + detail. We provide a brid's eye view to nagivate and a fisheye lens to magnify regions of interest for details. We also provide a Treemap to repre-sent the macrostructure of the network and providing a fast filtering mechanism to isolate each connected component of the network.

By combining both representations of a network and by interacting with them, MatrixExplorer supports an iterative and interactive exploration process. Users can create multiple views on a network, compare them and explore them at different levels of details.

Find a Consensus in the Data

Each visualization may lead to the discovery of different insights. While in many cases, these insights may be confirmed by searching those using different representations, layouts or order during the analysis. It is possible that they differ slightly or even contradict each other when observed under different conditions. This may happen when attempting to identify clusters for example. In this case, different tech-niques to reorder the matrix may lead to different cluster sets. To help analysts find a consensus and validate hypotheses, some support is needed.

MatrixExplorer allows analysts to find consensus in the data through simple interactions. For example, by associating visual variables such as colors to different cluster sets and by reordering the matrix several times, analysts can identify clusters appearing clearly in multiple orders as more valid. In addition, to mark the uncertainty of attribution of an actor to a given cluster, MatrixExplorer also provides a technique to indicate the degree of membership of the element to a given cluster. Analysts can mark elements less likely to belong to a cluster with a lighter color. Finally, we support overlapping clusters and multiple sets of clusters: elements may belong to multiple clusters at the same time.

Present Findings

While matrix representations may prove effective when exploring large networks, node-link diagrams are essential to communicate findings to a wide audience. Many node-link diagrams may be created for presenting results with different filters and possibly different aggregations. To ease this process. MatrixExplorer allows users to generate pictures while performing the exploration.

5.5.3 Hybrid Representations

Providing both matrix and node-link diagrams to the user has a number of advantages but also drawbacks. First, it requires a large amount of display space. At least two display monitors are required to comfortably use MatrixExplorer; a third one is strongly recommended to display textual and detailed views. Secondly, we observed that switching from one representation to the other may induce high cognitive load to the user and split attention is always tiresome. Indeed, a single node on the node-link diagram becomes both a row and column in the matrix and a link, visually represented by a line, becomes a cell, i.e. a rectangle, in the matrix. Switching representations between tasks require a few seconds of adjustment, disorienting momentarily users. To minimize the display space required and limit the cognitive cost when switching representations, we developed two hybrid representations: MatLink and NodeTrix. The goal of these hybrids is to augment one representation to overcome its drawbacks and enrich it with the advantages of the other one.

Augmenting Matrices

As Ghoniem et al. demonstrated in their study, matrices do not support well path-following tasks. For example, finding the shortest path between two given actors is far easier in a node-link diagram, in which users can quickly investigate the multiple paths and assess what the shortest path is. These tasks being very common in social network analysis, we proposed to create a hybrid representation to overcome the problem in matrices: MatLink (Fig. 5.5.3.1.

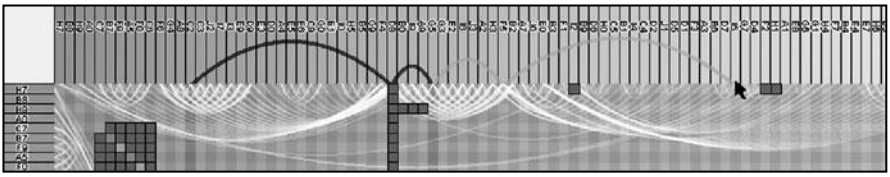


Fig. 5.5.3.1 Matlink support path-following tasks in standard matrices by adding links of their borders. *White* links are static and always shown. Links with a darker shades are interactive and follow the mouse pointer or selection

The principle of MatLink is to augment a standard matrix representation with links on its borders. These links provides a dual encoding of the connections between actors and ease path-following tasks since they use the visual representations of node-link diagrams. Two types of links are added to the representations: static links (in white on the figure) and interactive links (in a darker shade). The interactive links appear when the mouse cursor is moving over a specific row or column. When a row or column is selected, these links show a shortest path to any other row or column placed under the cursor.

Assessing the Readability of MatLink

To assess the performance of MatLink compare to traditional matrices, we performed a user study, borrowing the study design, low-level readability tasks and procedure from Ghoniem et al. In addition, we introduced specific tasks of social network analysis: find a cut point, find a clique and find communities (strongly connected groups). Our results show that MatLink significantly improve standard matrix representations. In particular, MatLink ease path-following tasks and performs even better than node-link diagrams for densely connected networks. The only task for which node-link diagrams still perform better is the identification of cut points. With MatLink, this task requires to identify specific visual patterns of the links. We believe this is possibly with more training, our participants having had only a few minutes of training with each technique for each task.

Using MatLink for Navigating in the Matrix

In addition to improving the readability of matrices, MatLink also supports navigation in large ones. Since matrices display actors in rows and columns, they require far more space than node-link diagrams to represent a network. Thus, it often happens that the neighbors of a given actor are placed outside of the current view; the reordering algorithm rarely offering strong guaranties regarding distances between connected nodes in the matrix. In standard matrices, visiting all neighbors of an actor placed in a row requires to review the whole set of columns, an extremely tedious task for large networks. In MatLink, all links connected to a given actor are displayed when this actor is selected. Thus, a direct visual feedback is provided on the number of neighbors and the curvature of the links provides an indication of their distance in the matrix as shown in Fig. 5.5.3.2.

In addition, to ease the navigation in very large matrix, we developed techniques helping users to navigate on these links and reach elements out of the view. The first technique M´elange folds the space between two far away nodes as if it was a piece of paper (Fig. 5.5.3.3). Thus, users may see side by side parts of the matrix that are far away, the intervening folded space providing context. M´elange also offers the possibility to specify a different zoom factor for each non-folded region. The two other techniques use links as navigation support. Bring-and-Go, brings

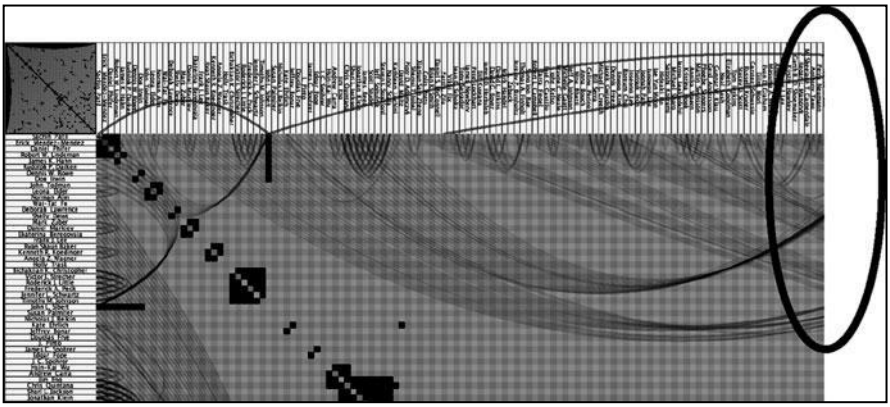


Fig. 5.5.3.2 Links in MatLink provide a visual cue that an actor on a path is outside the view. These links also provide a mean to quickly navigate to the neighbours by using Link Sliding

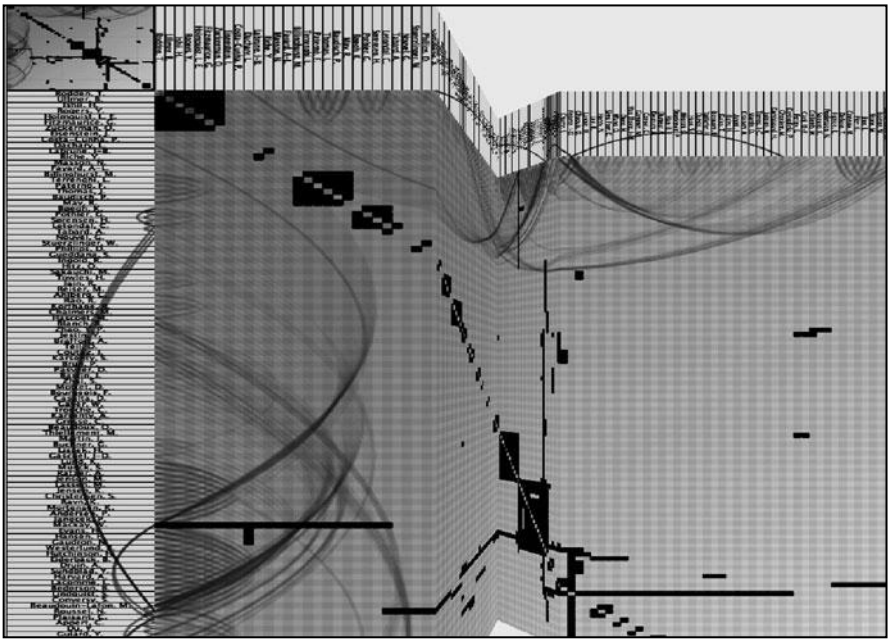


Fig. 5.5.3.3 Melange is a space folding technique designed to show far away parts of a matrix side by side while preserving the intermediate context

neighbors of an actor closer as if their links were elastic, by moving the cursor over one of the neighbor and releasing the mouse, the view and the node travel to its previous location. Link Sliding allows users to locks their cursor to a given link and travel very fast to its destination. These three techniques provide users with effective tools to navigate in large matrices with MatLink.

Merging Matrix and Node-Link Diagram

Node-link diagrams or matrices perform differently according to the types of visualized networks. For example, node-link diagrams or hybrids Treemap C links are well suited to represent tree-like networks. Conversely, for dense networks or bipartite networks, matrices are better suited, maximizing the use of space and remove any link crossing. For small-world networks, however, the choice of representation is not so clear. When visualizing such network with a node-link diagram, the dense regions (e.g. communities) suffer from link crossing and become difficult to read. However, when using a matrix representation, the visualization is very sparse and requires a lot of navigation for exploration.

To solve this problem, we created NodeTrix: a hybrid visualization merging node-link diagrams and matrices. The principle of NodeTrix is to represent the global network as a node-link diagram and the locally dense subparts as matrices (Fig. 5.5.3.4).

Interactive Exploration

To ease creation, exploration and edition of matrices in NodeTrix, we developed number of interactions based on traditional drag-and-drop of objects with the mouse cursor. The matrices may be generated automatically or created interactively. Performing a lasso selection on a group of nodes in the node-link diagrams transforms these nodes into a matrix representation. This representation on dense subparts of the networks allows identifying information such as the lack of connections between two actors. In the node-link representation, such information is difficult to read due to the high number of links and their crossings. It may also be useful to extract a set of communities from a standard matrix and place them in a NodeTrix view to better understand how they are connected.

Matrix representations have the advantage of placing actors of the network linearly (in rows and in columns), thus it becomes easy to identify the community members connected to external actors. To add or remove actors from the matrix,

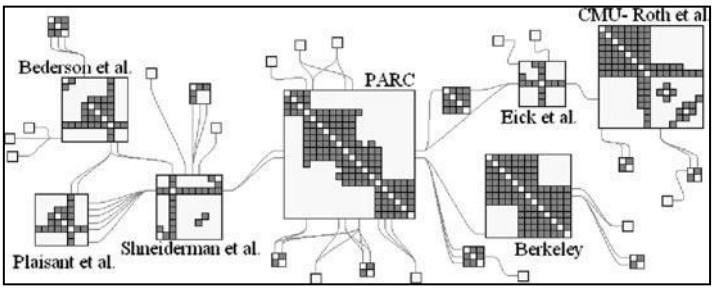


Fig. 5.5.3.4 NodeTrix representation of the collaboration network of researchers in information visualization, filtered down to a hundred actors

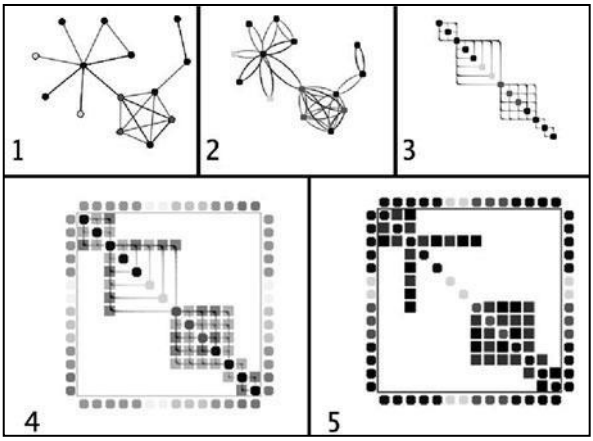


Fig. 5.5.3.5 Animation to transform a node-link diagram into a matrix

users simply select the node or row/column representing an actor and drag it in or out of the matrix. Other interactions include the possibility to merge two matrices or split them to get back to the original node-link representation. Finally, to help users understand the change of representation, we animate the transformation (see the steps of the animation in Fig. 5.5.3.5).

The main drawback of NodeTrix is the concrete representation of communities, making it impossible to place an actor in two different communities. To solve this problem, we provide users with the possibility to duplicate an actor and place it in two or more communities. Preliminary results of a user study suggest that duplications improve readability by providing non-biased view of each community. It becomes easier to identify actors acting as bridge between communities and understand the inter-community connections. Our results also show that confusion can be minimized by visually representing links between duplicates.

Presenting Findings

Because matrices can be expanded showing detailed information on actors and connections or compacted (their rows and columns headers retracted and their size minimized) showing higher-level connection patterns, NodeTrix can be used for both exploration and communication. Figure 5.5.3.4 shows an example of the compact representation of a network with more than a hundred actors. Figure 5.5.3.6 shows the same network with more details.

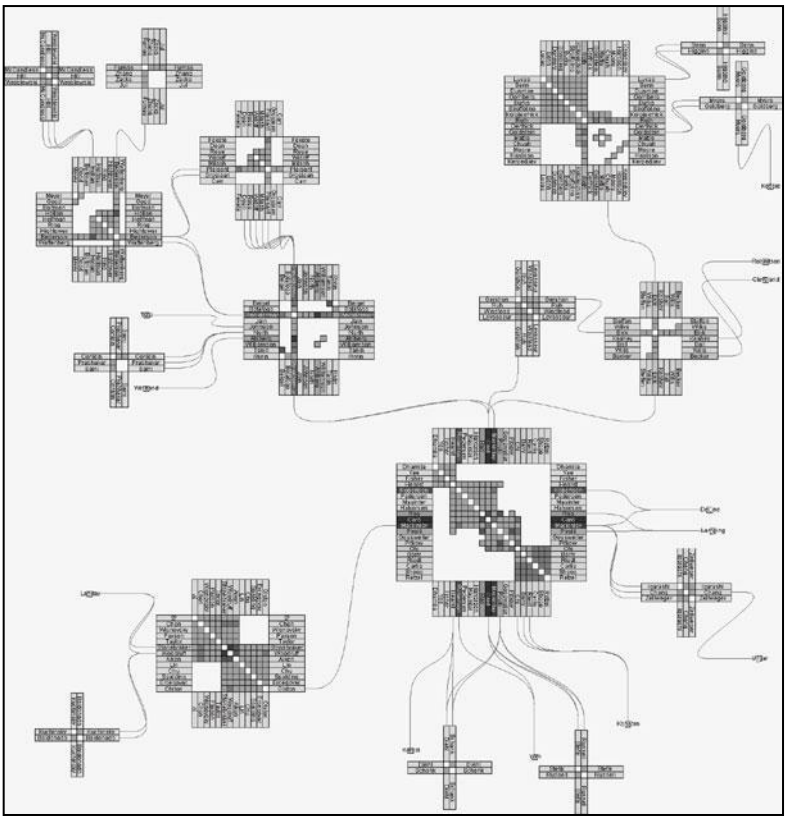


Fig. 5.5.3.6 NodeTrix showing the same collaboration network than Fig. 5.5.3.4 at a more detailed stage: including all labels of researchers and using shades of grey to indicate the number of publications

5.6 Applications of Social Network Analysis

Graph representation and usage of graph theoretic algorithms to analyze the interdependency characteristics of involved entities allows analysts to use the SNA techniques to study any type of interactions between any types of entities. Hence, SNA has been utilized as relationship analysis tool for broad fields.

The SNA has been used in organizations management to study employees' interactions and their impact on project success in order to take corrective measures to improve the performance of organizations. SNA may also be used to increase productivity, employee satisfaction and motivation, overall organization well-being, and capitalizes on the most important asset in an organization of the people.

Another major application of SNA is to discover terrorist's networks and to predict the terrorist activity. In various scientific research areas, SNA is used to understand the scope of any subject and to improve the research and innovation. Apart from human interactions, SNA is being successfully used in WWW and Information Sciences also.

5.6.1 Covert Networks

The covert networks are hidden, the actors of such network does not disclose their information to the external world. Covert groups have cellular networks structure which is different from hierarchical organizations. The terrorist and criminal networks are good examples of such networks. Generally, the information about actors, relationship among them and involvement in any external event is not easily available in terrorist networks. SNA has been successfully applied to such domains to understand covert cell operations and their organization. Intelligence analysis nor-mally focused on the identification of network vulnerabilities within the different types and forms of criminal networks. Hence, combating terrorism is another field where SNA techniques have important and successful application.

SNA has been used to understand the communication and structure of terrorist cells. SNA is applied on terrorism database for predicting node and link, discovering interesting patterns and actors involved in an event. In this context, SNA discovers who is central within organizations, which individual's removal would most effectively disrupt the network, what roles individuals are playing, and which relationships are vital to monitor.

Another vital application of SNA for terrorist database is to predict terrorism activities. Terrorist organizations have special structures on recruitment, evolution, and ideas diffusion in network. SNA tools has been used to identify these organization structures and provide critical information for terrorist detection and terrorism prediction.

Studies have shown that these types of networks can be well understood by mapping them. The Valdis Krebs has used social network analysis to map the terrorist network that attacked on 9/11. In spite of unavailability of complete and proper knowledge of all actors and connections in between them, his analysis has disclosed network which is almost near to real network.

Since terrorist groups and organizations are hidden networks which does not disclose their identity, generally the data to build and complete such networks is gathered from publicly available resources such as news papers. Sudhir Saxena has analyzed such public data to discover the terrorist cells in Jammu and Kashmir and relations among them. Now a days Web resources such as blogs, emails etc. are also used for hidden communication. Hence, various data mining and social network analysis techniques are employed to extract necessary information to detect terror.

SNA techniques applied to terrorist's network varies from basic measures to complex graph algorithms and data mining techniques. The basic measures includes betweenness and degree centrality measures, cohesion factors, closeness etc. Also, the network structure or topological measures are required in this type of analysis which may include node neighborhood search.

SNA considers terrorists networks analysis as a problem of connecting dots. Connecting multiple pairs of dots exposes the total network. Centrality is the most important and widely used measure in SNA used to identify key players in terrorist network and further the actors linked to these key players can be detected to reveals the whole network. To facilitate this, the regular day-to-day activities of the key players are monitored. The hidden actors are discovered by monitoring contact and the extend of contacts of known terrorists with other people. In order to measure the location of actors in the network, various measures such as centrality may give us

insight into the details of connector actors, mavens, leaders, bridge actors, isolates, clusters, core actor, peripheral actors etc.

Structural cohesion is also used to find connectors among group of actors. This measure is used to identify sub-groups in an organization having similar features, skills and involvement in particular event.

With the advanced graph theoretic and link analysis techniques, SNA is applied to terrorist's network to persecution of criminal activities.

5.6.2 Web Applications

From last two decades, the popularity of web has grown exponentially and web applications are becoming ubiquitous in nature. Web being a wealth of information, SNA finds a lot of applications in this domain.

Web is being used by different community for various purposes such academic improvement, knowledge sharing, interest sharing, communication and profiling, research, business etc. Hence, different techniques are required to improve and optimize the usability of web.

SNA has been used to solve various security problems in computer communica-tion system. It has been successfully employed to detect insider threats and security violation of email systems. This is done by collecting and analyzing data from net-works logs, network traffic, file shares and IM (Instant Messaging) logs. Individual's personal social network can be mapped and studied by tracing email flow. This helps to understand the person's comfortless and prestige.

Researches have been also employed to study the network of World Wide Web as a social network. It helps to understand how sociology evolves with respect to contents of the web. By analyzing the navigation and usage patterns of web, better improvements in web algorithms for gathering, searching and discovering infor-mation can be achieved. The evolution of web demands more computing power. SNA can be employed to infer the knowledge of web evolution in advance. By observing and analyzing current web usages, the evolution of web structure can be predicted. SNA has been also used for web to discover community. The web browsing or downloading patterns are analyzed to discover the individual's community and their interest in particular knowledge domain. SNA is also employed to predict the movie success and academy awards from IMDB network.

SNA models web as a graph where web pages are represented as nodes and hyperlinks as edges. Node similarity based SNA techniques are employed to classify the web based on its usage and contents in order to understand the scope of domain and density. Multi-mode network can be created by representing web resources, persons and community as nodes and web resources being used by community as edges to discover group of likely or similar persons by finding cliques with respect to each knowledge domain.

SNA is also used in search engines such as google to enhance keyword search quality. Google uses PageRank as a measure of popularity, which is obtained by simulating a random walk on network of web pages and computing prestige of web pages. Given a keyword query, matching documents are ordered by this score. Since this popularity score is precomputed independent of the query, Google can be as fast as any relevance-ranking search engine. Other type of search engines such as Alta Vista, are based on Hyperlink induced topic search does not crawl the web. Rather given a query it retrieves a subgraph of the Web whose nodes (pages) match the query. Pages citing or cited by these pages are also included.

5.6.3 Community Welfare

The SNA techniques are not limited to scientific and research areas, rather also used to improve the community welfare. SNA is used to analyze different types of relations such as communication patterns, physical contacts, sexual relationship etc. The SNA may reveal the patterns of human contact which may lead to spread of disease such as HIV in population. Considerable research has been done to analyze the spread of disease. It has been employed in epidemiology and has shown considerable results for community improvement. Another interesting application is to use SNA to examine and observe farm animal network to identify patterns of disease spread from one animal to another.

Mass surveillance is one of the modern practices undertaken by some organizations and governments to monitor the behavior of suspected people of population. This is done with the purpose of protecting people from criminals, terrorists or polit-ical subversives to maintain social control. In US, the Total Information Awareness program of the Information Awareness Office designed numerous technologies to be used to perform mass surveillance which made use of SNA tools. The community of practices involves the people to share their information to improve their own knowledge. The SNA tools have also been used to assess the communities of practices. This information can further be used to improve knowledge sharing in community.

Social Networks which are made for strengthening community resilience against disasters (natural or human-made) can reveal vulnerabilities within a network. These networks are analysed using SNA tools to study the changes that occur during disaster and further to improve disaster preparedness strategies.

5.6.4 Collaboration Networks

Collaboration network consists groups of persons working together to perform particular activity and studying human collaboration is an important topic in sociology. The widely studied collaboration network by researchers in context of SNs are science Coauthorship collaboration network and movie actor collaboration networks.

The co-authorship network is analyzed by various researchers to study dynamics in patterns of interactions between educational entities or communities. Further, these types of networks are analyzed to understand the influence of individual researchers. The structure of research collaboration in various scientific fields is disclosed by applying SNA methods to collaboration network of scientists or re-searchers which helps for strategic planning of research and development. SNA also identifies the most prominent actors in particular subject area and reveals their ego networks. The observations and results of time series and location based analysis captures the nature and characteristics of research subject over time and location. This helps to identify the scope of research discipline at particular location so that further new inventions in same can be promoted at respective region and using skills of subject experts.

The Co-authorship network analysis also helps to study and understand the interdisciplinary research which is key factor for innovation. Better way to improve

the interdisciplinary research is by identifying such current interactions and engaging involved institutions and researchers for future research.

The examples of co-authorship networks are Wikipedia article authors, network of the pacific Asia Conference on Information Systems, network of European Conference on Information Systems (ECIS) etc. SNA on these networks has been conducted to understand the research community which produces the research knowledge. Since, scope of research subject and persons subject interest may vary over time, these networks has been viewed as a dynamic social networks. So the development of such dynamic network is observed by using SNA techniques.

The required datasets for co-authorship network analysis is mostly extracted from sources including scientific journals, bibliographic records and digital libraries. The important SNA measures used for co-authorship network includes cohesion, network density and centrality. The cohesion is used to identify the sub-groups within network with respect to each research subject. The node similarity measure in this context represents extend of similar subject skills. This will help to identify group of persons to engage in particular research knowing few expertise in that area. The identified hub in sub-network can represent the key researcher in that sub-network. The scope and popularity of particular subject in its evolution is measured by computing density over time and location. The various centrality measures are also used to analyze the impact of collaboration of researchers on research in particular discipline.

Movie actor network is analyzed to study the interaction amongst themselves, to discover closely related actors. It is built based on Internet movie database (www.imdb.com) consisting of all movies and their casts. In this network nodes represents and ties represent two connected nodes acted together in some movie. These networks are large scale consisting of millions of vertices and edges, so appropriate analysis techniques are required. The advanced techniques along with traditional analysis model such as random walk are also used on movie actor database for disambiguation of name in list.

Another type of collaboration network studied in this context is knowledge collaboration network. The information about Open Source Software needs to be distributed amongst community or users because not all members have required knowledge or skills for such software usage and development. Hence, success of such software highly depends on distribution of knowledge using tools such as emails, discussion forums, web blogs etc. The extend and quality of such knowledge sharing is measured by applying SNA tools on knowledge collaboration networks.

5.6.5 Co-Citation Networks

In the area of analysis and computation, Co-citation is used as a measure of similarity between two objects. Co-citation analysis helps to understand the status and structure of scientific research. Basic two approaches of co-citation are author co-citation and document co-citation.

The Co-citation network can be viewed as a bipartite graph showing linkage between two different groups of documents. Basic application of co-citation analysis is to study the scientific communication.

There are different examples of co-citation analysis. In the field of methodological evaluation, co-citation analysis has been employed to search for invisible colleges. This reveals the research network consisting of different institutions linked to each other

informally by having indicators to each others documents/papers which can be used to get group of institutes having similar ongoing research. This may help to promote further research in respective area in those institutions.

The node similarity measure is used to find similarity between two articles or publications. In this case, the nodes represent papers and existence of link shows that two articles were cited in other articles. As we have seen in above sections, the centrality measure can also be used for co-citation analysis. Centrality represents the scope or importance of paper or respective research subject. While analyzing the relation between different disciplines, the cohesion property is used to observe how close two subjects are.