Post-print version of the article https://doi.org/10.1016/j.ins.2015.05.014

Discovering Missing Me Edges across Social Networks *

Francesco Buccafurri[§], Gianluca Lax, Antonino Nocera, Domenico Ursino
DIIES, Università Mediterranea di Reggio Calabria,
Via Graziella, Località Feo di Vito, 89122 Reggio Calabria, Italy
{bucca, lax, a.nocera, ursino}@unirc.it
[§]Corresponding Author

Abstract

Distinct social networks are interconnected via membership overlap, which plays a key role when crossing information is investigated in the context of multiple-social-network analysis. Unfortunately, users do not always make their membership to two distinct social networks explicit, by specifying the so-called `me` edge (practically, corresponding to a link between the two accounts), thus missing a potentially very useful information. As a consequence, discovering missing `me` edges is an important problem to address in this context with potential powerful applications. In this paper, we propose a common-neighbor approach to detecting missing `me` edges, which returns good results in real-life settings. Indeed, an experimental campaign shows both that the state-of-the-art common-neighbor approaches cannot be effectively applied to our problem and, conversely, that our approach returns precise and complete results.

Keywords: Social networks, identity management, membership overlap.

1 Introduction

In recent years, (on-line) social networks have been showing a rapid development growth becoming probably the main actor of the Web 2.0. The rapid and revolutionary diffusion of social networks among all segments of the population has attracted the interest of researchers from several fields of computer science, such as digital forensics [40, 24], user behaviour [6], trust and reputation [26], steganography [25, 28], also for the applications that the analysis of involved data can enable [61, 22, 13, 38, 14, 12]. In this landscape, Social Network Analysis and Social Network Mining have assumed an important role because both the large volume of data and their graph-based organization have enforced the development of specific models and methods allowing the study of social-network data to discover knowledge from them. Clearly, the graph-based data schema gives a great information

*A shorter abridged version of this paper appears in "Discovering Links among Social Networks", by F. Buccafurri, G. Lax, A. Nocera, and D. Ursino, in the Proc. of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD 2012), Bristol, United Kingdom, 2012. Springer. [17]

power to links among data, because it allows people profiles, resources, activities, and so on, to be directly (and indirectly) related. The crucial role of relationships in the expression of an individual's personality and social identity, traditionally recognized by social sciences, is even strengthened in the field of virtual societies, in which relationship links are the main form of expression of participation of individuals to the community. To make more challenging the analysis of this reality, consider that the reference scenario does not consist of a single, isolated, independent social network, but is a constellation of social networks, each forming a community with specific connotations, but strongly interconnected with each other. It is a matter of fact that, despite the inherent underlying heterogeneity, the interaction among distinct social networks is the basis of a new emergent internetworking scenario enabling a lot of strategic applications, whose main strength will be just the integration of possibly different communities yet preserving their diversity and autonomy. Clearly, social mining and analysis approaches may strongly rely on this huge multi-network source of information, which reflects multiple aspects of people personal life, thus enabling a lot of powerful discovering activities.

From this perspective, links among different social networks assume a fundamental role. They connect the same user on two different social networks who thus assumes the role of passing point of information from one social network the other. For this reason, we call this user *i-bridge*[1]. The link derives from the explicit user's declaration (sometimes supported and encouraged by specific tools) consisting in the insertion of `me` edges [1]. Unfortunately, for disparate reasons, users do not always make their membership to two distinct social networks explicit, by specifying the so-called `me` edge (practically corresponding to a link between the two accounts), thus missing a potentially very useful information. As a consequence, in the overall underlying (social internetworking) graph a big number of missed `me` edges exists, whose discovery represents a very important issue. In other words, an interesting problem of missing link detection arises, which partially overlaps with a link prediction issue, because we may expect that a portion of missing `me` edges will be inserted in a next stage in the graph.

In this paper, we deal with the above problem by proposing an effective solution experimentally tested in a real-life Social Internetworking Scenario (SIS, for short) [15]. To the best of our knowledge, the problem of detecting `me` edges has not been investigated in the literature, but the approach we adopt in this work, which exploits a recursive notion of common-neighbor similarity, suggested us to prior verify whether common-neighbor approaches for link prediction [47] can be directly applied to our problem. The answer to this question was definitely negative, as intuitively explained in Section 2 and experimentally confirmed in Section 6, thus motivating our work. Our solution is based on a notion of node similarity, whose usage allows us to detect whether a suitable threshold is exceeded and then a missing `me` edge between two nodes is detected. The similarity between two nodes is obtained by combining two contributions: a string similarity between the associated usernames, and a contribution based on a suitable recursive notion of common-neighbor similarity. The neighborhood

[1]The prefix "i-" stands for "internetworking" and is used to avoid ambiguity with the classic notion of "bridge" [33]. Observe that an i-bridge is a node, whereas a (classic) bridge is an edge.

similarity allows these errors to be detected and avoided. As a consequence, it is important to clarify that the problem we are addressing does not deal with the case in which a user with membership overlap between two social networks chooses the corresponding account names very different from each other. It is worth noting that, under this case, often falls the situation in which a user voluntarily keeps the two accounts separated in their respective social networks and thus avoid also to have common friends. Therefore, also neighborhood similarity fails.

The plan of this paper is as follows: in the next section, we examine related literature. In Section 3, we present our recursive notion of similarity. On the basis of this notion, we design the method we use to detect missing `me` edges. This is described in Section 4. In Section 5, we determine the computational complexity of our approach. In Section 6, we illustrate the experiments we have carried out to verify the performances of our technique. Finally, in Section 7, we draw our conclusions and sketch possible future evolutions of our research.

2 Related work

In this section, we survey the literature related to the topics addressed in this paper. It is discussed subdivided into three categories, one for each subsection.

2.1 Identifying users on the Web

The detection of `me` edges in a SIS can be seen as a special case of the problem of identifying users on the Web. As a matter of fact, it allows the features of i-bridge users to be detected. Identifying users on the Web has received a great attention in several application scenarios, such as personalization. A lot of work is devoted to verify whether user profile information can be sufficient to address this problem. In [23] the authors define and implement a framework that provides a common base for user identification for cross-system personalisation among Web-based user-adaptive systems. The corresponding user identification algorithm combines a set of identification properties, such as username, name, location or email address, and classifies a user as identified if such a combination exceeds a suitable threshold. In [42], a technique based on user profiles for identifying users across social systems is proposed. This technique has been successfully validated on three social tagging networks (Flickr, Delicious and StumbleUpon). The limit of this technique is that only few users make their profile available in social tagging platforms. A method to identify users on the basis of profile matching is proposed in [60]. In this paper, data from two popular social networks are used to evaluate the importance of fields in the Web profile and to develop a profile comparison tool. The authors of [64] provide evidence on the existence of mappings among usernames across different communities. Starting from the observation of the data in BlogCatalog, they infer 7 hypotheses on the relationships among the usernames selected by a single person in different communities. On the basis of such hypotheses, they propose an approach that, given a username u in 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 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.

In [48], the authors propose an approach to creating a digital footprint of a user joining multiple social media services by using username, display name, description, location, profile image and number of collections. An important task in this activity is played by the measurement of the similarity of the user profiles on different social networks. This task is performed by means of Jaro Winkler similarity and Wordnet-based ontologies. Profile disambiguation is then performed by means of automated classifiers. The authors prove also that userID and name are the most discriminative features for disambiguating user profiles. In [5], the authors propose an approach for user profile matching based on Conditional Random Fields that extensively combines the usage of profile attributes and social linkage. They demonstrate the importance of social links for identity resolution and show that some profiles can be matched only thanks to social relationships between online social network users. This approach is suitable when profile is poor, incomplete or hidden due to privacy settings. The authors show that their approach significantly outperforms common attribute-based approaches and can find matches that are not discoverable by using only profile information. In [65], the authors aim at addressing the cross-media user identification problem by proposing a methodology, called MOBIUS, for finding a mapping among identities of individuals across social media sites. This methodology consists of three key components: the first one identifies users' unique behavioral patterns that lead to information redundancies across sites; the second one constructs features that exploit information redundancies due to identified behavioral patterns; the third one employs machine learning for effective user identification. [36] proposes a framework that combines authorship analysis and machine learning techniques for the detection of multiple identities in social networks. In [44], the authors formalize the problem of identifying all the accounts of the same individual in different social networks and propose an algorithm to solve it. They theoretically prove the algorithm's performance on Random Graphs. In [43], the authors investigate how subtle correlations in a user's activity patterns across different sites may be exploited to infer that two accounts correspond to the same person. They analyze a variety of features, including similarity of temporal access patterns, textual content, geo-tags, and social connections.

2.2 Link prediction

From another point of view, the detection of `me` edges in a SIS is somewhat related to link prediction. Link prediction is a task of link mining aiming at predicting the (even future) existence of a link between two objects [47, 3]. In the contest of social networks, it focuses on predicting friendships among users. Often, social networks are represented as graphs [10]. As a consequence, several link prediction approaches are totally based on the structural properties of these graphs [46]. A first possibility to perform this task consists in analyzing common neighbors. Based on *preferential attachment*, [4] experimentally verifies that the probability of a relationship between two nodes is proportional

to the product of the number of their neighbors. Some approaches to link prediction rely on the notion of shortest-path distance that is computed by means of several similarity measures, such as the Katz coefficient, PageRank and SimRank. Due to the high computational cost of these measures, approximations have to be adopted to make them effective. In any case, whenever the number of nodes is considerable, the application of these methods may result in a too long running time. In conjunction with all the above techniques, strategies may be used to enhance the accuracy of predictions. Also the use of *unseen bigrams* [29] can help in the link detection task. Here, the similarity between a node A and a node B is computed by taking into account the similarity between the nodes B and C, where C is the node most similar to A. Furthermore, the quality of link detection can be improved by means of clustering techniques aiming at identifying the graph components that introduce noise in the similarity computation [46]. [54] proposes the application of statistical relational learning to link prediction in the domain of scientific literature citations. In this approach, statistical modeling and feature selection are integrated into a search mechanism over the space of database queries in such a way as to define feature candidates involving complex interactions among objects in a given relational database. [61] analyzes the localization in space and time of a large number of users by means of their call detail records. This analysis shows that users with similar movement routines are strongly connected in a social network and have intense direct interactions. This result allows implicit ties in the social network to be predicted with a significant accuracy starting from the analysis of the correlation between user movements (i.e., their mobile homophily).

In [41], the authors focus on the link prediction task which can be formulated as a binary classification problem in social network. In particular, they propose a sparse semi-supervised classification algorithm called STKPM, based on empirical feature selection. They show that STKPM outperforms several outstanding learning algorithms with smaller test errors and more stability. The dynamics of link creation and the social influence phenomenon that may trigger information diffusion in the social graph is investigated in [2]. In [39], the authors propose a Ordered Weighted Averaging (OWA) operator and a Prediction Ensemble Algorithm (LPEOWA). It assignes the aggregation weights for nine local information-based link prediction algorithms with three different OWA operators. Experiments show that LPEOWA obtains a more stable prediction performance and improves the prediction accuracy, in comparison with the nine individual prediction algorithms. The relationship between node closeness and link prediction is investigated in [45], where the authors use proximity to capture the degree of "closeness" of two nodes in the network. They introduce new metrics that model different types of interactions that can occur between network nodes. These metrics are evaluated on data about URL recommendation on Digg and Twitter. In [55], the authors propose a nonparametric link prediction algorithm for a sequence of graph snapshots over time. It predicts links based on the features of endpoints, as well as those of the local neighborhood around the endpoints themselves. The authors prove the consistency of their estimator and give a fast implementation based on locality-sensitive hashing. Experiments show that this approach behaves well when sharp fluctuations of nonlinearities are present. In [32], the authors provide a formal definition of link recommendation

across heterogeneous networks. They propose a ranking factor graph model (RFG) for predicting links in social networks. Starting from the intuition that people make friends in different networks with similar principles, they find several social patterns that are general across heterogeneous networks. These patterns are exploited to develop a transfer-based RFG model that combines them with network structure information. This way, they can investigate the fundamental principles that drive link formation and network evolution. In [52], the authors present a link prediction approach for friend recommendation that operates by traversing all paths of a limited length on the basis of the "algorithmic small world hypothesis". The experimental validation is carried out on both synthetic and real data sets (Epinions, Facebook and Hi5). In [57], the authors propose a link prediction approach that integrates classic node-similarity-based measures with community information obtained from community detection algorithms. They show that the inclusion of community information improves the accuracy of similarity-based link prediction methods. Heterogeneous and reciprocal link prediction, where links are reciprocally determined by both entities that heterogeneously belong to disjoint groups, is investigated in [22]. The nature and causes of interactions in these domains make the prediction of this kind of link significantly different from the one of conventional links. The authors propose a novel learnable framework called ReHeLP, which learns heterogeneous and reciprocal knowledge from collaborative information and demonstrate its impact on link prediction. In [30], the authors develop supervised learning approaches for link prediction in multi-relational networks and demonstrate that a supervised approach to link prediction can enhance performance. They also present results on three diverse, real-world heterogeneous information networks and discuss the trends and tradeoffs of supervised and unsupervised link prediction in multi-relational setting. To design efficient algorithms for large scale networks, researchers increasingly adapt methods from advanced matrix and tensor computation. [58] proposes a novel approach of link prediction for complex networks by means of multi-way tensors. In addition to structural data, they consider temporal evolution of a network. The proposed approach applies the canonical Parafac decomposition to reduce tensor dimensionality and to retrieve latent trends. Experiments show significant improvements for evolutionary networks in terms of prediction accuracy measured through mean average precision. In [37], the authors extend the SAN framework with several leading supervised and unsupervised link prediction algorithms and demonstrate performance improvement for each algorithm on both link prediction and attribute inference. They find that attribute inference can help inform link prediction. In [59], the authors study the prediction problem when a certain relationship will happen in the scenario of heterogeneous networks. They extend the link prediction problem to the relationship prediction one by defining both the target relation and the topological features. Then, they model the distribution of relationship building time with the use of the extracted topological features.

From the above analysis, it is evident that our approach can be related only with common-neighbor ones. However, despite their apparent closeness to ours, we can easily realize that they are not directly applicable to our context. Indeed, the notion of common-neighbors relies, in general, on the notion of common identity of the friends of a user. But discovering the common identity of users in different

social networks is for us the output of the problem, leading to a sort of recursive definition of the problem itself. We have experimentally confirmed the above claim by showing that the application of the state-of-the-art common-neighbors approaches to our problem returns very unsatisfying results. The results of these experiments are reported in Section 6.

2.3 Data de-anonymization

The problem addressed in this paper can be related to the problem of re-identification of anonymized social network data [50, 49, 53] faced in the field of privacy. In particular, in [50], de-anonymization of a network is done by using an auxiliary network only on the basis of membership overlap and structural similarity between the two networks. The authors give a demonstration of their solution by applying it to Flickr and Twitter, showing that a third of the users who are verifiable members of both Flickr and Twitter can be recognized in the completely anonymous Twitter graph with 12% error rate. Evidently, the overall precision of the above technique is definitely smaller than that of this paper. Indeed, whereas [50] detects only a third of the positive cases with 12% error rate, our technique can identify if two accounts in different social networks belong to the same person with a precision of 90% on the whole dataset (with a recall of 69%) instead of only a third as in [50]. On the other hand, the gap of precision and recall of the two techniques is not surprising, as [50] operates in conditions more disadvantageous than ours (i.e., anonymized data). From this point of view, our approach and those falling into the field of social data de-anonymization, even strongly related, are not significantly comparable.

In [63], the authors study privacy-sensitive information that are accessible from the Web and how it could be exploited to discover personal identities. The considered scenario assumes that an adversary knows a small piece of "seed" information about a targeted user and conducts extensive and intelligent search to identify the target over both the Web and an information repository collected from it. In particular, the authors model two kinds of attacker, namely tireless and resourceful attackers, analyze possible attacking mechanisms and quantify the threats of both types of attacks to general Web users. They show that a large portion of users are highly identifiable, even when only a small piece of (possibly inaccurate) seed information is known to the attackers. In [62], the authors propose SybilDefender, a sybil defense mechanism that leverages the network topologies to defend against sybil attacks in social networks, where an adversary creates multiple bogus identities to compromise the running of the system. The proposed approach is based on performing a limited number of random walks within the social graph. A de-anonymization algorithm based on node similarity measurement in proposed in [35]. This algorithm is exploited to evaluate the privacy risk of several "light" anonymization algorithms on real datasets. In [56], the authors propose an automated approach to re-identifying nodes in anonymized social networks, which uses machine learning. It uncovers artefacts and invariants of any black-box anonymization scheme from a small set of examples. The authors show that their approach is effective even when only small numbers of samples are used for training. The problem of user data de-anonymization in dynamic social networks (i.e., by taking user data updates into account) is

investigated in [31]. Here, the authors show that by utilizing correlations between sequential releases data can be de-anonymized more easily. In [38], the authors investigate conditions under which an adversary may de-anonymize a social network user by means of some auxiliary information. They consider two scenarios. In the former, the adversary has no information about the anonymized graph and they show that an adversary can predict when the node-overlap between the anonymized and auxiliary graphs is low. In the latter, the adversary is able to gain some information about the anonymized graph and they show that an adversary can identify pairs of anonymized and auxiliary graphs with high node-overlap.

Finally, we observe that social internetworking is related to multidimensional network analysis [7, 21], as each social network of a SIS can be considered as a layer (or dimension) of a single multi-layer network. Moreover, a `me` edge acts as an interconnection edge in interdependent networks. The interested reader can find a solid repertoire of basic concepts and analytical measures, which takes into account the general structure of multidimensional networks, in [8]. In the same paper a complete framework for multidimensional network analysis is proposed and tested on different real world networks. However, the specific problem addressed in this paper cannot be related to studies done in the above field.

3 The notion of similarity

Our approach operates in a Social Internetworking System (SIS) resulting from the interconnection of a number of distinct social networks. We start with the basic notion of underlying graph, which is the layer we deal with.

Definition 3.1 A *t-Social-Internetworking Graph* is a directed graph $G = \langle N, E \rangle$, where N is the set of *nodes*, E is the set of *edges* (i.e., ordered pairs of nodes) and N is partitioned into t subsets $S_1, \ldots, S_t$. The graphs corresponding to $S_1, \cdots, S_t$ are called *social graphs*. Given a node $a \in N$ we denote by $S(a)$ the social graph which a belongs to. E is partitioned into two subsets E_f and E_m. E_f is said the set of *friendship edges* and E_m is the set of `me` *edges*. E_f is such that for each $(a, b) \in E_f$, $S(a) = S(b)$, while E_m is such that for each $(a, b) \in E_m$, $S(a) \neq S(b)$. Given a node a we denote by $\Gamma(a)$ the set of nodes in $S(a)$ such that, for each $b \in \Gamma(a)$, $(a, b) \in E_f$. $\Gamma(a)$ is said the set of *neighbors* of a. Finally, given a node $a \in N$ we denote by $indegree(a)$ the cardinality of the set $\{(y, x) \in E_f \mid x = a\}$ and by $outdegree(a)$ the cardinality of the set $\{(x, y) \in E_f \mid x = a\}$.

A t-Social-Internetworking Graph $G = \langle N, E_f \cup E_m \rangle$ is the graph underlying a SIS composed of t social networks. Each node a of G is associated with a user joining the social network whose underlying graph is $S(a)$. An edge $(a, b) \in E_f$ means that the user b is a friend of the user a in the social network of $S(a)$. An edge $(c, c') \in E_m$ means that the user c in the social network of $S(c)$ has declared a `me` edge between herself and the user c' in the social network of $S(c')$. In other words, c is an i-bridge and, therefore, c and c' are actually associated with the same user, and a `me` edge

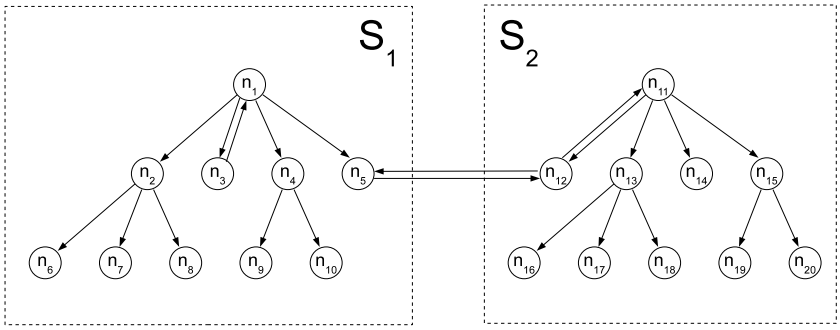

Figure 1: An example of a 2-Social-Internetworking Graph.

Node	*Username*
n_1	johnsmith
n_2	william89.hernandez
n_3	andersoncamera
n_4	diamond
n_5	brown78
n_6	obviouslywilliams
n_7	jonespict
n_8	mary_richardson
n_9	zmiller
n_{10}	wilsondj

Node	*Username*
n_{11}	dj_smith
n_{12}	brownontwitter
n_{13}	william.hernandez
n_{14}	taylormusic32tv
n_{15}	moore816
n_{16}	maria_richardson
n_{17}	hotjackson
n_{18}	t.davis
n_{19}	wildboy
n_{20}	rodriguez_sr

Table 1: Nodes and usernames into consideration.

interconnect different the two social networks $S(c)$ and $S(c')$. From now on, throughout this section, consider given a t-Social-Internetworking Graph $G = \langle N, E_f \cup E_m \rangle$.

Example 3.1 Figure 1 shows an example of a 2-Social-Internetworking Graph. In this figure, S_1 and S_2 are two social graphs corresponding to two social networks, say SN_1 and SN_2. As a consequence, the nodes $n_1 \dots n_{10}$ represent accounts of users joining SN_1, whereas $n_{11} \dots n_{20}$ are associated with users belonging to SN_2. Table 1 associates each node with the corresponding username. An example of an edge belonging to E_f is (n_1, n_2); it indicates that n_2 is a friend of n_1. The edge (n_5, n_{12}) is an example of an edge belonging to E_m; it indicates that n_5 and n_{12} are two accounts of the same user in different social networks (SN_1 and SN_2, respectively). Finally, the set of neighbors of n_1 is $\{n_2, n_3, n_4, n_5\}$.

Our approach is based on a recursive notion of "inter-social-network" similarity aimed to detect missing i-bridges of G. The similarity between two nodes a and b (belonging to two different social networks) is obtained by combining two contributions: a string similarity between the usernames

associated with a and b 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.

Concerning string similarity, several functions have been proposed in the literature, such as Jaro-Winkler, Levenshtein, QGrams, Monge-Elkan, Soundex [34]. Any of them may be adopted to measure username similarity in our approach. As a consequence, our technique is parametric w.r.t. the string-similarity function. We have evaluated the application of the different functions in Section 6.2.

Before defining our notion of similarity, we have to introduce a preliminary notion, which the common-neighbors contribution relies on. Indeed, we detect a missing `me` edge between a and b if a suitable combination of the string similarity between the usernames associated with them, according to the metric Q, and the (recursive) similarity of the top-k similar pairs each composed of a friend of a and a friend of b, is greater than a suitable threshold, for a given k.

Thus, we have preliminarily to define how to select such top-k pairs. This is related to the next definition.

Definition 3.2 Given a positive integer k_0, a pair of nodes $a, b \in N$ such that $S(a) \neq S(b)$, a string-similarity metric Q, and a non-negative integer n, we inductively define $Top^n_Q(a, b, k_0)$ as follows:

1. $Top^0_Q(a, b, k_0)$ is any subset of $C = \{(x^a, y^b) \mid x^a \in \Gamma(a), y^b \in \Gamma(b)\}$ containing the top-k_0 elements of C w.r.t. the metric Q.

2. For any $0 < i \leq n$, $Top^i_Q(a, b, k_0)$ is any subset of $C = \{(x^z, y^w) \mid (z, w) \in Top^{i-1}_Q(a, b, k_0), x \in \Gamma(z), y \in \Gamma(w), (x^z, s) \notin Top^j_Q(a, b, k_0), (s, y^w) \notin Top^j_Q\ (a, b, k_0), 0 \leq j \leq i-1\}$ containing the top-k_i elements of C w.r.t. the metric Q, where $k_i = \lceil \frac{k_0}{(1+i)^{1+i}} \rceil$ and s denotes any node in N.

In this definition, $Top^i_Q(a, b, k_0)$ indicates the set of the k_i most similar pairs of nodes (x, y) such that the distance between x (resp., y) and a (resp., b) is i.

Example 3.2 Consider the nodes n_1 and n_{11} of Figure 1. Assume that Q is the Levenshtein metric [34]. First we compute $Top^0_Q(n_1, n_{11}, 2)$. For this purpose $\Gamma(n_1) = \{n_2, n_3, n_4, n_5\}$, $\Gamma(n_{11}) = \{n_{12}, n_{13}, n_{14}, n_{15}\}$. The values of the similarities between the pairs belonging to C are shown in Table 2. The two selected pairs for $Top^0_Q(n_1, n_{11}, 2)$ are (n_2, n_{13}) and (n_5, n_{12}) because they have the highest similarities.

Now, we compute $Top^1_Q(n_1, n_{11}, 2)$. For this purpose, we must consider the friends of n_2 and n_5, on the one hand, and the friends of n_{13} and n_{12}, on the other hand. Then, we construct C by taking all the possible pairs of nodes such that the former belongs to $\Gamma(n_2) \cup \Gamma(n_5)$ and the latter belongs to $\Gamma(n_{13}) \cup \Gamma(n_{12})$. After this, we compute the corresponding similarities. They are reported in Table 3. Now, we compute $k_1 = \lceil \frac{2}{(1+1)^{1+1}} \rceil = 1$. Finally, $Top^1_Q(n_1, n_{11}, 2) = \{(n_8, n_{16})\}$ because this is the pair showing the highest similarity in Table 3.

	n_{12}	n_{13}	n_{14}	n_{15}
n_2	0.0526	0.8947	0.1053	0.1053
n_3	0.1429	0.1176	0.1333	0.0714
n_4	0.1429	0.2941	0.0667	0.0000
n_5	0.3571	0.1176	0.0667	0.1250

Table 2: Similarities between the usernames associated with $\Gamma(n_1)$ and $\Gamma(n_{11})$.

	n_{16}	n_{17}
n_6	0.0588	0.0588
n_7	0.1250	0.1000
n_8	0.8750	0.2667

Table 3: Similarities between the usernames associated with $\Gamma(n_2) \cup \Gamma(n_5)$ and $\Gamma(n_{13}) \cup \Gamma(n_{12})$.

Concerning the contribution of neighbors, we have to consider a particular situation that may significantly affect the precision of our technique. Suppose we have two nodes z and w belonging to different social networks and consider $x \in \Gamma(z)$ and $y \in \Gamma(w)$. Assume that x (resp., y) has a very high indegree in the corresponding social network (e.g., x is a public figure). In this case, the presence of x in $\Gamma(z)$ (y in $\Gamma(w)$, respectively) is not significant in defining the real life relationships of z and w. To prevent this effect, we introduce the scaling coefficient $\delta_t(x)$, for a given node x, which plays the role of deleting the contribution of x in affecting the similarity of a pair of nodes whenever x belongs to the neighborhood of one of these two nodes but x has a public-figure indegree (which the parameter t is related to).

Definition 3.3 Let $x \in N$ be a node of G and $t > 0$ be an integer number. We define $\delta_t(x) = 1 - \frac{indegree(x)}{max(indegree(x), t_h)}$, where t_h is said *public-figure threshold*.

In the definition of $\delta_t(x)$, the parameter t_h represents the indegree threshold above which a user can be considered a public figure in the social network of x. A possible way to estimate t_h is to set a suitable number of magnitude orders to increase the average indegree, say ψ, of the considered social network. For example, a reasonable value for t is $t_h = 100 \cdot \psi$. Observe that $\delta_t(x)$ tends to 0 as much as the indegree of x tends to be equal or higher than the public-figure threshold t_h, obtaining the expected effect.

Example 3.3 In Figure 1, consider the nodes n_1 and n_{11}, and the nodes $n_2 \in \Gamma(n_1)$ and $n_{13} \in \Gamma(n_{11})$. To be realistic, assume that the involved social networks SN_1 and SN_2 are LiveJournal and Twitter, respectively. Their average indegree has been estimated to 37.29 and 105.51, respectively [11]. Assume

that t_h is computed as $t_h = 100 \cdot \psi$, where ψ represents the average indegree of the considered social network. Now, $\delta_t(n_2) = 1 - \frac{1}{max(1,100 \cdot 37.29)} = 0.9997$, because n_2 belongs to LiveJournal and $\delta_t(n_{13}) = 1 - \frac{1}{max(1,100 \cdot 105.51)} = 0.9999$, as n_{13} belongs to Twitter. Thus, $min(\delta_t(n_2), \delta_t(n_{13})) = 0.9997$. In fact, in our leading example, we have no public figure and, consequently, the effects of the scaling coefficient are negligible. Conversely, consider a scenario in which n_{13} is the node associated with President Obama. Currently, President Obama has about 27M followers in Twitter. In this case, $\delta_t(n_{13}) = 1 - \frac{27 \cdot 10^6}{max(27 \cdot 10^6, 10551)} = 0$, and this implies that the presence of n_{13} in $\Gamma(n_{11})$ makes the pair (n_2, n_{13}) not significant (no matter the value of $\delta_t(n_2)$) in characterizing the real life relationship between n_1 and n_{11}.

Now we introduce the notion of *me-aware similarity* $Q^{me}(x, y)$ between two nodes whose role will be clear in the following. In words, $Q^{me}(x, y)$ takes the presence of a `me` edge between x and y into account, by upgrading to 1 their similarity independently of their string similarity.

Definition 3.4 Given a string similarity metric Q and a pair of nodes $x, y \in N$ such that $S(x) \neq S(y)$, we define the *me-aware similarity* $Q^{me}(x, y)$ between x and y w.r.t. Q, as $Q^{me}(x, y) = 1$ if $(x, y) \in E_m$, $Q^{me}(x, y) = Q(x, y)$ otherwise.

The rationale underlying this definition is the following: our approach exploits string similarity as an indicator to verify if two nodes in two different social networks are associated with the same user. The presence of a `me` edge between these two nodes directly guarantees this condition, with no need of further evaluations.

Example 3.4 Consider the nodes n_5 and n_{12} of Figure 1. We have previously seen that, by applying the Levenshtein similarity metric, $Q(n_5, n_{12}) = 0.3571$. In this case, we have that $Q^{me}(n_5, n_{12}) = 1$ because there exists a `me` edge between these two nodes.

We are ready to define our similarity function. As said above, it is obtained as a combination of two contributions, namely, the string-similarity component and the common-neighbors one. The underlying intuition is that if two accounts, belonging to different social networks, are associated with the same user, even though there is no `me` edge between them, they will have usernames somewhat related each other and, moreover, they share a (even low) number of friends. This condition can be verified by exploring real-life social networks and does not mean that the two users have a strong overlap of their neighbors, congruently with the diversity among the communities included in different social networks. However, we argue that it is highly probable that an even little neighbor overlap will occur[2].

Definition 3.5 Given a pair of nodes $a, b \in N$ such that $S(a) \neq S(b)$, a string-similarity metric Q, a public-figure threshold t_h, two integers $n \geq 0$ and $k_0 > 0$, we inductively define the *similarity operator* $T_Q^n(a, b, k_0)$ as follows:

[2]Recall that we are not interested in cases of users who voluntarily keep two accounts separated in their respective social networks, in which the above conditions are not verified, as explained in the introduction of this paper.

1. $T^0_Q(a,b,k_0) = Q(a,b)$.

2. For any $0 < i \leq n$,

$$T^i_Q(a,b,k_0) = (1-\beta_i)\cdot T^{i-1}_Q(a,b,k_0) + \beta_i \cdot \frac{\sum_{(x,y)\in Top^{i-1}_{\widetilde{Q}}(a,b,k_0)} \widetilde{Q}(x,y)}{|Top^{i-1}_{\widetilde{Q}}(a,b,k_0)|}$$

where $\beta_i = \frac{1}{(i+1)^{i+1}}$ and $\widetilde{Q}(x,y) = min(\delta_t(x),\delta_t(y))\cdot Q^{me}(x,y)$, for any $x, y \in N$ such that $S(x) \neq S(y)$.

The above definition of similarity is recursive. At the basic step, only the direct string-similarity value concurs to define the similarity between two nodes a and b. At step i, the similarity is obtained as a linear combination of the similarity of the step $i-1$, and the new common-neighbors contribution. This is obtained as the average of the reduced (by δ_t – see Definition 3.3) me-aware similarity between the top-k_i pairs w.r.t. the same metric. k_i is derived, at each step, as an exponential reduction of k_0, which is an input parameter allowing us to modulate the size of the neighbors overlapping considered relevant for the similarity computation. Observe that the above linear combination depends on the β_i parameter, which is exponentially decreasing as i increases, making quickly less important the common-neighbors contribution when leaving from the root nodes a and b.

Example 3.5 Consider the nodes n_1 and n_{11} of Figure 1. Assume that Q is the Levenshtein metric, that $n = 1$ and $k_0 = 2$, as in the previous examples, and assume given a public-figure threshold t_h.

Now, $T^0_Q(n_1,n_{11},2) = Q(n_1,n_{11}) = 0.5556$. $\beta_1 = \frac{1}{2^2} = 0.25$.

$Top^0_{\widetilde{Q}}(n_1,n_{11},2) = \{(n_5,n_{12}),(n_2,n_{13})\}$.

$T^1_Q(n_1,n_{11},2) = (1-0.25)\cdot T^0_Q(n_1,n_{11},2) + 0.25 \cdot \frac{\widetilde{Q}(n_5,n_{12})+\widetilde{Q}(n_2,n_{13})}{2}$.

Moreover, $\widetilde{Q}(n_5,n_{12}) = min(\delta_t(n_5),\delta_t(n_{12}))\cdot Q^{me}(n_5,n_{12}) = min(\delta_t(n_5),\delta_t(n_{12}))\cdot 1 =$
$min(0.9995, 0.9998) = 0.9995.$

$\widetilde{Q}(n_2,n_{13}) = 0.9997 \cdot 0.8947 \simeq 0.8944.$

Therefore, $T^1_Q(n_1,n_{11},2) = 0.75\cdot 0.5556 + 0.25 \cdot \frac{0.9995+0.8984}{2} = 0.6534.$

$Top^1_{\widetilde{Q}}(n_1,n_{11},2) = \{(n_8,n_{16})\}$.

$T^2_Q(n_1,n_{11},2) = (1-0.11)\cdot T^1_Q(n_1,n_{11},2) + 0.11\cdot \widetilde{Q}(n_8,n_{16}) =$.

Now, $\widetilde{Q}(n_8,n_{16}) = 0.8750$.

Therefore $T^2_Q(n_1,n_{11},2) = (1-0.11)\cdot T^1_Q(n_1,n_{11},2) + 0.11\cdot \widetilde{Q}(n_8,n_{16}) =$
$0.8889 \cdot 0.6534 + 0.1111 \cdot 0.8750 = 0.6781.$

Now we are ready to define the effective tool we provide to detect `me` edges, obviously based on the above notion of similarity. Indeed, Definition 3.5 leads to an ineffective computation, covering in the worst case the whole graph G. Anyway, we can observe that when, during the computation, we reach a step h whose contribution to the overall similarity is under a given small ϵ, we expect that from now on involved neighbors do not give us any meaningful information. Thus, we can stop here the iteration. This is encoded into the next definition.

Definition 3.6 Given a pair of nodes $a, b \in N$ such that $S(a) \neq S(b)$, a string-similarity metric Q, an integer number $k_0 > 0$, and a number ϵ ranging in the real interval $(0, 1]$, we define the *ϵ-similarity* $S^{\epsilon}_{Q}(a, b, k_0)$ between a and b w.r.t. Q as $T^{h}_{Q}(a, b, k_0)$, where $h > 0$ is the least number (if any) such that $|T^{h}_{Q}(a, b, k_0) - T^{h-1}_{Q}(a, b, k_0)| < \epsilon$.

Example 3.6 Consider the nodes n_1 and n_{11} in Figure 1. Assume that $k_0 = 2$, $\epsilon = 0.03$ and that Q is the Levenshtein metric. Then, the 0.01-*similarity* $S^{\epsilon}_{Q}(n_1, n_{11}, 2)$ between n_1 and n_{11} w.r.t. Q is $T^{2}_{Q}(n_1, n_{11}, 2) = 0.6781$, because $T^{2}_{Q}(n_1, n_{11}, 2)$ - $T^{1}_{Q}(n_1, n_{11}, 2) = 0.0247$.

From this example and the previous one we can draw an important conclusion. In fact, we can see that nodes n_1 and n_{11} have a "moderate" string similarity (i.e., equal to 0.5556) that does not allow us to say if they are associated with the same user or not. The examination of their neighbors leads to an increase in their similarity that becomes first equal to 0.6534 and then equal to 0.6781. This allows us to solve the uncertainty caused by the "moderate" string similarity and to conclude that n_1 and n_{11} are associated with the same user and that a hidden me edge exists between them.

Clearly, our approach is really effective if the ϵ-similarity $S^{\epsilon}_{Q}(a, b, k_0)$ between two nodes a and b always exists. This is ensured by the following theorem.

Theorem 3.1 *Given a pair of nodes $a, b \in N$ such that $S(a) \neq S(b)$, a string-similarity metric Q, an integer number $k_0 > 0$, and a number ϵ ranging in the real interval $(0, 1]$, then the ϵ-similarity $S^{\epsilon}_{Q}(a, b, k_0)$ between a and b w.r.t. Q exists.*

Proof. We have to prove that there exists $h > 0$ such that $|T^{h}_{Q}(a, b, k_0) - T^{h-1}_{Q}(a, b, k_0)| < \epsilon$. From Definition 3.5 it follows that, for any $i > 0$:

$$|T^{i}_{Q}(a, b, k_0) - T^{i-1}_{Q}(a, b, k_0)| = \beta_i \cdot \left| \left(\frac{\sum_{(x,y) \in Top^{i}_{\widetilde{Q}}(a,b,k_0)} \widetilde{Q}(x, y)}{|Top^{i}_{\widetilde{Q}}(a, b, k_0)|} - T^{i-1}_{Q}(a, b, k_0) \right) \right|.$$

Because $\beta_i \cdot \left| \left(\frac{\sum_{(x,y) \in Top^{i}_{\widetilde{Q}}(a,b,k_0)} \widetilde{Q}(x,y)}{|Top^{i}_{\widetilde{Q}}(a,b,k_0)|} - T^{i-1}_{Q}(a, b, k_0) \right) \right| < \beta_i$, it suffices to prove that there exists $h > 0$ such that $\beta_h < \epsilon$. Recall that $\beta_i = \frac{1}{(1+i)^{1+i}}$. Consider the inequality $\frac{1}{(1+i)^{1+i}} < \epsilon$, i.e., $(i+1) > log_{i+1}\frac{1}{\epsilon}$. Being $\frac{1}{\epsilon} > 1$, the function $log_{i+1}\frac{1}{\epsilon}$ is monotonically decreasing as i increases. Conversely, $(i + 1)$ is monotonically increasing. As a consequence, a number $h > 0$ exists such that $\beta_h < \epsilon$, and, therefore, the proof is concluded. □

4 Discovering links among social networks

In this section we focus on the main goal of this paper, that is the detection of `me` edges in a SIS. Consider given a Social Internetworking System (SIS) composed of t social networks, whose underlying Social Internetworking Graph is $G = \langle N, E_f \cup E_m \rangle$. Due to the hugeness of the search space and the

high sparsity of me-edges [11], we risk to meet prohibitive obstacles of infeasibility if we do not pose the problem in a suitable way. Accordingly, we consider the following two problems:

P1 (*node-seed crystallization problem*): finding missing `me` edges in G, starting from a seed a, where $a \in N$ is a node of G.

P2 (*edge-seed crystallization problem*): finding missing `me` edges in G, starting from a seed (a, b), where $(a, b) \in E_m$ is a `me` edge occurring in G.

Observe that problem **P2** could be reduced to problem **P1**. However, we will address **P2** using a specific heuristics, which takes into account that the seed is a `me` edge instead of a single node and allows us to gain in terms of efficiency. This explains why we keep the two problems separated.

Both problems above require the solution of two sub-problems, reflecting the fact that their nature allows us to use only a generate-and-test approach:

S1: Find a suitable set of candidate pairs (a, b), such that $b \notin S(a)$ and $(a, b) \notin E_m$, and

S2: For each candidate (a, b) achieved at the previous step, decide if there exists a missing `me` edge between a and b.

Clearly, the solution of sub-problem **S2** is independent of the problem which **S2** belongs to (among **P1** and **P2**). For step **S2**, our idea is to use our recursive notion of similarity introduced in the previous section. Recall that the similarity between two nodes a, b defined in Section 3 is obtained by combining two contributions: a string similarity between the usernames associated with a and b, and a contribution based on a suitable notion of common-neighbors similarity. Concerning the first contribution, as pointed out in Section 3, there exist several already defined functions for computing the similarity between two strings, each characterized by specific features (e.g., Jaro-Winkler, Levenshtein, QGrams, Monge-Elkan, Soundex, etc. [34]). The function $Q(a, b)$ (receiving two nodes a and b) in Algorithm 1 can be considered parametric w.r.t. the string-similarity function. We can choose one of the above functions on the basis of the desired target. For instance, QGrams is very severe and assigns quite low similarity degrees, Jaro-Winkler is more permissive whereas Soundex is very permissive. In Section 6.2 the application of the different functions in our technique is experimentally evaluated.

The function implementing the resolution of sub-problem **S2** is reported in Algorithm 1. It starts from a pair of nodes a, b candidate to have a `me` edge. This pair is discarded if the value of $Q(a, b)$ is lower than a suitable threshold th_c. Otherwise, a function S is called, which implements the computation of $S^{\epsilon}_{Q}(a, b, k_0)$ (see Definitions 3.5 and 3.6). If the value returned by this function is greater than a suitable threshold th_d, then (a, b) is detected as `me` edge. Otherwise, it is discarded. The function S is reported in Algorithm 2. It is recursive because it implements the operator $T^{n}_{Q}(a, b, k_0)$ of Definition 3.6. It receives as arguments: an integer $k_0 > 0$, a (small) real $\epsilon > 0$, a list L of triplets $\langle a, b, s \rangle$, where a and b are nodes and s is a $[0, 1]$ real value, the value of S at the previous step (at the initial step of the recursion this value coincides with $Q(a, b)$), and an integer $i > 0$ representing

ALGORITHM 1: $\mathbf{S_2}$

Input (a, b): the candidate `me` edge
Input th_c: a candidate threshold
Input th_d: a detection threshold
Input k_0: an integer
Input ϵ: a number in the real interval $(0, 1]$
Variable L: a list of triplets of the form $\langle a, b, s\rangle$

```
1: L := ∅;
2: if (Q(a,b) > th_c) then
3:     insert the triplet ⟨a, b, Q(a,b)⟩ into L
4:     if (S(k_0, ε, L, Q(a,b), 1) > th_d) then
5:         return true;
6:     end if
7:     L := ∅
8: end if
9: return false;
```

the step of the recursion. To explain what is s, observe that, to implement $T^n_{\widetilde{Q}}(a, b, k_0)$, we need as argument the list L storing, at the step $i > 1$ of the recursion, the top-k_i node pairs w.r.t. the metric $\widetilde{Q}^i(a, b, k_0)$. Thus, in this case, s represents just $\widetilde{Q}^i(a, b, k_0)$. At the initial step of the recursion ($i = 1$), s is just the string similarity value $Q(a, b)$.

The correspondence between Algorithm 2 and Definitions 3.5 and 3.6 is quite clear. We just have to highlight that the result of the computation of the operator $Top^i_{\widetilde{Q}}(a, b, k_0)$ (see Definitions 3.2 and 3.6) is embedded in the list L, as described above.

The computation of $Top^i_{\widetilde{Q}}(a, b, k_0)$ is implemented by Algorithm 3. The function Top receives two nodes a and b, a positive integer k_i, and the *public-figure threshold* t_h. It returns a list L of k_i triplets $\langle a', b', \widetilde{Q}^i(a', b', k_0)\rangle$, where $a' \in \Gamma(a)$, $b' \in \Gamma(b)$ and (a', b') is one of the top-k_i pairs w.r.t. the metric $\widetilde{Q}$. We remark that the metric here used includes the scaling coefficient δ of Definition 3.3.

4.1 Node-seed crystallization problem

In this section, we deal with the solution of problem $\mathbf{P_1}$. Let a be a node in G. The goal is to find the missing `me` edges involving a. Concerning sub-problem $\mathbf{S_1}$, we observe that it is not a trivial task, because an exhaustive generate-and-test approach is prohibitive, because the number of $b \notin S(a)$ such that $(a, b) \notin E_m$ is at most the number of nodes of all the social networks of the SIS but $S(a)$. We know that we have to leave the ambition of obtaining a complete technique, so the exhaustive generate-and-test approach is not necessary. But this is in not enough, because a blind generate-and-test approach has very low chances to succeed, due to the size of the universe we deal with and the very low density of explicit and implicit `me` edges. To solve this problem we have heuristically identified a nice property of social networks: With a high probability, some of the nodes belonging to the neighbors of two nodes linked by a `me` edge are in turn linked by a `me` edge, showing an assortative behavior of on-line social networks, already observed in literature for the node degree [51]. We call this property *bridge-assortativity* [18]. We tested this property by analyzing a sample which includes

ALGORITHM 2: S

Input k_0: an integer
Input ϵ: a number in the real interval $(0, 1]$
Input L_i: a list of triplets $\langle a, b, s\rangle$
Input S_{i-1}: the similarity value of a and b at step $i-1$
Input i: an integer
Output S_i: the similarity value of a and b at step i
Variable β: a [0,1] real
Variable k_i: an integer
Variable $avgS$: a real
Variable L_{i+1}: a list of triplets $\langle a, b, s\rangle$

1: $L_{i+1} := \emptyset$; $\beta := \frac{1}{i^i}$; $k_i := \lceil k_0 \cdot \beta \rceil$
2: **if** (k=1) **then**
3: $S_i := S_{i-1}$
4: **else**
5: assign to $avgS$ the average value of the similarities of the node pairs of L_i
6: $S_i := (1-\beta) \cdot S_{i-1} + \beta \cdot avgS$
7: **end if**
8: **if** $(|S_i - S_{i-1}| \geq \epsilon)$ **then**
9: **for each** $\langle a, b, s\rangle$ in L_i **do**
10: add the list returned by $Top(a, b, k_i)$ to L_{i+1}
11: **end for**
12: **return** $S(i+1, L_{i+1}, S_i)$
13: **else**
14: **return** S_i
15: **end if**

473,121 nodes, 403 i-bridges, and an average degree of 222.3261. We computed the expected value of the random variable: number of i-bridges in the neighborhood of a node. This can be obtained as the expected value of a hypergeometric distribution, i.e. $E(N) = \frac{r \cdot h}{n}$, where r represents the number of i-bridges in the sample, h is the average degree and n is the number of nodes in the sample. As a consequence, the expected value is 0.189. By contrast, we experimentally found that the average number of i-bridges occurring in the neighborhood of an i-bridge is 8.275, showing a high biasing w.r.t. the random case.

On the basis of the above result, we consider the neighbors of a and we select those having a `me` edge. Let b be one of them and let (b, b') be the corresponding `me` edge. Thus, the neighbors of b' are promising nodes for the detection of `me` edges with a (see Figure 2).

We are ready to describe the algorithm that solve the node-seed crystallization problem. It uses the above heuristic observation for sub-problem $\mathbf{S_1}$ to find promising candidates and then uses Algorithm 1 presented earlier to solve sub-problem $\mathbf{S_2}$.

Our algorithm receives a node $a \in N$ and builds a set M' of candidate `me` edges, such that $M' \cap (E_f \cup E_m) = \emptyset$. Then, for each candidate pair $(a, c) \in M'$ it uses Algorithm 1 to detect missing `me` edges. The function P_1 implementing our approach is reported in Algorithm 4.

Synthetically, Algorithm 4 proceeds as follows. It receives as argument the starting node a. For each node $b \in \Gamma(a)$, we consider its `me` edges (b, b'), if any. For each node $c \in \Gamma(b')$, we consider (a, c)

ALGORITHM 3: *Top*

Input a, b: a node
Input k_i: an integer
Input t_h: an integer
Output L: a list of triplets $\langle a, b, s\rangle$
Variable t: a triplet $\langle a, b, s\rangle$
Variable c: an integer
Variable $\delta_t(a), \delta_t(b), \widetilde{Q}_{(a,b)}$: a real

$L := \emptyset$; $c := 0$
for each a' in $\Gamma(a)$ **do**
  for each b' in $\Gamma(b)$ **do**
    $\delta_t(a) := 1 - \frac{indegree(a)}{max(indegree(a), t_h)}$
    $\delta_t(b) := 1 - \frac{indegree(b)}{max(indegree(b), t_h)}$
    $\widetilde{Q}_{(a,b)} := min(\delta_t(a), \delta_t(b)) * Q(a', b')$
    if $(c < k_i)$ **then**
      insert the triplet $\langle a', b', \widetilde{Q}_{(a,b)}\rangle$ into L
      sort L in a descending order
      $c := c + 1$
    else
      $t := L.get(k_i - 1)$
      if $(\widetilde{Q}_{(a,b)} > t.s))$ **then**
        replace the triplet in the position $(k_i - 1)$ of L with the triplet $\langle a', b', \widetilde{Q}_{(a,b)}\rangle$
        sort L in a descending order
      end if
    end if
  end for
end for
return L

ALGORITHM 4: P_1

Input a: the starting node
Output M'': the set of detected `me` edges
Variable M': a set of candidate `me` edges

$M' := \emptyset$; $M'' := \emptyset$
for each *node* $b \in \Gamma(a)$ **do**
  for each *edge* $(b, b') \in E_m$ **do**
    for each *node* $c \in \Gamma(b')$ **do**
      if (*the edge* $(a, c) \notin E_m$) **then**
        insert the `me` edge (a, c) in M'
      end if
    end for
  end for
end for
for each *node pair* $(a, c) \in M'$ **do**
  if $(S2(a, c))$ **then**
    insert the `me` edge (a, c) in M''
  end if
end for
return M''

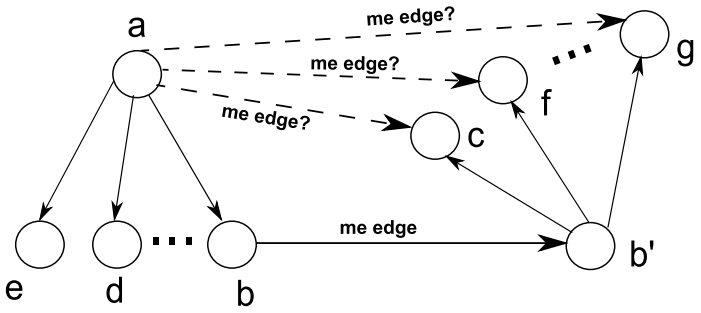

Figure 2: An example of the `me` edge detection approach.

as a candidate `me` edge. Then, for each candidate `me` edge (a, c) Algorithm 1 is used to verify whether this pair is a missing `me` edge or not. Finally, Algorithm 4 returns the set M'' of detected `me` edges.

4.2 Edge-seed crystallization problem

In this section we deal with the solution of problem $\mathbf{P_2}$. Let $(a, b) \in E_m$ a `me` edge in G. The goal is to find missing `me` edges in G starting from the seed (a, b). The solution of the sub-problem $\mathbf{S_2}$ is the same as problem $\mathbf{P_1}$. So, the reader may refer Algorithm 1 presented earlier. For the sub-problem $\mathbf{S_1}$ we use again the bridge-assortativity property described in the previous section. Recall that, according to this property, with a high probability, some of the nodes belonging to the neighbors of two nodes linked by a `me` edge are in turn linked by a `me` edge. In this case, we can apply the assortativity property in a direct way, by considering as promising candidate all the pairs (a', b') such that $a' \in \Gamma(a)$ and $b' \in \Gamma(b)$ and the edge $(a', b') \notin E_m$.

Observe that the assortativity property may lead us to consider as promising candidates every pair (a', x), for each $a' \in \Gamma(a)$, and every pair (b', y), for each $b' \in \Gamma(b)$, where x and y are found as shown in the solution of problem $\mathbf{P_1}$, thus reducing problem $\mathbf{P_2}$ on (a, b) to $|\Gamma(a)| + |\Gamma(b)|$ problems $\mathbf{P_1}$. Clearly, this may increase the recall of the solution, but is more costly in terms of efficiency. This issue is discussed in detail in Section 5.

The solution to problem $\mathbf{P_2}$ is implemented in Algorithm 5. It receives an existing `me edge` node (a, b) and builds a set M' of candidate `me` edges, such that $M' \cap (E_f \cup E_m) = \emptyset$. Then, for each candidate `me` edges (a', b') it uses Algorithm 1 to verify whether it is a missing `me` edge or not. Finally, it returns the set M'' of detected `me` edges.

5 Complexity issues

In this section, we analyze the computational complexity of our approach. We distinguish the operations that involve network accesses from those executed in memory, remarking that they differ in time

ALGORITHM 5: P_2

Input (a, b): the starting `me` edge
Output M'': the set of detected `me` edges
Variable M': a set of candidate `me` edges

```
1: M' := ∅; M'' := ∅
2: for each node pair (a', b') ∈ Γ(a) × Γ(b) do
3:     if (the edge (a', b') ∉ E_m) then
4:         insert the edge (a', b') in M'
5:     end if
6: end for
7: for each node pair (a', b') ∈ M' do
8:     if (S2(a', b')) then
9:         insert the me edge (a', b') in M''
10:    end if
11: end for
12: return M"
```

of (at least) nine orders of magnitude. We denote by $\mathcal{O}(f(n))$ I/Os, for a given function $f(n)$, the asymptotical evaluation of the number of operations involving network accesses vs the input variable n.

The first result concerns the computational complexity of the operator $T_Q^h(a, b, k_0)$. Intuitively, the number of operations performed by $T_Q^h(a, b, k_0)$ depends on k_0 because it represents an upper bound of the number of pairs that the operator selects at each step for the further computation. In other words, k_0 sets the maximum degree of the computation tree produced by the operator. Moreover, the number of operations performed by $T_Q^h(a, b, k_0)$ depends on the maximum degree of $S(a)$ and $S(b)$, denoted by d, because the operator, at the i-th step, visits the whole neighborhood of the nodes selected at the previous step. The dependance on k_0 and d of the complexity of $T_Q^h(a, b, k_0)$ is stated by the following theorem.

Theorem 5.1 *Given a pair of nodes $a, b \in N$ such that $S(a) \neq S(b)$, a string-similarity metric Q, an integer number $k_0 > 0$, and denoting by d the maximum degree of $S(a)$ and $S(b)$, it holds that: $T_Q^h(a, b, k_0)$ is $\mathcal{O}(k_0)$ I/Os $+\mathcal{O}(k_0 \cdot d^2)$ in the worst case.*

Proof. From Definition 3.5, clearly there exists an algorithm for the computation of $T_Q^h(a, b, k_0)$ proceedings as follows:

1. It visits the nodes a and b. This costs 2 I/Os.
2. It computes $Q(a', b')$ for each pair $(a', b') \in \Gamma(a) \times \Gamma(b)$. This costs d^2 in-memory operations (*in-memory* for short).
3. It selects the top k_0 node pairs among those of Step (2) above. This requires $(2 \cdot k_0)$ I/Os + d^2 in-memory.

4. It iterates the previous steps for h times. This costs:

$$\sum_{i=1}^{h} \frac{(2 \cdot k_0)}{(1+i)^{(1+i)}} I/Os + \sum_{i=1}^{h} \frac{d^2}{(1+i)^{(1+i)}} \text{in-memory}.$$

As a consequence, the overall cost for the computation of $T_Q^h(a, b, k_0)$ is:

$$\left(2 + 2 \cdot k_0 + \sum_{i=1}^{h} \frac{(2 \cdot k_0)}{(1+i)^{(1+i)}}\right) \text{I/Os} + \left(d^2 + d^2 \cdot k_0 + \sum_{i=1}^{h} \frac{(d^2 \cdot k_0)}{(1+i)^{(1+i)}}\right) \text{in-memory}.$$

Now, we have that:

$$\sum_{i=1}^{h} \frac{1}{(1+i)^{(1+i)}} = \sum_{i=1}^{h} \frac{1}{(1+i)} \cdot \frac{1}{(1+i)^i} < \sum_{i=1}^{h} \frac{1}{(1+i)^i}.$$

Recalling that, for $c > 0$,

$$\sum_{j=1}^{n} \frac{1}{(1+c)^j} = \left(1 + \frac{1}{c} \cdot (1 - \frac{1}{(1+c)^n})\right)$$

we have that:

$$\sum_{i=1}^{h} \frac{1}{(1+1)^i} = 1 + \frac{1}{1} \cdot (1 - \frac{1}{(1+1)^h}) = 2 - \frac{1}{2^{(h-1)}} < 2.$$

As a consequence, the overall cost for the computation of is:

$$T_Q^h(a, b, k_0) = \left(2 + 2 \cdot k_0 + \sum_{i=1}^{h} \frac{(2 \cdot k_0)}{(1+i)^{(1+i)}}\right) \text{I/Os} + \left(d^2 + d^2 \cdot k_0 + \sum_{i=1}^{h} \frac{(d^2 \cdot k_0)}{(1+i)^{(1+i)}}\right) \text{in-memory}$$

Thus,it holds that:

$$T_Q^h(a, b, k_0) < (2 + 2 \cdot k_0 + 2 \cdot k_0 \cdot 2)\text{I/Os} + (d^2 + k_0 \cdot d^2 + 2 \cdot k_0 \cdot d^2) = \mathcal{O}(k_0)\text{I/Os} + \mathcal{O}(k_0 \cdot d^2)\text{in-memory}.$$

As far as this result is concerned, we observe that the contribution $\mathcal{O}(k_0 \cdot d^2)$ to the overall complexity of the operator is negligible for realistic values of d (typically, ranging from 50 to 200) by taking into account that $\mathcal{O}(k_0 \cdot d^2)$ concerns in-memory operations. As a consequence, the cost of $T_Q^h(a, b, k_0)$ is in practice dominated by $\mathcal{O}(k_0)$ I/Os.

The next theorem determines the worst-case complexity of the operator $S_Q^\epsilon(a, b, k_0)$. Again, the analysis highlights the dependance of the computational cost on both k_0 and d, because $S_Q^\epsilon(a, b, k_0)$ is implemented by iterating the computation of $T_Q^h(a, b, k_0)$. Moreover, the complexity cost depends on ϵ which the number of iteration steps derives from.

Theorem 5.2 *Given a pair of nodes $a, b \in N$ such that $S(a) \neq S(b)$, an integer number $k_0 > 0$, a number ϵ in the real number $(0, 1]$, and denoting by d the maximum degree of the nodes of $S(a)$ and $S(b)$, it holds that: $S^{\epsilon}_{Q}(a, b, k_0)$ is $\mathcal{O}(k_0 \cdot ln(\frac{1}{\epsilon}))$ I/Os $+\mathcal{O}(k_0 \cdot d^2 \cdot ln(\frac{1}{\epsilon}))$ in the worst case.*

Proof. In Theorem 3.1, we proved that the maximum number of steps h necessary for the computation of $S^{\epsilon}_{Q}(a, b, k_0)$ is strictly linked to β_i. We recall that $\beta_i = \frac{1}{(i+1)^{(i+1)}}$ is a parameter tuning the contribution of the common-neighbors in the computation of the similarity between a and b. Specifically, in Theorem 3.1, we showed that $h > 0$ and h must be set in such a way that $\beta_h < \epsilon$.

Thus: $\beta_h < \epsilon \Leftrightarrow (1+h)^{(1+h)} > \frac{1}{\epsilon} \Leftrightarrow (1+h) > log_{(1+h)}\frac{1}{\epsilon} \Leftrightarrow 1 + h > ln(\frac{1}{\epsilon}) \cdot \frac{1}{ln(1+h)} \Leftrightarrow h > ln(\frac{1}{\epsilon}) \cdot \frac{1}{ln(1+h)} - 1$.

This inequality can be made stricter by requiring that $h > ln(\frac{1}{\epsilon}) \cdot \frac{1}{ln(1+h)}$. Now, recall that h is a discrete positive number. As a consequence, it can be either equal to 1 or higher than 1. In the latter case we have that $\frac{1}{ln(1+h)} < 1$; therefore, the inequality $h > ln(\frac{1}{\epsilon}) \cdot \frac{1}{ln(1+h)}$ can be made stricter by requiring that $h > ln(\frac{1}{\epsilon})$.

As a consequence, we have that the maximum number of steps necessary for the computation of $S^{\epsilon}_{Q}(a, b, k_0)$ is $max(1, ln(\frac{1}{\epsilon})) = ln(\frac{1}{\epsilon})$, being $0 < \epsilon \leq 1$. Theorem 5.1 states that the worst case complexity for each step is $\mathcal{O}(k_0)$ I/Os $+\mathcal{O}(k_0 \cdot d^2)$, then the proof is concluded. □

Clearly, by applying the reasonings seen for Theorem 5.1, we can say that the cost of $S^{\epsilon}_{Q}(a, b, k_0)$ is in practice dominated by $\mathcal{O}(k_0)$ I/Os.

The next theorem specifies the worst-case complexity of the sub-problem $\mathbf{S_2}$. Again, the analysis evidences the dependance of the computational cost on k_0, d, and ϵ, because $\mathbf{S_2}$ relies on $S^{\epsilon}_{Q}(a, b, k_0)$.

Theorem 5.3 *Given a pair of nodes $a, b \in N$ such that $S(a) \neq S(b)$, an integer number $k_0 > 0$, a number $\epsilon >$ in the real interval $(0, 1]$, and denoting by d the maximum degree of the nodes of $S(a)$ and $S(b)$, it holds that: $\mathbf{S_2}$ is $\mathcal{O}(k_0 \cdot ln(\frac{1}{\epsilon}))$ I/Os $+\mathcal{O}(k_0 \cdot d^2 \cdot ln(\frac{1}{\epsilon}))$ in the worst case.*

Proof. A direct algorithm solving $\mathbf{S_2}$ first verifies, in constant time, if the candidate pair (a, b) has a string similarity higher than a certain threshold. If this is not the case, the algorithm halts. Thus, we have to consider the other case, in which the algorithm computes $S^{\epsilon}_{Q}(a, b, k_0)$, which costs $\mathcal{O}(k_0 \cdot ln(\frac{1}{\epsilon}))$ I/Os $+\mathcal{O}(k_0 \cdot d^2 \cdot ln(\frac{1}{\epsilon}))$ in-memory operations according to Theorem 5.2. As a consequence, the worst-case complexity coincides with that of $S^{\epsilon}_{Q}(a, b, k_0)$. □

Now, we can compute the complexity of problem $\mathbf{P_1}$. This analysis highlights the dependance of this complexity on (1): k_0, d and ϵ, as $\mathbf{P_1}$ relies on $\mathbf{S_2}$, and (2): on the maximum number d_m of `me` edges for an i-bridge, because for each of these nodes, an algorithm solving $\mathbf{P_1}$ must visit all the nodes linked to it by means of a `me` edge.

Theorem 5.4 *Given a pair of nodes $a, b \in N$ such that $S(a) \neq S(b)$, an integer number $k_0 > 0$, a number ϵ in the real interval $(0, 1]$, and denoting by d the maximum degree of the nodes of $S(a)$ and $S(b)$, and by d_m the maximum number of* `me` *edges for an i-bridge, it holds that: $\mathbf{P_1}$ is $\mathcal{O}(k_0 \cdot d_m \cdot d^2 \cdot ln(\frac{1}{\epsilon}))$ I/Os $+\mathcal{O}(k_0 \cdot d_m \cdot d^4 \cdot ln(\frac{1}{\epsilon}))$ in the worst case.*

Proof. Clearly, there exists an algorithm solving $\mathbf{P_1}$ that proceeds as follows:

1. It visits a. This costs 1 I/O.
2. It visits the neighbors of a to verify if they have `me` edges. This requires d I/Os.
3. It visits the nodes linked to the neighbors of a by means of a `me` edge. These nodes are $d \cdot d_m$ and, then, their visits cost $d \cdot d_m$ I/Os.
4. For each neighbor of a of Step (3) above, it solves the sub-problem $\mathbf{S_2}$ to verify if a `me` edge exists between it and a. The number of neighbors of the nodes of Step (3) is $d^2 \cdot d_m$, whereas, from Theorem 5.3, the cost of the resolution of $\mathbf{S_2}$ is $\mathcal{O}(k_0 \cdot ln(\frac{1}{\epsilon}))$ I/Os $+\mathcal{O}(k_0 \cdot d^2 \cdot ln(\frac{1}{\epsilon}))$ in-memory operations. As a consequence, the overall cost of this step is: $\mathcal{O}(k_0 \cdot d^2 \cdot d_m \cdot ln(\frac{1}{\epsilon}))$ I/Os $+\mathcal{O}(k_0 \cdot d^4 \cdot d_m \cdot ln(\frac{1}{\epsilon}))$ in-memory operations.

Finally, the overall cost of $\mathbf{P_1}$ is 1 I/O$+d$ I/Os$+d \cdot d_m$ I/Os$+\mathcal{O}(ln(\frac{1}{\epsilon}) \cdot 2 \cdot k_0 \cdot d^2 \cdot d_m)$ I/Os $+\mathcal{O}(k_0 \cdot d^4 \cdot d_m \cdot ln(\frac{1}{\epsilon}))$ in-memory operations $= \mathcal{O}(k_0 \cdot d_m \cdot d^2 \cdot ln(\frac{1}{\epsilon}))$ I/Os $+\mathcal{O}(k_0 \cdot d_m \cdot d^4 \cdot ln(\frac{1}{\epsilon}))$ in-memory operations. □

Also in this case, by applying the reasonings seen for Theorem 5.1, we can say that in the reality d^4 in-memory operations cost much less than $2 \cdot d^2$ I/Os. As a consequence, the cost of problem $\mathbf{P_1}$ is in practice dominated by $\mathcal{O}(k_0 \cdot d_m \cdot d^2 \cdot ln(\frac{1}{\epsilon}))$ I/Os.

Now, the following theorem analyzes the worst case complexity of problem $\mathbf{P_2}$. This analysis evidences the dependance of the computational cost on k_0, d, and ϵ, because $\mathbf{P_2}$ uses $\mathbf{S_2}$.

Theorem 5.5 *Given a pair of nodes $a, b \in N$ such that $S(a) \neq S(b)$, an integer number $k_0 > 0$, a number ϵ in the real interval $(0, 1]$, and denoting by d the maximum degree of the nodes of $S(a)$ and $S(b)$, it holds that: $\mathbf{P_2}$ is $\mathcal{O}(k_0 \cdot d^2 \cdot ln(\frac{1}{\epsilon}))$ I/Os $+\mathcal{O}(k_0 \cdot d^4 \cdot ln(\frac{1}{\epsilon}))$ in the worst case.*

Proof. A direct algorithm solving $\mathbf{P_2}$ first visits a and b, and this costs 2 I/Os. Then, for each of the d^2 pairs $(a', b') \in \Gamma(a) \times \Gamma(b)$, it solves the sub-problem $\mathbf{S_2}$. From Theorem 5.3, this step costs $d^2 \cdot \mathcal{O}(k_0 \cdot ln(\frac{1}{\epsilon}))$ I/Os $+d^2 \cdot \mathcal{O}(k_0 \cdot d^2 \cdot ln(\frac{1}{\epsilon}))$ in-memory operations. As a consequence, the overall cost of $\mathbf{P_2}$ is: 2 I/Os$+d^2 \cdot \mathcal{O}(k_0 \cdot ln(\frac{1}{\epsilon}))$ I/Os $+d^2 \cdot \mathcal{O}(k_0 \cdot d^2 \cdot ln(\frac{1}{\epsilon}))$ in-memory operations $= \mathcal{O}(k_0 \cdot d^2 \cdot ln(\frac{1}{\epsilon}))$ I/Os $+\mathcal{O}(k_0 \cdot d^4 \cdot ln(\frac{1}{\epsilon}))$ in-memory operations. □

By applying the reasonings done for the previous theorems, the cost of problem $\mathbf{P_2}$ is in practice dominated by $\mathcal{O}(k_0 \cdot d^2 \cdot ln(\frac{1}{\epsilon}))$ I/Os.

Finally, we observe that if $\mathbf{P_2}$ is solved by reducing it to $(|\Gamma(a)| + |\Gamma(b)|)$ problems $\mathbf{P_1}$, its cost is $\mathcal{O}(k_0 \cdot d_m \cdot d^3 \cdot ln(\frac{1}{\epsilon}))$ I/Os that is $d \cdot d_m$ times the cost of the direct algorithm solving $\mathbf{P_2}$ described above.

6 Experiments

In this section, we present our experimental campaign aimed at determining the performances of our approach. Because it operates on a SIS, we had to extract not only the connections among the accounts of different users in the same social network but also the connections among the accounts of the same user in different social networks. To handle these connections, two standards encoding human relationships are generally exploited. The former is XFN (XHTML Friends Network) [27]. It simply uses an attribute, called `rel`, to specify the kind of relationship between two accounts. Possible values of `rel` are `me`, `friend`, `contact`, `co-worker`, `parent`, and so on. A (presumably) more complex alternative to XFN is FOAF (Friend-Of-A-Friend) [9]. In both of them information about `me` edges is that explicitly declared by users. To handle all the technicalities linked to this task, we leverage on SNAKE [20], a tool supporting the extraction of data from social network accounts. The SISs considered in our experiments are composed by several social networks (e.g., Twitter, LiveJournal, YouTube, Flickr, and Advocato), which are highly representative (they are among the top-10 social networks in terms of population).

6.1 Application of the state-of-the-art common-neighbor approaches

As described in Section 2, we have to prior verify whether common-neighbor approaches for link prediction [47] can be directly applied to our problem. However, a first aspect has to be considered. In our scenario, the notion of common neighbors cannot be the classical one, because the neighbors of examined pairs belong to different social networks. As a consequence, we cannot expect that two examined neighbors have common nodes in strict sense. To overcome this drawback we have just to re-define the notion of node identity. Coherently with our setting, it simply suffices to consider as *identical* two nodes linked by a `me` edge. At this point, classical common-neighbor techniques can be directly applied.

Given two nodes a and b in G such that $S(a) \neq S(b)$, the considered (state-of-the-art) techniques [47] are reported in Table 4. In the first and second columns of this table, we include the definition of the similarity index they rely on.

We tested all the above techniques in our SIS by preliminarily constructing a set M of 100 node pairs linked by a `me` edge and then by running them on M. For each technique, we obtained a set M' of detected `me` edges. Clearly, M' represents a set of true positives. Finally, we measured the *sensitivity* of the techniques as the ratio $\frac{|M'|}{|M|}$, obtaining the results reported in the third column of Table 4.

The analysis of such values clearly shows that no effective result can be obtained if common-neighbor techniques are adopted. This has motivated our further study, whose experimental validation is reported in the next sections.

Index Name	*Definition*	*Sensitivity*
Salton Index (SAI)	$s_{ab}^{SAI} = \frac{\lvert\Gamma(a)\bigcap\Gamma(b)\rvert}{\sqrt{\lvert\Gamma(a)\rvert\times\lvert\Gamma(b)\rvert}}$	0.01
Jaccard Index (JAI)	$s_{ab}^{JAI} = \frac{\lvert\Gamma(a)\bigcap\Gamma(b)\rvert}{\lvert\Gamma(a)\bigcap\Gamma(b)\rvert}$	0.01
Sorensen Index (SOI)	$s_{ab}^{SOI} = \frac{2\lvert\Gamma(a)\bigcap\Gamma(b)\rvert}{\lvert\Gamma(a)\rvert+\lvert\Gamma(b)\rvert}$	0.01
Hub Promoted Index (HPI)	$s_{ab}^{HPI} = \frac{\lvert\Gamma(a)\bigcap\Gamma(b)\rvert}{min(\lvert\Gamma(a)\rvert,\lvert\Gamma(b)\rvert)}$	0.00
Hub Depressed Index (HDI)	$s_{ab}^{HDI} = \frac{\lvert\Gamma(a)\bigcap\Gamma(b)\rvert}{max(\lvert\Gamma(a)\rvert,\lvert\Gamma(b)\rvert)}$	0.01
Leicht-Holme-Newman Index (LHNI)	$s_{ab}^{LHNI} = \frac{\lvert\Gamma(a)\bigcap\Gamma(b)\rvert}{\lvert\Gamma(a)\rvert\times\lvert\Gamma(b)\rvert}$	0.01
Resource Allocation Index (RA)	$s_{ab}^{RA} = \sum_{z\in\Gamma(a)\bigcap\Gamma(b)}\frac{1}{\lvert\Gamma(z)\rvert}$	0.01
Local Path Index (LPI)	$s_{ab}^{LPI} = A^2 + \epsilon A^3$ (A is the adjacency matrix of G)	0.03

Table 4: The tested common-neighbor approaches.

6.2 Similarity notion validation

A first experiment aims at validating our similarity notion that represents the basis of our approach. As a further result of this experiment, we investigate the effectiveness of the string similarity functions in our context. We started from the set M introduced in the previous section (i.e., a set of 100 real `me` edge-connected pairs). Then, we found another set, denoted by $\neg M$, of 100 node pairs not connected by a `me` edge. To find two elements, say (c_1, d_1) and (c_2, d_2), of $\neg M$, we started from a `me` edge-connected pair (a, b) and then we required that $c_1 = a$, $d_1 \in \Gamma(b)$, $c_2 = b$, $d_2 \in \Gamma(a)$. This way, both (c_1, d_1) and (c_2, d_2) are not linked by a `me` edge, because d_1 is a friend of a user (i.e., b) who is identical to c_1. The dual situation occurs for (c_2, d_2).

Then, we applied our approach on the pairs of M and $\neg M$ and we obtained the sets TP, FP, and FN, which are true positives, false positives, and false negatives, resp. For clarity, an element in TP is a pair of M detected as `me` edge-connected pair by our technique, an element in FP is a pair of $\neg M$ detected as `me` edge-connected pair by our technique, and an element of FN is a pair of M not detected as `me` edge-connected pair by our technique.

To compute the performance of our approach we adopted three classical measures, namely *precision* (as measure of correctness), *recall* (as measure of completeness) and *F-measure* (as the harmonic mean of precision and recall). They are defined as: $precision = \frac{|TP|}{|TP|+|FP|}$, $recall = \frac{|TP|}{|TP|+|FN|}$, and $F\text{-}measure = 2 \cdot \frac{precision \cdot recall}{precision+recall}$.

Because the behavior of our approach (and, consequently, the values of precision, recall and F-measure) depends on the function adopted for computing string similarity, we considered the most common of these functions and, for each of them, we computed precision, recall and F-measure. This way, we were able to determine the function(s) maximizing these measures. Obtained results are shown in Table 5.

The main conclusion we can draw from the analysis of this table is that our approach presents in general a very satisfying performance both in correctness and in completeness. Moreover, we observe that we are free to choose the string-similarity function in a rich set. Indeed, 5 functions led

Function	*Precision*	*Recall*	*F-measure*
Jaro-Winkler	0.558	0.920	0.694
QGrams	0.908	0.690	0.784
Levenshtein	0.877	0.710	0.785
Smith-Waterman	0.840	0.790	0.814
Smith-Waterman-Gotoh	0.779	0.810	0.794
Monge-Elkan	0.779	0.810	0.794
Needleman-Wunch	0.500	1.000	0.667
Jaro	0.555	0.910	0.689
Soundex	0.500	0.990	0.664

Table 5: Precision, recall and F-measure of our approach for each string similarity function.

our approach to obtain a precision higher than 0.77 and 6 functions led it to obtain a recall higher than 0.81. However, among the considered functions, QGrams (Needleman-Wunch, Smith-Waterman, resp.,) proved to be that capable of assuring the best precision (recall, F-measure, resp.). The high performance level of our approach is even more evident if we compare Tables 4 and 5 and if we consider that, in this testbed, the definitions of recall and sensitivity coincide and, consequently, the corresponding columns can be compared[3].

6.3 Experiments on problem $\mathbf{P_1}$

In this section, we validate our approach for the resolution of problem $\mathbf{P_1}$ (see Section 4.1). For this purpose, we consider a set M of already existing `me` edges. To derive M we applied BDS, a crawling technique specifically conceived to operate on a SIS, instead of on a single social network, which is highly capable of finding and returning *explicitly* declared `me` edges [19] (note that, to the best of our knowledge, no other technique with this feature is available in literature). For each edge (a, c) of M we give the node a as input to our algorithm and obtain a set M' of detected `me` edges having a as source node. For each edge $(a, \hat{c}) \in M'$, we test whether $\hat{c}$ coincides with c. In the affirmative case, we can conclude that our strategy has been able of correctly reconstructing the original `me` edge.

The starting set M consists of 100 explicitly declared `me` edges and we consider QGrams as string-similarity function, because it proved to assure the best precision (see Section 6.2). After running this experiment, we obtained 57 cases in which the original `me` edge has been reconstructed. This implies a percentage of success in detecting the original `me` edges equal to 57%. Observe that, at a first glance, it is possible to conclude that this percentage is not very high. However, to correctly evaluate this result, it is necessary to deepen the problem we are investigating. Indeed, given a node a, the nodes that should be examined for the possible presence of a `me` edge with a are $|N| - |S(a)|$ (i.e., the number of all the users of the SIS minus the number of the users of the social network of a). For instance, in a SIS comprising the most famous social networks (such as the SIS considered in our experiments), this

[3]Observe that, owing to the extremely low values of sensitivity, the computation of precision in Table 4 makes no sense.

number is higher than 10^9. This implies that, in 57% of cases, our approach is able to find the right node among a set of 10^9 candidates.

6.4 Experiments on problem $\mathbf{P_2}$

The experiment presented in this section aims at computing the accuracy of our approach for the resolution of problem $\mathbf{P_2}$ discussed in Section 4.2. In this case, we benefited from the support of a human expert. We first applied a crawling technique to derive a sample of the SIS. As in the previous experiment, to construct this sample, we applied BDS [16]. Our sample consisted of 93,169 nodes and 146,325 edges. 745 out of 146,325 were `me` edges. We randomly selected 160 `me` edges and put them in a set M. We gave this set as input to our technique. The adopted string-similarity function was QGrams, because it proved to assure the best precision. Our approach returned a set M' of 22 `me` edges and a set of 133 non-`me` edges, from which we randomly selected a set $\neg M'$ of 22 non-`me` edges in such a way that `me` edges and non-`me` edges had the same weight. After this, we asked the human expert to verify whether the elements of M' were actually `me` edges and the elements of $\neg M'$ were actually non-`me` edges. For each edge, the possible answers were *true*, *false* and *unknown*. Observe that the value *unknown* reflects both uncertain cases and unreachable-page ones. At the end of the experiment, we obtained that, as for M', the human expert returned $t_p = 16$ *true*, $f_p = 4$ *false* and 2 *unknown*. As for $\neg M'$, she returned $t_n = 18$ *true*, $f_n = 2$ *false* and 2 *unknown*. Finally, we computed the *accuracy* as the ratio $\frac{t_p+t_n}{t_p+t_f+t_n+f_n}$, obtaining the value 0.85, which denotes a very good performance of our method.

6.5 Running time

In this section, we evaluate the running time of our approach. We carry out our experiments on a server equipped with a 2 Quad-Core E5440 processor and 16 GB of RAM with the CentOS 6.0 Server operating system. Specifically, we measure the time necessary for operations that involve network accesses and in-memory operations. This allows us to quantify the value of the constants (I/Os and in-memory operations) of the computational complexity defined in Section 5.

As a first experiment, we consider the I/O time versus the degree of nodes. We define five degree bins according to the equi-depth binning strategy in such a way as to have approximately the same number of nodes in each of them. This partition of nodes on the basis of their degree generated the following five bins: (0..100], (100..250], (250..500], (500..750], (750..∞). Then, we visit each node and, for each bin, we average the measured running time. Obtained results are reported in Figure 3.

From the analysis, it is evident that the I/O time has an almost linear trend w.r.t. the degree of nodes.

As for the time of in-memory operations we know that typically they differ in time of (at least) nine orders of magnitude w.r.t. the I/O time. For this reason, rather than measuring the time of a single in-memory operation, we consider as an atomic operation the computation of the string

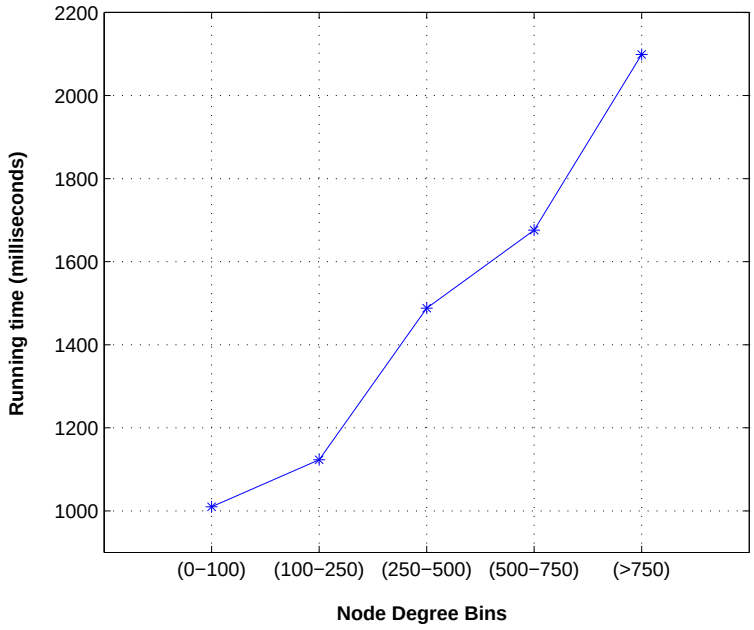

Figure 3: Average I/O time vs node degree.

similarity (by adopting the QGrams similarity function) between the usernames of two nodes (which is the most frequent operation that our approach performs) and we measure the time necessary for this computation. Specifically, we estimate the average time for the computation of the string similarity between the usernames of all the possible pair of nodes that we considered in our experiments. The average running time for this atomic operation is 0.009 which still remains negligible w.r.t. the I/O time.

The results of the experiments described in this section along with the computation complexity analysis described in Section 5 allow us to conclude that our system is scalable. This statement becomes even stronger if we consider that the discovery of missing `me` edge is a task that can be performed in background.

7 Conclusion and future work

In this paper, we studied the problem of discovering missing `me` edges in a Social Internetworking Scenario. The most evident information we can use to detect missing `me` edges, besides usernames, concerns neighbors. By means of experimental campaign, we showed that state-of-the-art common-neighbor techniques cannot be applied to solve this problem. Thus, the need of studying the problem as new and finding a specific solution arises. We defined a suitable notion of "inter-social-network" similarity that is recursive. Indeed, the common-neighbor notion has to necessarily rely on the same notion because neighbors belong to different social networks. On the basis of this notion, we defined

an algorithm to detect whether there is a missing `me` edge between two given nodes. The experimental analysis of this approach on a real-life data set showed its correctness and completeness.

The results obtained in this paper can be useful for further investigations in the context of multiple social networks. Indeed, the role of `me` edges is relevant for any phenomenon of information crossing through different social networks, so discovering new `me` edges may strongly enrich the analysis capabilities of social data, and strengthen multi-context analysis of people profiles. We believe that a number of future directions of our research can be undertaken. A first research line we plan to investigate is to apply our approach to privacy-aware solutions, in which user activities in a social network are driven (through simple notifications or recommendations) in such a way that the privacy requirement of a user of keeping separated two profiles can be autonomously controlled. Another extension we are considering is to improve the effectiveness of our identification technique by using also additional information such as behavioral contents (posts, messages, events, groups, etc.). A third direction of further investigation is to study whether the empirical observation adopted in this paper to smartly find good candidates is based on some theoretical property of social networks such as assortativity of membership overlap.

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] Google Social Graph. `http://code.google.com/p/itswhoyouknow/wiki/SocialGraph`, 2012.

[2] L. Aiello, A. Barrat, C. Cattuto, R. Schifanella, and G. Ruffo. Link creation and information spreading over social and communication ties in an interest-based online social network. *EPJ Data Science*, 1(1):1–31, 2012. Springer.

[3] M. Al Hasan and M. Zaki. A Survey of Link Prediction in Social Networks. In *Social Network Data Analytics*, pages 243–276. Elsevier, 2011.

[4] A. Barabási, H. Jeong, Z. Neda, E. Ravasz, A. Schubert, and T. Vicsek. Evolution of the social network of scientific collaborations. *Physica A: Statistical Mechanics and its Applications*, 311(3-4):590–614, 2002.

[5] S. Bartunov, A. Korshunov, S. Park, W. Ryu, and H. Lee. Joint link-attribute user identity resolution in online social networks. In *Proc. of the Workshop on Social Network Mining and Analysis (SNA-KDD'12)*, Beijing, China, 2012. ACM.

[6] F. Benevenuto, T. Rodrigues, M. Cha, and V. Almeida. Characterizing user navigation and interactions in online social networks. *Information Sciences*, 195:1–24, 2012.

[7] M. Berlingerio, M. Coscia, F. Giannotti, A. Monreale, and D. Pedreschi. Foundations of Multi-dimensional Network Analysis. In *Proc. of the International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2011)*, pages 485–489, Kaohsiung, Taiwan, 2011. IEEE.

[8] M. Berlingerio, M. Coscia, F. Giannotti, A. Monreale, and D. Pedreschi. Multidimensional networks: foundations of structural analysis. *World Wide Web*, 16(5-6):567–593, 2013. Springer.

[9] D. Brickley and L. Miller. The Friend of a Friend (FOAF) project. *http://www.foaf-project.org/*, 2013.

[10] M. Bröcheler, A. Pugliese, and V. S. Subrahmanian. Probabilistic Subgraph Matching on Huge Social Networks. In *Proc. of the International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2011)*, pages 271–278, Kaohsiung, Taiwan, 2011. IEEE.

[11] F. Buccafurri, V. Foti, G. Lax, A. Nocera, and D. Ursino. Bridge Analysis in a Social Internetworking Scenario. *Information Sciences*, 224:1–18, 2013. Elsevier.

[12] F. Buccafurri, G. Lax, S. Nicolazzo, and A. Nocera. A model to support multi-social-network applications. In *On the Move to Meaningful Internet Systems: OTM 2014 Conferences*, pages 639–656. Springer, 2014.

[13] F. Buccafurri, G. Lax, S. Nicolazzo, A. Nocera, and D. Ursino. Measuring Betweennes Centrality in Social Internetworking Scenarios. In *Proc. of International Workshop on Social and Mobile Computing for collaborative environments (SOMOCO'13)*, pages 666–673, Gratz, Austria, 2013. Springer Verlag.

[14] 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.

[15] F. Buccafurri, G. Lax, A. Nocera, and D. Ursino. Supporting Information Spread in a Social Internetworking Scenario. *Post-Proceedings of the International Workshop on New Frontiers in Mining Complex Knowledge Patterns at ECML/PKDD 2012 (NFMCP 2012)*, 200–214. Lecture Notes in Artificial Intelligence, Springer.

[16] F. Buccafurri, G. Lax, A. Nocera, and D. Ursino. Crawling Social Internetworking Systems. In *Proc. of the International Conference on Advances in Social Analysis and Mining (ASONAM 2012)*, pages 505–509, Istanbul, Turkey, 2012. IEEE Computer Society.

[17] F. Buccafurri, G. Lax, A. Nocera, and D. Ursino. Discovering Links among Social Networks. In *Proc. of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD 2012)*, pages 467–482, Bristol, United Kingdom, 2012. Lecture Notes in Computer Science. Springer.

[18] F. Buccafurri, G. Lax, A. Nocera, and D. Ursino. Internetworking assortativity in Facebook. In *Proc. of the International Conference on Social Computing and its Applications (SCA 2013)*, pages 335–341, Karlsruhe, Germany, 2013. IEEE Computer Society.

[19] F. Buccafurri, G. Lax, A. Nocera, and D. Ursino. Moving from social networks to social internetworking scenarios: The crawling perspective. *Information Sciences*, 256:126–137, 2014. Elsevier.

[20] F. Buccafurri, G. Lax, A. Nocera, and D. Ursino. A system for extracting structural information from social network accounts. *Software: Practice and Experience*, 2014. DOI: 10.1002/spe.2280.

[21] S. V. Buldyrev, R. Parshani, G. Paul, H. Stanley, and S. Havlin. Catastrophic cascade of failures in interdependent networks. *Nature*, 464(7291):1025–1028, 2010.

[22] X. Cai, M. Bain, A. Krzywicki, W. Wobcke, Y. Kim, P. Compton, and A. Mahidadia. Reciprocal and heterogeneous link prediction in social networks. In *Advances in Knowledge Discovery and Data Mining*, pages 193–204. Springer, 2012.

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

[24] A. Castiglione, G. Cattaneo, and A. De Santis. A forensic analysis of images on online social networks. In *Intelligent Networking and Collaborative Systems (INCoS), 2011 Third International Conference on*, pages 679–684. IEEE, 2011.

[25] A. Castiglione, B. D'Alessio, and A. De Santis. Steganography and secure communication on online social networks and online photo sharing. In *Broadband and Wireless Computing, Communication and Applications (BWCCA), 2011 International Conference on*, pages 363–368. IEEE, 2011.

[26] J. Caverlee, L. Liu, and S. Webb. The socialtrust framework for trusted social information management: Architecture and algorithms. *Information Sciences*, 180(1):95–112, 2010.

[27] T. Celik, E. Meyer, and M. Mullenweg. XHTML Friends Network. In *Proc. ACM Hypertext*, volume 4, 2004.

[28] M. Conti, A. Hasani, and B. Crispo. Virtual private social networks. In *Proceedings of the first ACM conference on Data and application security and privacy*, pages 39–50. ACM, 2011.

[29] I. Dagan, F. Pereira, and L. Lee. Similarity-based estimation of word cooccurrence probabilities. In *Proc. of the annual meeting on Association for Computational Linguistics*, pages 272–278. Association for Computational Linguistics, 1994.

[30] D. Davis, R. Lichtenwalter, and N. Chawla. Supervised methods for multi-relational link prediction. *Social Network Analysis and Mining*, 3(2):127–141, 2013.

[31] X. Ding, L. Zhang, Z. Wan, and I. Gu. De-Anonymization of Dynamic Social Networks. *Information Technology Journal*, 12(19):4882–4888, 2013. Asian Network for Scientific Information.

[32] Y. Dong, J. Tang, S. Wu, J. Tian, N. Chawla, J. Rao, and H. Cao. Link prediction and recommendation across heterogeneous social networks. In *Proc. of the International Conference on Data Mining (ICDM 2012)*, pages 181–190, Brussels, Belgium, 2012. IEEE Computer Society.

[33] D. Easley and J. Kleinberg. *Networks, crowds, and markets*, volume 8. Cambridge University Press, Cambridge, UK, 2010.

[34] A. Elmagarmid, P. Ipeirotis, and V. Verykios. Duplicate Record Detection: A Survey. *IEEE Transactions on Knowledge and Data Engineering*, 19(1):1–16, 2007.

[35] H. Fu, A. Zhang, and X. Xie. De-anonymizing social graphs via node similarity. In *Proc. of the companion publication of the International Conference on World Wide Web (WWW Companion '14)*, pages 263–264, Seoul, Korea, 2014. International World Wide Web Conferences Steering Committee.

[36] K. Gani, H. Hacid, and R. Skraba. Towards multiple identity detection in social networks. In *Proc. of the International Conference Companion on World Wide Web (WWW 2012 Companion)*, pages 503–504, Lyon, France, 2012. ACM.

[37] N. Gong, A. Talwalkar, L. Mackey, L. Huang, E. Shin, E. Stefanov, E. Shi, and D. Song. Joint link prediction and attribute inference using a social-attribute network. *ACM Transactions on Intelligent Systems and Technology*, 5(2):27, 2014.

[38] P. Govindan, S. Soundarajan, and T. Eliassi-Rad. Finding the Most Appropriate Auxiliary Data for Social Graph Deanonymization. In *Proc. of the KDD14 Data Ethics Workshop*, New York, NY, USA, 2014. ACM.

[39] Y. He, J. Liu, Y. Hu, and X. Wang. OWA Operator Based Link Prediction Ensemble for Social Network. *Expert Systems with Applications*, 42(1):21–50, 2015. Elsevier.

[40] C. Howden, L. Liu, Z. Ding, Y. Zhan, and K. Lam. Moments in time: A forensic view of twitter. In *Green Computing and Communications (GreenCom), 2013 IEEE and Internet of Things (iThings/CPSCom), IEEE International Conference on and IEEE Cyber, Physical and Social Computing*, pages 899–908. IEEE, 2013.

[41] L. Huang, R. Li, K. Wen, and X. Gu. A Self Training Semi-Supervised Truncated Kernel Projection Machine for Link Prediction. *Advanced Materials Research*, 580:369–373, 2012. Trans Tech Publications Inc.

[42] 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.

[43] M. Korayem and J. Crandall. De-Anonymizing Users Across Heterogeneous Social Computing Platforms. In *Proc. of the International AAAI Conference on Weblogs and Social Media (ICWSM'13)*, pages 689–692, Boston, MA, USA, 2013. Association for the Advancement of Artificial Intelligence.

[44] N. Korula and S. Lattanzi. An efficient reconciliation algorithm for social networks. In *Proc. of the International Conference on Very Large Data Bases (VLDB'14)*, pages 377–388, Hangzhou, Cina, 2014. VLDB Endowment.

[45] K. Lerman, S. Intagorn, J. Kang, and R. Ghosh. Using proximity to predict activity in social networks. In *Proc. of the International Conference Companion on World Wide Web (WWW '12 Companion)*, pages 555–556, New York, NY, USA, 2012. ACM.

[46] D. Liben-Nowell and J. Kleinberg. The link-prediction problem for social networks. *Journal of the American society for information science and technology*, 58(7):1019–1031, 2007.

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

[48] A. Malhotra, L. Totti, W. Meira Jr, P. Kumaraguru, and V. Almeida. Studying user footprints in different online social networks. In *Proc. of the International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2012), year = 2012, address = Istanbul, Turkey pages=1065–1070, publisher=IEEE Computer Society.*

[49] A. Narayanan, E. Shi, and B. I. Rubinstein. Link prediction by de-anonymization: How we won the kaggle social network challenge. In *Proc. of the International Joint Conference on Neural NetworksNeural Networks (IJCNN 2011)*, pages 1825–1834, San Jose, California, USA, 2011. IEEE.

[50] 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.

[51] M. Newman. Assortative mixing in networks. *Physical Review Letters*, 89(20):208701, 2002.

[52] 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. Elsevier.

[53] P. Pedersani and M. Grossglauser. On the privacy of anonymized networks. In *Proc. of the International Conference on Knowledge Discovery and Data Mining (KDD 2011)*, pages 1235–1243, San Diego, CA, USA, 2011. ACM.

[54] A. Popescul and L. Ungar. Statistical relational learning for link prediction. In *Proc. of the International Workshop on Learning Statistical Models from Relational Data*, volume 149, pages 172–179, Acapulco, Mexico, 2003.

[55] P. Sarkar, D. Chakrabarti, and M. Jordan. Nonparametric Link Prediction in Dynamic Networks. In *Proc. of the International Conference on Machine Learning (ICML 2012)*, Edinburgh, Scotland, UK, 2012. Omnipress.

[56] K. Sharad and G. Danezis. An Automated Social Graph De-anonymization Technique. *arXiv preprint arXiv:1408.1276*, 2014.

[57] S. Soundarajan and J. Hopcroft. Using community information to improve the precision of link prediction methods. In *Proc. of the International Conference Companion on World Wide Web (WWW'12 Companion)*, pages 607–608, New York, NY, USA, 2012. ACM.

[58] S. Spiegel, J. Clausen, S. Albayrak, and J. Kunegis. Link prediction on evolving data using tensor factorization. In *New Frontiers in Applied Data Mining*, pages 100–110. Springer, 2012.

[59] Y. Sun, J. Han, C. Aggarwal, and N. Chawla. When will it happen?: relationship prediction in heterogeneous information networks. In *Proc. of the ACM International Conference on Web Search and Data Mining*, pages 663–672, Seattle, WA, USA, 2012. ACM.

[60] 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.

[61] X. Wang, F. Wei, X. Liu, M. Zhou, and M. Zhang. Topic Sentiment Analysis in Twitter: A Graph-based Hashtag Sentiment Classification Approach. In *Proc. of the ACM International Conference on Information and Knowledge Management (CIKM 2011)*, pages 1031–1040, Glasgow, Scotland, UK, 2011. ACM.

[62] W. Wei, F. Xu, C. Tan, and W. Li. Sybildefender: Defend against sybil attacks in large social networks. In *Proc. of the International Conference on Computer Communications (INFOCOM 2012)*, pages 1951–1959, Orlando, FL, USA, 2012. IEEE Computer Society.

[63] Y. Yang, J. Lutes, F. Li, B. Luo, and P. Liu. Stalking online: on user privacy in social networks. In *Proc. of the ACM International Conference on Data and Application Security and Privacy (CODASPY'12)*, pages 37–48, San Antonio, TX, USA, 2012. ACM.

[64] 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.

[65] R. Zafarani and H. Liu. Connecting users across social media sites: a behavioral-modeling approach. In *Proc. of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 41–49, Chicago, IL, USA, 2013.