UNIT - V

What is social Networks analysis, development of the social networks analysis, Electronic Sources for Network Analysis – Electronic Discussion networks, Blogs and Online Communities, Web Based Networks. Building Semantic Web Applications with social network features.

What is social Networks 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 actors.

Network analysis takes a global view on social structures based on the belief that types and patterns of relationships emerge from individual connectivity and that the presence (or absence) of such types and patterns have substantial effects on the network and its constituents. In particular, the network structure provides opportunities and imposes constraints on the individual actors by determining the transfer or flow of resources (material or immaterial) across the network.

The focus on relationships as opposed to actors can be easily understood by an example. When trying to predict the performance of individuals in a scientific community by some measure (say, number of publications), a traditional social science approach would dictate to look at the attributes of the researchers such as the amount of grants they attract, their age, the size of the team they belong to etc. A statistical analysis would then proceed by trying to relate these attributes to the outcome variable, i.e. the number of publications.

In the same context, a network analysis study would focus on the interdependencies within the research community. For example, one would look at the patterns of relationships that scientists have and the potential benefits or constraints such relationships may impose on their work.

The patterns of relationships may not only be used to explain individual performance but also to hypothesize their impact on the network itself (network evolution). Attributes typically play a secondary role in network studies as control variables.

SNA is thus a different approach to social phenomena and therefore requires a new set of concepts and new methods for data collection and analysis. 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.

It is interesting to note that the formalization of network analysis has brought much of the same advantages that the formalization of knowledge on the Web (the SemanticWeb) is expected to bring to many application domains.

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.

Development of Social Network Analysis

The field of Social Network Analysis today is the result of the convergence of several streams of applied research in sociology, social psychology and anthropology.

Many of the concepts of network analysis have been developed independently by various researchers often through empirical studies of various social settings. **For example,** many social psychologists of the 1940s found a formal description of social groups useful in depicting communication channels in the group when trying to explain processes of group communication. Already in the mid-1950s anthropologists have found network representations useful in generalizing actual field observations, **For example** when comparing the level of reciprocity in marriage and other social exchanges across different cultures.

Some of the concepts of network analysis have come naturally from social studies. In an influential early study at the Hawthorne works in Chicago, researchers from Harvard looked at the workgroup behavior (e.g. communication, friendships, helping, controversy) at a specific part of the factory, the bank wiring room [May33].

The investigators noticed that workers themselves used specific terms to describe who is in ―our group‖. The researchers tried to understand how such terms arise by reproducing in a visual way the group structure of the organization as it emerged from the individual relationships of the factory workers (see below Figure).

Despite the various efforts, each of the early studies used a different set of concepts and different methods of representation and analysis of social networks. However, from the 1950s network analysis began to converge around the unique world view that distinguishes network analysis from other approaches to sociological research. (The term ―social network‖ has been introduced by Barnes in 1954.)

This convergence was facilitated by the adoption of a graph representation of social networks usually credited to Moreno. What Moreno called a *sociogram* was a visual representation of social networks as a set of nodes connected by directed links.

The nodes represented individuals in Moreno‘s work, 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.

The following decades have seen a tremendous increase in the capabilities of network analysis mostly through new applications. SNA gains its relevance from applications and these settings in turn provide the theories to be tested and greatly influence the development of the methods and the interpretation of the outcomes.

For example, one of the relatively new areas of network analysis is the analysis of networks in entrepreneurship, an active area of research that builds and contributes to organization and management science. The vocabulary, models and methods of network analysis also expand continuously through applications that require to handle ever more complex data sets.

The increasing variety of applications and related advances in methodology can be best observed at the yearly Sunbelt Social Networks Conference series, which started in 1980. The field of Social Network Analysis also has a journal of the same name since 1978, dedicated largely to methodological issues.

While the field of network analysis has been growing steadily from the beginning, there have been ***two developments*** in the last two decades that led to an explosion in network literature. **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.

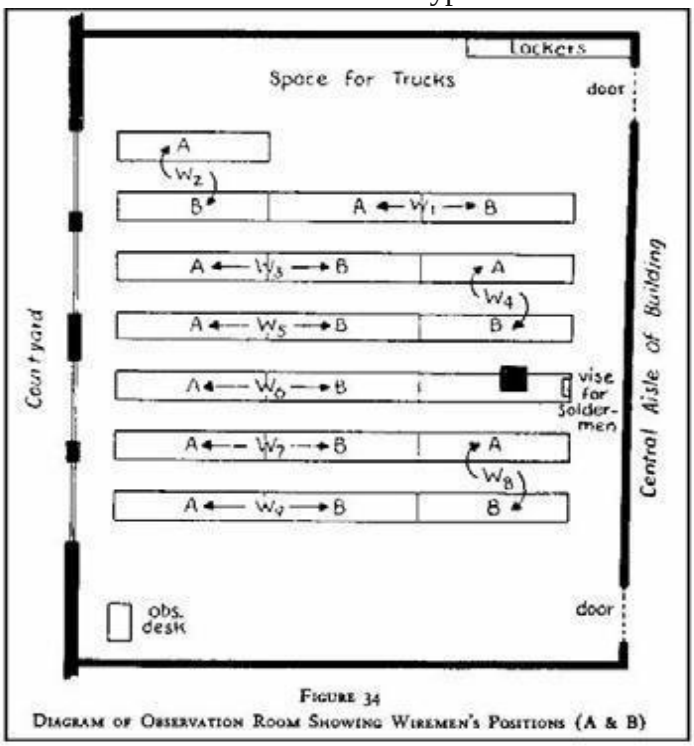

FIGURE 34
DIAGRAM OF OBSERVATION ROOM SHOWING WIREMEN'S POSITIONS (A & B)

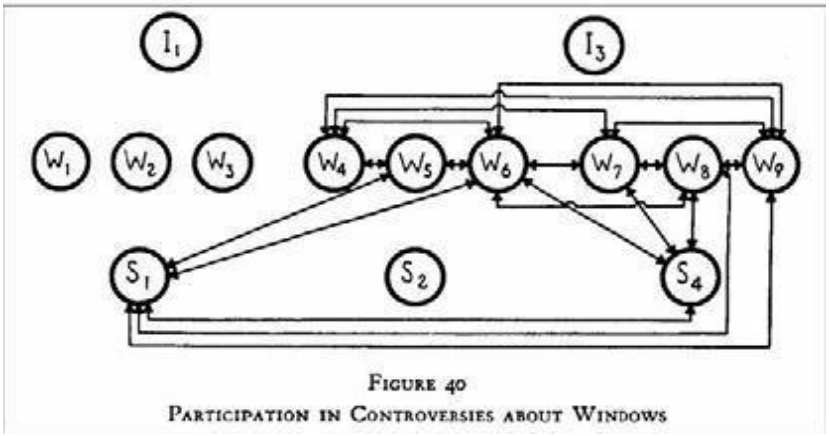

FIGURE 40
PARTICIPATION IN CONTROVERSIES ABOUT WINDOWS

Illustrations froman early social network study at the Hawthorne works ofWestern 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).

Electronic sources for network analysis

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.

Tyler, Wilkinson and Huberman analyze communication among employees of their own lab by using the corporate email archive. They recreate the actual discussion networks in the organization by drawing a tie between two individuals if they had exchanged at least a minimum number of total emails in a given period, filtering out one-way relationships. Tyler et al. find the study of the email network useful in identifying leadership roles within the organization and finding formal as well as informal communities. (Formal communities are the ones dictated by the organizational structure of the organization, while informal communities are those that develop across organizational boundaries.)

The authors verify this finding through a set of interviews where they feed back the results to the employees of the Lab.

Wu, Huberman, Adamic and Tyler use this data set to verify a formal model of information flow in social networks based on epidemic models [WHAT04]. In yet another study, Adamic and Adar revisits one of the oldest problems of network research, namely the question of *local search*: how do people find short paths in social networks based on only local information about their immediate contacts?

Their findings support earlier results that additional knowledge on contacts such as their physical location and position in the organization allows employees to conduct their search much more efficiently than using the simple strategy of always passing the message to the most connected neighbor. Despite the versatility of such data, the studies of electronic communication networks based on email data are limited by privacy concerns.

Public forums and mailing lists can be analyzed without similar concerns. Starting from the mid-nineties, Marc Smith and colleagues have published a series of papers on the visualization and analysis of USENET newsgroups, which predate Web-based discussion forums

In the work of Peter Gloor and colleagues, the source of these data for analysis is the archive of the mailing lists of a standard setting organization, the World Wide Web Consortium (W3C)

The W3C —which is also the organization responsible for the standardization of Semantic Web technologies—is unique among standardization bodies in its commitment to transparency toward the general public of the Internet and part of this commitment is the openness of the discussions within the working groups.

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 re-create the discussion networks of the working group.1 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.

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.2 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.

A slight difference is that unlike with personal emails messages to a mailing list are read by everyone on the list. Nevertheless individuals interactions can be partly recovered by looking at ***To:*** and ***CC:*** fields of email headers as well as the Reply ***Figure 3.1*** shows some of the features of blogs that have been used in various studies to establish the networks of bloggers.

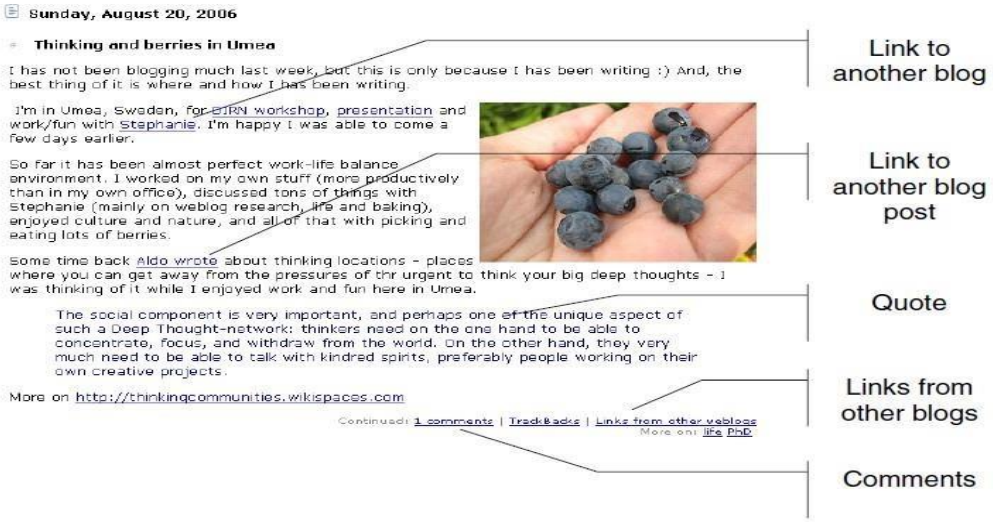

Figure 3.1. Features of blogs that can be used for social network extraction. Note also that —unlike web pages in general— blog entries are timestamped, which allows to study network dynamics, e.g. the spread of information in online communities.

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.

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. Blog analysis has suddenly shed its image as relevant only to marketers interested in understanding product choices of young demographics;

Online community spaces and social networking services such as MySpace, LiveJournal cater to socialization even more directly than blogs with features such as social networking (maintaining lists of friends, joining groups), messaging and photo sharing.

As they are typically used by a much younger demographic they offer an excellent opportunity for studying changes in youth culture. Paolillo, Mercure and Wright offer a characterization of the LiveJournal community based on the electronic data that the website exposes about the interests and social networks of its users.

Backstrom et al. also study the LiveJournal data in order to answer questions regarding the influence of certain structural properties on community formation and community growth, while also examining how changes in the membership of communities relates to (changes in) the underlying discussion topics. These studies are good examples of how directly available electronic data enables the longitudinal analysis of large communities (more than 10,000 users).

Most online social networking services (Friendster, Orkut, LinkedIn and their sakes) closely guard their data even from their own users.

A technological alternative to these centralized services is the FOAF network (see also Chapter 5). FOAF profiles are stored on the web site of the users and linked together using hyperlinks. The drawback of FOAF is that at the moment there is a lack of tools for creating and maintaining profiles as well as useful services for exploiting this network. Nevertheless, a few preliminary studies have already established that the FOAF network exhibits similar characteristics to other online social networks.

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.

There are two features of web pages that are considered as the basis of extracting social relations: links and co-occurrences (see Figure 3.2). 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.

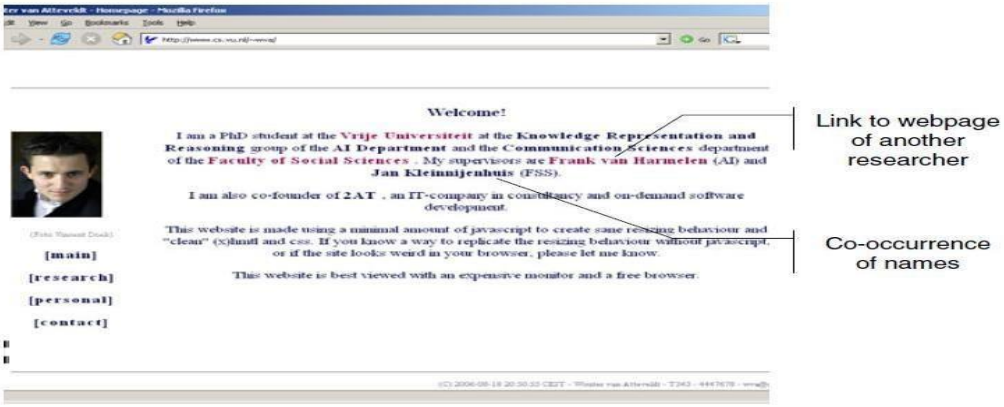

Figure 3.2. Features in web pages that can be used for social network extraction.

For this reason most social studies based on the linking structure of the web are looking at relationships at higher levels of aggregation. Unfortunately, search engines such as Google or Yahoo! typically limit the number of queries that can be issued a day.

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.

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. (Web mining is the application of text mining to the content of web pages.)

Web mining has been first tested for social network extraction from the Web in the work of Kautz el al. on the ReferralWeb project in the mid-90s [KSS97]. The goal of Kautz et al. was not to perform sociological experiments but to build a tool for automating what he calls *referral chaining*: looking for experts with a given expertise, who are close to the user of the system, i.e. experts who can be accessed through a chain of referrals.

Tie strength was calculated by dividing the number of co-occurrences with the number of pages returned for the two names individually (see Figure 3.3). 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 [Sal89]. 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.

Although Kautz makes no mention of it we can assume that he also filtered ties also based on support, i.e. the number of pages that can be found for the given individuals or combination of individuals. The reason is that the Jaccard-coefficient is a relative measure of co-occurrence and it does not take into account the absolute sizes of the sets. In case the absolute sizes are very low we can easily get spurious results:

for example, if two names only occur once on the Web and they occur on the same page, their co-efficient will be one. However, in this case the absolute sizes are too low to take this as an evidence for a tie.

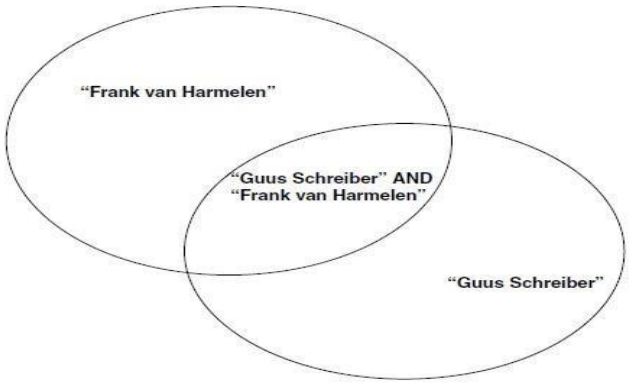

Figure 3.3. The Jaccard-coefficient is the ratio of the intersection and the union of two sets. In the case of co-occurrence analysis the two sets contain the pages where the individual names occur. The intersection is formed by the pages where both names appear.

The 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 nameswere 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.

Kautz never evaluated his system in the sense of asking whether the networks he extracted are an accurate reflection of real world networks. He notes that the system as a recommender system performed well on both the research domain and in the corporate setting, although ―the recommendations made by (any) recommender system tend to be either astonishingly accurate, or absolutely ridiculous [, which is] true for any AI-complete problem‖. However, he suggest that the system is able to keep the trust of the user provided that it is made transparent. For example, the system can show the evidence on which the recommendation is based and indicate the level of confidence in its decisions.

A disadvantage of the Jaccard-coefficient is that it penalizes ties between an individual whose name often occurs on theWeb and less popular individuals (see Figure 3.4). 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.

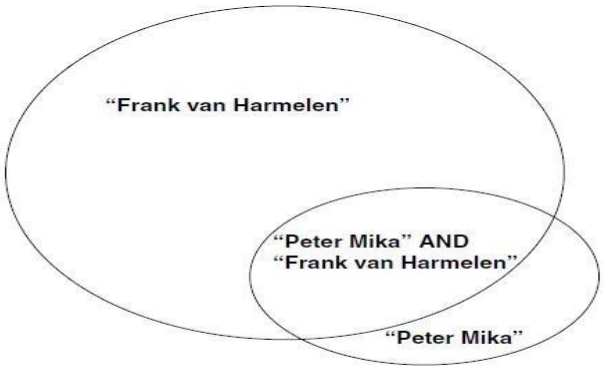

Figure 3.4. The Jaccard-coefficient does not show a correlation in cases where there is a significant difference in the sizes of the two sets such as in the case of a student and a professor.

Kautz already notes that the biggest technical challenge in social network mining is the disambiguation of person names. Person‘s names exhibit the same problems of polysemy and synonymy that we have seen in the general case of web search.

In our work in extracting information about the Semantic Web community we also add a disambiguation term our queries. We use a fixed disambiguation term (*Semantic Web OR ontology*) instead of a different disambiguation term for every 'name. This is a safe (and even desirable) limitation of the query as we are only interested in relations in the Semantic Web context.

We also experimentwith a second method based on the concept of *average precision*. When computing the weight of a directed link between two personswe 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 3.5. 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 ($1 \leq n \leq N$).

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:

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

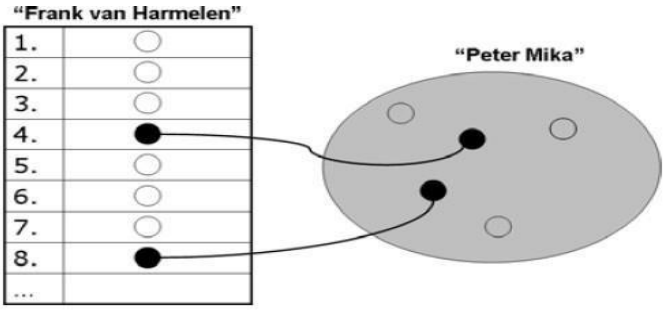

Figure 3.5. The average precision method considers also the position of the pages related to a second person in the list of results for the first person.

The average precision method is more sophisticated in that it takes into account the order in which the search engine returns document for a person:

It assumes that names of other persons that occur closer to the top of the list represent more important contacts than names that occur in pages at the bottom of the list. The method is also more scalable as it requires only downloading the list of top ranking pages once for each author.

The drawback of this method is that most search engines limit the number of pages returned to at most a thousand. In case a person and his contacts have significantly more pages than that it is likely that some of the pages for some the alters will not occur among the top ranking pages.

Lastly, we would note that one may reasonably argue against the above methods on the basis that a single link or co-occurrence is hardly evidence for any relationship. In fact, not all links are equally important nor every co-occurrence is intended.

For example, it may very well happen that two names co-occur on a web page without much meaning to it (for example, they are on the same page of the corporate phone book or appear in a list of citations).

Building Semantic Web applications with social network features

Sesame is a general database for the storing and queryingRDF data. Along with Jena, Redland and the commercial offerings of Oracle, Sesame is one of the most popular triple stores among developers, appreciated in particular for its performance. Sesame has been developed by Aduna (formerly Aidministrator), but available as open source (currently under LGPL license).

Next, we describe the **Elmo API**, a general purpose ontology API for **Sesame**. **Elmo** allows to manipulate RDF/OWL data at the level of domain concepts, with specific tools for collecting and aggregating RDF data from distributed, heterogeneous information sources. Elmo has been developed in part by the author and is available under the same conditions as Sesame, using the same website.

Lastly, we introduce a simple utility called GraphUtil which facilitates reading FOAF data into the graph object model of the Java UniversalNetwork Graph (JUNG) API. GraphUtil is open source and available as part of Flink.

The generic architecture of SemanticWeb applications

Semantic Web applications have been developed in the past years in a wide range of domains from cultural heritage to medicine, from music retrieval to e-science. Yet, almost all share a generic architecture as shown in Figure 6.1. By the definition above, all Semantic Web applications are mashups in that they build on a number of heterogeneous data sources and services under diverse ownership or control.

Before external, heterogeneous data sources can be reused, they need to be normalized syntactically as well as semantically. The first refers to transforming data into RDF syntax such as RDF/XML, while the latter means that the ontologies (schemas and instances) of the data sources need to be reconciled.

Needless to say; the first step can be skipped if the data is exposed as an RDF or OWL document, or can be queried dynamically using the SPARQL query language and protocol.

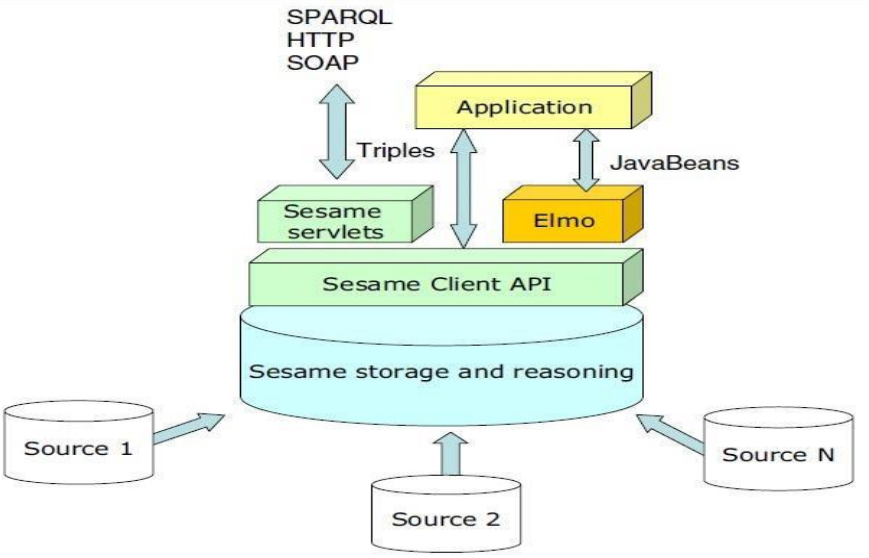

Figure 6.1. The generic design of Semantic Web applications using Sesame and Elmo. Developing with other triple stores results in similar architectures, but in general application code is not portable among triple stores due to proprietary APIs.

Most current Semantic Web applications are based on a fixed, small number of data sources selected by the application developer. In this case, the schemas of the data sources are known in advance and their mapping can be performed manually. In the future, it is expected that SemanticWeb applications will be able to discover and select new data sources and map them automatically.

Semantic Web applications persist information in ontology stores (also known as triple stores), databases specifically designed for the storage, manipulation and querying of RDF/OWL data. Ontology stores are almost always equipped with a reasoner or can be connected to an external reasoning service. Reasoning is used to infer new information based on the asserted facts or to check the existing information for consistency.

Some triple stores also allow to define customrules that are evaluated by the reasoner along with the rules prescribed by the ontology language itself. Reasoning can take place either when the data is added to a repository (forward-chaining) or at query time (backward-chaining).

Most Semantic Web applications have a web interface for querying and visualization and thus considered by all as web applications. However, this is not a requirement:

Semantic Web applications may have a rich client interface (desktop applications) or other forms of access.

Queries are given as a combination of triple patterns and return a table (a set of variable bindings) as a result. This is similar to accessing a relational database. the above mentioned SPARQL protocol, which provides limited, read-only access and is only suitable for accessing remote data sources.

In other words, what is lacking is an equivalent of the ODBC and JDBC protocols for relational databases. This means that without additional abstraction layers (such as the one provided by Elmo), all application code is specific to a particular triple store.

Further, in most cases it is desirable to access a triple store on an ontological level, i.e. at the level of classes, instances and their properties. This is also the natural level of manipulating data in object-oriented frameworks.

The Elmo library to be introduced facilitates this by providing access to the data in the triple store through Java classes that map the ontological data in the triple store. Setting and reading attributes on the instances of these classes result in adding and removing the corresponding triples in the data store.

Elmo is a set of interfaces that have been implemented for the specific case of working with data in Sesame triple stores. Sesame is one of the most popular RDF triple stores and it is to be introduced next. We note that the Elmo interfaces can be implemented for other, Java-based triples stores such as Jena.

Sesame

Sesame is a triple store implemented using Java technology. Much like a database for RDF data, Sesame allows creating repositories and specifying access privileges, storing RDF data in a repository and querying the data using any of the supported query languages. In the case of Sesame, these include Sesame‘s own SeRQL language and SPARQL.

The data in the repository can be manipulated on the level of triples:

Individual statements can be added and removed from the repository.

RDF data can be added or extracted in any of the supported RDF representations including the RDF/XML and Turtle languages.

Sesame can persistently store and retrieve the data from a variety of back-ends: data can persist in memory, on the disk or in a relational database. As most RDF repositories, Sesame is not only a data store but also integrates reasoning. Sesame has a built-in inferencer for applying the RDF(S) inference rules.

While Sesame does not support OWL semantics, it does have a rule language that allows to capture most of the semantics of OWL, including the notion of inverse-functional properties and the semantics of the *owl*: *sameAs* relationship.

An important, recently added feature of Sesame is the ability to store and retrieve context information. In distributed scenarios, it is often necessary to capture metadata about statements. For example, in the case of collecting FOAF profiles from the Web, we might want to keep track of where the information came from (the URL of the profile) and the time it was last crawled.

Context information is important even for centralized sites with user contributed content. Every triple becomes a *quad*, with the last attribute identifying the context. Contexts are identified by resources, which can be used in statements as all other resources. Contexts (named graphs) can also be directly queried using the SPARQL query language supported by this version of Sesame. The above mentioned functionalities of ***Sesame can be accessed in three ways.***

First, Sesame provides an HTML interface that can be accessed through a browser.

Second, a set of servlets exposes functionality for remote access through HTTP, SOAP and RMI.

Lastly, Sesame provides a Java client library for developers which exposes all the above mentioned functionality of a Sesame repository using method calls on a Java object called *SesameRepository*.

This object can provide access to both local Sesame servers (running in the same Java Virtual Machine as the application) or and remote servers (running in a different JVM as the application or on a remote machine.

Working with the Sesame client API is relatively straightforward. Queries, for example, can be executed by calling the *evaluateTableQuery* method of this class, passing on the query itself and the identifier of the query language.

Elmo

Elmo is a development toolkit consisting of two main components. The first one is the Elmo API, providing the above mentioned interface between a set of JavaBeans representing ontological classes and the underlying triple store containing the data that is manipulated through the JavaBeans.

The API also includes the tool for generating JavaBeans from ontologies and vice versa. The second main component consists of a set of tools for working with RDF data, including an RDF crawler and a framework of smushers.

The Elmo API

The core of the Elmo API is the *ElmoManager* a JavaBean pool implementation that is responsible for creating, loading, renaming and removing ElmoBeans. ElmoBeans are a composition of *concepts* and *behaviors*.

Concepts are Java interfaces that correspond one-to-one to a particular ontological class and provide getter and setter methods corresponding to the properties of the ontological class. (The mapping is maintained using annotations on the interface.) The inheritance hierarchy of the ontological classes is mapped directly to the inheritance hierarchy of concepts. Elmo concepts are typically generated using a code-generator.

An instance of ElmoBeans corresponds to instances of the data set. As resources in ontology may have multiple types, ElmoBeans themselves need to be composed of multiple concepts. ElmoBeans implement particular combinations of concept interfaces.

ElmoBeans can be generated runtime as the types of resources may change during the run-time of the application. ElmoBeans may also implement behaviors. Behaviors are concrete or abstract classes that can be used to give particular implementations of the methods of a concept (in case the behavior should differ from the default behavior), but can also be used to add additional functionality. Behaviours can be mixed-in to ElmoBeans the same way that additional types can be added runtime.

The separation of concepts and behaviors, and the ability to compose them at will support the distributed application development, which is the typical scenario in case of Web applications.

Lastly, Elmo helps developers to design applications that are robust against incorrect data, which is a common problem when designing for the Web.

In general, Semantic Web applications processing external data typically have few guarantees for the correctness of the input. In particular, many of the RDF documents on the Web —especially documents written by hand—, are either syntactically incorrect, semantically inconsistent or violate some of the assumptions about the usage of the vocabularies involved. Most of these problems result from human error.

Syntax can be easily checked by syntax validators such as the online RDF validation service of theW3C1112 Inconsistency can be checked by OWL DL reasoners. Elmo provides solutions for performing checks that can only be carried out programmatically.

Elmo tools

Elmo also contains a number of tools to work with RDF data. The Elmo *scutter* is a generic RDF crawler that follows ***rdfs:seeAlso*** links in RDF documents, which typically point to other relevant RDF sources on the web.

Several advanced features are provided to support this scenario:

- **Blacklisting:** Sites that produce FOAF profiles in large quantities are automatically placed on a blacklist. This is to avoid collecting large amounts of uninteresting FOAF data produced by social networking and blogging services or other dynamic sources.
- **Whitelisting:** the crawler can be limited to a domain (defined by a URL pattern).
- **Metadata:** the crawler can optionally store metadata about the collected statements. This metadata currently includes provenance (what URL was the information coming from) and timestamp (time of collection)

- **Filtering:** incoming statements can be filtered individually. This is useful to remove unnecessary information, such as statements from unknown namespaces.
- **Persistence:** when the scutter is stopped, it saves its state to the disk. This allows to continue scuttering from the point where it left off. Also, when starting the scutter it tries to load back the list of visited URLs from the repository (this requires the saving of metadata to be turned on).
- **Preloading from Google:** the scutter queue can be preloaded by searching for FOAF files using Google
- **Logging:** The Scutter uses Simple Logging Facade for Java (SLF4J) to provide a detailed logging of the crawler.

The smushers report the results (the matching instances) by calling methods on registered listeners. We provide several implementations of the listener interface, for example to write out the results in HTML, or to represent matches using the ***owl*:*sameAs*** relationship and upload such statements to a Sesame repository.

Smushers can also be used as a *wrapper*. The difference between a wrapper and a smusher is that a smusher finds equivalent instances in a single repository, while a wrapper compares instances in a source repository to instances in a target repository.

If a match is found, the results are lifted (copied) from the source repository to the target repository. This component is typically useful when importing information into a specific repository about a certain set of instances from a much larger, general store.

GraphUtil

GraphUtil is a simple utility that facilitates reading FOAF data into the graph object model of the Java Universal Network Graph (JUNG) API. GraphUtil can be configured by providing two different queries that define the nodes and edges in the RDF data.

These queries thus specify how to read a graph from the data. For FOAF data, the first query is typically one that returns the ***foaf* :*Person*** instances in the repository, while the second one returns ***foaf* :*knows*** relations between them. However, any other graph structure that can be defined through queries (views on the data) can be read into a graph.

JUNG16 is a Java library (API) that provides an object-oriented representation of different types of graphs (sparse, dense, directed, undirected, k-partite etc.) JUNG also contains implementations for the most well known graph algorithms such as Dijkstra‘s shortest path.

We extended this framework with a new type of ranker called ***PermanentNodeRanker*** which makes it possible to store and retrieve node rankings in an RDF store.

Lastly, JUNG provides a customizable visualization framework for displaying graphs.Most importantly, the framework let‘s the developer choose the kind of layout algorithm to be used and allows for defining interaction with the graph visualization (clicking nodes and edges, drag-and-drop etc.) The visualization component can be used also in applets as is the case in Flink and openacademia.