Soc. Netw. Anal. Min. (2014) 4:154
DOI 10.1007/s13278-014-0154-z

ORIGINAL ARTICLE

Interactive discovery of influential friends from social networks

Carson Kai-Sang Leung · Syed K. Tanbeer · Juan J. Cameron

Received: 4 June 2013 / Accepted: 22 October 2013 / Published online: 12 February 2014

Abstract Social networks, which are made of social entities (e.g., individual users) linked by some specific types of interdependencies such as friendship, have become popular to facilitate collaboration and knowledge sharing among users. Such interactions or interdependencies can be dependent on or influenced by user characteristics such as connectivity, centrality, weight, importance, and activity in the networks. As such, some users in the social networks can be considered as highly influential to others. In this article, we propose a computational model that integrates data mining with social computing to help users discover influential friends from a specific portion of the social networks that they are interested in. Moreover, our social network analysis and mining model also allows users to interactively change their mining parameters (e.g., scopes of their interested portions of the social networks).

Keywords Applications on social networks · Computational aspects of social networks · Influential person · Knowledge discovery and data mining · Social network analysis and mining · Social network computing

1 Introduction and related work

Recent advancements in internet technology and social awareness have increased the popularity and momentum for the social network analysis in many different types of social networks (Ding et al. 2010 ; Leung and Carmichael 2010; Horak et al. 2011; Bhagat et al. 2012; Lee et al. 2012) such as co-authorship, friendship, professional and organizational networks. Social networking sites such as Facebook, Google+, LinkedIn, MySpace, Twitter and Weibo (Fan and Yeung 2010; Nasirifard and Hayes 2011; Lin et al. 2012; Yang et al. 2012) are providing valuable social information about their entities (e.g., individuals, corporations, collective social units, or organizations), contacts and their relationships (e.g., kinship, friendship, common interest, beliefs, or financial exchange). For instance, a social network user p can create a personal profile in Facebook, add other Facebook users as friends, and exchange messages with these friends. Similarly, p can connect with friends and family members (e.g., find and connect with people from p's high school or hometown) in Google+, categorize his friends into different circles, and share ideas and thoughts with other Google+ users. Moreover, p can also create a professional profile in LinkedIn, establish connections to other LinkedIn users, endorse the skills of his connections, and be recommended by someone in his contact network. There are many users in these social networks, and each user may have different number of users connected/linked to his network. Among all users in these social networks, some can (i) have a strong relationship or (ii) be very significant or important to others depending on different parameters such as the number of interactions made between them.

Over the past few years, social computing and its applications have become an emerging research area in the field of computer science. Data mining techniques have been applied to social computing to extract implicit, previously unknown, and potentially useful information or interesting knowledge from social networks (Alsaleh et al. 2011; Baatarjav and Dantu 2011; Ferreira et al. 2012). Examples include clustering and classification of tweets, mining and analysis of co-authorship networks,

C. K.-S. Leung (✉) · S. K. Tanbeer · J. J. Cameron
Department of Computer Science, University of Manitoba, Winnipeg, MB R3T 2N2, Canada
e-mail: kleung@cs.umanitoba.ca

visualization of social networks (Leung and Carmichael 2010; Górecki et al. 2011), prediction of social links (Liaghat et al. 2013), and mining opinions (Muhammad et al. 2013). The current article, on the other hand, focuses on a different but also important aspect—namely, pattern mining on social networks.

In recent years, there have been a few publications on pattern mining from social networks. Examples include the mining of strong friends (Cameron et al. 2011; Leung et al. 2013) and significant friends (Leung and Tanbeer 2012; Tanbeer et al. 2013). The mining of many of these types of friends depends on the number of messages posted by users in social networking sites such as Facebook. However, there are situations in which one may want to find friends based on their relevant information (e.g., status of a friend in a social network) other than the number of messages or wall postings. For instance, a Facebook user may want to identify those prominent friends who have high impact (e.g., in terms of knowledge or expertise about a subject matter) in the social network. As another example, a LinkedIn user may want to get introduced to those second-degree connections who have rich experience in some profession. Similarly, a Twitter user may also be interested in following (and subscribing to a Twitter feed from) those who are highly influential in the whole network. Furthermore, finding influential friends from social networks may also help corporations and business organizations in making important business decisions. For example, when promoting and marketing a product to potential customers in a social network, it may be helpful to consider influential people (i.e., person having high impact) in a social network because products advertised to these people are likely to reach a large target group (e.g., his followers). Hence, it is desirable to discover influential friends from the social network.

A key contribution of the current article is our integration of data mining with social computing to build a social network mining model—called DIFSoN—for discovering the group of influential friends from a large volume of social network data. Here, we focus on the social entities who have high impact and/or are linked to many other entities in the network. The DIFSoN uses (i) a prefix-tree based data structure called Influential Friend tree (IF-tree) to effectively capture the social network data and (ii) a mining routine to efficiently discover the set of influential friend groups from the IF-tree.

Moreover, it is not unusual that users are interested in exploring some portions of their social networks to find groups of influential friends. Another key contribution of the current article is our extension of DIFSoN for handling efficient interactive mining of influential friends when users change the mining parameters (e.g., change the scope of their interested portions of social networks). In other words, the current article deals with interactive mining. In contrast, the recent publications on mining popular friends (Jiang et al. 2012), significant friends (Leung and Tanbeer 2012; Tanbeer et al. 2013), and diverse friends (Tanbeer and Leung 2013) have not yet handled the interactive mining.

As the current article is an expanded version of our CASoN 2012 paper (Tanbeer et al. 2012), additional novel contributions beyond the basic DIFSoN model for mining influential friends from social networks includes (i) an enhancement to the basic DIFSoN model to speed up the mining process by reducing the number of generated candidates, (ii) an extension to handle the interactive broadening of users' interested portions of social networks, (iii) an extension to handle the interactive narrowing of users' interested portions of social networks, (iv) additional theoretical results (e.g., proofs to lemmas), as well as (v) additional experimental results on the basic, enhanced and extended models for efficient discovery of influential friends from an interesting focused portion of social networks and when the scope of the focused portion of social networks is broadened and/or narrowed.

The remainder of the current article is organized as follows. We introduce the concept of influential friends in the next section. In Sect. 3, we propose the IF-tree structure and the corresponding mining routine, which first mines potentially influential friends from the IF-tree and then verifies if they are truly influential. Section 4 discusses how to speed up the mining process by bringing the number of potentially influential friends closer to that of the truly influential ones. Section 5 explains how we discover influential friends when the scope of the focused portion of social networks is broadened and/or narrowed. Section 6 shows the experimental results. Finally, conclusions are presented in Sect. 7.

2 Notion of influential friends

In this section, we present the basic definitions and notations for discovering influential friends from social networks. Recall that social networks are made of social entities (e.g., individual users) linked by some specific types of interdependencies such as friendship. Let us consider a sample portion of a social network as shown in Table 1, which consists of $n = 7$ lists of friends (i.e., friend lists L_1–L_7). Each friend list L_j records all the friends of an individual social entity. For example, L_1 records that *Ana*, *Beto*, *Carlos* and *Eva* are friends of some individual social entity (say, *Davi*). L_2 records that *Ana*, *Beto* and *Carlos* are friends of another individual social entity (say, *Eva*).

Note that we define the friend lists in an unrestrictive way such that the lists can be—but do not need to be—symmetric. For applications such as capturing mutual

Table 1 A collection LC of friend lists

List ID	Friend list (L)
L_1	{*Ana*, *Beto*, *Carlos*, *Eva*}
L_2	{*Ana*, *Beto*, *Carlos*}
L_3	{*Beto*, *Eva*, *Fabio*}
L_4	{*Ana*, *Beto*, *Davi*}
L_5	{*Ana*, *Beto*, *Carlos*, *Eva*}
L_6	{*Beto*, *Eva*, *Fabio*}
L_7	{*Ana*, *Beto*, *Carlos*, *Eva*}

Table 2 Prominence of friends in a social network

Friend (f_i)	Prominence (Prom(f_i))
Ana	0.60
Beto	0.50
Carlos	0.40
Davi	0.70
Eva	0.42
Fabio	0.57
Gil	0.11

friends in social networks like Facebook, the friend lists are symmetric. For instance, if *Ana* is a friend of *Beto*, then *Beto* is a friend of *Ana*. In other words, as *Ana* and *Beto* are mutual friends, they are on each other's list (i.e., *Ana* is on *Beto*'s friend list and *Beto* is on *Ana*'s friend list). However, for applications such as capturing publish–"subscribe" relationships in Facebook or the "follow" relationships in Twitter, the friend lists are not necessarily symmetric. For instance, if *Ana* follows *Beto*, then *Beto* is on *Ana*'s friend list or watch list but *Ana* may not be on *Beto*'s list unless *Ana* is also followed by *Beto*.

Moreover, our notion of friend lists is not confined to the lists of friends of individual social entities. It can be defined as the lists of friends in interest-groups. For instance, Table 1 may consist of $n = 7$ interest-group lists (L_1–L_7). Each list L_j captures all individual social entities who are connected as friends due to some common interests. For example, L_1 records members of a common interest group (e.g., on social computing)—namely, *Ana*, *Beto*, *Carlos* and *Eva*. L_2 records members of another common interest group (e.g., on data mining)—namely, *Ana*, *Beto* and *Carlos*.

More formally, given a set $F = \{f_1, f_2, \ldots, f_m\}$ of m users in a social network media, a friend list (i.e., a list of friends of an individual user or a list of friends in a common interest group) $L_j \subseteq F$ of a person $p \in F$ contains all social network users who are connected with p as friends. Let $G = \{f_1, f_2, \ldots, f_k\} \subseteq F$ be a group of friends (or a friend group) with k friends. The size size(G) of G indicates the number of friends in G (e.g., size(G) = k). Usually, users are interested in some portions of the social network. These user interested portions are represented by a list collection LC of friend lists (e.g., $\{L_1, L_2, \ldots, L_n\}$). The projected list of G (denoted as LC^G) is the set of friend lists in LC that contain the group G. We use Freq(G, LC) to represent the frequency of G (i.e., number of lists containing or supporting G) in LC.

Example 1 Consider the LC shown in Table 1, which consists of $n = 7$ friend lists for the $m = 7$ friends in Table 2. For group $G = \{Ana, Carlos\}$, its size is 2 (i.e., size(G) = $|\{Ana, Carlos\}| = 2$), and its projected list $\text{LC}^G = \{L_1, L_2, L_5, L_7\}$ with frequency Freq(G, LC) of 4 (i.e., {*Ana*, *Carlos*} appears in four lists—namely, L_1, L_2, L_5 & L_7—out of the $n = 7$ lists in LC).

Similarly, if group $G = \{Ana, Carlos, Eva\}$, then (i) size(G) = $|\{Ana, Carlos, Eva\}| = 3$ and (ii) $\text{LC}^G = \{L_1, L_5, L_7\}$ implying that Freq(G, LC) = 3.

Recall that Table 1 shows a collection LC of $n = 7$ lists. These lists involve $m = 7$ social entities (i.e., *Ana*, *Beto, Carlos, Davi, Eva, Fabio* and *Gil*). Table 2 shows the individual prominence of these $m = 7$ social entities. Here, we assume that the conflict of same name for different users (if any) has been resolved using a unique identification scheme. The prominence, which is represented by a non-negative number, indicates the status (such as importance, weight, value, reputation, belief, position, or significance) of a friend in a social network. The prominence values can be measured using a common scale, which could be user-defined or automatically calculated based on user-centric parameters (e.g., connectivity, centrality, expertise in the domain of the network, membership duration in the network, and the activity in the network). In this article, we consider the prominence in a scale between 0.0 and 1.0. For example, the prominence of *Ana* (denoted as Prom(*Ana*), which equals to 0.60) is higher than Prom(*Beto*) = 0.50, which implies that *Ana* is a more influential friend than *Beto* in this social network.

Definition 1 The prominence Prom(G) of a group G measures the average of all prominence values for all friends in the group:

$$\text{Prom}(G) = \frac{\sum_{i=1}^{\text{size(G)}} \text{Prom}(f_i)}{\text{size}(G)}. \quad (1)$$

Example 2 Consider the prominence table shown in Table 2. The prominence of friend group {*Ana*, *Carlos*} can be computed as $\frac{\text{Prom}(Ana)+\text{Prom}(Carlos)}{2} = \frac{0.60+0.40}{2} = 0.5$. Similarly, the prominence of friend group {*Ana*, *Carlos*, *Eva*} is $\frac{0.60+0.40+0.42}{3} \approx 0.473$.

Definition 2 The influence Inf(G, LC) of a group G in a collection LC of friend lists measures an aggregated prominence degree of G in all friend lists in LC. It is defined as the product of the prominence value of G and its appearance frequency in LC [i.e., Freq(G, LC)]:

$$\text{Inf}(G, \text{LC}) = \text{Prom}(G) \times \text{Freq}(G, \text{LC}). \quad (2)$$

Example 3 Recall from (i) Example 2 that Prom({*Ana*, *Carlos*}) = 0.5 and (ii) Example 1 that Freq({*Ana*, *Carlos*}, LC) = 4. So, the overall social network influence of {*Ana*, *Carlos*} can be calculated as Inf({*Ana*, *Carlos*}, LC) = Prom({*Ana*, *Carlos*}) × Freq({*Ana*, *Carlos*}, LC) = 0.5 × 4 = 2.0.

Similarly, recall from Example 2 that Prom({*Ana*, *Carlos*, *Eva*}) ≈ 0.473. Then, we observe from Table 1 that {*Ana*, *Carlos*, *Eva*} appears together in L_1, L_5 and L_7. So, Freq({*Ana*, *Carlos*, *Eva*}, LC) = 3. Hence, we compute Inf({*Ana*, *Carlos*, *Eva*}, LC) = 0.473 × 3 = 1.42.

A group of friends is considered influential in a social network media if its influence in the user-interested portion LC of the social network is no less than a user-specified minimum influence threshold *minInf*, which is a non-negative real number. The influence threshold can also be expressed as a percentage in terms of the number of lists in LC.

Definition 3 Given a user-specified minimum social network influence threshold *minInf*, a group G is considered influential in a collection LC of friend lists if its influence value is at least *minInf*:

$$\text{Inf}(G, \text{LC}) \geq minInf. \quad (3)$$

Example 4 Recall from Example 3 that (i) Inf({*Ana*, *Carlos*}, LC) = 2.0 and (ii) Inf({*Ana*, *Carlos*, *Eva*}, LC) = 1.42. If the user-specified *minInf* = 2.0, then group {*Ana*, *Carlos*} is influential in LC as shown in Table 1 because Inf({*Ana*, *Carlos*}, LC) = 2.0 ≥ 2.0 = *minInf*. But, group {*Ana*, *Carlos*, *Eva*} is *not* influential because Inf({*Ana*, *Carlos*, *Eva*}, LC) = 1.42 < 2.0 = *minInf*. In other words, based on the influence of aggregated weights and links with other friends in the network, friend group {*Ana*, *Carlos*} plays an important (or influential) role in the network, but the group {*Ana*, *Carlos*, *Eva*} does not.

Example 5 Recall from Table 2 that Prom({*Ana*}) = 0.60. As {*Ana*} appears in five friend lists (namely, L_1, L_2, L_4, L_5 and L_7), Freq({*Ana*}, LC) = 5. Thus, Inf({*Ana*}, LC) = 0.60 × 5 = 3.0.

Similarly, recall from Table 2 Prom({*Carlos*}) = 0.40. As {*Carlos*} appears in four friend lists (namely, L_1, L_2, L_5 and L_7), Freq({*Carlos*},LC) = 4. Thus, Inf({*Carlos*}, LC) = 0.40 × 4 = 1.6.

With the same *minInf* = 2.0, {*Ana*} is an influential friend group, but {*Carlos*} is not. However, as observed from Example 4, their superset {*Ana*, *Carlos*} (with Inf({*Ana*, *Carlos*}, LC) = 2.0) is an influential friend group. In other words, the group influence does not satisfy the downward closure property.

3 The DIFSoN model for mining influential friends

When mining frequent patterns, the frequency measure (Agrawal and Srikant 1994; Han et al. 2000) satisfies the downward closure property: if a pattern is frequent, then all its subsets are also frequent. Equivalently, if a pattern is infrequent, then all its supersets are also infrequent. Knowing that the frequency measure satisfies the downward closure property helps reduce the search/solution space by pruning infrequent patterns, and thus speeds up the mining process. However, when mining influential friend groups, one may observe from Example 5 that influence does not satisfy the downward closure property. For instance, a group (e.g., {*Carlos*}) is not influential, but its superset (e.g., {*Ana*, *Carlos*}) can be influential. Hence, the mining of influential friend groups can be challenging.

To handle the challenge, we calculate the upper bound $\text{Inf}^{\text{UB}}(G, \text{LC})$ to the influence of a group G in a collection LC of friend lists by multiplying its frequency with the highest prominence value available in the prominence table. We denote the highest prominence value among all the social entities in LC (in the prominence table) as global maximum prominence value Prom*G*Max(LC):

$$\text{Inf}^{\text{UB}}(G, \text{LC}) = \text{Prom}G\text{Max}(\text{LC}) \times \text{Freq}(G, \text{LC}) \quad (4)$$

where

$$\text{Prom}G\text{Max}(\text{LC}) = \max\{\text{Prom}(f_i) \mid f_i \text{ in LC}\} \quad (5)$$

and

$$\text{Inf}(G, \text{LC}) \leq \text{Inf}^{\text{UB}}(G, \text{LC}). \quad (6)$$

Equation (4)—on the upper bound $\text{Inf}^{\text{UB}}(G, \text{LC})$ to the influence of a group G in a collection LC of friend lists—leads to the following lemma.

Lemma 1 The upper bound $\text{Inf}^{\text{UB}}(G, \text{LC})$ to the influence satisfies the downward closure property.

Proof As the frequency measure satisfies the downward closure property,

$$\text{Freq}(G', \text{LC}) \leq \text{Freq}(G, \text{LC}) \text{ where } G \subseteq G' \quad (7)$$

for any given collection LC of friend lists. Based on Eq. (4) that $\text{Inf}^{\text{UB}}(G, \text{LC}) = \text{Prom}G\text{Max}(\text{LC}) \times \text{Freq}(G, \text{LC})$, we get

$$\text{Inf}^{\text{UB}}(G', \text{LC}) \leq \text{Inf}^{\text{UB}}(G, \text{LC}) \text{ where } G \subseteq G'. \quad (8)$$

If the upper bound $\text{Inf}^{\text{UB}}(G, \text{LC}) < minInf$ (i.e., a friend group G is uninfluential in LC), then $\text{Inf}^{\text{UB}}(G', \text{LC}) < minInf$ (i.e., every super-group G' of G is also uninfluential in LC) because

$$\text{Inf}^{\text{UB}}(G', \text{LC}) \leq \text{Inf}^{\text{UB}}(G, \text{LC}) < minInf.$$

Hence, the upper bound $\text{Inf}^{\text{UB}}(G, \text{LC})$ to the influence satisfies the downward closure property.

Knowing that the upper bound $\text{Inf}^{\text{UB}}(G, \text{LC})$ to the influence satisfies the downward closure property helps reduce the search/solution space by pruning uninfluential friend groups, and thus speeds up the mining process.

Example 6 In Table 2, Prom$G$Max(LC) = 0.70, which is Prom(*Davi*). Either recall from Example 5 or read directly from Table 1 that Freq($\{Ana\}$, LC) = 5 and Freq($\{Carlos\}$, LC)=4; recall from Example 3 or read directly from Table 1 that Freq ($\{Ana, Carlos\}$, LC) = 4. Then, we compute upper bounds to influence of three groups: (i) $\text{Inf}^{\text{UB}}(\{Ana\}, \text{LC})$ = Prom$G$Max(LC) × Freq($\{Ana\}$, LC) = 0.70 × 5 = 3.5, (ii) $\text{Inf}^{\text{UB}}(\{Carlos\}, \text{LC})$ = Prom$G$Max(LC) × Freq($\{Carlos\}$, LC) = 0.70 × 4 = 2.8, and (iii) $\text{Inf}^{\text{UB}}(\{Ana, Carlos\}, \text{LC})$ = Prom$G$Max(LC) × Freq($\{Ana, Carlos\}$, LC) = 0.70 × 4 = 2.8. With user-specified threshold *minInf*=2.0, we do not prune the group $\{Carlos\}$ at the early stage. Otherwise, we could have missed $\{Ana, Carlos\}$.

At the final stage, after finding all potentially influential friend groups, all such overestimated groups (e.g., $\{Carlos\}$) will be pruned by calculating their actual influence (using the prominence table and the frequency obtained from the mining phase).

In the remaining part of this section, we present our proposed DIFSoN model, which consists of two key components: (i) a memory-effective prefix tree structure to capture the important contents of the collection LC of friend lists representing the user-interested portion of the social network and (ii) an efficient routine to mine influential friend groups from the tree.

3.1 An IF-tree

A key component of our proposed DIFSoN model is a tree structure—called Influential Friend tree (IF-tree)—which captures the complete information from the collection LC of friend lists with only one scan of LC. To construct an IF-tree, we scan the friend lists in LC one-by-one and insert each list into the IF-tree in a fixed predefined friend order. Because influential friend groups will be identified based on the global maximum prominence value, we use the prominence value in ascending order for friends in our IF-tree.

The basic construction process of an IF-tree is similar, but not identical, to that of an FP-tree (Han et al. 2000). There are several differences between the two trees. First, the FP-tree captures only the frequent itemsets and is constructed with two database scans, while our IF-tree captures the complete collection LC of friend lists with a single scan. Second, each node in an IF-tree maintains the friend information and the appearance frequency in the respective path. We use the following example to demonstrate the IF-tree construction process.

Example 7 Let us show how to construct an IF-tree for the collection LC of friend lists shown in Table 1 with $minInf = 2.0$. At the first stage of constructing the IF-tree, a header table is built with the information available in the prominence table (i.e., Table 2). This table includes all the friends in LC according to their prominence value in ascending order. Thus, we obtain the header table order as $\langle Gil, Carlos, Eva, Beto, Fabio, Ana, Davi \rangle$.

Next, we scan each list, sort it according to the header table order, and insert it into the IF-tree. As such, the first list (i.e., L_1) in LC is inserted in $\langle Carlos, Eva, Beto, Ana \rangle$ order. Similar to an FP-tree, the IF-tree also maintains the horizontal node traversal pointers from the header table so as to facilitate a fast tree traversal. The status of the IF-tree after capturing L_1 is shown in Fig. 1a, where the header table includes the frequency (Freq) and the first pointer (Ptr) information for each friend f_i.

Figure 1b shows the contents of the header table, horizontal node traversal pointers, as well as the IF-tree structure after inserting L_2. Note that L_1 and L_2 share a common prefix *Carlos*. Hence, the common prefix part of L_2 (i.e., *Carlos*) is inserted by following the existing path and the remaining part (i.e., *Beto*, *Ana*) is inserted by creating a new path from *Carlos*. The frequency count of node *Carlos* is also updated (i.e., 2) to indicate that this node is shared by two paths. In addition, we update the header table for the frequencies of friends in L_2. The next list (i.e., L_3) does not share any common prefix path with the existing tree (in Fig. 1b), hence it is inserted as a new branch from the root, as shown in Fig. 1c. (For the simplicity of figures, though the horizontal node traversal pointers are maintained in trees, we do not show them in Fig. 1c and subsequent figures.) The remaining lists are inserted in the IF-tree in similar fashion and the resultant IF-tree after inserting all lists is shown in Fig. 1d.

Once all lists in LC are inserted into the IF-tree, we obtain the frequency of each friend from the header table. This frequency information can be used to identify the set of friends who will not form influential friend groups with other friends. Knowing the value of Prom$G$Max(LC) from Table 2, we calculate the upper bound $\text{Inf}^{\text{UB}}(f_i, \text{LC})$ to the influence value of each friend f_i in the header table. So, the

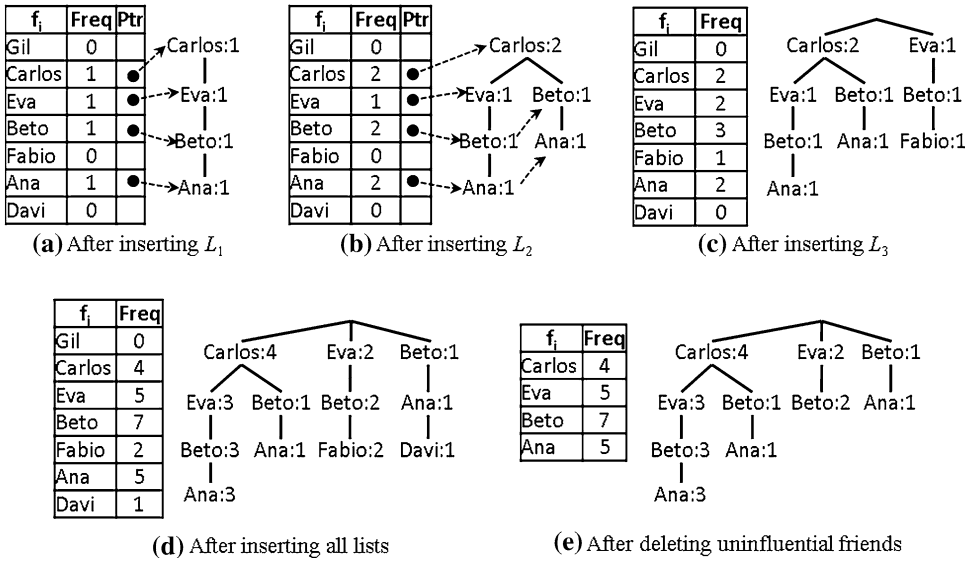

Fig. 1 The IF-tree construction

upper bound to the influence of all friends in the header table will be *Gil*: $0.11 \times 0 = 0$, *Carlos*: $0.70 \times 4 = 2.8$, *Eva*: $0.70 \times 5 = 3.5$, *Beto*: $0.70 \times 7 = 4.9$, *Fabio*: $0.70 \times 2 = 1.4$, *Ana*: $0.70 \times 5 = 3.5$, and *Davi*: $0.70 \times 1 = 0.7$. *Gil*, *Fabio* and *Davi* are found not to be influential friends and they will not appear in any influential friend group, as their upper bound to the influence values (i.e., 0, 0.7 and 1.4, respectively) is less than *minInf*. Therefore, we can safely remove *Gil*, *Fabio* and *Davi* from the IF-tree. In the next phase of IF-tree construction, we removed all such uninfluential friends from the IF-tree. Figure 1e shows the final IF-tree after removing *Davi* and *Fabio* from the header table and the tree structure as well.

Let $F(L_j)$ be the set of friends in a friend list L_j that are found influential after constructing the IF-tree. Based on the above IF-tree construction procedure, we observed the following:

Observation 1 An IF-tree registers the projection of $F(L_j)$ for L_j in LC only once.

Observation 2 The total frequency of any node in an IF-tree is greater than or equal to the sum of frequencies of its children.

Lemma 2 The size of an IF-tree on LC for *minInf* is bounded above by $\sum_{L\_j \in \text{LC}} |F(L_j)|$.

Proof Following the steps in the IF-tree construction, all friend lists in LC are inserted into an IF-tree. Guaranteed uninfluential friends in any friend list $L_j \in$ LC are then identified and safely removed. As each remaining friend (i.e., influential friend) in each L_j is represented by a tree node, the number of tree nodes is bounded above by the total number of influential friends in all friend lists in LC. Moreover, as IF-tree is a tree structure, its tree paths representing the common prefix of friend lists are shared. This further reduces the number of tree nodes, and thus reduces the size of the IF-tree. Hence, the size of an IF-tree on LC for *minInf* is bounded above by $\sum_{L\_j \in \text{LC}} |F(L_j)|$.

Lemma 3 Given LC and *minInf*, the complete set of all influential friend groups can be obtained from an IF-tree for the *minInf* on LC.

Proof An IF-tree can be considered as an alternative representation of LC because the IF-tree captures all friend lists in LC with only guaranteed uninfluential friends f_i (i.e., $\text{Inf}^{\text{UB}}(f_i, \text{LC}) < minInf$) safely removed. We know that the complete set of all influential friend groups can be obtained from LC. Consequently, given LC and *minInf*, the complete set of all influential friend groups can be obtained from the corresponding IF-tree representing LC with *minInf*.

Lemma 3 proves the completeness of an IF-tree for mining influential friend groups. Based on this lemma, influential friend groups can be found by mining our IF-tree.

3.2 A routine for mining influential friend groups

Another key component of our proposed DIFSoN model is a mining routine. Once the IF-tree is constructed, this mining routine then mines influential friend groups by applying a tree-based pattern mining technique (Han et al. 2000) to the IF-tree. Based on this IF-tree, the mining routine recursively constructs the projected tree for potential influential friend groups and mines their extensions. It examines all the

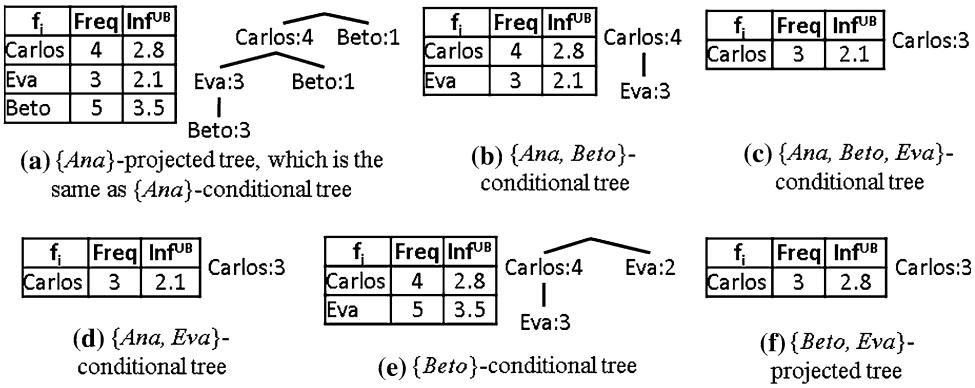

Fig. 2 The IF-tree mining with Prom*G*Max

f_i	Freq	Inf^{UB}
Carlos	4	2.8
Eva	3	2.1
Beto	5	3.5

(a) {*Ana*}-projected tree, which is the same as {*Ana*}-conditional tree

f_i	Freq	Inf^{UB}
Carlos	4	2.8
Eva	3	2.1

(b) {*Ana, Beto*}-conditional tree

f_i	Freq	Inf^{UB}
Carlos	3	2.1

(c) {*Ana, Beto, Eva*}-conditional tree

f_i	Freq	Inf^{UB}
Carlos	3	2.1

(d) {*Ana, Eva*}-conditional tree

f_i	Freq	Inf^{UB}
Carlos	4	2.8
Eva	5	3.5

(e) {*Beto*}-conditional tree

f_i	Freq	Inf^{UB}
Carlos	3	2.8

(f) {*Beto, Eva*}-projected tree

conditional IF-trees consisting of the set of potential influential friend groups occurring with a suffix group. In other words, the mining routine proceeds to recursively mine the IF-tree of decreasing size to generate candidate influential friend groups without additional scans of the interested portion LC of social networks.

More specifically, the mining routine recursively mines the projected trees of all friends in the header table starting from the bottommost friend in the header table. Unlike the header table in the IF-tree, the header table in a projected tree maintains the upper bound to influence value for each friend f_i in the projected tree using Eq. (4). Uninfluential friends are then removed from the projected tree. The resulting tree becomes the corresponding conditional tree. See the following example.

Example 8 Let us mine influential friend groups from the IF-tree in Fig. 1e with $minInf = 2.0$. Again, as observed from Table 2 that Prom*G*Max $= 0.70$, which is Prom(*Davi*). The projected tree for {*Ana*} (i.e., {*Ana*}-projected tree) is constructed first by accumulating the contents in the tree paths ⟨*Carlos*:3 *Eva*: 3 *Beto*: 3⟩, ⟨*Carlos*:1 *Beto*:1⟩, and ⟨*Beto*: 1⟩. The resulting {*Ana*}-projected tree is shown in Fig. 2a. The upper bound to influence values for other friends with {*Ana*} (i.e., *Carlos*, *Eva*, and *Beto*) are calculated using this Prom*G*Max, and are shown in the last column of the header table in Fig. 2a. Then, the {*Ana*}-projected conditional tree is constructed by removing all uninfluential friends (none in this case) from the {*Ana*}-projected tree. Thus, the {*Ana*}-projected conditional tree is identical to the {*Ana*}-projected tree. Potentially influential friend groups {*Ana*, *Beto*}:3.5, {*Ana*, *Eva*}:2.1 and {*Ana*, *Carlos*}:2.8 are generated from this conditional tree. In the next iteration, the {*Ana*, *Beto*}-projected conditional tree (as shown in Fig. 2b) is constructed in the same fashion. Figure 2c shows the {*Ana, Beto, Eva*}-projected conditional tree, which generates group {*Ana*, *Beto, Eva, Carlos*}:2.1. Figure 2d shows the {*Ana, Eva*}-projected conditional tree, which generates group {*Ana*, *Eva,Carlos*}: 2.1.

Figure 2e shows the next friend {*Beto*}-projected conditional tree. Again, Prom*G*Max is used for calculating the upper bounds to influence values for all friends in the subtree. The {*Beto*, *Eva*}-projected tree, which is identical to its projected conditional tree, is shown in Fig. 2f. The remaining mining process is performed in a similar manner for all the friends in the header table, and it terminates when we reach the top of the header table of the original IF-tree.

After the generation of all potentially influential friend groups, we eliminate those uninfluential friend groups from the list by calculating the true influence value for each group. The set of 15 potentially influential friend groups generated by the mining routine and the seven truly influential groups among them are presented in Table 3.

4 The enhanced DIFSoN model with tightened upper bounds to influence values

Recall from Sect. 3 that DIFSoN uses Prom*G*Max to compute the upper bound to the influence value for removing the set of uninfluential friends from further consideration in our IF-tree. As a loose upper bound may lead to many potentially influential friends who are not truly influential, we enhance DIFSoN by tightening such an upper bound with the use of a local maximum prominence value—called Prom*L*Max—instead of Prom*G*Max. For LC^G, Prom*L*Max is calculated as

$$\text{Prom}L\text{Max}(G) = \max\{\text{Prom}(f_i) \,|\, f_i \in G\}. \quad (9)$$

Let $F = \{f_1, \ldots, f_h\}$ and $G' = \{f_{h+1}, \ldots, f_k\}$ such that $G = F \cup G'$. Then, for any F-projected tree containing a set of friends G', the Prom*L*Max value for the F-projected tree is the maximum of the prominence values among all friends in $G' = \{f_1, f_2, \ldots, f_k\}$ and that among all friends in F. With the use of Prom*L*Max, we obtain a tighter upper bound $\text{Inf}^{\text{TUB}}(G, \text{LC})$ to the influence of a group G in a collection LC of friend lists:

Table 3 Influential friend group calculation

G:Freq(G, LC)	$\text{Inf}^{\text{UB}}(G, \text{LC})$	$\text{Inf}^{\text{TUB}}(G, \text{LC})$	Prom(G)	Inf(G, LC)	Truly influential?
{*Ana*}:5	3.5	3.0	0.60	$\times 5 = 3.0$	Yes
{*Ana*, *Beto*}:5	3.5	3.0	$\frac{0.6+0.5}{2} = 0.55$	$\times 5 = 2.75$	Yes
{*Ana*, *Beto*, *Eva*}:3	2.1	1.8	$\frac{0.6+0.5+0.42}{3} \approx 0.51$	$\times 3 = 1.52$	No*
{*Ana*, *Beto*, *Eva*, *Carlos*}:3	2.1	–	$\frac{0.6+0.5+0.42+0.4}{4} = 0.48$	$\times 3 = 1.44$	No*
{*Ana*, *Beto*, *Carlos*}:4	2.8	2.4	$\frac{0.6+0.5+0.4}{3} = 0.50$	$\times 4 = 2.0$	Yes
{*Ana*, *Eva*}:3	2.1	1.8	$\frac{0.6+0.42}{2} = 0.51$	$\times 3 = 1.53$	No*
{*Ana*, *Eva*, *Carlos*}:3	2.1	–	$\frac{0.6+0.42+0.4}{3} \approx 0.47$	$\times 3 = 1.42$	No*
{*Ana*, *Carlos*}:4	2.8	2.4	$\frac{0.6+0.4}{2} = 0.50$	$\times 4 = 2.0$	Yes
{*Beto*}:7	4.9	3.5	0.50	$\times 7 = 3.5$	Yes
{*Beto*, *Eva*}:5	3.5	2.5	$\frac{0.5+0.42}{2} = 0.46$	$\times 5 = 2.3$	Yes
{*Beto*, *Eva*, *Carlos*}:3	2.1	1.5	$\frac{0.5+0.42+0.4}{3} = 0.44$	$\times 3 = 1.32$	No*
{*Beto*, *Carlos*}:4	2.8	2.0	$\frac{0.5+0.4}{2} = 0.45$	$\times 4 = 1.8$	No
{*Eva*}:5	3.5	2.1	0.42	$\times 5 = 2.1$	Yes
{*Eva*, *Carlos*}:3	2.1	1.26	$\frac{0.42+0.4}{2} = 0.41$	$\times 3 = 1.23$	No*
{*Carlos*}:4	2.8	1.6	0.40	$\times 4 = 1.6$	No*

* These seven potentially influential friend groups are generated by the original, but not the enhanced, DIFSoN model

$$\text{Inf}^{\text{TUB}}(G, \text{LC}) = \text{Prom}L\text{Max}(G) \times \text{Freq}(G, \text{LC}) \quad (10)$$

and

$$\text{Inf}(G, \text{LC}) \leq \text{Inf}^{\text{TUB}}(G, \text{LC}) \leq \text{Inf}^{\text{UB}}(G, \text{LC}). \quad (11)$$

The number of potentially influential friends would be closer to that of the truly influential ones. This, in turn, speeds up the mining process.

As a global maximum, Prom$G$Max(LC) can be computed once and used multiple times during the mining process. One concern about using the local maximum Prom$L$Max(LC) for the prominence value is its computation cost. Fortunately, because we arrange the friends in the IF-tree in ascending order of prominence values and we mine the tree in a bottom-up manner, it is guaranteed that Prom(f_i) is always the Prom$L$Max in the F-projected tree. This property helps us avoid repeatedly calculating Prom$L$Max while constructing projected trees. Getting this advantage during the mining process is the primary reason of arranging our IF-tree in ascending order of prominence values. To describe our mining routine, in Example 9, we revisit our running example (as shown in Example 8) of the LC in Table 1 and the IF-tree in Fig. 1e.

Example 9 Let us mine influential friend groups from the IF-tree in Fig. 1e with $minInf = 2.0$. The projected tree for {*Ana*} (i.e., {*Ana*}-projected tree) is constructed first by accumulating the contents in the tree paths ⟨*Carlos*:3 *Eva*:3 *Beto*:3⟩, ⟨*Carlos*:1 *Beto*:1⟩, and ⟨*Beto*:1⟩. For the {*Ana*}-projected tree presented in Fig. 3a, Prom$L$Max = 0.6 (i.e., Prom(*Ana*)). Thus, the potential influence values for other friends with {*Ana*} (i.e., *Carlos*, *Eva*, and *Beto*) are calculated using this Prom$L$Max, and are shown in the last column of the header table in Fig. 3a. Then, the {*Ana*}-projected conditional tree is constructed by removing all uninfluential friends (i.e., *Eva* because *Eva*'s potential influence value = 1.8 < *minInf*) from the {*Ana*}-projected tree. Fig. 3b illustrates the {*Ana*}-projected conditional tree after removing *Eva*. The influential friend groups {*Ana*, *Beto*}:3.0 and {*Ana*, *Carlos*}:2.4 are generated from this conditional tree. In the next iteration, the {*Ana*, *Beto*}-projected conditional tree is constructed in the same fashion. See Fig. 3c, which shows the {*Ana*, *Beto*}-projected conditional tree that generates group {*Ana*, *Beto,Carlos*}:2.4.

Afterwards, Fig. 3d shows the next friend {*Beto*}-projected conditional tree where Prom(*Beto*) (instead of Prom(*Ana*)) is considered as Prom$L$Max to calculate the potential influence values for all friends in the sub-tree. Fig. 3e shows the {*Beto*, *Eva*}-projected tree, in which the only friend (i.e., *Carlos*) is uninfluential. Hence, the {*Beto*, *Eva*}-projected conditional tree is not formed. The remaining mining process is performed in a similar manner for all the friends in the header table, and it terminates when we reach the top of the header table of the original IF-tree.

After the generation of all potentially influential friend groups, we eliminate those uninfluential friend groups from the list by calculating the true influence value for each group. The set of eight potentially influential friend groups generated by this enhanced DIFSoN model (cf. 15 potentially influential friend groups generated by the original

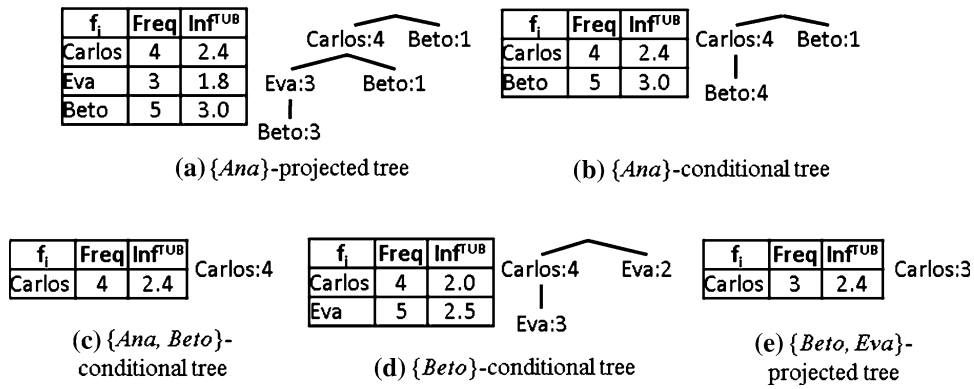

Fig. 3 The IF-tree mining with Prom*L*Max

DIFSoN model) and the seven truly influential groups among them are presented in Table 3.

If the user-specified *minInf* were set to 2.4 (or 2.5), the number of potentially influential friend groups generated by the enhanced DIFSoN model would be substantially smaller.

Our mining routine is efficient because it applies a pattern-growth based mining technique on an IF-tree.

5 The extended DIFSoN model for handling interactive mining

In the previous section, we presented how we find influential friend groups from a collection LC of friend lists in a portion of a social network with the user-specified *minInf* threshold. In many social networks, users are classified into different categories based on their status, degree of connectivity, sociability, activity (e.g., first-degree vs. second-degree connection, circle vs. extended circles, friends vs. friends-of-friends, basic vs. premium or expert users). In addition, there are common-interest groups of different categories as well. When mining influential friend groups, it is not uncommon that a user may want to interactively expand or shrink their scope or the interested portion of the social network so as to include or exclude some lists of friends of certain individual social entities from other categories (e.g., second-degree connection, extended circles, friends-of-friends) or some lists of friends in certain common-interest groups from other categories (e.g., database group in addition to the social computing and the data mining groups). A naive solution to handle these interactive changes is to rebuild an IF-tree from scratch. However, we can do better. In this section, we present how we effectively maintain the IF-tree when LC is changed interactively.

5.1 Mining influential friends from the expanded scope

First, let us consider the case in which users expand the scope of their interested portion of the social network (i.e., to include additional friend lists LC^+ beyond the old LC). As the new LC subsumes the old LC and the collection LC^+ of additional friend lists, instead of rebuilding a new IF-tree (corresponding to this new LC) from scratch, we build the new IF-tree based on the old IF-tree (corresponding to the LC). Note that we cannot just trivially insert those friend lists in LC^+ into the old IF-tree to form the new IF-tree because of the following issue. Recall from previous sections that we removed those uninfluential individual friends from the IF-tree. Such pruning is sound in a static environment, in which users are interested in a specific portion of the social network. However, in a dynamic environment in which users may interactively change their scope of interested portion of the social network, an individual friend who was uninfluential in the old LC may become influential in the new LC (due to the additional friend lists in LC^+). Hence, to ensure the soundness of our extended DIFSoN model, we safely skip the extra work of pruning those uninfluential individual friends when handling interactive user changes of scope.

Once we established the soundness of our extended DIFSoN model, we further improved its performance as follows. Recall from previous sections that two main purposes of the first scan of LC in the static environment are to (i) prune uninfluential individual friends and (ii) sort influential individual friends in ascending order of their prominence values. Here, in the dynamic environment, to ensure the soundness of our extended DIFSoN model, we no longer need to prune those uninfluential individual friends. Regarding the sorting of individual friends in ascending order of their prominence values, it can be done once and for all mining. To elaborate, the prominence values of individual social entities are independent of the changes in LC or *minInf*. Hence, these prominence values can be read and sorted in the ascending order once. No more sorting is needed for subsequent mining, even if the scope of LC were changed (e.g., broaden or narrow the scope) or the user-specified *minInf* were changed (e.g., raise or lower the threshold). As such, we improve the performance of our extended DIFSoN model by skipping the first scan of LC for subsequent execution of the model.

Next, we scan LC^+ and insert every friend list $L_j \in LC^+$ into the old IF-tree to form a new IF-tree. During the insertion, in addition to inserting each friend $f_i \in L_j$ into the IF-tree, we also flag f_i in the header table. So, at the end of the insertion process, the flagged friends in the header table are those whose frequencies got updated (i.e., increased) due to insertion of friend lists in LC^+. For the unflagged friends, their frequency values (and thus their influential values or upper bounds to their influential values) remain unchanged. With this information from the construction of IF-tree, we improve the performance of our extended model in mining influential friends by projecting only those flagged friends (instead of projecting every flagged and unflagged friend). The rationale is that potentially influential friend groups mined from projections of these unflagged friends would have the same (upper bounds to) influential values in the new LC as those mined from the old LC. We safely avoid the redundant mining on these unflagged friends for the same potentially influential friend groups having the same upper bounds to influential values.

Afterwards, we then check each potentially influential friend group to verify if it is truly influential (i.e., eliminate false positives). More specifically, if the user-specified *minInf* threshold is expressed in relative percentage of the new LC, we check each potentially influential friend group G—mined from (i) the new LC for the above flagged friends and (ii) the old LC for those unflagged friends—to see if its Inf(G, new LC) $\geq$ *minInf*. On the other hand, if *minInf* is expressed in absolute number (and does not change during the interactive mining), we improve the performance of our extended DIFSoN model by checking only those potentially influential friend groups mined from the new LC for the above flagged friends.

5.2 Mining influential friends from the shrunk scope

Next, let us consider the case in which users shrink the scope of their interested portion of the social network (i.e., to exclude some friend lists LC^- from the old LC). As the old LC subsumes the new LC and the collection LC^- of excluded friend lists, instead of rebuilding a new IF-tree (corresponding to this new LC) from scratch, we build the new IF-tree based on the old IF-tree (corresponding to the old LC) by removing those friend lists in LC^- from the old IF-tree. Again, we first improve the performance of our extended model by skipping one scan of LC^- as we do not need to remove uninfluential friends.

Next, we scan LC^-, delete every friend list $L_j \in LC^-$ from the old IF-tree, and adjust the tree path to form a new IF-tree. During the deletion, we also flag each friend $f_i \in L_j$ in the header table. These flagged friends in the header table are those whose frequency values (and thus influence values) got updated (i.e., decreased). We improve the performance of our extended model in mining influential friends by projecting only those flagged friends (instead of projecting every flagged and unflagged friend).

This safely avoids the redundant mining on those unflagged friends for the same potentially influential friend groups having the same upper bounds to influential values. Finally, we eliminate false positives by verifying each potentially influential friend group.

6 Experimental results

In this section, we present the experimental results on DIFSoN. To the best of our knowledge, our IF-tree is the first to mine such influential friend groups from a user-interested portion of social networks. Hence, there is no existing work to compare with our DIFSoN model. However, as mining weighted frequent patterns can be related to our mining of influential friend groups, we compare our DIFSoN model with a well-known weighted frequent itemset mining algorithm WFIM (Yun and Leggett 2005). The WFIM is an FP-tree based weighted frequent pattern mining algorithm that requires two database scans. Similar to our approach, it uses a minimum weight value. However, the main difference between the two approaches is that we use a fixed weight for each friend in the social network, while the WFIM uses different weights for each domain item which are given randomly from a weight range. Moreover, WFIM uses a secondary support threshold to calculate weighted frequent patterns.

Since the weighted frequent pattern mining algorithms were not designed for social network mining, we used the datasets that are mostly used in frequent pattern mining domain for fair comparison. Fortunately, we observed similar characteristics of lists (as we defined in this article) in the datasets where each transaction consists of an ID and a set of items. We map the ID and the set of items of each transaction as list ID and the set of friends in the list, respectively. More specially, we used (i) IBM synthetic datasets (e.g., T10I4D100K) from http://www.almaden.ibm.com/cs/quest (or http://www.cs.loyola.edu/~cgiannel/assoc_gen.html) and (ii) real datasets (e.g., mushroom, pumsb*, kosarak). from the Frequent Itemset Mining Dataset Repository (http://fimi.ua.ac.be/data). See Table 4 for the characteristics of the datasets. These datasets do not provide the prominence values of each item. Hence, we generated random numbers, ranging from 0.0 to 1.0, for the prominence values of each item in a dataset.

All programs were written in C++ and run on the Windows XP operating system with a 2.13 GHz CPU and 2 GB main memory. The runtime specified indicates the total execution time (i.e., CPU and I/Os). The reported results are based on the average of multiple runs for each

Table 4 Dataset characteristics

Dataset	#Transactions	#Items	Max trans length	Avg trans length	Density
T10I4D100K	100,000	870	29	10.10	Sparse
pumsb*	49,046	2,088	63	50.48	Sparse
kosarak	990,002	41,270	2,498	8.10	Sparse
mushroom	8,124	119	23	23.00	Dense

case. We obtained consistent results for all of these datasets.

6.1 Runtimes of our DIFSoN model

In the first experiment, we study the execution time performance of our DIFSoN over WFIM for datasets of different types and changes in *minInf*. The execution time for DIFSoN includes all the steps of IF-tree building, cleaning the tree for uninfluential friends, the corresponding mining, and final influence value calculation for all generated groups. The results on two sparse datasets (e.g., T10I4D100K and pumsb*) and one dense dataset (e.g., mushroom) are presented in Fig. 4. As expected, both approaches required more execution time when mining larger datasets and/or for lower thresholds. As size of LC increased and *minInf* decreased, the tree structure size and number of influential friend groups increased. Hence, a comparatively longer time was required to generate a large number of influential friend groups from larger trees.

However, it can be observed from the graphs that DIFSoN outperforms the WFIM in all three datasets. The primary reason of this performance improvement of DIFSoN is constructing the tree with a single scan of LC, whereas the WFIM scanned the database twice to construct the tree. As both trees organized the tree nodes in the same order, the overall mining process of the two approaches were similar. However, WFIM took longer mining time for handling multiple weight range (for items) and the additional support threshold. This was another issue for the comparatively poor performance of WFIM.

6.2 Compactness of the IF-tree

We tested the memory requirement of our IF-tree to capture contents of the collection LC of friend lists. In Fig. 5, we show the amount of memory required by our IF-tree when capturing the entire LC without considering any *minInf* (i.e., before the removal of uninfluential friends having upper bounds to the influence $<$ *minInf*), which could be the worst-case size of our IF-tree. From this result, we can observe that the IF-tree can be handled within a very reasonable amount of memory—low enough for recently available gigabyte-range memory.

6.3 Scalability of our DIFSoN model

To test the scalability of DIFSoN by varying the number of transactions, we used the kosarak dataset, since it is a huge sparse dataset with a large number of distinct items and transactions (ref. Table 4). We divided this dataset into five

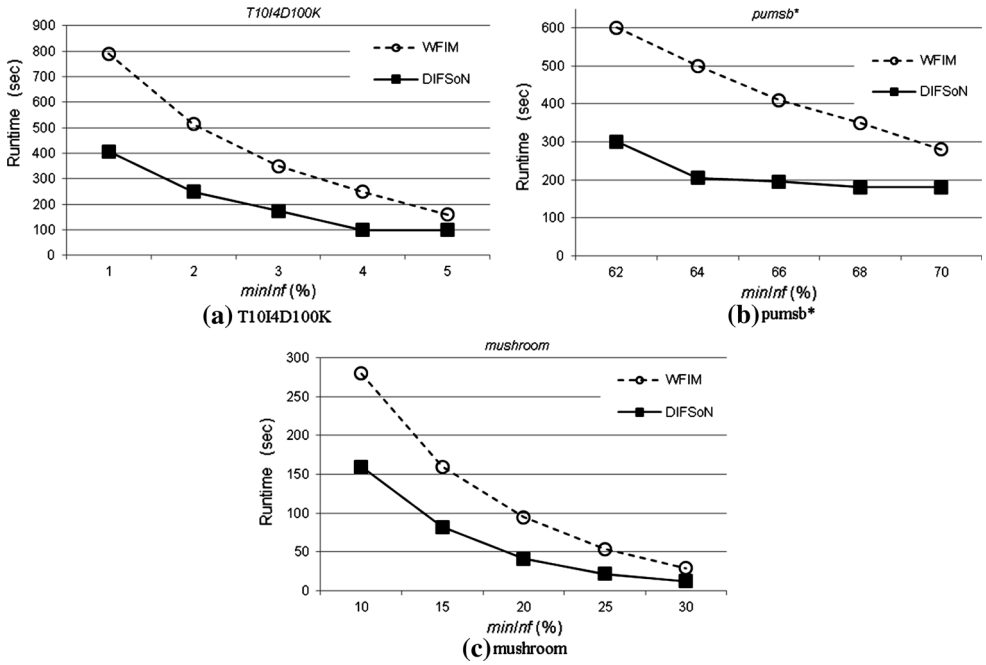

Fig. 4 Runtimes of our DIFSoN model

portions of 0.2 million transactions each. Then, we tested the performance of DIFSoN after accumulating each portion with previous parts and performing the mining each time with *minInf* = 5 %. The time in the y-axis of Fig. 6 specifies the total required time with increasing database size. Clearly, as the size of the database increases, the overall time increases. However, as shown in the figure, DIFSoN showed a linear scalability over the database size.

To recap, the above experimental results show that the proposed DIFSoN approach can mine the set of influential friend groups in both time- and memory-efficient manners over different types of datasets. Furthermore, it is scalable for dataset size.

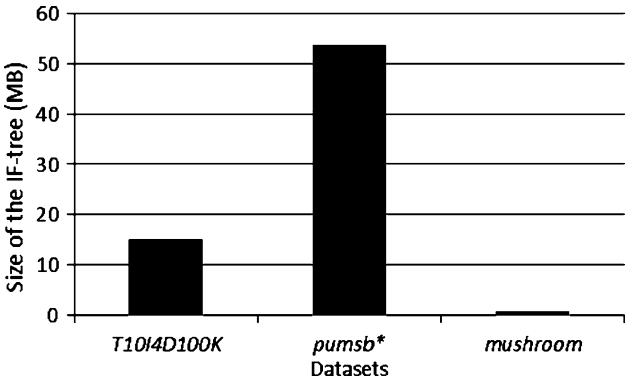

Fig. 5 Compactness of the IF-tree

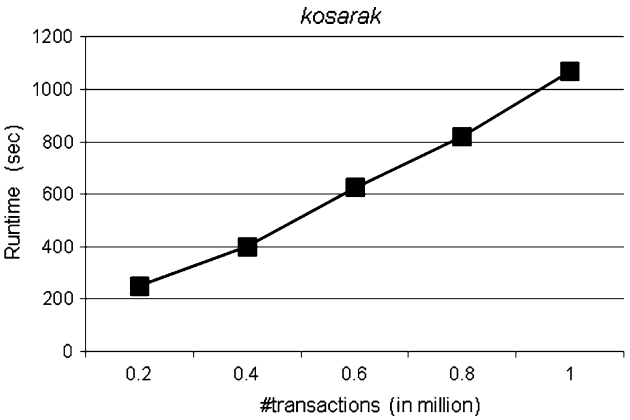

Fig. 6 Scalability of our DIFSoN model

6.4 Maintenance of IF-trees for interactive mining

To test the performance of interactive mining and maintenance of our IF-tree, we divided the kosarak dataset into five equal-sized portions, each consisting of 200 K transactions. Then, we applied our influential friend mining algorithm on one portion, and then interactively expanded our LC. At each stage, we mined influential friends with various *minInf* thresholds (e.g., 4, 5 and 6 %) of the accumulative portions. Experimental results in Fig. 7a show that the overall runtime increased when the size/scope of interest networks increased.

Similarly, to test the effect of shrinking of the size/scope of interested networks, we also used the kosarak dataset. This time, we started with all transactions (i.e., all friend lists) in the dataset and gradually reduced its size by deleting 100 K transactions at every stage. Again, at each stage, we mined influential friends with various *minInf* thresholds (e.g., 4, 5 and 6 %) of the accumulative portions. Figure 7b shows that, when the size/scope of interest networks decreased, the runtime decreased. As an ongoing work, we plan to add experimental results on real social network data.

7 Conclusions

In this article, we introduced a new notion of influential friends for social network databases and presented the DIFSoN model to discover influential friends (or entities) from social networks. The DIFSoN comprises the IF-tree and a mining routine. Information about the lists of friends of individual social entities or lists of friends in common-interest groups are captured in the IF-tree, from which sets of influential friend groups can be mined efficiently. Although the notion of influential friends does not satisfy

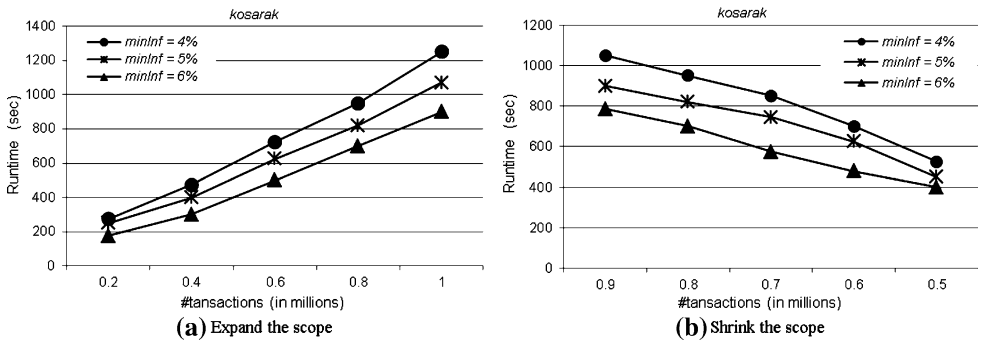

Fig. 7 Runtimes of our extended DIFSoN model for interactive mining

the downward closure property, we addressed this issue using the global maximum prominence values of users. To enhance the model, we proposed to use the local maximum prominence values, which give a tighter upper bound to values for influence measures and thus reduce the number of generated potentially influential friend groups. Moreover, we further extended our DIFSoN model by allowing users to change the mining parameters (i.e., broaden or shrink the scope of their interested portion of social networks) and efficiently handling their changes. Experimental results showed that (i) the IF-tree is compact and space efficient and (ii) the tree-based mining routine within the DIFSoN model is fast and scalable for both sparse and dense data. For ongoing and future work, we plan to incorporate other computational metrics (e.g., popularity, significance, strength) with prominence to discover useful information from social networks.

Acknowledgments This project is partially supported by NSERC (Canada) and University of Manitoba.

References

Agrawal R, Srikant R (1994) Fast algorithms for mining association rules in large databases. In: VLDB 1994, pp 487–499

Alsaleh S, Nayak R, Xu Y (2011) Finding and matching communities in social networks using data mining. In: ASONAM 2011, pp 389–393

Baatarjav EA, Dantu R (2011) Unveiling hidden patterns to find social relevance. In: IEEE SocialCom 2011, pp 242–249

Barbosa EM, Moro MM, Lopes GR, de Oliveira JPM (2012) VRRC: web based tool for visualization and recommendation on co-authorship network. In: ACM SIGMOD 2012, p 865

Bhagat S, Goyal A, Lakshmanan LVS (2012) Maximizing product adoption in social networks. In: ACM WSDM 2012, pp 603–612

Cameron JJ, Leung CKS, Tanbeer SK (2011) Finding strong groups of friends among friends in social networks. In: SCA 2011, pp 824–831

Dalvi N, Kumar R, Pang B (2012) Object matching in tweets with spatial models. In: ACM WSDM 2012, pp 43–52

Ding X, Zhang L, Wan Z, Gu M (2010) A brief survey on de-anonymization attacks in online social networks. In: CASoN 2010, pp 611–615

Fan W, Yeung KH (2010) Virus propagation modeling in Facebook. In: ASONAM 2010, pp 331–335

Ferreira LN, Pinto AR, Zhao L (2012) QK-means: a clustering technique based on community detection and K-means for deployment of cluster head nodes. In: IJCNN 2012. doi:10.1109/IJCNN.2012.6252477

Górecki J, Slaninová K, Snášel V (2011) Visual investigation of similarities in global terrorism database by means of synthetic social networks. In: CASoN 2011, pp 255–260

Han J, Pei J, Yin Y (2000) Mining frequent patterns without candidate generation. In: ACM SIGMOD 2000, pp 1–12

Hong L, Ahmed A, Gurumurthy S, Smola AJ, Tsioutsiouliklis K (2012) Discovering geographical topics in the Twitter stream. In: WWW 2012, pp 769–778

Horak Z, Kudelka M, Snasel V, Abraham A, Rezankova H (2011) Forcoa.NET: an interactive tool for exploring the significance of authorship networks in DBLP data. In: CASoN 2011, pp 261–266

Jiang F, Leung CKS, Tanbeer SK (2012) Finding popular friends in social networks. In: SCA 2012, pp 501–508

Lee W, Leung CKS, Song JJ, Eom CSH (2012) A network-flow based influence propagation model for social networks. In: SCA 2012, pp 601–608

Leung CKS, Carmichael CL (2010) Exploring social networks: a frequent pattern visualization approach. In: IEEE SocialCom 2010, pp 419–424

Leung CKS, Carmichael CL, Teh EW (2011) Visual analytics of social networks: mining and visualizing co-authorship networks. In: HCII-FAC 2011, pp 335–345

Leung CKS, Medina IJM, Tanbeer SK (2013) Analyzing social networks to mine important friends. In: Social media mining and social network analysis: emerging research, pp 90–104

Leung CKS, Tanbeer SK (2012) Mining social networks for significant friend groups. In: DASFAA Workshops 2012, pp 180–192

Liaghat Z, Rasekh AH, Mahdavi A (2013) Application of data mining methods for link prediction in social networks. SNAM 3(2):143–150

Lin W, Kong X, Yu PS, Wu Q, Jia Y, Li C (2012) Community detection in incomplete information networks. In: WWW 2012, pp 341–350

Muhammad M, Missen S, Boughanem M, Cabanac G (2013) Opinion mining: reviewed from word to document level. SNAM 3(1):107–125

Nasirifard P, Hayes C (2011) Tadvise: a Twitter assistant based on Twitter lists. In: SocInfo 2011, pp 153–160

Tanbeer SK, Leung CKS (2013) Finding diverse friends in social networks. In: APWeb 2013, pp 301–309

Tanbeer SK, Leung CKS, Cameron JJ (2012) DIFSoN: discovering influential friends from social networks. In: CASoN 2012, pp 120–125

Tanbeer SK, Jiang F, Leung CKS, MacKinnon RK, Medina IJM (2013) Finding groups of friends who are significant across multiple domains in social networks. In: CASoN 2013, pp 21–26

Valverde-Rebaza JC, Lopes AA (2012) Structural link prediction using community information on Twitter. In: CASoN 2012, pp 132–137

Yang X, Ghoting A, Ruan Y, Parthasarathy S (2012) A framework for summarizing and analyzing Twitter feeds. In: ACM KDD 2012, pp 370–378

Yun U, Leggett JJ (2005) WFIM: weighted frequent itemset mining with a weight range and a minimum weight. In: SIAM SDM 2005, pp 636–640