A model to support design and development of multiple-social-network applications

Post-print version of the article https://doi.org/10.1016/j.ins.2015.10.042

A Model to Support Design and Development of Multiple-Social-Network Applications

Buccafurri Francesco, Lax Gianluca, Nicolazzo Serena, Nocera Antonino

DIIES, University Mediterranea of Reggio Calabria,
Via Graziella, Località Feo di Vito, 89122 Reggio Calabria, Italy

Abstract

Online social networks have become so pervasive in people's lives that they can play a crucial role in design and development processes of applications. At moment, a gap exists w.r.t. standard networking programming to support social-network-based programming in large, according to software engineering principles of genericity and polymorphism. This drawback is made evident when applications should be built on top of multiple social networks and the user-centered vision should be kept. Indeed, heterogeneity of social networks does not allow us to produce software with suitable abstraction. In this paper, we cover the above gap by defining and implementing a model aimed at generalizing concepts, actions and relationships of existing social networks. The effectiveness of our approach is shown by two case studies.

Keywords: Multiple Social Networks, User-Centered Models, Facebook, Twitter

1. Introduction

Over the past decade, online social networks have became part of people's live. Nowadays, most people have a profile in one or more online social networks like `Facebook`, `Twitter`, `Linkedin`, `MySpace`, in which they spend a lot of time. This is recognized as an important phenomenon from a social and economic point of view, and, thus, in design and development processes of

Email addresses: `bucca@unirc.it` (Buccafurri Francesco), `lax@unirc.it` (Lax Gianluca), `s.nicolazzo@unirc.it` (Nicolazzo Serena), `a.nocera@unirc.it` (Nocera Antonino)

(Web) applications. Indeed, often applications should be based on behaviors of a community, or take advantage from these, so that modern Web applications should be social by default. In many cases, both personal information and social interactions coming from social network profiles can be part of innovative solutions. Among these, social Web applications are the most significative example, in which both people's identities and contents they produced are involved in the business process and data are mostly owned by users, strongly interlinked and inherently polymorphic [4]. The polymorphic nature of data and functionalities of applications built on top of social networks has different sources. It is related to the dynamics of social-based applications, making the meaning of concepts context and situation dependent. There is a more technical reason related to the need of delaying the binding between abstract concepts and concrete API calls, when applications operate across multiple social networks. On this aspect we focus our attention in this paper. Indeed, despite the conceptual uniformity of the social-network universe in terms of structure, basic mechanisms, main features, etc., each social network has in practice its own terms, resources, actions. This is a strong handicap for the design and implementation of applications enabling internetworking functions among multiple social networks, and, then, for the achievement of the above goal. As a matter of fact, little exists in terms of models and languages to support social-network-based programming in large, according to software engineering principles of genericity and polymorphism.

On the other hand, the power of the social-network substrate can be fully exploited only if we move from a single-social-network to a multiple-social-network perspective, still keeping the user-centered vision, so that the above issue becomes crucial. The recent literature has highlighted that the aforementioned multiple-social-network perspective opens a lot of new problems in terms of analysis [11, 43, 13, 10] but also new opportunities from the application point of view [12, 47, 32, 51, 9, 8, 65].

Consider, for example, the possibility of building the complete profile of users by merging all the information they spread out over the joined social networks. This could give a considerable added value to market analysis and job recruitment strategies, as membership overlap among social networks is often an expression of different traits of users personality (sometimes almost different identities). Again, consider the field of identity management [17, 28, 71]: To trust identity of a user or to identify fake profiles a cross check involving different social networks can be used.

From the above observations, it clearly follows that even though each

single social network is an extraordinary source of knowledge, the information power of the social-network Web can be considerable increased if we see it as a huge global social network, composed of autonomous components with strong correlation and interaction. Thus, social-network-based programming should work at this abstraction level.

In this paper, we do an important step to cover the gap highlighted above, by defining and implementing a model aimed at generalizing concepts, actions and relationships of existing social networks. We remark that our aim is not just the development of a sort of APIs working over all social networks (as done in [50]), but an approach allowing us to keep the typical semantics structure of a social network in this new multiple social network perspective. From this point of view, the user-centered vision assumes a crucial role because, besides maintaining all entities and relationships of single social networks, allows us to transparently associate with a user the information coming from all the social networks he belongs to.

This paper is organized as follows. The background necessary to understand the topic is presented in Section 2. Section 3 surveys the related work. Section 4 introduces the characteristics of the multiple-social-network scenario that we model. We give a formal definition of the graph-based conceptual model in Section 5. In Section 6, the model is implemented by defining suitable mappings among concepts and social network functionalities. To validate our approach, in Section 7, we show how our model is profitably applied to two very relevant applications in the context of social network analysis. Finally, our conclusions and possible future work are summarized in Section 8.

2. Background

This section provides the background necessary to fully understand the concepts presented in this paper. First, it discusses the main features that differentiate a social network from a regular website, then it lists the social networks we analyze to build our model and, finally, it describes the reference scenario of this paper, which involves social networks altogether.

Online social networks (OSNs) provide powerful technical features to make communication among users easy. Their backbone consists of public profiles, which collect personal information and interests, and an articulated list of friends who are other users of the system. When a user joins a social network, usually he has to fill his own profile with descriptors, such as

age, location, interests, photos and multimedia contents. Moreover, an OSN models entities and connections among them. Entities are often individuals who connected to each other by personal relationships, interactions, or information flows. The collection of friends is not simply a list of close profiles. It represents a microcosm inside the social network, where each user can interact with others. Because a friend list is visible to everyone, users can trace friend links. A new participant can find and add a new friend using the friend lists of the other users.

Profiles and friend lists are only two key features of social networks. The third feature allows users to write comments, which are prominently displayed and are visible to anyone accesses the profile of the user who generates it.

The three features (profiles, friends lists and comments) represent the basic structure of a social network. Moreover, all social networks can have a set of basic functionalities which are considered essential to qualify them as a social networking service. These functionalities are:

- the ability to set up and customize a personal profile by simple forms;
- an utility that allows members to reference other users in their posts;
- a feature allowing users to make a granular control of shared information (privacy settings);
- the ability to block an unwanted member in order to exclude him from the friend list;
- a homepage containing personal information, notes and individual picture albums.

Most of the OSNs include also many other proprietary functionalities, such as instantaneous messages, photo tagging tools, notifications, photo and video sharing, the ability to own, form or be member of a group or a community within the network, and to include new "social applications" or gadgets.

In our article, we focus on some specific social sites chosen according to their popularity and specificities. However, most of the other social networks not mentioned here, have functionalities similar to that described below.

`Twitter` is a microblogging and an online social networking service that allows users to exchange short (140-character) messages called *tweets*. We

choose it because it currently ranks as one of the leading social networks worldwide based on active users. As of the fourth quarter of 2014, `Twitter` has 288 million monthly active users [59]. The peculiarity of `Twitter` lies in its efficiency to spread out information instantaneously: it allows one person to inform millions of people in seconds, and suddenly to see responses and direct replies.

`Facebook` is the biggest social network in the world and allows people to connect to each other, upload an unlimited number of photos, post links and videos. At the end of 2014, it has more than 1.39 billion global monthly active users [60]. One of the winning factor of `Facebook` is its user centric vision, as it is the first social network to literally focus all of the attention on the user and what he wants to express and portray about himself.

`LinkedIn` is a business-oriented social networking service. With close to 347 million members worldwide in December 2014 [61], it is one of the most popular social network in terms of active users and the most trustworthy source of professional content according to UK business professionals. Available in more than 200 countries, its website focuses on business connections and industry contacts for employers and working professionals, allowing companies to present themselves and users to find job listings, to build their career and to stay in touch with their connections in their area of expertise.

`Flickr` is an image hosting and video hosting website and an online community and a multimedia networks. The main aim of Flickr is to allow users to upload their photos as well as organize and share them with other users. We choose it because it is one of the most used online photo management and sharing application in the world. It provides users with a massive online photo storage allotment of a whole terabyte.

`Google+` is a social networking and identity service owned and operated by Google. Its 359 millions of active users make it a leading social networks worldwide [62]. It is intended to integrate all Google services (Gmail, Google Maps, search, Google Calendar, etc.) into one cohesive network, incorporating everything that searchers use at Google into a comprehensive social and content dashboard.

`LiveJournal` is a community publishing platform and a social networking service where users can keep a blog, journal or diary. It has more than 50 million journals on different topics like politics, entertainment, fashion, literature and design. We choose `LiveJournal` because it has been well studied by social network analyzers in the past.

`Advogato` is an online community and social networking site dedicated to

the development of free software. It represents a resource for free software developers because it provides a research testbed for group trust metrics and other social networking technologies. This site is mentioned for its early adoption of the FOAF ontology [6] as an alternative method for showing user information.

`about.me` is a personal web hosting service that ties together users of other social-networking sites. It also includes analytics that let users track things like how many people viewed their `about.me` page and which other social-networking profiles they viewed from there. We choose it because of its main feature of linking together in the user profile relevant external sites and multiple social networking websites such as `Facebook`, `Flickr`, `Google+`, `LinkedIn`, `Twitter`, `Tumblr` and `YouTube`.

Social networks altogether form a more complex scenario in which users interaction assumes a relevant importance. This interaction is enabled by the presence of users who have multiple profiles in different social networks and adopt particular kind of edges, called `me` edges, to link them [**?**]. Figure 1 shows a graphical representation of a possible scenario involving three social networks, namely `Twitter`, `Facebook`, and `LinkedIn`. In this example, we have that nodes from 1 to 6 are `Twitter` accounts, nodes from 7 to 12 are `Facebook` accounts and, finally, nodes from 13 to 18 are `LinkedIn` accounts. As for edges, they represent friendship relationships among users. However, while edges among `Facebook` and `LinkedIn` actors are bidirectional, those among `Twitter` users are directed, according to the typology of relationship allowed by the social network. Finally, edges $(14, 3)$, $(12, 15)$ and $(7, 6)$ represent `me` edges and connect accounts of the same user on different OSN.

3. Related work

Traditionally, social networks have been mainly represented through two kinds of mathematical tools: matrices and graphs. These structures allow the modeling of information about tie patterns among social actors.

The approaches that adopt matrices representation to model social networks [36, 31, 63] belong to the second group. Specifically, the approach of [36] incorporates social influence processes in the specification of a weight matrix W, whereas the approach of [63] uses a tensor to model the interaction between resources and users.

Examples of the second group are: Kronecker graphs model [38]; the class of model networks presented in [46], which are generalizations of the much-

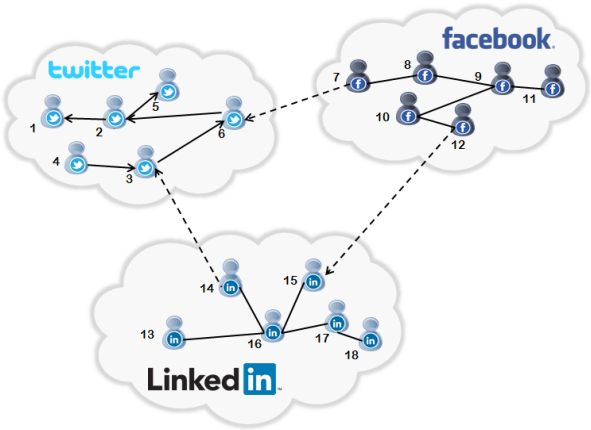

Figure 1: An example of a multiple social network scenario.

studied random graph of Erdös and Rényi [19] to model social networks; the multiplicative attribute graph model presented in [35], which considers information about properties of the nodes of the network; the approach of [55], which tries to model users interests through an *interest map* obtained by partitioning an individual's social graph and others, such as [16, 68, 48, 67, 2], which model their application scenarios with graphs.

A minor trend is to formalize social networks through a three phase model [54, 25]. This model was developed in [69] to identify critical social networks activities.

The hypergraph theory [34] allows a hyperedge to connect an arbitrary number of vertices instead of two in regular graphs. For instance, Ghoshal et al. [22] introduce a random hypergraph model to describe the ternary relationship among one user, one resource and one tag, thus making the model more flexible in the representation of many peculiar properties of folksonomies. In [72], the authors propose an hypergraph model to illustrate the emergence of some statistical properties in a folksonomy such as: degree distribution, clustering coefficients and average distance between nodes.

Other approaches adopt suitable models with the purpose of creating global user profiles by means of deep analyses of their behavior accessing multiple social networks. Often, the application scenarios of these approaches are those of ontologies and folksonomies.

Still in the field of ontologies, [18, 29, 44, 33] present ontology-based applications concerning social aspects. In particular, [18] deals with team

building, whereas TAGMAS (TAG MAnagement System) [29] relies on an ontology to uniformly describing tags and tagging actions in several distinct folksonomies. Furthermore, the author of [44] formulates an abstract model of semantic-social networks, in the form of a tripartite graph of persons, concepts and instances. Hence, incorporating actors in this model, he extends the traditional concept of ontologies (composed by concepts and instances). Because the referring scenario of [44] is that of folksonomies, the adopted model represents only one action (i.e., *tagging*). It is defined as a ternary association between user, concept and object. More in detail, the set of shared object and the set of keywords defined by users themselves are extracted from social networks. These collections are, then, used to obtain the emergence of a community-based ontologies.

Other interesting approaches in this context are [63, 20, 27, 26]. For instance, [63] proposes a cross-tagging approach, whose goal is to create a system capable of improving the set of tags of a social site with the tags used in the other sites. The enriched set of tags allows two main applications: the automatic annotation of resources, which were not originally labelled and the enrichment of user profiles. As a side effect, the more refined profiles introduce an higher precision in the computation of user similarities.

Some studies focus on the problem of integrating data of different social sites [58], [49] and [24]. The basic idea of [58] is to collect data from different folksonomies and, then, to create groups of tags by adopting suitable clustering algorithms. These groups are mapped onto the concepts of an ontology. In [49], the authors propose an approach that gathers data about user activities on social sites. Suitable ontologies are used both to analyze these data and model user interests. In [24] the authors provide an unsupervised method for integrating multiple data views of a user in a single social network to produce a unified graph. They carry out this task using a form of rank aggregation applied to nearest neighbor sets.

Other recent works take advantage of social network modeling like [21, 1]. In [21], the authors try to find a method to model and simulate interactive behavior in OSNs. Their aim is to predict what users post or reply with regard to sentiments and to analyze how information spreads across the network. Finally, a comparative analysis of four mining tools based on social network graphs is presented in [1]. These tools can easily model the structure of social networks.

The system proposed in [50] has some relation with our own, as a set of meta-APIs working on social networks is provided. However, while the

	User	Resource	Tag	Comments	Friendship	Multi Networks	Like	Post
[38]	√	–	–	–	√	–	–	–
[72]	√	√	√	–	√	–	–	–
[63]	√	√	√	√	–	√	–	√
[58]	√	√	√	–	–	√	–	√
This paper	√	√	√	√	√	√	√	√

Table 1: A comparative analysis of our model.

approach of [50] aims at creating external web services allowing the retrieval of information coming from different social networks via a unique and comprehensive platform, our approach, instead, provides a framework focused on social network users (thus, with a user-centered vision) allowing the aggregation of the information concerning a user coming from all the social networks he belongs to. The technical counterpart of the above feature is that our approach, besides APIs used also in [50], relies on further technologies, such as FOAF, XFN and HTML parsing. Moreover, new (user-centered) entities are considered, like `me` edges, which link two accounts belonging to a single physical person. As a consequence, differently from [50], the aim of our paper is to support the development of models and languages for user-centered social-network-based programming in large, according to software engineering principles of genericity and polymorphism.

Our approach has some common aspects with these proposals. However, none of them consider the possibility of integrating information coming from different and heterogeneous social networks. This additional feature makes our model strongly different from the approaches presented above, because the uniform representation of all the peculiarities of different social networks is a non-trivial task and needs ad-hoc solutions to be pursued. Indeed, the solution adopted by our approach is to build a suitable middleware on top of social networks to support internetworking applications. To accomplish this task, we create a complete XML model, mapping all social network actions with abstract concepts. Therefore, the approach followed in this paper is practical, as we solve the trade-off between complexity/expressiveness of the conceptual model and implementation issues in favor of the latter. The resulting benefits from the implementation perspective appears considerable.

To show the significance of our contribution, we pose this question. To

reach the goal of this paper, could one of the models proposed in the literature be used, even though it is defined for different purposes? Obviously, if the answer is yes the significance of our proposal is compromised. Among the models proposed in the literature analyzed to answer the above question, we choose [38] and [72], which are the most recent and representative models using graphs and hypergraphs, respectively. In addition, we consider [63] and [58] because they are the most suitable models based on matrices and ontologies, respectively.

We compare our model with those mentioned above with respect to the following functionalities:

1. modeling of user accounts and/or profile information (e.g., screen name, user picture, etc.);
2. representing Web assets such as photos, external links, videos, etc.;
3. supporting the labeling of social entities (user profiles, resources, and so on) with tags;
4. modeling the comments expressed by users on social entities;
5. storing information about user relationships (e.g., friendship);
6. distinguishing information coming from different social networks (multiple social networks);
7. supporting the *I Like it*, by means of which users express that they like, enjoy or support a given content;
8. storing information about *who posts what*.

We report the results of this comparison in Table 1. As it emerges from the analysis of this table, our model is the only one supporting all the considered features. It is worth remarking that this comparison is obviously not aimed at stating some form of superiority of our approach w.r.t. the considered ones. Indeed, all these models are defined for different purposes, so that a comparison in this sense is meaningless.

Finally, we observe that our paper extends the preliminary work presented in [7]. The additional contributions of this paper can be summarized as follows. We provide a deep requirement analysis of the reference scenario to identify the functionalities that the model has to support. Moreover, we enrich the model and made available the XML schema that allows the serialization of our model. Furthermore, we describe two case studies for which the application of our model is important allowing a more precise handling of social network information. Finally, we enrich and update the

related work discussion showing that no existing proposal can be directly applied to solve our problem.

4. Design specification

In Section 2, we focused on the general services provided by the most popular OSNs. Their study is one of the targets of our paper. As it can be recognized by analyzing the technical details described in the sequel of the section, there is strong heterogeneity in the representation of concepts among different social networks. For instance, contacts are represented by *friends* in Facebook and the relationship is symmetric, while they are represented by *followers* and *followings* in Twitter and the corresponding relationship is not symmetric. Again, the concept of appreciation becomes *+1* in Google+ and *endorsement* in about.me. Importantly, similar concepts can mapped to each other but they have in general different features. Thus, an integration step is necessary for our purpose. In this section, we prepare this integration step by grouping the main technical entities into a number of categories to which the formal model presented in the next section maps. In particular, we aim at modeling the following entities.

4.1. Profile

Social network sites are built around user profiles, a form of individual (or group) homepage, which provides a description of each registered user. Profiles are constructed by filling out forms on the site.

As for `Twitter`, at the moment of registration, a user can create his profile typing his name, username, password and email address in the registration form. People often use their real name without the spaces as username. After typing in the CAPTCHA words from the image, a user can create his account. When a user is logged in, he can upload a profile picture and start following other people. Moreover he can complete his profile adding a short biography, a position (the place where he lives) and a link to his website or to one of his account on other social networks.

Similarly, if a user wants to sign up for `Facebook` he has to enter in the suitable page his full name, a valid email address, a password, his gender and birthday. As a second step, the user can complete his timeline, which is his personal profile. Timeline includes everything from uploading a profile picture and cover photo to outlining user employment history, determining

his relationship status, declaring web link toward the profile of the same user in other social networks.

To join `LinkedIn`, a user has to fill out his biography with information, such as past and present employment, education, skills and web links. It is also possible to add a user profile photo. The "Headline" and the "Professional Summary" section of the profile are useful to highlight user experiences.

Whereas if a user wants to register a `Flickr` account, first he has to create a Yahoo! Mail account. Then the user has to choose his `Flickr` screen name that will be the user name for the site. After the first access, a user can fill out his profile adding some personal information as gender, birthday, web link, occupation, hometown, relationship status, interests.

As for `Google+`, a user with a gmail account can automatically sign in, otherwise, he has to create it filling in his name, preferred username, password, birthday, gender, mobile Phone and other email address. Once the user logs in, he can choose his username, upload a picture and complete further information about himself, such as where he went to school or where he works.

A user can sign in `LiveJournal` platform via one of his account (`Facebook`, `Google+`, `Twitter`, etc.). After this step, he has to complete his profile adding a profile picture, username, gender, birthday, education, web links, interests, biography and position.

`Advogato` allows users to create a profile page and a blog. At the moment of the registration, the user has to provide a valid email address, a username and a password. The rest of information (such as name, surname, notes) is optional, but is useful in order to be certified by other `Advogato` users.

Finally, `about.me` is characterized by its one-page user profiles, each with a large background image and short biography. At the moment of registration a user has to fill the suitable form with his username, email, password for the site and at a second step short biography, a short description, a profile image and a background image.

4.2. Links to external social networks

An important feature provided by all the social networks considered in this paper is the possibility for a user to add in his profile a link toward one of his accounts in another social site or external website. This feature is typically enabled during the creation of the user profile. It is of particular interest in this paper because it encodes the basic information allowing the

possibility of seeing different social sites as members of a Multiple-Social-Network environment.

4.3. Friendship

After creating a profile, participants are asked to invite their friends to the site or to look at others' profiles and add those people to their list of friends.

In `Twitter`, a user can follow another user, becoming his *follower*. Only if this user follows him back the relationship is bidirectional.

Differently from `Twitter`, `Facebook` requires approval for two people to be linked as friends. When someone links another as a friend, the recipient receives a message asking for confirmation. Indeed, `Facebook` friendship is bidirectional, hence, once a user accepts a friendship request of another user they become mutual friends.

`LinkedIn` allows registered users to maintain a list of contact details of people with whom they have some level of work relationship, called *connections*. When a user establishes a relationship with another user, he declares a sort of mutual friendship and, from this moment on, he will see all the updates of this new connection in his homepage.

`Flickr` follows a strategy similar to that of `Twitter` about contacts. These social ties may or may not be reciprocated. Only users who include each other as contact have a reciprocal relationship. Once a user adds another user as contact, he can further distinguish the relationship with this user by labeling him as friend and/ or member of his family or just keeping him only as a following contact.

Among the functionalities of `Google+` there is the possibility of adding users in a friend list. In particular, a user can assign his contacts to one or more "circles" (such as friends, colleagues or acquaintances), which is a way of categorizing and organizing people.

In `LiveJournal` two users can list each other as friends mutually, or one of them can *follow* the other without reciprocation, like in `Facebook`.

Finally, neither `Advogato` nor `about.me` provide the user with the possibility to create any list of friends.

4.4. Resources

A Social network resource is a Web asset such as a status update, a photo, a web link or a video created and loaded by a user in his profile.

`Twitter` resources can be shared only inside a *tweet*. They can be photos, videos, web links or comments. Once a user generates a tweet, it is publicly posted on his `Twitter` profile. Moreover, each *tweet* can be associated with one or more hashtags describing it, and/or some references to other user profiles. The stream of tweets of a user is called *timeline*.

Concerning `Facebook`, a user can publish one or more resources in a *post* in his timeline. The post can contain photos, videos, comments, resources coming from other social platforms, hashtags and references to other users. A user can also add in his "diary" (his `Facebook` profile) notes, photos or videos without publishing them in a post. Photos and videos can be uploaded in specific albums.

As for `LinkedIn`, a user can add a resource like a new item or a new file in his profile. He can also embed a comment, a photo, a web link or a video in a new status update. Also skills representing specific technical expertise can be seen as a typology of resource, which are posted by users to describe their ability. This way, his *connections* can like it, comment it and share it on their "wall".

In `Flickr`, when a user upload an image or a video he can optionally add a text description to the resource. The images a `Flickr` photographer uploads are stored into his sequential "photostream", which is the basis of a `Flickr` account.

As far as `Google+` is concerned, a user can share messages, links, photos or videos with everyone or only with those within designated circles, but he can also create his own photo albums and add his photos or videos inside them.

In `LiveJournal`, examples of resources are images, videos and audio files. Resources can be uploaded by users in their blog post.

In `Advogato`, once a user account has been certified by other trusted users, he will be able to post to the news flow, create projects, or syndicate his blog to the `Advogato` recentlog from his existing blogging site. The only form of resource available in `Advogato` is the *article*.

Finally, in `about.me` there is not an explicit concept of resource. Indeed comments and appreciations are allowed directly on the user profile and users cannot upload any photos or videos, except for the profile and background picture.

4.5. Actions on resources

So far, we stated that in addition to the content that members add when they create their own profiles, social network sites typically provide the possibility to share resources. After a resource is published by a user, several actions can be performed on this resource: other users can appreciate it, or re-share it, or it can be associated with a user through a mention on his profile.

Hereafter, we list the main possible actions a user can do on a resource according to the different social networks analyzed in this paper.

Once a user write a *tweet* in `Twitter`, it will appear on the homepage of all his followers, who can *reply* to it, make it one of their *favourites* or *retweet* it (that is, forwarding it again on their own timeline). A *tweet* can contain also a user *mention*. It can be done using the symbol `@` followed by the referenced username. To categorize *tweets* by keyword, people use the hashtag symbol `#` before a relevant keyword or phrase (no spaces) in their *tweets*. Hashtags are indexed to make it easier to find conversations about that topic.

As for `Facebook`, when a resource is posted, a user can comment it and/or give a positive feedback through the *like* button. Users can *like* all types of resources, such as: status updates, comments, photos, other user profiles, links posted by their friends and adverts by clicking the *like* button at the bottom of the content. This makes the content appear in their friends' "News Feeds". Moreover, users who are interested in or agree to a post, can share it again in their timeline (*re-post*). This allows a very fast propagation of posts inside `Facebook`.

The concept of referencing users in status updates has been introduced as an attempt to imitate `Twitter`. This means putting the name of a user, a brand, an event or a group in a post in such a way that it is linked to the wall of the `Facebook` page being tagged. Thus, the post appears in news feeds for that page, as well as those of selected friends. This is done by using the `@` symbol followed by a person's name.

The same symbol allows users to tag people in photos or videos taken of them. This functionality is a peculiar feature of `Facebook`. Whereas using the `#` symbol followed by a tag word in a status allows a user to create a hashtag. As explained before, this metadata tag allows grouping of similarly tagged messages and support the search for messages referring to a specific topic represented by that hashtag. Every hashtag on `Facebook` has its own

unique URL and this allows to search for specific topics from the Facebook search bar.

Clicking on *like* option on LinkedIn presents some differences w.r.t. the Facebook *like* function. Indeed, on LinkedIn, when users click on the *like* link underneath the various updates, this immediately forwards that particular update out to all of the user first level *connections*. The *share* option, instead, allows users to either redistribute the article (and partially modify it) as an update to their *connections*, post it to a group (or multiple groups), or forward it in a private message. The *comment* link allows users to comment on someone's update.

Similarly to what happens in Twitter, also in LinkedIn while a user publishes a resource he can mention one of his *connections* with the @ symbol. He can also use a keyword as hashtag using the # symbol.

Furthermore, companies can post information about themselves, list jobs and search for potential candidates. Finally, LinkedIn allows users to endorse each other's skills.

As for Flickr, by clicking on a photostream image, it is possible to open it in the interactive *photopage*, thus allowing users to comment it and to embed it on external websites. Moreover, images can be added to a user favourite list or to user galleries.

Users may label their uploaded images with titles and descriptions, and images may be tagged either by the uploader or by other users, if the uploader permits it.

The main Google+ page consists of a "stream" of updates, conversations and shared content. A user can make comments underneath content shared by other users, and he can appreciate contents clicking "+1" on it. A user can also re-share contents within his circles. Google+ provides the referencing functionality in its posts. A user can mention another user using the + or @ signs.

Moreover, a user can insert some hashtag in his comments similarly to what happens for Twitter. The main differences with the hashtag of Twitter is that here the system automatically adds hashtags (recognizable by different colors), too.

As for LiveJournal, users can interact with resources in different ways. For instance, a user can leave a comment on a post of another user or share it in his blog. He can also add to "Memories" a post. The Memories feature on LiveJournal allows the organization of favorite resources with a keyword-based archive system. Thanks to this functionality, a user can also add tags,

or descriptive keywords, to his own resources.

As far as `about.me` is concerned, an interesting characteristic is the possibility to make *compliments*, which is a form of *like* made on user pages. There are different kinds of compliments a user can do and `about.me` let users choose among them, whether it was a professional compliment or a more personal one. The company provides also a service called *collections*: a user can organize various profiles into public or private *collections*. In order to bring *collections* and *compliments* together, `about.me` introduced the Dashboard, which included "PeopleFeed". This let users see activities on their page, including views, who visited and complimented their page. *Replies*, instead, let users respond to activity on their page.

All the features of the OSNs described in this section are mapped by our model, which is formalized in the next section.

5. The conceptual model

In the previous section, we have identified eleven technical entities, of which three concepts and eight relationships. Now, we want to formalize the so described environment into an abstract multiple-social-network model. To do this, we adopt a direct graph $G = \langle N, E \rangle$, in which nodes represent the concepts and edges encode the relationships. Therefore, the set of nodes is partitioned into three disjoint sets P, R, and B, which correspond to the set of social profiles, the set of resources, and the set of bundles (which are resource containers), respectively. Further, the set of edges is partitioned into eight disjoint sets F, M, Pu, S, T, Re, L, and Co, each corresponding to one of the eight relationships identified in Section 4.

Let us start with the description of nodes. An element of P models the *profile* of a user on a social network and consists in the tuple ⟨`url, socialNetwork, screen-name, [personalInformation], [picture]`⟩. In this tuple `url` is the Web address that identifies and localizes the profile, and `socialNetwork` is the commercial name of the social network which the profile belongs to, `screen-name` is the name chosen by the user who registered the profile to appear in the home-page of the profile or when posting a resource, and, finally, `personalInformation` and `picture` are the information and the image which the user inserted as related to the profile. The two last elements of the tuple are optional (i.e., they can be null).

The set R models *resources* of the Web or created by users. A resource is represented by a tuple ⟨`url, type, [description], [date]`⟩,

where `url` is the Web address to access the resource, `type` indicates the type of the resource content, and finally, `description` and `date`, which are optional, represent the string, inserted by the who published the resource, describing the resource itself and the publishing date, respectively. For example, the most viewed video on YouTube is a resource represented as ⟨`'https://www.youtube.com/watch?v=9bZkp7q19f0', 'video/mp4', 'PSY - GANGNAM STYLE', '07/15/2012'`⟩.

Our model includes the *bundle* set B. Indeed, commonly users do not handle a single resource, but most of the actions they do (e.g., publishing or sharing) involve more resources simultaneously. For example, a user can publish more photos or videos, can include a comment, and so on. In our model, we include all resources handled simultaneously by a user in a bundle. A bundle is represented by a tuple ⟨`uri, [description], [date]`⟩, where `uri` is the identifier of the bundle, `description`, which is optional, is the string chosen by the user to be shown with those resources and, finally, `date` represents the publishing date. As we will see next, we represent the inclusion of a resource into a bundle by means of *containing* edges.

Relationships among profiles, resources and bundles are represented by direct edges of a graph. As already stated earlier, the set E of edges is partitioned into eight disjoint sets, named F, M, Pu, S, T, Re, L, and Co, each corresponding to one of the eight relationships identified in Section 4.

The *follow* edge set $F \subseteq E = \{p_s, p_t \mid p_s, p_t \in P\}$ models the fact that in the (source) profile p_s, it has been declared a certain type of relationship towards the (target) profile p_t. This kind of edge models different relationships. For example, on `Facebook` or `Flickr`, it models friendships, on `LinkedIn`, job contacts, and, on `Twitter`, *followers*. Observe that, typically, this kind of relationship occurs between users of the same social network, because it is presumable that a social network does not have interest in promoting links to profiles of another (competitor) social network.

The *me* edge set $M \subseteq E = \{p_s, p_t \mid p_s, p_t \in P\}$ denotes that the user with profile p_s has declared in this profile to have a second profile p_t. This edge allows a user to provide a link to its profile (typically) on a different social network or (sometimes) on the same social network (as a sort of alias).

The *publishing* edge set $Pu \subseteq E = \{p_s, b_t \mid p_s \in P, b_t \in B\}$ indicates that the user with profile p_s has published in this profile a bundle b_t. This edge models one of the typical actions a user does when enriches his/her profile by publishing resources.

The *shared* edge set $S \subseteq E = \{b_s, b_t \mid b_s, b_t \in B\}$ specifies that the bundle

b_s (published by a user) is derived from an already published bundle b_t. This type of edge is used when a user shares an existing bundle. Indeed, this action is represented by two edges: a publishing edge (as described before) and a shared edge from the new bundle to the existing one.

The *tagging* edge set $T \subseteq E = \{p_s, br_t, w \mid p_s \in P, br_t \in B \cup R$ and w is a word$\}$, denotes that the user with profile p_s assigned the word w to describe a bundle or a resource br. By means of the tag mechanism, users contribute to resource labelling, which is necessary to carry out several actions on resources, such as searching or classification.

The *referencing* edge set $Re \subseteq E = \{b_s, p_t \mid b_s \in B, p_t \in P\}$ models the fact that a bundle b_s includes a reference to the profile p_t. For example, this occurs when a *tweet* includes a user account name.

The *like* edge set $L \subseteq E = \{p_s, pbr_t \mid p_s \in P, pbr_t \in B \cup R \cup P\}$ describes the information that a user with the profile p_s expressed a preference/appreciation for a bundle, a resource or another user profile pbr_t.

The *containing* edge set $Co \subseteq E = \{b_s, r_t \mid b_s \in B, r_t \in R\}$ indicates that a bundle b_s contains the resource r_t. For example, when a user publishes a photo p and includes a comment c, this action is modeled by creating a bundle b with a description c, a resource p, and finally, a *containing* edge from b to p.

The model defined above is able to represent data coming from multiple social networks. As a consequence, this model may appears more complex than those typically adopted in Social Network Analysis. However, it is worth noting that some measures used in this field can be still calculated in a easy way by suitably *pruning* the graph $G = \langle N, E \rangle$ underlying the model. For example, the friendship degree distribution of social network users has to be computed on the graph $G' = \langle N', E' \rangle$, in which $N' = P$ and $E' = F$, that is, the subgraph obtained from G by maintaining only Profile nodes and Follow edges. Analogously, other topological features, such as clustering coefficient and assortative coefficient, are computed also on the same subgraph G'.

After defining the conceptual model, we will show how to practically map real-life data from social networks to each component of the model, in such a way to build a data structure that can be used at application level (as we will show in Section 7).

6. Building the model

Information necessary to build the model can be extracted from social networks via four technologies: (i) APIs provided by the social network; (ii) FOAF datasets; (iii) XFN microformat; and (iv) HTML parsing.

As for the first technology, social network APIs are a platform available for developers which allow the access to social-networks data so as to create applications on top of them. Usually, there are different kinds of APIs each providing specific services. Among them, the most commons are the REST API, the Search API and the Streaming API. Specifically, the REST APIs allow operations such as insert, update or deletion to be performed. The Search APIs, instead, are useful to query the database and, finally, the Streaming APIs are conceived for applications that need to receive real-time updates (such as, new posts or feeds).

The second possible strategy to extract information from social network relies on FOAF datasets. The FOAF project focuses on the creation of a machine-readable ontology describing friendship relationships among users. FOAF data sources allow the representation of a whole social network without the need of a centralized database. As a matter of fact, by relying on this technology, it is possible to represent the information concerning a user account, along with the corresponding contacts and activities, through an RDF graph serialized as an XML document, according to the W3C RDF/XML syntax.

The third option makes use of XFN microformat. It allows for the representation of the kind of relationship existing between two user accounts. This is obtained by empowering the set of values that the `rel` attribute of the HTML tag `<a>` (which represents a link) can assume. In our case, we focus on the value "me" (`rel='me'`) which indicates that the corresponding link represents a `me` edge.

The last data extraction strategy leverages on HMTL parsing. Processing HTML to obtain social data is the most intricate procedure. Parsing requires much time because it needs to analyze all context information from the page source code. It is a low-level way of dealing with social data. Because the code written depends on the HTML page structure, it is not stable (due to the frequent graphical changes). For this reason, this strategy needs continue maintenances. However, it remains a valid alternative when other more practical solutions (like APIs, for instance) are not available.

Now we will show some significant examples on how the information rep-

```
1  {
2     "id": "1587099156",
3     "first_name": "Serena",
4     "gender": "female",
5     "last_name": "Nicolazzo",
6     "locale": "en_GB",
7     "name": "Serena Nicolazzo",
8     "username": "serena.nicolazzo"
9  }
```

Figure 2: An example of the output of the `Facebook` Graph API

resented by our model are extracted from social networks.

As for the user profile P described in Section 5, we recall that it consists in a tuple ⟨url, socialNetwork, screen-name, personalInformation, picture⟩. For example, to extract the information to build the profile of a `Facebook` user, we use the `Graph APIs`, accessible through the url `http://graph.facebook.com/{user-id}` or `http://graph.facebook.com/{screen-name}`. The output of this API is a JSON file (see, for instance, Figure 2).

We can extract from this JSON the user id, his username (which correspond to our notion of screen-name) and his personal information like: first name, last name, gender, locale (chosen language). The field *url* available in our social profile object can be obtained as `http://www.facebook.com/{screen-name}`, whereas the field *picture* can be obtained by another call to the `Graph APIs`, specifically by accessing the url `http://graph.facebook.com/{user-id}/picture`.

Many social networks are equipped also with FOAF datasets. As an example, we show how *follow* edges can be obtained for the social networks `LiveJournal` and `Advogato`. The FOAF datasets for both social networks are reachable through the specific URLs `http://{screen-name}.livejournal.com/data/foaf` (for `LiveJournal`) and `http://www.advogato.org/person/{screen-name}/foaf.rdf` (for `Advogato`). An example of an XML serialization of a FOAF document is shown in Figure 3. In this document, the information needed to build an edge of the set *follow* can be extracted from lines 11 to 29. Specifically, the element `<foaf:Person>` indicates the beginning of the portion of the document where information about a user, his contacts and, often, his activities are reported. The information about each contact is encoded as a `<foaf:Person>` nested inside a tag `<foaf:knows>`.

Concerning the information about *me* edges, it can often be extracted through the XFN microformat. Some examples of social networks adopting this standard to represent *me* edges are `about.me`, `Advogato`, `Facebook`,

```
1  <?xml version='1.0'?>
2  <rdf:RDF
3    xml:lang="en"
4    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
5    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
6    xmlns:foaf="http://xmlns.com/foaf/0.1/"
7    xmlns:ya="http://blogs.yandex.ru/schema/foaf/"
8    xmlns:lj="http://www.livejournal.org/rss/lj/1.0/"
9    xmlns:geo="http://www.w3.org/2003/01/geo/wgs84_pos#"
10   xmlns:dc="http://purl.org/dc/elements/1.1/">
11   <foaf:Person>
12     <foaf:nick>antoninonocera</foaf:nick>
13     <foaf:name>real_name</foaf:name>
14     <foaf:openid rdf:resource="http://antoninonocera.livejournal.com/"/>
15     <foaf:weblog rdf:resource="http://antoninonocera.livejournal.com/"/>
16     <foaf:homepage rdf:resource="*****" dc:title=""/>
17     ...
18     <foaf:knows>
19       <foaf:Person>
20         <foaf:nick>contact1</foaf:nick>
21         <foaf:member_name>contact_real_name</foaf:member_name>
22         <foaf:tagLine></foaf:tagLine>
23         <foaf:image>http://l-userpic.livejournal.com/****/****</foaf:image>
24         <rdfs:seeAlso rdf:resource="http://contact1.livejournal.com/data/foaf"/>
25         <foaf:weblog rdf:resource="http://contact1.livejournal.com/"/>
26       </foaf:Person>
27     </foaf:knows>
28       ...
29   </foaf:Person>
30 </rdf:RDF>
```

Figure 3: An XML-serialized FOAF document

```
1 <a class="OLa url Xvc" href="http://www.youtube.com/channel/UCIUcwh3yFufPSnCbyrUyTBQ"
2 rel="me" target="_blank" title="UCIUcwh3yFufPSnCbyrUyTBQ">YouTube Channel of Antonino Nocera</a>
```

Figure 4: An example of a *me* edge using XFN.

`Flickr`, `Google+`, and `Twitter`. Figure 4 shows the code representing a *me* edge in the social network `Google+`. The code at line 2 represents the explicit declaration that the corresponding link encodes a relationship of type *me*.

Another interesting example regards the extraction of the information needed to build a *publishing* edge. Consider the social network `LinkedIn`. It provides a search API, called Job Lookup API, to obtain information about jobs that can be accessed at the address `http://api.linkedin.com/v1/jobs/{job_id}:(id,company,posting-date)`. The XML output produced by the call to this API is reported in Figure 5. In this case, when a company proposes a new job position (publishing), we model this event by adding two objects: (i) a bundle and (ii) a *publishing* edge between the profile of the company and the bundle just created (see Section 5). As for the bundle, the field `uri` is mapped to the element `<id>` (line 3), whereas

```
1  <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2  <job>
3    <id>1511685</id>
4    <expiration-timestamp>1304030488000</expiration-timestamp>
5    <company>
6      <id>229433</id>
7      <name>Cloudera</name>
8    </company>
9    <position>
10     <title>Technical Writer</title>
11     <location>
12       <name>San Francisco Bay Area</name>
13       <country>
14        <code>us</code>
15       </country>
16     </location>
17   </position>
18   <location-description>San Francisco or Palo Alto,CA</location-description>
19   <job-poster>
20     <id>hQ4ruu3J2q</id>
21     <first-name>Paul</first-name>
22     <last-name>Battaglia</last-name>
23     <headline>Technical Writer at Cloudera</headline>
24   </job-poster>
25 </job>
```

Figure 5: An example of the output of the `LinkedIn` Job Lookup API.

the field `description` is obtained from the elements `<company>` (lines 5-8), `<position>` (lines 9-17), and `<localition-description>` (line 18). The *publishing* edge is associated with the user profile whose identifier is specified by the element `<job-poster>` (lines 19-24) and has the new created bundle as target.

Now, consider the case of the publishing of a new tweet containing a resource and referencing another user in the `Twitter` social network. Our model represents this action by adding the following objects: (i) a bundle, (ii) a resource, (iii) a *publishing* edge, (iv) a *containing* edge, and (v) a *referencing* edge. In this case, all information required by our model is extracted from `Twitter` by means of the method *GET statuses/user_timeline* of the `Twitter` APIs. Figure 6 shows an example of the output of this API. Specifically, line 6 is the tweet identifier and is mapped to the field `uri` of the bundle. The bundle field `description` is obtained by suitably parsing lines 24-30. The bundle is linked to the publisher user by means of a *publishing* edge. As mentioned above, a new resource is added and associated with the bundle by means of a *containing* edge. Information needed to create the resource is extracted from lines 11-23. In particular, the field `url` is obtained from line 15, the field `type` from line 20, and, finally, the field `description` is extracted from lines 19, 20 and 21. The tweet in the example references another user and this action is modeled by adding a *referencing* from the

```
1 {
2  "coordinates": null,
3  "favorited": false,
4  "truncated": false,
5  "created_at": "Wed Aug 29 17:12:58 +0000 2012",
6  "id_str": "240859602684612608",
7  "entities": {
8    "hashtags": [ ],
9    "user_mentions":[{"indices": [3, 10],"id_str": "<user_id>","screen_name":
10   "<user_screen_name>","name": "<user_real_name>", "id": <user_id>}]
11   "media": [{
12     "id": 266031293949698048,
13     "id_str": "266031293949698048",
14     "indices": [17, 37],
15     "media_url": "http://pbs.twimg.com/media/A7EiDWcCYAAZT1D.jpg",
16     "media_url_https": "https://pbs.twimg.com/media/A7EiDWcCYAAZT1D.jpg",
17     "url": "http://t.co/bAJE6Vom",
18     "display_url": "pic.twitter.com/bAJE6Vom",
19     "expanded_url":"http://twitter.com/BarackObama/status/266031293945503744/photo/1",
20     "type": "photo",
21     "sizes": {...}
22   }]
23 },
24 "in_reply_to_user_id_str": null,
25 "contributors": null,
26 "text": "Introducing the Twitter Certified Products Program: https://t.co/MjJ8xAnT",
27 "retweet_count": 121,
28 "in_reply_to_status_id_str": null,
29 "id": 240859602684612608,
30 "geo": null,
31 "retweeted": false,
32 "possibly_sensitive": false,
33 "in_reply_to_user_id": null,
34 "place": null,
35 "user": {...},
36 "in_reply_to_screen_name": null,
37 "source": "<a href=\"http://sites.google.com/site/yorufukurou/\" rel=\"nofollow\">YoruFukurou</a>",
38 "in_reply_to_status_id": null
39 }
```


Figure 6: An example of the output of the API method *user_timeline*.

bundle to the user specified on lines 9 and 10.

An important feature, common to almost all social networks, is the possibility to appreciate a resource or another user profile. In our model, this concept is represented by means of the *like* edge. Consider the social network `about.me`, in which a user is allowed to favor another user profile, thus making an "endorsement". Information about this action is obtained by calling the method `http://api.about.me/api/v2/json/user/view/` of the `about.me` API followed by the desired user *screen_name*. Figure 7 reports an example of the output of this method called on the profile of one of the authors of this paper. The returned information has to be seen from the "caller" point of view (i.e., the authenticated user), therefore the line 21 indicates that the user *snicolazzo* is in the favourite list of the user calling this API method. According to our model, a *like* edge from the authenticated user to the user with the given *snicolazzo* is created. A similar reasoning can be applied also

```
{
 "status": 200,
 "profile": "http://about.me/snicolazzo",
 "user_name": "test account",
 "first_name": "test22",
 "last_name": "tester",
 "display_name": "test22 tester",
 "header": "my headline",
 "bio": "test this is one!!!!",
 "background": "http://about.me/.../snicolazzo_1326415784_79.jpg",
 "mobile_background": "",
    "email_searchable": true,
    "email_public": false,
    "avatar": "http://about.me/.../snicolazzo_1325746595_83.jpg",
    "img_base_url": "http://about.me/.../thumbnail",
    "thumbnail_291x187": "http://about.me/.../291x187/snicolazzo.jpg",
    "thumbnail1": "http://about.me/.../803x408/snicolazzo.jpg",
    "thumbnail2": "http://about.me/.../260x176/snicolazzo.jpg",
    "thumbnail3": "http://about.me/.../198x134/snicolazzo.jpg",
 "thumbnail4": "http://about.me/.../161x109/snicolazzo.jpg",
 "is_fav": true
}
```


Figure 7: An example of the output of the API method *view*.

for "g+1" of `Google+` and "Like" of `Facebook`.

So far, we have seen how to extract information from different social networks and how to map them to the concepts defined in our model. Once this mapping has been done, a data-structure is obtained. It can be serialized using the XML language. In the following, we will show some details about the XML schema designed for our model.

Figures 8 shows the mind map of this XML schema [3]. The root element is *SocialGraph* and contains two unbounded sets of elements, namely *SocialNode* and *SocialEdge*. An element *SocialNode* is specialized in one of the following complex types: *SocialProfile*, *Resource*, or *Bundle*. The element *SocialEdge* is specialized in one of the following complex types: *Follow*, *Me*, *Publishing*, *Tagging*, *Shared*, or *Referencing*. Each complex-type in this XML Schema is defined according to the corresponding objects defined in Section 5. Figure 9 reports a fragment of the XML schema allowing the serialization of our model. The complete XML Schema is available at `http://www.infolab.unirc.it/OSNmodel.html`.

We conclude this section by showing in Figure 10 an example of a fragment of an XML document derived from the XML Schema described above.

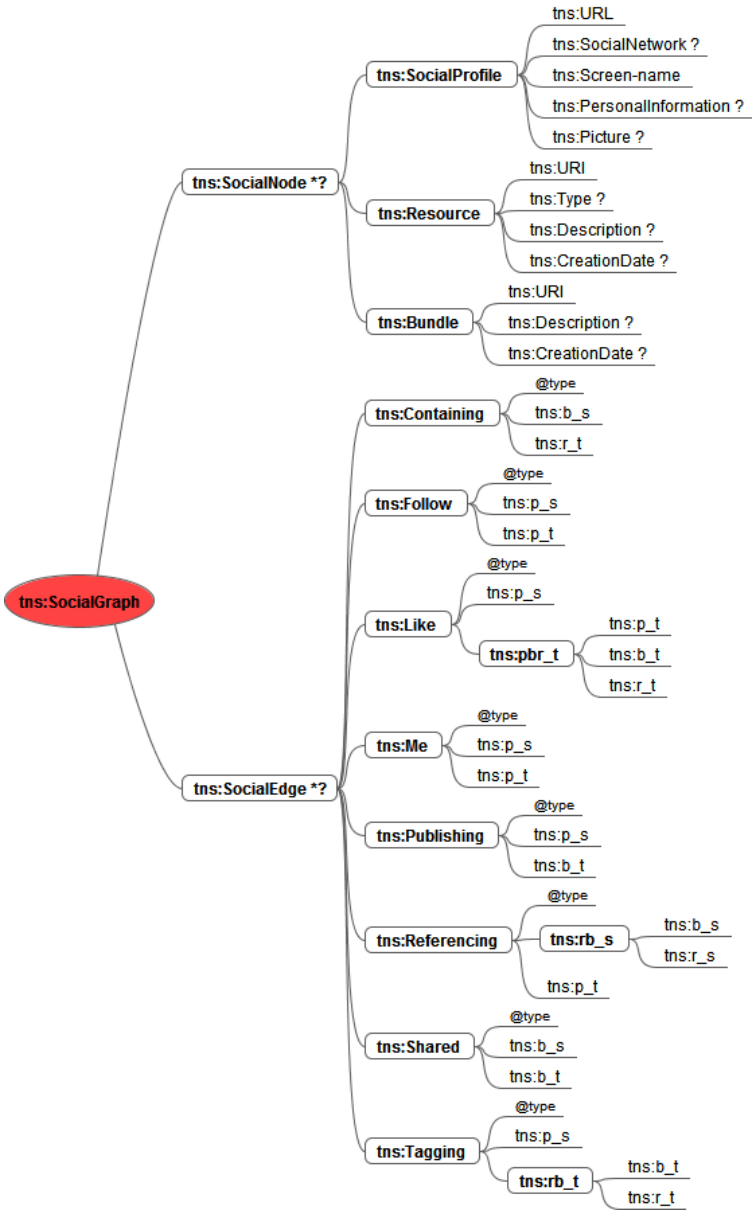

Figure 8: The mind map of our XML Schema.

```
1 ...
2  <element name="SocialGraph">
3   <complexType>
4    <sequence>
5     <element ref="tns:SocialNode" minOccurs="0" maxOccurs="unbounded" />
6     <element ref="tns:SocialEdge" minOccurs="0" maxOccurs="unbounded" />
7    </sequence>
8   </complexType>
9  </element>
10
11 <element name="SocialNode">
12  <complexType>
13   <choice>
14    <element ref="tns:SocialProfile" />
15    <element ref="tns:Resource" />
16    <element ref="tns:Bundle" />
17   </choice>
18  </complexType>
19 </element>
20
21 <element name="SocialProfile">
22  <complexType>
23   <sequence>
24    <element name="URL" type="string" />
25    <element name="SocialNetwork" type="string" minOccurs="0" />
26    <element name="Screen-name" type="string" />
27    <element name="PersonalInformation" type="string" minOccurs="0" />
28    <element name="Picture" type="string" minOccurs="0" />
29   </sequence>
30  </complexType>
31  <xs:key name="spkey">
32   <xs:selector xpath="SocialGraph/SocialNode/SocialProfile" />
33   <xs:field xpath="@URL" />
34  </xs:key>
35 </element>
...
43 <element name="SocialEdge">
44  <complexType>
45   <choice>
46    <element ref="tns:Containing" />
47    <element ref="tns:Follow" />
48    <element ref="tns:Like" />
49    <element ref="tns:Me" />
50    <element ref="tns:Publishing" />
51    <element ref="tns:Referencing" />
52    <element ref="tns:Shared" />
53    <element ref="tns:Tagging" />
54   </choice>
55  </complexType>
56 </element>
...
```

Figure 9: A portion of our XML Schema.

```
<?xml version="1.0" encoding="UTF-8"?>
<tns:SocialGraph xmlns:tns="http://www.unirc.it/SocialGraph"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://www.unirc.it/SocialGraph SocialGraph.xsd">

 <tns:SocialNode>
  <tns:SocialProfile>
   <tns:URL>twitter.com/antoninonocera</tns:URL>
   <tns:SocialNetwork>Twitter</tns:SocialNetwork>
   <tns:Screen-name>AntoninoNocera</tns:Screen-name>
  </tns:SocialProfile>
 </tns:SocialNode>

 <tns:SocialNode>
  <tns:SocialProfile>
   <tns:URL>twitter.com/serenanicolazzo</tns:URL>
   <tns:SocialNetwork>Twitter</tns:SocialNetwork>
   <tns:Screen-name>serenanicolazzo</tns:Screen-name>
  </tns:SocialProfile>
 </tns:SocialNode>

 <tns:SocialNode>
  <tns:Bundle>
   <tns:URI>240859602684612608</tns:URI>
   <tns:Description>beautiful photo :-)</tns:Description>
   <tns:CreationDate>2014-12-29</tns:CreationDate>
   </tns:Bundle>
 </tns:SocialNode>

 <tns:SocialNode>
  <tns:Resource>
   <tns:URI>http://pbs.twimg.com/media/A7EiDWcCYAAZT1D.jpg</tns:URI>
   <tns:Type>photo</tns:Type>
   <tns:Description>A pic of us</tns:Description>
   <tns:CreationDate>2014-12-29</tns:CreationDate>
  </tns:Resource>
 </tns:SocialNode>

 <tns:SocialEdge>
  <tns:Follow type="F">
   <tns:p_s>twitter.com/antoninonocera</tns:p_s>
   <tns:p_t>twitter.com/serenanicolazzo</tns:p_t>
  </tns:Follow>
 </tns:SocialEdge>

 <tns:SocialEdge>
  <tns:Publishing type="Pu">
   <tns:p_s>twitter.com/antoninonocera</tns:p_s>
   <tns:b_t>240859602684612608</tns:b_t>
  </tns:Publishing>
 </tns:SocialEdge>

 <tns:SocialEdge>
  <tns:Containing type="Co">
   <tns:b_s>240859602684612608</tns:b_s>
   <tns:r_t>http://pbs.twimg.com/media/A7EiDWcCYAAZT1D.jpg</tns:r_t>
  </tns:Containing>
 </tns:SocialEdge>
 <tns:SocialNode>
  <tns:SocialProfile>
   <tns:URL>www.facebook.com/antonino.nocera.35</tns:URL>
   <tns:SocialNetwork>Facebook</tns:SocialNetwork>
   <tns:Screen-name>Antonino Nocera</tns:Screen-name>
   <tns:PersonalInformation>Male</tns:PersonalInformation>
  </tns:SocialProfile>
 </tns:SocialNode>
 <tns:SocialEdge>
  <tns:Me>
   <tns:p_s>twitter.com/antoninonocera</tns:p_s>
   <tns:p_t>www.facebook.com/antonino.nocera.35</tns:p_t>
  </tns:Me>
 </tns:SocialEdge>
 ...
</tns:SocialGraph>
```

Figure 10: An example of an XML document.

Lines 6-20 show the definition of the `Twitter` profiles of two of the authors of this paper. Lines 22-37 represent the definition of a new bundle and a new resource of type "photo" (line 33). In lines 39-44, a *follow* edge among the two `Twitter` profiles is defined. The bundle described in lines 22-28 is published by one of the authors of this paper and the *publishing* action is encoded in lines 46-51. The resource is contained in the bundle as shown in lines 53-58 with the definition of a *containing* edge. Finally, the second account in `Facebook` of one of the authors is modeled in lines 59-66 and the information that this account and that of `Twitter` belong to the same person is encoded in lines 67-72.

7. Case studies

Evaluating the accuracy of a model is a difficult task because often a golden standard misses [5]. In these cases, evaluation can be done by humans (e.g., [44, 41]) or by applying the model to an application and evaluating the results (e.g., [52]). In this section, following the latter approach, we describe how our model has been profitably applied to two applications very relevant in the context of social network analysis. The first application regards the extraction of information from a multiple-social-network scenario, the second one concerns a particular analysis done on social network data.

7.1. Information extraction

It is well known that any analysis activity on social network users needs a preliminary task implementing the extraction of data from social networks. In the past, several crawler-based strategies have been adopted to extract data, such as Breadth First Search [70], Random Walk [40] or Metropolis-Hastings Random Walk [64].

A crawling task should implement the following steps.

1. Selecting the starting account (seed). This step is very important to provide data useful to the specified application. Usually, the starting account is randomly selected from an available pool of accounts. For particular analysis, the seed can be selected from those accounts having some characteristics, for example, being a power user (i.e., they have a number of contacts much higher than the average user [39]).

2. Building the sub-graph. In this step, the information about this account is created: it includes the user account, contacts, published resources, and so on. This step is strongly dependent on the data model used.
3. Selecting the next account. There exist several strategies to implement this step. A first possibility is to randomly select another profile (uniform sampling), and this is feasible whenever a social network uses an identifier for accounts and the domain of identifiers is known and limited. This occurs for example for `Facebook` and `Twitter` [23]. Another possibility consists in selecting one profile (i.e., a node of the graph) connected with the last visited profile by a *follow* edge or a *me* edge (see, for example, [40, 64]). Again, it is also possible to select more than one (even all) among the profiles referred above, as done for example in [70]. Once one or more profiles have been selected, Steps 2 and 3 are iterated until the desired amount of data have been extracted or a stop condition has been reached.

As it emerges from the previous description of the crawling task, data extraction is not simple to be performed because there is the problem of receiving data from different sources. In this case, we need a model that is able to handle indifferently data from different social networks.

For instance, if we need a Breadth First Search crawler operating over multiple social networks, our model allows us to implement the solution described by Algorithm 1.

The algorithm performs a Breadth First Search over user profiles and gathers all the information related to each profile visited. Specifically, it starts by adding the seed profile p_0 to the FIFO queue and executes a loop until the queue has any element. At each iteration, first, it extracts a profile from the queue (Line 4), and all its information are gathered from its social network (Line 5). Then, according to the Breadth First Search logic, the visit continues by considering the neighbors of the currently visited profile. For this purpose, all current-profile neighbors (i.e., the profiles linked by means of a *follow* edge) are stored and added to the FIFO queue (Lines 6-10). Moreover, because we are operating in a multiple social network scenario, also the other profiles of the considered user in other social networks are visited. This is obtained by storing and adding all the profiles linked by means of a *me* edge to the FIFO queue (Lines 11-15). Finally, all the information related to the actions performed by the current account (i.e., profile) are also stored

Algorithm 1 *BFS*

Input p_0: a seed profile
Variable *ProfileQueue*: a FIFO queue of profiles
1: *ProfileQueue*:=$\emptyset$
2: insert p_0 into *ProfileQueue*
3: **while** *ProfileQueue* !=$\emptyset$ **do**
4: poll a profile p_s from the *ProfileQueue*
5: store p_s
6: store *follow* edges of (p_s, p_t)
7: **for each** *follow* edge (p_s, p_t) **do**
8: insert p_t into *ProfileQueue*
9: store p_t
10: **end for**
11: store *me* edges of (p_s, p_t)
12: **for each** `me` edge (p_s, p_t) **do**
13: insert p_t into *ProfileQueue*
14: store p_t
15: **end for**
16: store *resources* r_s of p_s
17: store *bundles* b_s of p_s
18: store *containing* edges (b_s, r_s)
19: store *publishing* edges (p_s, b_s)
20: store *shared* edges (b_s, b_t)
21: store *tagging* edges (p_s, br_s)
22: store *referencing* edges (b_s, p_t)
23: store *like* edges (p_s, pbr_t)
24: **end while**

(Line 16-23). The store operations are implemented according to the technicalities presented in Section 6, which regards how information is extracted from social networks. For the sake of presentation, such technicalities are not explicitly reported in the algorithm.

Observe that the algorithm presented above can be simplified: indeed, it is possible to set the maximum number of iterations performed by the algorithm and to reduce the information gathered for a profile on the basis of that strictly requested by the application built on top of the crawler. The former goal can be obtained by modifying the loop-stop condition at Line 3 of the previous algorithm, whereas the latter goal can be fulfilled by removing the unnecessary instructions at Lines 16-23.

It is worth mentioning that a solution very close to the one described above has been implemented in the SNAKE system [15], a tool supporting the extraction of data from social network accounts. In this system, our model has been successfully used to allow the extraction and storage of the information about the neighborhood and the alternative accounts in other social networks of a seed profile.

In order to test the correctness and effectiveness of the data extraction

procedure above, we designed a prototype implementing the techniques described in Section 6. After the prototype implementation phase, in which we fixed several errors in data extraction, we performed a final prototype validation step[1]. For this purpose, we gathered a controlled dataset from real-life social network accounts. We run a social network crawler (as described in Section 7.1) starting from a randomly selected seed. As for seed selection, we observe that many social networks, such as Facebook and Twitter, associate each account with an incremental integer (specifically, a 64-bit number). Thanks to this feature, to obtain a random seed it suffices to generate numbers uniformly at random in a suitable interval and, for each number, to verify whether it corresponds to an existing account (because an account could have been deleted). Fortunately, the most of the identifiers are associated with active accounts, so that few attempts are enough to find a valid seed. In the implementation of the crawler, to have a feasible stop condition, we slightly modified Algorithm 1 by enforcing that when the `ProfileQueue` (see variable of contains Algorithm 1) contains 200 profiles, no more profile is inserted, so that the crawler ends by visiting such profiles.

At the end of the crawling activity, we obtained a dataset containing information (concerning profiles, resources, friendship, etc.) coming from several social networks. Observe that not all profiles included in the `ProfileQueue` appear in the dataset because some of them were private or inactive. Specifically, the dataset is composed of 265 data from Twitter, 234 from Facebook, 195 from LinkedIn, 196 from Flickr, 270 from Google+, 124 from LiveJournal, 54 from Advogato, and 44 from about.me. Table 2 reports the number of data extracted for each feature subdivided by social network.

We analyzed this dataset to find possible extraction faults, due to many practical issues in processing the social network data. We manually verified that the information extracted in the dataset corresponded to the actual online data. In Table 3, we explicit the checked attributes for each type of data. We considered *correct* the operation of extraction of a profile, a resource, and so on, if and only if the value of the extracted attributes matched that actually reported on the Web. In particular:

- a profile is considered correctly extracted if the data about the url, the belonging social network, and the screen-name match the corresponding

[1]Clearly, changes in the technologies used by the social networks (e.g., API) should need suitable adaptations of the prototype.

	User Profiles	Resources	Tags	Comments	Friendships	me edges	Likes	Posts
Twitter	12	22	41	28	81	35	15	31
Facebook	14	26	35	23	78	29	11	18
LinkedIn	11	21	34	17	54	24	13	21
Flickr	9	28	39	19	42	18	9	23
Google+	13	29	40	31	80	31	18	28
LiveJournal	9	14	21	14	36	16	N.A.	14
Advogato	8	11	N.A.	12	N.A.	11	N.A.	12
about.me	8	N.A.	11	N.A.	N.A.	9	6	10

Table 2: Dataset composition: number of data per feature and social network (N.A. indicates that a feature is not available for that social network).

Type of data	User Profiles	Resources	Tags	Comments
Checked Attributes	Screen name OSN name url	url resource type	source profile target label	source profile resource uri

Type of data	Friendships	me edges	Likes	Posts
Checked Attributes	source profile target profile	source profile target profile	source profile entity	source profile referred resource

Table 3: The attributes compared to verify the correctness of the extracted information.

values on the Web;

- for resources, the verification test regards the url and the resource type;
- the correctness of a tag is verified by checking the source profile, the target resource or bundle, and the label itself;
- a comment is considered correct if the source profile and the content (i.e., the bundle uri) match the online ones;
- the correctness of friendship and `me` edges involves the target and the source profiles;
- to assess if a like is correct, we checked the source profile and the entity (i.e., a resource, a bundle, or a profile) to which the like referred to;
- and, finally, we verified the correctness of a post by analyzing the source profile and the bundle or the resource which it refers to.

The evaluation showed that the whole dataset has been extracted with no error: therefore, the number of successful tests reached 100% at the final stage of the prototype development.

Finally, we recall that the comparison among our model and the state of the art approaches on the basis of the concepts and relationships to extract has been discussed in Section 3 (see, in particular, Table 1).

7.2. Matching accounts on social networks

The problem of matching user accounts is receiving a great attention in several application scenarios, such as personalization [17, 28, 71]. Indeed, people have accounts on diverse social networks, where they disseminate several traits of their personality. Gathering these traits in a unique profile is extremely useful in disparate application contexts. Unfortunately, automatically connecting the different social-network identities of users is not a trivial task due to the heterogeneity of the users' information representation in different social networks.

A common approach to address this problem utilizes profile matching techniques typically based on a set of identification properties, such as username, to find correspondences between user identities. For instance, the authors of [28] describe a technique relying on usernames and tags used in three collaborative tagging systems: Flickr, Delicious and StumbleUpon. In particular, for the identification of candidate users to be matched across social systems, the authors extract information about tagging activities to measure the frequency of use of each tag. Then, they compare tagging behavior in the different social networks to identify candidates. Finally, they extract information about the usernames of these candidates and use popular string similarity functions to match them. Another approach is that of [66], which makes use of machine learning classification techniques to match user profiles on the basis of user real name, birthday, interests, attended high school, job, and so forth, extracted from social networks. Moreover, they consolidate their results by analyzing the structural properties of the direct-friend network. Another approach is described in [71], in which the authors infer 7 hypotheses on the relationships among the usernames chosen by a single person in different communities, starting from the observation of data in BlogCatalog. They propose an approach that, given a username u suitably extracted from a source community, and a target community c, generates a set of candidate usernames in c corresponding to u. The approach first generates a set of usernames from u by adding and removing suitable

prefixes and suffixes. Then, it exploits the information extracted through a Web search on Google aimed at checking for the existence of each candidate username in such a way as to reduce the returned set of usernames. Also the approach of [45] can be collocated in this context, even if its purpose is little different. Indeed, in this paper the authors focus on the de-anonymization of a network by using auxiliary information from a different social site on the basis of membership overlap and structural similarity. The authors give a demonstration of their solution on Flickr and Twitter. For this purpose, first, they extract and anonymize information from Twitter. After this, they extract information from Flickr and use it to apply their de-anonymization strategy on the modified Twitter graph.

All the above approaches require the comparison of profile information coming from different social networks, which could be quite heterogeneous. Handling such an heterogeneity is a very hard task and requires a pre-phase in which data extraction and concept-matching have to be performed. In this context, the adoption of our model can be very useful because it avoids this preliminary step.

Observe that, in the context of identity matching, our model has been successfully adopted in [14]. The approach proposed computes the similarity between two accounts belonging to two different social networks by combining two contributions: a string similarity between the usernames of the two accounts and a contribution based on a suitable notion of common-neighbors similarity. The latter component leads to a recursive definition of the overall inter-social-network similarity. Indeed, the common-neighbors notion has to rely on the same notion because neighbors belong to different social networks, and, hence, common nodes have to be detected too.

The identification of matching accounts can be seen from two different points of view leading to the formulation of two sub-problems. The former concerns the identification of accounts of the same user in different social networks starting from a seed account. This is an intrinsically-hard task due to the wideness of the search space, which could potentially involve the entire social Web. The latter, instead, receives two accounts and aims at verifying whether these belong to the same user.

The use of our model allowed us to simplify these issues and to handle all profiles in a uniform way. We carried out an analysis of the performance of the identity matching approach in [14] adopting our model: concerning the first sub-problem, starting from a seed account of a user in a social network, the approach was able to find, among all the user accounts of the other

Name	*Sensitivity*
Salton Index [56]	0.01
Jaccard Index [30]	0.01
Sorensen Index [57]	0.01
Hub Promoted Index [53]	0.00
Hub Depressed Index [42]	0.01
Leicht-Holme-Newman Index [37]	0.01
Resource Allocation Index [73]	0.01
Local Path Index [73]	0.03
Our model applied to [14]	0.87

Table 4: Comparison among our approach and the state of the art.

covered social networks (which is a set of more than 10^9, i.e., almost all Web users), the alternative accounts of the same user in 57% of cases in a time interval ranging from 1 to 2.1 seconds.

As for the second sub-problem, we compared the identity matching approach in [14] adopting our model with the most meaningful local and quasi-local similarity indices (namely, Salton Index, Jaccard Index, Sorensen Index, Hub Promoted Index, Hub Depressed Index, Leicht-Holme-Newman Index, Resource Allocation Index, and Local Path Index). In our evaluation, we preliminarily extracted information about a set M of 100 social accounts, each having a secondary account in another social network. Then, we run each of the above techniques and obtained a set M' of the accounts that the technique detected as matching accounts. Clearly, M' represents a set of true positives. Finally, we measured the *sensitivity* of the techniques as the ratio $\frac{|M'|}{|M|}$. Table 4 summarizes the results of this comparison.

The results of Table 4 show that, when applied to our application domain, our approach outperforms classical state-of-the-art approaches based on common neighbors techniques. Indeed, while those approaches have a sensitivity less than or equal to 0.03, our approach reaches a sensitivity of 0.87.

It is worth noting that the low performance of common-neighbors-based approaches can be justified by the fact that they detect as similar only accounts having exactly the same information. However, in the considered scenario, typically users create accounts on social networks for different purposes and, thus, they can share different contents. Due to this reason, the intersection of neighbors can be low. As discussed previously, to overcome this problem, our approach does not rely on syntactic intersection but on similarity-based intersection. For the interested reader, further and deep-

ened details on these experiments and comparisons are available in [14].

In summary, we can state that the success of this technique strongly relies on the model described in this paper. Moreover, it is worth noting that the paper referred above (i.e., [14]) is completely not focused on the modeling (and implementation) aspects faced in this paper, so that this paper adds original relevant work.

8. Conclusion

It is a matter of fact that the multiplicity of social networks together with users' membership overlap, result in a multiplicative effect in terms of information power. Indeed, correlation, integration, negotiation of information coming from different social networks offer a lot of strategic knowledge whose benefits are still unexplored.

Starting from this awareness, in this paper, we addressed an important issue: Social-network-based programming should work at the multiple-social-network abstraction level, that of a global social network formed by autonomous components with strong correlation and interaction.

To accomplish the above goal, our proposal goes beyond a mere definition of APIs working over all social networks, as it operates at a conceptual level, by building a semantic middleware that abstracts real-life social networks towards a truly multiple-social-network perspective. At this level, all entities and relationships of single social networks are maintained, including the user-centered vision allowing us to transparently associate with a user the information coming from all the social networks he belongs to. However, the approach followed in this paper is also practical, in the sense that we solved the trade-off between complexity/expressiveness of the conceptual model and implementation issues in favor of the latter. In other words, despite a somewhat simple model focused on the crucial concepts underlying real-life social networks, the resulting benefits from the implementation perspective appears considerable. The effectiveness of our approach is shown by means of two case studies, but we believe, as a future work, that a lot of further applications may confirm the relevance of our approach. Another possible future research direction, following this proposal, could be the design of multiple-social-network-oriented languages based on our model.

Acknowledgment

This work has been partially supported by the TENACE PRIN Project (n. 20103P34XC) funded by the Italian Ministry of Education, University and Research, by the Program "Programma Operativo Nazionale Ricerca e Competitività" 2007-2013, project BA2Kno (Business Analytics to Know) PON03PE_00001_1, in "Laboratorio in Rete di Service Innovation", and by the Program "Programma Operativo Nazionale Ricerca e Competitività" 2007-2013, Distretto Tecnologico CyberSecurity funded by the Italian Ministry of Education, University and Research.

References

[1] N. Akhtar. Social network analysis tools. In *Communication Systems and Network Technologies (CSNT), 2014 Fourth International Conference on*, pages 388–392. IEEE, 2014.

[2] A. Alkouz, E. W. D. Luca, and S. Albayrak. Latent Semantic Social Graph Model for Expert Discovery in Facebook. In *IICS*, pages 128–138, 2011.

[3] J. Beel and B. Gipp. Enhancing search applications by utilizing mind maps. In *Proceedings of the 21st ACM Conference on Hypertext and Hypermedia*, pages 303–304. ACM, 2010.

[4] G. Bell. *Building social web applications.* "O'Reilly Media, Inc.", 2009.

[5] J. Brank, M. Grobelnik, and D. Mladenić. A survey of ontology evaluation techniques. In *In Proceedings of the Conference on Data Mining and Data Warehouses (SiKDD 2005)*, 2005.

[6] D. Brickley and L. Miller. FOAF Vocabulary Specification 0.91. Technical report, Tech. rep. ILRT Bristol. Available at `http://xmlns.com/foaf/spec/20071002.html`, 2000.

[7] F. Buccafurri, G. Lax, S. Nicolazzo, and A. Nocera. A Model to Support Multi-Social-Network Applications. In *Proc. of the International Conference Ontologies, DataBases, and Applications of Semantics (ODBASE 2014)*, pages 639–656, Amantea, Italy, 2014. Springer.

[8] F. Buccafurri, G. Lax, S. Nicolazzo, and A. Nocera. Fortifying tripadvisor against reputation-system attacks. In *Internet Security (WorldCIS), 2014 World Congress on*, pages 20–21. IEEE, 2014.

[9] F. Buccafurri, G. Lax, S. Nicolazzo, and A. Nocera. A privacy-preserving solution for tracking people in critical environments. In *Computer Software and Applications Conference Workshops (COMPSACW), 2014 IEEE 38th International*, pages 146–151. IEEE, 2014.

[10] F. Buccafurri, G. Lax, S. Nicolazzo, and A. Nocera. Comparing twitter and facebook user behavior: privacy and other aspects. *Computers in Human Behavior*, 52:87–95, 2015.

[11] F. Buccafurri, G. Lax, S. Nicolazzo, A. Nocera, and D. Ursino. Measuring betweenness centrality in social internetworking scenarios. In *On the Move to Meaningful Internet Systems: OTM 2013 Workshops*, pages 666–673. Springer, 2013.

[12] F. Buccafurri, G. Lax, S. Nicolazzo, A. Nocera, and D. Ursino. Driving Global Team Formation in Social Networks to Obtain Diversity. In *Proc. of the International Conference on Web Engineering (ICWE 2014)*, pages 410–419, Toulouse, France, 2014. Springer.

[13] F. Buccafurri, G. Lax, and A. Nocera. A new form of assortativity in online social networks. *International Journal of Human-Computer Studies*, 80:56–65, 2015.

[14] F. Buccafurri, G. Lax, A. Nocera, and D. Ursino. Discovering Missing Me Edges across Social Networks. *Information Sciences*, 319:18–37, 2015. Elsevier.

[15] F. Buccafurri, G. Lax, A. Nocera, and D. Ursino. A system for extracting structural information from social network accounts. *Software: Practice and Experience*, 2015. DOI: 10.1002/spe.2280.

[16] G. Caldarelli. *Scale-Free Networks: Complex Webs in Nature and Technology*. Number 9780199211517 in OUP Catalogue. Oxford University Press, 2007.

[17] F. Carmagnola and F. Cena. User identification for cross-system personalisation. *Information Sciences*, 179(1-2):16–32, 2009.

[18] C. Diamantini, D. Potena, and E. Storti. Semantically-supported team building in a kdd virtual environment. In *Collaboration Technologies and Systems (CTS), 2012 International Conference on*, pages 45–52. IEEE, 2012.

[19] P. Erdös and A. Rényi. On Random Graphs, I. *Publicationes Mathematicae*, 6:290–297, 1959.

[20] Y. Gao, M. Wang, Z.-J. Zha, J. Shen, X. Li, and X. Wu. Visual-textual joint relevance learning for tag-based social image search. *Image Processing, IEEE Transactions on*, 22(1):363–376, 2013.

[21] M. Gatti, P. Cavalin, S. B. Neto, C. Pinhanez, C. dos Santos, D. Gribel, and A. P. Appel. Large-scale multi-agent-based modeling and simulation of microblogging-based online social network. In *Multi-Agent-Based Simulation XIV*, pages 17–33. Springer, 2014.

[22] G. Ghoshal, V. Zlatić, G. Caldarelli, and M. E. J. Newman. Random hypergraphs and their applications. *Physical Review E*, 79(6):066118, 2009.

[23] M. Gjoka, M. Kurant, C. Butts, and A. Markopoulou. Walking in Facebook: A case study of unbiased sampling of OSNs. In *Proc. of the International Conference on Computer Communications (INFOCOM'10)*, pages 1–9, San Diego, CA, USA, 2010. IEEE.

[24] D. Greene and P. Cunningham. Producing a unified graph representation from multiple social network views. In *Proceedings of the 5th Annual ACM Web Science Conference*, pages 118–121. ACM, 2013.

[25] A. Greve and J. W. Salaff. Social networks and entrepreneurship. *Entrepreneurship theory and practice*, 28(1):1–22, 2003.

[26] T. Guan, Y. He, L. Duan, J. Yang, J. Gao, and J. Yu. Efficient bof generation and compression for on-device mobile visual location recognition. *MultiMedia, IEEE*, 21(2):32–41, 2014.

[27] T. Guan, Y. He, J. Gao, J. Yang, and J. Yu. On-device mobile visual location recognition by integrating vision and inertial sensors. *Multimedia, IEEE Transactions on*, 15(7):1688–1699, 2013.

[28] T. Iofciu, P. Fankhauser, F. Abel, and K. Bischoff. Identifying users across social tagging systems. In *Proc. of the International Conference on Weblogs and Social Media (ICWSM'11)*, Barcelona, Catalonia, Spain, 2011. The AAAI Press.

[29] J. Iturrioz, O. Diaz, and C. Arellano. Towards federated web2. 0 sites: The tagmas approach. In *Tagging and Metadata for Social Information Organization Workshop, WWW07*, 2007.

[30] P. Jaccard. *Etude comparative de la distribution florale dans une portion des Alpes et du Jura*. Impr. Corbaz, 1901.

[31] M. Jamali and M. Ester. A matrix factorization technique with trust propagation for recommendation in social networks. In *Proceedings of the fourth ACM conference on Recommender systems*, pages 135–142. ACM, 2010.

[32] M. N. Jelassi, C. Largeron, and S. B. Yahia. Efficient unveiling of multi-members in a social network. *Journal of Systems and Software*, 94:30–38, 2014.

[33] R. Ji, L.-Y. Duan, J. Chen, L. Xie, H. Yao, and W. Gao. Learning to distribute vocabulary indexing for scalable visual search. *Multimedia, IEEE Transactions on*, 15(1):153–166, 2013.

[34] G. Karypis, R. Aggarwal, V. Kumar, and S. Shekhar. Multilevel hypergraph partitioning: applications in VLSI domain. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 7(1):69–79, 1999.

[35] M. Kim and J. Leskovec. Modeling social networks with node attributes using the multiplicative attribute graph model. *arXiv preprint arXiv:1106.5053*, 2011.

[36] R. T. Leenders. Modeling social influence through network autocorrelation: constructing the weight matrix. *Social Networks*, 24(1):21–47, 2002.

[37] E. Leicht, P. Holme, and M. E. Newman. Vertex similarity in networks. *Physical Review E*, 73(2):026120, 2006.

[38] J. Leskovec, D. Chakrabarti, J. Kleinberg, C. Faloutsos, and Z. Ghahramani. Kronecker graphs: An approach to modeling networks. *The Journal of Machine Learning Research*, 11:985–1042, 2010.

[39] S.-H. Lim, S.-W. Kim, S. Park, and J. H. Lee. Determining content power users in a blog network: an approach and its applications. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 41(5):853–862, 2011.

[40] L. Lovász. Random walks on graphs: A survey. *Combinatorics, Paul Erdos is Eighty*, 2(1):1–46, 1993.

[41] A. Lozano-Tello and A. Gómez-Pérez. Ontometric: A method to choose the appropriate ontology. *Journal of Database Management*, 2(15):1–18, 2004.

[42] L. Lü and T. Zhou. Link Prediction in Complex Networks: A Survey. *Physica A: Statistical Mechanics and its Applications*, 390(6):1150–1170, 2011.

[43] V. S. A. Menezes, G. Zimbrão, and J. M. Souza. Group and link analysis of multirelational scientific social networks. *Journal of Systems and Software*, 86(7):1819–1830, 2013.

[44] P. Mika. Ontologies are us: A unified model of social networks and semantics. In *The Semantic Web–ISWC 2005*, pages 522–536. Springer, 2005.

[45] A. Narayanan and V. Shmatikov. De-anonymizing social networks. In *Proc. of the International IEEE Symposium on Security and Privacy*, pages 173–187, Oakland, California, USA, 2009. IEEE Computer Society.

[46] M. E. J. Newman, D. J. Watts, and S. H. Strogatz. Random graph models of social networks. *Proceedings of the National Academy of Sciences*, 99(suppl 1):2566–2572, 2002.

[47] D. T. Nguyen, H. Zhang, S. Das, M. T. Thai, and T. N. Dinh. Least cost influence in multiplex social networks: Model representation and analysis. In *Data Mining (ICDM), 2013 IEEE 13th International Conference on*, pages 567–576. IEEE, 2013.

[48] A. Nocera and D. Ursino. PHIS: a system for scouting potential hubs and for favoring their "growth" in a Social Internetworking Scenario. *Knowledge-Based Systems*, 36:288–299, 2012. Elsevier.

[49] S. Noor and K. Martinez. Using social data as context for making recommendations: an ontology based approach. In *Proceedings of the 1st Workshop on Context, Information and Ontologies*, page 7. ACM, 2009.

[50] G. Papadakis, K. Tserpes, E. Sardis, M. Kardara, A. Papaoikonomou, and F. Aisopos. Social media meta-api: leveraging the content of social networks. In *Proceedings of the 21st international conference companion on World Wide Web*, pages 271–274. ACM, 2012.

[51] A. Papadimitriou, P. Symeonidis, and Y. Manolopoulos. Fast and accurate link prediction in social networking systems. *Journal of Systems and Software*, 85(9):2119–2132, 2012.

[52] R. Porzel and R. Malaka. A task-based approach for ontology evaluation. In *ECAI Workshop on Ontology Learning and Population, Valencia, Spain*. Citeseer, 2004.

[53] E. Ravasz, A. L. Somera, D. A. Mongru, Z. N. Oltvai, and A.-L. Barabási. Hierarchical organization of modularity in metabolic networks. *science*, 297(5586):1551–1555, 2002.

[54] C. Romm, N. Pliskin, and R. Clarke. Virtual communities and society: toward an integrative three phase model. *International Journal of Information Management*, 17(4):261–270, 1997.

[55] B. L. Y. Rowe. Using social network graph analysis for interest detection. *arXiv preprint arXiv:1410.0316*, 2014.

[56] G. Salton and M. J. McGill. Introduction to modern information retrieval. 1986.

[57] T. Sørensen. {A method of establishing groups of equal amplitude in plant sociology based on similarity of species and its application to analyses of the vegetation on Danish commons}. *Biol. Skr.*, 5:1–34, 1948.

[58] L. Specia and E. Motta. Integrating folksonomies with the semantic web. In *The semantic web: research and applications*, pages 624–639. Springer, 2007.

[59] Statista. The statistics portal. `http://www.statista.com/statistics/282087/number-of-monthly-active-twitter-users/`, 2015.

[60] Statista. The statistics portal. `http://www.statista.com/statistics/264810/number-of-monthly-active-facebook-users-worldwide/`, 2015.

[61] Statista. The statistics portal. `http://www.statista.com/statistics/274050/quarterly-numbers-of-linkedin-members/`, 2015.

[62] Statista. The statistics portal. `http://www.statista.com/statistics/283870/google-plus-monthly-active-users-worldwide/`, 2015.

[63] A. Stewart, E. Diaz-Aviles, W. Nejdl, L. B. Marinho, A. Nanopoulos, and L. Schmidt-Thieme. Cross-tagging for personalized open social networking. In *Proceedings of the 20th ACM conference on Hypertext and hypermedia*, pages 271–278. ACM, 2009.

[64] D. Stutzback, R. Rejaie, N. Duffield, S. Sen, and W. Willinger. On unbiased sampling for unstructured peer-to-peer networks. In *Proc. of the International Conference on Internet Measurements*, pages 27–40, Rio De Janeiro, Brasil, 2006. ACM.

[65] Z. Sun, L. Han, W. Huang, X. Wang, X. Zeng, M. Wang, and H. Yan. Recommender systems based on social networks. *Journal of Systems and Software*, 99:109–119, 2015.

[66] J. Vosecky, D. Hong, and V. Shen. User identification across multiple social networks. In *Proc. of the International Conference on Networked Digital Technologies (NDT'09)*, pages 360–365, Ostrava, the Czech Republic, 2009. IEEE Press.

[67] A. H. Wang. Don't follow me: Spam detection in twitter. In *Security and Cryptography (SECRYPT), Proceedings of the 2010 International Conference on*, pages 1–10. IEEE, 2010.

[68] X. Wang, F. Wei, X. Liu, M. Zhou, and M. Zhang. Topic sentiment analysis in twitter: a graph-based hashtag sentiment classification approach. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, pages 1031–1040. ACM, 2011.

[69] P. H. Wilken. *Entrepreneurship: A comparative and historical study.* Ablex Norwood, NJ, 1979.

[70] S. Ye, J. Lang, and F. Wu. Crawling online social graphs. In *Proc. of the International Asia-Pacific Web Conference (APWeb'10)*, pages 236–242, Busan, Korea, 2010. IEEE.

[71] R. Zafarani and H. Liu. Connecting corresponding identities across communities. In *Proc. of the International Conference on Weblogs and Social Media (ICWSM'09)*, San Jose, CA, USA, 2009. The AAAI Press.

[72] Z. Zhang and C. Liu. A hypergraph model of social tagging networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2010(10):P10005, 2010.

[73] T. Zhou, L. Lü, and Y.-C. Zhang. Predicting missing links via local information. *The European Physical Journal B*, 71(4):623–630, 2009.