Analysis of a Real Online Social Network Using Semantic Web Frameworks

Guillaume Erétéo[1], Michel Buffa[2], Fabien Gandon[3], and Olivier Corby[3]

[1] Orange Labs, Sophia Antipolis, 06921 France
guillaume.ereteo@orange-ftgroup.com
[2] KEWI, I3S, University of Nice, France
buffa@unice.fr
[3] INRIA - Edelweiss, 2004 rt des Lucioles, BP 93, 06902 Sophia Antipolis
{fabien.gandon,olivier.corby}@sophia.inria.fr

Abstract. Social Network Analysis (SNA) provides graph algorithms to characterize the structure of social networks, strategic positions in these networks, specific sub-networks and decompositions of people and activities. Online social platforms like Facebook form huge social networks, enabling people to connect, interact and share their online activities across several social applications. We extended SNA operators using semantic web frameworks to include the semantics of these graph-based representations when analyzing such social networks and to deal with the diversity of their relations and interactions. We present here the results of this approach when it was used to analyze a real social network with 60,000 users connecting, interacting and sharing content.

Keywords: semantic web, social network analysis, graph representations.

1 Introduction

We are witnessing the deployment of a social media landscape where "expressing tools allow users to express themselves, discuss and aggregate their social life", "sharing tools allow users to publish and share content", and "networking tools allow users to search, connect and interact with each other" [4]. Social platforms, like Facebook, Orkut, Hi5, etc., are at the center of this landscape as they enable us to host and aggregate these different social applications. You can publish and share your del.icio.us bookmarks, your RSS streams or your microblog posts via the Facebook news feed, thanks to dedicated Facebook applications. This integration of various means for publishing and socializing enables us to quickly share, recommend and propagate information to our social network, trigger reactions, and finally enrich it.

More and more social solutions (e.g. Socialtext[1]) are deployed in corporate intranets to reproduce information sharing success stories from the web into organizations' intranets. However, the benefit of these platforms is often hindered when the social network becomes so large that relevant information is frequently lost in an

[1] http://www.socialtext.com/

A. Bernstein et al. (Eds.): ISWC 2009, LNCS 5823, pp. 180–195, 2009.

overwhelming flow of activity notifications. Organizing this huge amount of information is one of the major challenges of web 2.0[2] for its acceptance in corporate contexts and to achieve the full potential of Enterprise 2.0, i.e., the efficient use of Web 2.0 technologies like blogs and wikis within the Intranet [17].

Social Network Analysis (SNA) proposes graph algorithms to characterize the structure of a social network, strategic positions, specific sub-networks and networking activities [22]. Collaborative applications now capture more and more aspects of physical social networks and human interactions in a decentralized way. Such rich and diffuse data cannot be represented using only raw graphs as in classical SNA algorithms without some loss of knowledge. Semantic Web frameworks answer this problem of representing and exchanging knowledge on such social networks with a rich typed graph model (RDF[3]), a query language (SPARQL[3]) and schema definition frameworks (RDFS[3] and OWL[3]). Reporting on the experiments of SNA on online social networks, we have shown in [10] the lack of techniques for applying SNA on these rich typed representations of social networks. We proposed a framework to exploit directly the RDF representations of social networks using semantic web search engines, in order to take advantage of the rich information they hold and in particular the typed relations that form these labeled graphs [1] [11].

Today, most common analyses of social networks rely directly on graph theory or on algebraic approaches. Mika [18] showed that folksonomies can be exploited using graph theory in order to identify user groups and interest emergence. An approach by [19] uses FOAF profiles in order to identify communities of interest from the network of LiveJournal.com. [14] studied trust propagation in social networks using semantic web frameworks. [12] verified the power law of the degrees and community structures in FOAF profiles. [15] worked on merging FOAF profiles and identities used on different sites. Other researchers like [8] (see [10] and [11]) have extended tools, e.g., the SPARQL query language, in order to find paths between semantically linked resources in RDF-based graphs. These works will be a basis for us to work on graph-based and ontology-based social network representation and analysis.

Many algorithms are available for detecting social structures, roles and positions. But one aspect of our work that differentiates us from other approaches is in going beyond the application of classical algorithms to social networks described with semantic web technologies. We propose to extend these algorithms and to create new ones, in order to manage communities' life cycles, taking into account not only the graph structure but also the semantics of the ontological primitives used to label its nodes and arcs.

We first introduce our framework for exploiting the graph models underlying RDF representations of social networks. We provide formal definitions in SPARQL of SNA operators parameterized by the ontologies underlying these representations. In addition we detail SemSNA, an ontology of SNA characteristics used to annotate social networks. Finally we present how we conducted a semantic social network analysis on an anonymized dataset from Ipernity.com and the results we obtained.

[2] http://oreilly.com/web2/archive/what-is-web-20.html
[3] Semantic Web, W3C, http://www.w3.org/2001/sw/

2 Semantic Social Network Analysis

We use the RDF graphs to represent social networks, and we type those using existing ontologies together with specific domain ontologies if needed. Some social data are already readily available in a semantic format (RDF, RDFa, μformats, etc.). However, today, most of the data are still only accessible through APIs (flickr, Facebook, etc.) or by crawling web pages and need to be converted. To annotate these social network representations with SNA indices, we designed SemSNA (Fig. 1), an ontology that describes SNA notions, e.g., centrality. With this ontology, we can (1) abstract social network constructs from domain ontologies to apply our tools on existing schemas by having them extend our primitives; and we can (2) enrich the social data with new annotations (Fig. 2) such as the SNA indices that will be computed. These annotations enable us to manage more efficiently the life cycle of an analysis, by pre-calculating relevant SNA indices and updating them incrementally when the network changes over time. On top of SemSNA we propose SPARQL formal definitions of SNA operators handling the semantics of the representations. The current tests use the semantic search engine Corese [7] that supports powerful SPARQL extensions particularly well suited for SNA features such as path computations [8].

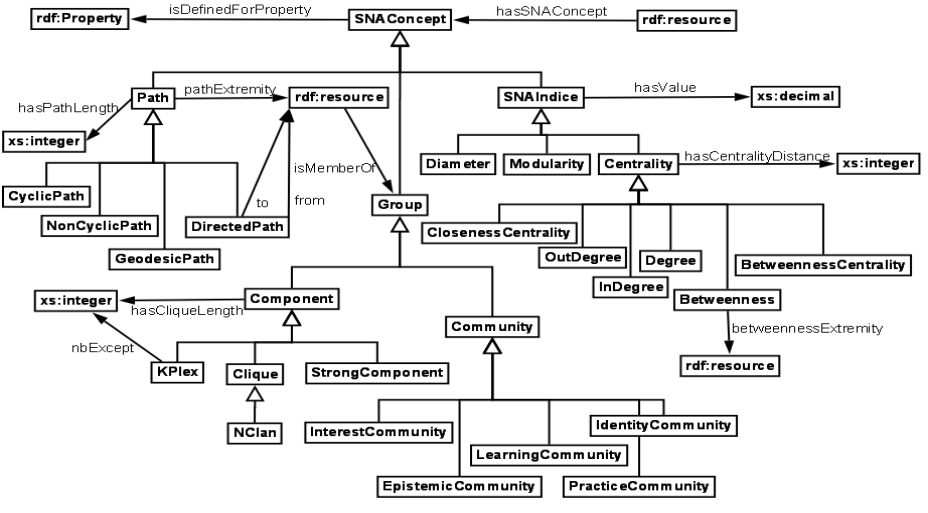

Fig. 1. Schema of SemSNA: the ontology of Social Netwtok Analysis

2.1 A New Version of SemSNA: The Ontology of Social Network Analysis

We have designed a new version of SemSNA[4], an ontology of Social Network Analysis. The first version [11] was focused on strategic position based on Freeman's definition of centrality [13]. Since then, we introduced many new primitives to annotate social data with different definitions of groups and useful indices to characterize their properties.

[4] http://ns.inria.fr/semsna/2009/06/21/voc

The main class `SNAConcept` is used as the super class for all SNA concepts. The property `isDefinedForProperty` indicates for which relationship, i.e., sub-network, an instance of the SNA concept is defined. An SNA concept is attached to a social resource with the property `hasSNAConcept`. The class `SNAIndice` describes valued concepts such as centrality, and the associated value is set with the property `hasValue`. We have slightly modified the modelling of centralities of the previous version [11]. We created a super class `Centrality` for all centralities defined by the classes `Degree`, `InDegree`, `OutDegree`, `Betweenness`, `BetweennessCentrality` and `ClosenessCentrality`. The property `hasCentralityDistance` defines the distance of the neighbourhood taken into account to measure the centrality.

We propose a set of primitives to define and annotate groups of resources linked by particular properties. The class `Group` is a super class for all classes representing an alternative concept of group of resources. The class `Component` represents a set of connected resources. The class `StrongComponent` defines a component of a directed graph where the paths connecting its resources do not contain any change of direction.

The `Diameter`, subclass of `Indice`, defines the length of the longest geodesics (shortest paths between resources) of a component. The property `maximumDistance` enables us to restrict component membership to a maximum path length between members. A clique is a complete sub graph, for a given property according to our model. An n-clique extends this definition with a maximum path length (n) between members of the clique; the class `Clique` represents this definition, and the maximum path length is set by the property `maximumDistance`. Resources in a clique can be linked by shortest paths going through non clique members. An `NClan` is a restriction of a clique that excludes this particular case. A `KPlex` relaxes the clique definition to allow connecting to k members with a path longer than the clique distance, k is determined by the property `nbExcept`. Finally the concept `Community` supports different community definitions: `InterestCommunity`, `LearningCommunity`, `GoalOrientedCommunity`, `PraticeCommunity` and `EpistemicCommunity` [16] [5]. These community classes are linked to more detailed ontologies like [23] used to represent communities of practice.

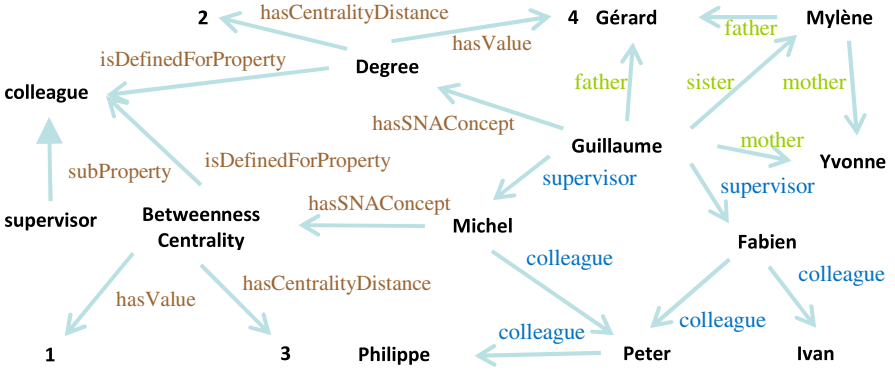

Fig. 2. A social network annotated with SemSNA indices (Degree, Betweenness)

2.2 Extract SNA Concepts with SPARQL

In [21], researchers have shown that SPARQL "is expressive enough to make all sensible transformations of networks". However, this work also shows that SPARQL is not expressive enough to meet SNA requirements for global metric querying (density, betweenness centrality, etc.) of social networks. Such global queries are mostly based on result aggregation and path computation which are missing from the standard SPARQL definition. The Corese search engine [7] provides such features with result grouping, aggregating function like `sum()` or `avg()` and path retrieving [8] [11]. We present here how to perform social network analysis combining structural and semantic characteristics of the network with queries performed with Corese, e.g., Figure 3 illustrates the calculation of a parameterized degree where only family relations are considered.

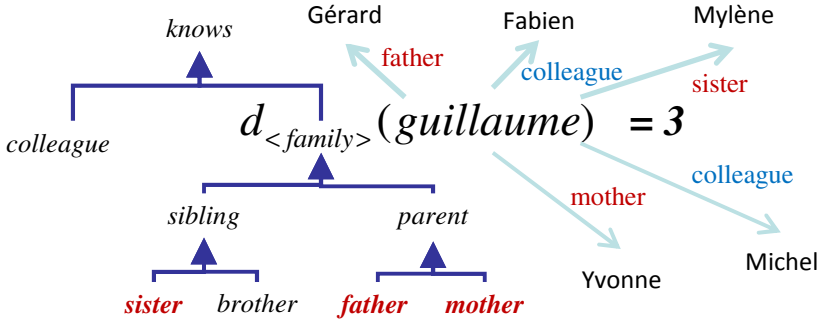

Fig. 3. A Parameterized degree that considers a hierarchy of relations

We introduced a new syntactic convention in Corese for path extraction. A regular expression is now used instead of the property variable to specify that a path is searched and to describe its characteristics. The regular expression operators are: `/` (sequence), `|` (or), `*` (0 or more), `?` (optional), `!` (not). We can bind the path with a variable specified after the regular expression. The following example retrieves a path between two resources `?x` and `?y` starting with zero or more `foaf:knows` properties and ending with the `rel:worksWith` property; the path length must be equal to or less than 3:

```
?x foaf:knows*/rel:worksWith::$path ?y
filter(pathLength($path) <= 3)
```

Path characteristics are defined by adding options before the regular expression: `'i'` to allow inverse properties, `'s'` to retrieve one shortest path, `'sa'` to retrieve all shortest paths, e.g., `?x i sa foaf:knows*/worksWith ?y`.

Path retrieval enables us to exploit the hierarchy of relation, by taking into account sub-properties at each step. Consequently we propose parameterized queries that accept as argument a regular expression of properties. Figure 3 shows a parameterized degree that only considers the hierarchy of family relationships.

Table 1. Definition of SNA notions in labeled oriented graphs and notations used

SNA indices and definition	Notation
Graph: defined as in [1] and [11]	$G=(E_G, R_G, n_G, l_G)$ where : • E_G and R_G are two disjoint finite sets respectively, of nodes and relations. • $n_G : R_G \rightarrow E_G^*$ associates to each relation a couple of entities called the arguments of the relation. If $n_G(r)=(e_1,e_2)$ we note $n_G^i(r)=e_i$ the ith argument of r. • $l_G : E_G \cup R_G \rightarrow L$ is a labelling function of entities and relations.
Number of actors: the number of actors of a given type.	$nb^{actor}_{<type>}(G)$
Number of actors: the number of actors involved in a given relation as subject or object.	$nb^{actor}_{<rel>}(G)$
Number of subject actors: the number of actors involved in a given relation as subject.	$nb^{subject}_{<rel>}(G)$
Number of object actors: the number of actors involved in a given relation as object.	$nb^{object}_{<rel>}(G)$
Number of relations: number of pairs of resources linked by a relation *rel.*	$nb^{relation}_{<rel>}(G)$
Path: a list of nodes of a graph G each linked to the next by a relation belonging to G.	$path\ p_{<rel>} =< x_0, x_1, x_2 ... x_n >$ $\wedge \forall i < n \exists r \in R_G ; n_G(r) = (x_i, x_{i+1}) \wedge l_G(r) \leq rel$
Length: the number of relations/links/arcs involved in a path	$length(p =< x_0, x_1, x_2 ... x_n >) = n$
Density: proportion of the maximum possible number of relationships. It represents the cohesion of the social network.	$Den_{<rel>}(G) = \frac{nb^{relation}_{<rel>}(G)}{nb^{actor}_{<domain(rel)>}(G) * nb^{actor}_{<range(rel)>}(G)}$
Component: a connected sub graph for a given property (and sub-properties) with no link to resources outside the component	$Comp_{<rel>}(G) = (G_1, G_2, ..., G_k)$ where G_k is a subgraph of G such that for every pair of nodes n_i, n_j of G_k there exist a path from n_i to n_j in G_k.
Degree: number of paths of properties of type *rel* (or subtype) having y at one end and with a length smaller or equal to *dist*. It highlights local popularities.	$D_{<rel,dist>}(y) = \left\lvert \left\{ x_n ; \exists path\ p_{<rel>} =< x_0, x_1, x_2 ... x_n > \right. \right.$ $\left. \left. \wedge ((x_0 = y) \vee (x_n = y)) \wedge n \leq dist \right\} \right\rvert$

Table 1. (*continued*)

In-Degree: number of paths of properties of type *rel* (or subtype) ending by *y* and with a length smaller or equal to *dist*. It highlights supported resources.	$D^{in}_{<rel,dist>}(y) = \left\lvert \left\{ x_n ; \exists path\ p_{<rel>} =< x_0, x_1, x_2 ... x_n > \right. \right.$ $\left. \left. \wedge (x_n = y) \wedge n \leq dist \right\} \right\rvert$
Out-Degree: number of paths of properties of type *rel* (or subtype) starting by *y* and with a length smaller or equal to *dist*. It highlights influencing resources.	$D^{out}_{<rel,dist>}(y) = \left\lvert \left\{ x_n ; \exists path\ p_{<rel>} =< x_0, x_1, x_2 ... x_n > \right. \right.$ $\left. \left. \wedge (x_0 = y) \wedge n \leq dist \right\} \right\rvert$
Geodesic between *from* and *to*: a geodesic is a shortest path between two resources, an optimal communication path.	$g_{<rel>}(from, to) =< from, x_1, x_2 ..., x_n, to >$ such that: $\forall p_{<rel>} =< from, y_1, y_2 ..., y_m, to >$ n•m
Diameter: the length of le longest geodesic in the network.	$Diam_{<rel>}(G)$
Number of geodesics between *from* and *to*: the number of geodesics between two resources shows the dependency of their connectivity with their intermediaries.	$nb^{g}_{<rel>}(from, to)$
Number of geodesics between *from* and *to* going through node *b*: the number of geodesics between two resources going through a given intermediary node; it shows the connectivity dependency of these resources w.r.t. *b*.	$nb^{g}_{<rel>}(b, from, to)$
Closeness centrality: the inverse sum of shortest distances to each other resource. It represents the capacity of a resource to access or to be reached.	$C^{c}_{<rel>}(k) = \left[\sum_{x \in E_G} length\left(g_{<rel>}(k, x)\right) \right]^{-1}$
Partial betweenness: the probability for *k* to be on a shortest path of properties of type *rel* (or subtype) between *x* and *y*. It represent the capacity of *k* to be an intermediary or a broker between *x* and *y* for this type (or subtype) of property.	$B_{<rel}(b, x, y) = \dfrac{nb^{g}_{<rel>}(b, x, y)}{nb^{g}_{<rel>}(x, y)}$
Betweenness centrality: the sum of the partial betweenness of a node between each other pair of nodes. It represents the capacity of a node to be an intermediary in the whole network.	$C^{b}_{<rel>}(b) = \sum_{x, y \in E_G} B_{<rel>}(b, x, y)$

Table 2. Formal definition in SPARQL of semantically parameterized SNA indices

SNA indices	SPARQL formal definition
$nb^{actor}_{<type>}(G)$	`select merge`[5] `count(?x) as ?nbactor from <G> where{ ?x rdf:type param[type]`[6] `}`
$nb^{actor}_{<rel>}(G)$	`select merge count(?x) as ?nbactors from <G> where{ {?x param[rel] ?y} UNION{?y param[rel] ?x} }`
$nb^{subject}_{<rel>}(G)$	`select merge count(?x) as ?nbsubj from <G> where{ ?x param[rel] ?y }`
$nb^{object}_{<rel>}(G)$	`select merge count(?y) as ?nbobj from <G> where{ ?x param[rel] ?y }`
$nb^{relation}_{<rel>}(G)$	`select cardinality(?p) as ?card from <G> where { { ?p rdf:type rdf:Property filter(?p ^ param[rel]) } UNION { ?p rdfs:subPropertyOf ?parent filter(?parent ^ param[rel]) } }`
$Comp_{<rel>}(G)$	`select ?x ?y from <G> where { ?x param[rel] ?y }group by any`[7]
$D_{<rel,dist>}(y)$	`select ?y count(?x) as ?degree where { {?x (param[rel])*::$path ?y filter(pathLength($path) <= param[dist])} UNION {?y param[rel]::$path ?x filter(pathLength($path) <= param[dist])} }group by ?y`
$D^{in}_{<rel,dist>}(y)$	`select ?y count(?x) as ?indegree where{ ?x (param[rel])*::$path ?y filter(pathLength($path)`[8] `<= param[dist]) }group by ?y`

[5] The *merge* keyword merges all results in one with distinct values for each variable.
[6] Corese accepts a list of parameters at the execution of a query.
[7] The keyword *any* enables to group results having the same value for any result variables
[8] The *pathLength()* function returns the length a the path given in parameter.

Table 2. (*continued*)

$D^{out}_{<rel,dist>}(y)$	`select ?x count(?y) as ?outdegree where { ?x (param[rel])*::$path ?y filter(pathLength($path) <= param[dist]) }group by ?x`
$g_{<rel>}(from,to)$	`select ?from ?to $path pathLength($path) as ?length where{ ?from sa (param[rel])*::$path ?to }group by ?from ?to`
$Diam_{rel>}(G)$	`select pathLength($path) as ?length from <G> where { ?y s (param[rel])*::$path ?to }order by desc(?length) limit 1`
$nb^{g}_{<rel>}(from,to)$	`select ?from ?to count($path) as ?count where{ ?from sa (param[rel])*::$path ?to }group by ?from ?to`
$nb^{g}_{<rel>}(b, from,to)$	`select ?from ?to ?b count($path) as ?count where{ ?from sa (param[rel])*::$path ?to graph $path{?b param[rel] ?j} filter(?from != ?b) optional { ?from param[rel]::$p ?to } filter(!bound($p)) }group by ?from ?to ?b`
$C^{c}_{<rel>}(y)$	`select distinct ?y ?to pathLength($path) as ?length (1/sum(?length)) as ?centrality where{ ?y s (param[rel])*::$path ?to }group by ?y`
$B_{<rel>}(b, from,to)$	`select ?from ?to ?b (count($path)/count($path2)) as ?betweenness where{ ?from sa (param[rel])*::$path ?to graph $path{?b param[rel] ?j} filter(?from != ?b) optional { ?from param[rel]::$p ?to } filter(!bound($p)) ?from sa (param[rel])*::$path2 ?to }group by ?from ?to ?b`
$C^{b}_{<rel>}(b)$	Non SPARQL post-processing on shortest paths.

We propose a set of queries (Table 2) to compute SNA metrics adapted to the oriented labeled graphs described in RDF (Table 1). These queries exploit the path retrieval features as well as grouping and aggregating functions. We implemented and tested all the operators that we present.

3 Linking Online Interaction Data to the Semantic Web

Ipernity.com, the social network we analyzed, offers users several options for building their social network and sharing multimedia content. Every user can share pictures, videos, music files, create a blog, a personal profile page, and comment on other's shared resources. Every resource can be tagged and shared. To build the social network, users can specify the type of relationship they have with others: friend, family, or simple contact (like a favorite you follow). Relationships are not symmetric, *Fabien* can declare a relationship with *Michel* but *Michel* can declare a different type of relationship with *Fabien* or not have him in his contact list at all; thus we have a directed labeled graph. Users have a homepage containing their profile information and pointers to the resources they share. Users can post on their profile and their contacts' profiles depending on access rights. All these resources can be tagged including the homepage. A publisher can configure the access to a resource to make it public, private or accessible only for a subset of its contacts, depending on the type of relationship (family, friend or simple contact), and can monitor who visited it. Groups can also be created with topics of discussion with three kinds of visibility, public (all users can see it and join), protected (visible to all users, invitation required to join) or private (invitation required to join and consult).

3.1 SemSNI: Extending SIOC to Model Social Networking Interactions

Several Ontologies already exist to represent online social networks [10] [11], and we use them as a basis for our model. We use FOAF[9] to represent persons' profiles and their relationships in combination with the RELATIONSHIP[10] ontology that extends the `foaf:knows` property with sub-properties. RELATIONSHIP proposes many family properties (`parentOf`, `siblingOf`, etc.) but it does not have the super property we need for all these relations: family. Thus we extend the RELATIONSHIP ontology with the property `SemSNI:family` as a super property for all these family relationships. We modelled the favorite and friend relationship respectively with the property `knowsInPassing` and `friendOf` of this ontology. The SIOC[11] [3] ontology provides the basis for defining a user (class `User`), the content he produces (class `Item`, property `has_creator`) and the actions of others users on this content (property `has_reply`). SIOC types[12] extend `sioc:Item` to specify different types of resources produced online. We use `sioc:Post`, `sioct:ImageGallery`, `sioct:Weblog` and `sioct:Comment` to model respectively posts on homepages, photo albums, blogs and comments on resources. We use SCOT[13] to model tagging.

In order to model homepages, private messages, discussion topics, and documents that do not exist in SIOC types with the required semantics, we designed SemSNI (Semantic Social Network Interactions, Fig. 4). SemSNI defines the class `UserHome`, `PrivateMessage`, `Topic` and `Document` as subclasses of `sioc:Item`

[9] http://www.foaf-project.org/
[10] http://vocab.org/relationship/
[11] http://sioc-project.org/
[12] http://rdfs.org/sioc/types
[13] http://scot-project.org/

(SemSNI:Document also extends foaf:Document). The class Visit and the properties visitedResource and hasVisitor enable us to describe the visits of a user to a resource. In order to infer new relationships between users from their interactions and the content they share, SemSNI defines the class Interaction and the property hasInteraction (domain: sioc:User, range: sioc:User). The classes representing exchanges on a content (sioct:Comment, SemSNI:Visit and SemSNI:PrivateMessage) are defined as subclasses of SemSNI:Interaction and we can infer a relation hasInteraction between the creator of the resource and its consumer. We did not type more precisely such relations, but we can extend this property in order to increase the granularity in the description of interactions between users. We use the types of access defined in AMO[14] (Public, Private, Protected) in combination with the property SemSNI:sharedThroughProperty to model the kind of sharing for resources.

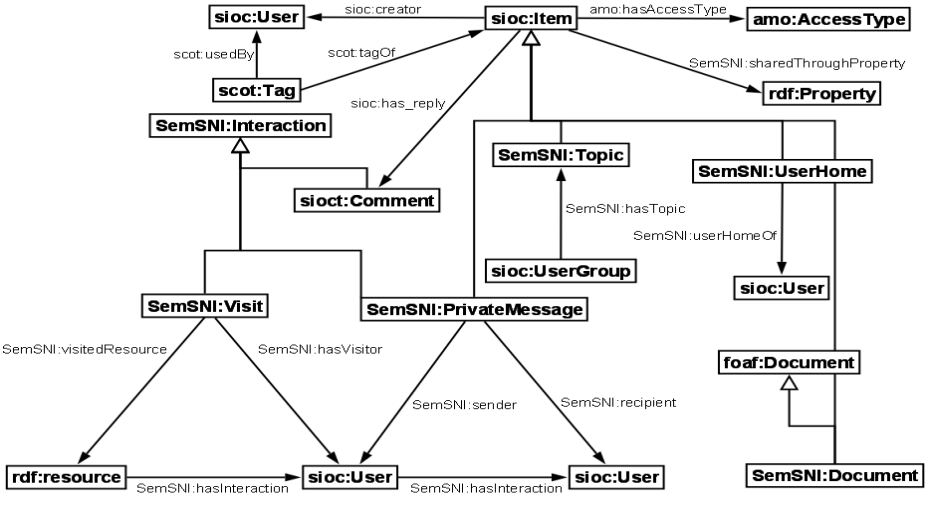

Fig. 4. Schema of SemSNI; an ontology of Social Network Interactions

3.2 Generating RDF from a Relational Database with Corese

We used another feature of Corese to transform the relational database of the Ipernity.com social network into RDF/XML. Indeed, Corese has an extension that enables us to nest SQL queries within SPARQL queries. This is done by means of the sql() function that returns a sequence of results for each variable in the SQL select clause. Corese has an extension to the standard SPARQL select clause that enables us to bind these results to a list of variables [9]. In the following example, we show how we retrieve the friend relationships from the relational database, using this sql() function and another one (genIdUrl()) that generates URIs from relational database primary keys (ids):

[14] Access Management Ontology, currently developed by the KEWI team at I3S.

```
construct { ?url_user1 rel:friendOf ?url_user2 }
select sql('jdbc:mysql://localhost:3306/mysql',
'com.mysql.jdbc.Driver', 'user', 'pwd',
'SELECT user1_id, user2_id from relations where rel = 2 limit
100000' ) as (?id1, ?id2)
fun:genIdUrl(?id1, 'http://semsni.fr/people/') as ?url_user1
fun:genIdUrl(?id2, 'http://semsni.fr/ people/') as ?url_user2
where {  }
```

Like in [21], once exported in RDF, this network can be processed to leverage the social network representation with a SPARQL query using a `construct` block. Corese can automate some transformations with inference rules [6]. As an example we can infer a property `SemSNI:hasInteraction` between two actors when one commented on the other's resource using the following rule (future improvements include using the Rule Interchange Format syntax):

```
<cos:if>
  { ?doc sioc:has_creator ?person1
    ?doc sioc:has_reply ?comment
    ?comment sioc:has_creator ?person2 }
</cos:if>
<cos:then>
  {?person1 semsni:hasInteraction ?person2  }
</cos:then>
```

4 Results

We tested our algorithms and queries on an bi-processor quadri-core Intel(R) Xeon(R) CPU X5482 3.19GHZ, 32.0Gb of RAM. We applied the defined queries on relations and interactions from Ipernity.com. We analyzed the three types of relations separately (*favorite*, *friend* and *family*) and also used polymorphic queries to analyze them as a whole using their super property: `foaf:knows`. We also analyzed the interactions produced by exchanges of private messages between users, as well as the ones produced by someone commenting someone else's documents.

We first applied quantitative metrics to get relevant information on the structure of the network and activities: the number of links and actors, the *components* and the *diameters*. 61,937 actors are involved in a total of 494,510 relationships. These relationships are decomposed in 18,771 *family* relationships between 8,047 actors, 136,311 *friend* relationships involving 17,441 actors and 339,428 *favorite* relationships

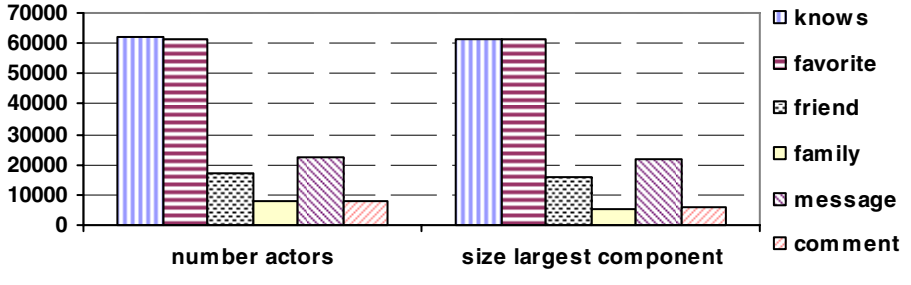

Fig. 5. Number of actors and size of the largest component of the studied networks

Table 3. Performance of queries

Indice	Relation	Query time	Nb of graph projections
$Diam_{<rel>}(G)$	Knows	1 m 41.93 s	10,000,000
	Favorite	1 m 51.37 s	10,000,000
	Friend	1 m 42.62 s	10,000,000
	Family	1 m 54.28 s	10,000,000
	Comment	35.06 s	1,000,000
	Message	1 m 50.84 s	10,000,000
$nb_{actors<rel>}(G)$	Knows	1 m 9.62 s	989,020
	Favorite	2 m 35.29 s	678,856
	Friend	11.67 s	272,622
	Family	0.68 s	37,542
	Message	17.62 s	1,448,225
	Comment	8 m 27.25 s	7,922,136
$Comp_{<rel>}(G)$	Knows	0.71 s	494,510
	Favorite	0.64 s	339,428
	Friend	0.31 s	136,311
	Family	0.03 s	18,771
	Message	1.98 s	795,949
	Comment	9.67 s	2,874,170
$D_{<rel,1>}(y)$	Knows	20.59 s	989,020
	Favorite	18.73 s	678,856
	Friend	1.31 s	272,622
	Family	0.42 s	37,542
	Message	16.03 s	1,591,898
	Comment	28.98 s	5,748,340
Shortest paths used to calculate $C_{b<rel>}(b)$	Knows	Path length <= 2: 2h 56m 34.13s	1,000,000
	Favorite	Path length <= 2: 5h 33m 18.43s	2,000,000
	Friend	Path length <= 2: 1m 12.18 s	1,000,000
	Family	Path length <= 2 : 27.23 s Path length <= 3 : 1m 10.71 s	1,000,000 1,000,000

for 61,425 actors. These first metrics show that the semantics of relations are globally respected, as *family* relations are less used than *friend* and *favorite*. 7,627 actors have interacted through 2,874,170 comments and 22,500 have communicated through 795,949 messages. All these networks are composed of a largest *component* containing most of the actors (Fig. 5) and few very small components (less than 100 actors) that show "the effectiveness of the social network at doing its job" [20], in connecting people. The interaction sub networks have a very small diameter (3 for comments and 2 for messages) due to their high density. The *family* network has a high diameter

(19), consistent with its low density. However the *friend* and *favorite* networks have a low density and a low diameter revealing the presence of highly intermediary actors.

The betweenness and degree centralities confirm this last hypothesis. The *favorite* network is highly centralized, with five actors having a betweenness centrality higher than 0, with a dramatically higher value for one actor: one who has a betweenness centrality of 1,999,858 and the other 4 have a value comprised between 2.5 and 35. This highest value is attributed to the official animator of the social network who has a *favorite* relationship[15] with most actors of the network, giving him the highest degree: 59,301. In the *friend* network 1,126 actors have a betweenness centrality going from 0 to 96,104 forming a long tail, with only 12 with a value higher than 10,000. These actors do not include the animator, showing that the *friend* network has been well adopted by users. The *family* network has 862 actors with a betweenness centrality from 0 to 162,881 with 5 values higher than 10,000. Only one actor is highly intermediary in both *friend* and *family* networks. The centralization of these three networks presents significant differences showing that the semantics of relations have an impact on the structure of the social network. The betweenness centralities of all the relations, computed using the polymorphism in SPARQL queries with the `knows` property, highlight both the importance of the animator who has again the significantly highest centrality and the adoption by users with 186 actors playing a role of intermediary. The employees of Ipernity.com have validated these interpretations of the metrics that we computed, showing the effectiveness of a social network analysis that exploits the semantic structure of relationships.

The Corese engine works in main memory and such an amount of data is memory consuming. The 494,510 relations declared between 61,937 actors use a space of 4.9 Gb. Annotations of all messages use 14.7 Gb and the representation of documents with their comments use 27.2 Gb. On the other hand working in main memory allows us to process the network very rapidly. The path computation is also time and space consuming and some queries had to be limited to a maximum number of graph projections when too many paths could be retrieved. However, in that case approximations are sufficient to obtain relevant metrics (Table 3) on a social network, i.e., for centralities [2]. Moreover, we can limit the distance of the paths we are looking for by using others metrics. For example, we limit the depth of paths to be smaller or equal to the diameter of the components when computing shortest paths.

5 Conclusion

The huge amounts of social activities and user-generated contents have to be properly organized and filtered to preserve the benefits of online collaboration. While SNA provides relevant metrics to understand the structure of online activities, Semantic Web technologies enable us to represent, to mash and to query social data from applications spanning both internet and intranet networks. The directed labeled graph structure of RDF is well suited to represent such social knowledge and such socially produced metadata. Our framework allows analyzing these rich typed representations of social networks and handling the diversity of interactions and relationships with parameterized SNA metrics. Classical SNA ignores the semantics of richly typed

[15] This animator is an employee of the company that animates the social network. He declares as *favorite* every user who just created an account and sends him welcome messages.

graphs like RDF and classical Relational Database approaches do not offer simple mechanisms for handling the semantics of type lattices. Subsumption relations are natively taken into account when querying the RDF graph in SPARQL with an engine like CORESE. Parameterized operators formally defined in SPARQL rely on this to allow us to adjust the granularity of the analysis of relations. Moreover, a new range of pre-processing can be used such as rules crawling the network to add types or relations whenever they detect a pattern, e.g., an actor frequently commenting on posts by another actor is linked to him by a relation "monitors".

New queries that compute new operators can be defined at anytime and SemSNA can be extended. Network assortativity [20] is an example of future operators that could both leverage the semantics of the schemas (e.g., similarity between two nodes) and extension mechanisms of SPARQL (e.g., counting the number of shared connections). In addition, using a schema to add the results of our queries (or rules) to the network also allows us to decompose complex processing into two or more stages and to factorize some computation among different operators, e.g., we can augment the network with in-degree calculation and betweenness calculation and then run a query on both criteria to identify nodes with an in-degree > y and a betweenness > x.

Furthermore we validated this framework on a real social network and revealed the importance of considering the diversity of relationships and their semantic links. The sub-networks we analyzed present different characteristics that highlight in particular the strategic actors and the partioning of the different activities. The approach is applied as batch processing on large RDF triple store (CORESE is a freeware handling millions of nodes but other engines with the same extensions could be used just as well). Consequently we annotate the social data with the results of these parameterized SNA metrics using SemSNA ontology to provide services based on this analysis (e.g. filter social activity notifications), to use them in the calculation of more complex indices or (in the future) to support iterative or parallel approaches in the computations. Computation is time consuming and even if CORESE runs in main memory, experiments reported in the paper show that handling a network with millions of actors is out of our reach today. We started to study different approaches for addressing that problem: (1) identifying computation techniques that are iterative, parallelizable, etc. (2) identifying approximations that can be used and under which conditions they provide good quality results (3) identifying graph characteristics (small worlds, diameters, etc.) that can help us cut the calculation space and time for the different operators. Our perspectives include also the development of a semantic based community detection algorithm and methods to manage the evolution of such ever-changing networks. More precisely, we plan to exploit these semantic based SNA metrics to structure overwhelming flows of corporate social data and to foster social interactions.

Acknowledgments. We thank the ANR for funding the ISICIL project ANR-08-CORD-011 that led to these results. We thank Peter Sander for proof-reading this article.

References

1. Baget, J.-F., Corby, O., Dieng-Kuntz, R., Faron-Zucker, C., Gandon, F., Giboin, A., Gutierrez, A., Leclère, M., Mugnier, M.-L., Thomopoulos, R.: Griwes: Generic Model and Preliminary Specifications for a Graph-Based Knowledge Representation Toolkit. In: Eklund, P., Haemmerlé, O. (eds.) ICCS 2008. LNCS (LNAI), vol. 5113, pp. 297–310. Springer, Heidelberg (2008)

2. Brandes, U., Pich, C.: Centrality estimation in large networks. Bifurcation and Chaos in Applied Sciences and Engineering 17(7), 2303–2318 (2007)
3. Breslin, J.G., Harth, A., Bojars, U., Decker, S.: Towards Semantically-Interlinked Online Communities. In: Gómez-Pérez, A., Euzenat, J. (eds.) ESWC 2005. LNCS, vol. 3532, pp. 500–514. Springer, Heidelberg (2005)
4. Cavazza, F.: http://www.fredcavazza.net/2009/04/10/social-media-landscape-redux/ (2009)
5. Conein, B.: Communautés épistémiques et réseaux cognitifs: coopération et cognition distribuée. Revue D'Economie Politique 113, 141–159 (2004)
6. Corby, O., Faron-Zucker, F.: Corese: A Corporate Semantic Web Engine. In: Workshop on Real World RDF and Semantic Web Applications (2002)
7. Corby, O., Dieng-Kuntz, R., Faron-Zucker, C.: Querying the Semantic Web with the Corese Search Engine. In: ECA/PAIS 2004 (2004)
8. Corby, O.: Web, Graphs and Semantics. In: Eklund, P., Haemmerlé, O. (eds.) ICCS 2008. LNCS (LNAI), vol. 5113, pp. 43–61. Springer, Heidelberg (2008)
9. Corby, O., Kefi-Khelif, L., Cherfi, H., Gandon, F., Khelif, K.: Querying the Semantic Web of Data using SPARQL, RDF and XML. INRIA Research Report n°6847 (2009)
10. Erétéo, G., Buffa, M., Gandon, F., Grohan, P., Leitzelman, M., Sander, P.: A State of the Art on Social Network Analysis and its Applications on a Semantic Web. In: SDoW 2008, Workshop at ISWC 2008 (2008)
11. Erétéo, G., Gandon, F., Corby, O., Buffa, M.: Semantic Social Network Analysis. In: Web Science 2009 (2009)
12. Finin, T., Ding, L., Zou, L.: Social networking on the semantic web. Learning organization journal 5(12), 418–435 (2005)
13. Freeman, L.C.: Centrality in Social Networks: Conceptual Clarification. Social Networks 1, 215–239 (1979)
14. Golbeck, J., Parsia, B., Hendler, J.: Trust network on the semantic web. In: Proceedings of cooperative information agents (2003)
15. Goldbeck, J., Rothstein, M.: Linking social Networks on the web with FOAF. In: Proceedings of the twenty-third conference on artificial intelligence, AAA 2008 (2008)
16. Henri, F., Pudelko, B.: Understanding and analyzing activity and learning in virtual communities. Journal of Computer Assisted Learning 19, 474–487 (2003)
17. McAfee, A.-P.: Enterprise 2.0: The Dawn of Emergent Collaboration, MIT Sloan Management Review, Management of Technology and Innovation (April 1, 2006)
18. Mika, P.: Ontologies are Us: a unified Model of Social Networks and Semantics. In: Gil, Y., Motta, E., Benjamins, V.R., Musen, M.A. (eds.) ISWC 2005. LNCS, vol. 3729, pp. 522–536. Springer, Heidelberg (2005)
19. Paolillo, J.C., Wright, E.: Social Network Analysis on the Semantic Web: Techniques and Challenges for Visualizing FOAF. In: Book Visualizing the semantic Web Xml-based Internet and Information (2006)
20. Newman, M.E.J.: The Structure and Function of Complex Networks. SIAM rev. 45, 167–256 (2003)
21. San Martin, M., Gutierrez, C.: Representing, Querying and Transforming Social Networks with RDF / SPARQL. In: ESWC 2009 (2009)
22. Scott, J.: Social Network Analysis, a handbook, 2nd edn. Sage, Thousand Oaks (2000)
23. Vidou, G., Dieng-Kuntz, R., El Ghali, A., Evangelou, C., Giboin, A., Tifous, A., Jacquemart, S.: Towards an Ontology for Knowledge Management in Communities of Practice (2006)