Mining of Diverse Social Entities from Linked Data

Alfredo Cuzzocrea
ICAR-CNR & Univ. of Calabria
Rende (CS), Italy
cuzzocrea@si.deis.unical.it

Carson K. Leung
University of Manitoba
Winnipeg, MB, Canada
kleung@cs.umanitoba.ca

Syed K. Tanbeer
University of Manitoba
Winnipeg, MB, Canada
tanbeer@cs.umanitoba.ca

ABSTRACT

Nowadays, high volumes of valuable data can be easily generated or collected from various data sources at high velocity. As these data are often related or linked, they form a web of linked data. Examples include semantic web and social web. The social web captures social relationships that link people (i.e., social entities) through the World Wide Web. Due to the popularity of social networking sites, more people have joined and more online social interactions have taken place. With a huge number of social entities (e.g., users or friends in social networks), it becomes important to analyze high volumes of linked data and discover those diverse social entities. In this paper, we present (i) a tree-based mining algorithm called **DF-growth**, along with (ii) its related data structure called **DF-tree**, which allow users to effectively and efficiently mine diverse friends from social networks. Results of our experimental evaluation showed both the time- and space-efficiency of our scalable DF-growth algorithm, which makes good use of the DF-tree structure.

Categories and Subject Descriptors

E.2 [**Data**]: Data Storage Representations—*linked representations*; H.2.8 [**Database Management**]: Database Applications—*data mining*; J.2 [**Computer Applications**]: Social and Behavioral Sciences

General Terms

Algorithms; Design; Experimentation; Management; Performance; Theory

Keywords

Data mining, friendship patterns, diverse friends, linked data, social networks, extending database technology, database theory

1. INTRODUCTION & RELATED WORK

Nowadays, with the advances in technology, high volumes of valuable data—such as blogs, forums, wikis, and users' reviews—can be easily generated or collected from various data sources. These data are often related or linked, and thus form a web of linked data [3]. Over the few years, researchers have modelled, queried, and reasoned these linked web data. For instance, Pernelle and Saïs [15] focused on classification rule learning for linked data. Ferrara et al. [9] proposed a feature-based approach to classify linked data.

A social web is an instance of a web of linked data. Such a social web can be viewed as a collection of social relationships that link social entities (e.g., users). In recent years, researchers have exploited the social perspective or social phenomena [4, 8] in this web of linked data (e.g., for the relevant problem of detecting communities over social and information networks [6, 7, 20]). Intuitively, social networks [14] are made of social entities who are linked by some specific types of relationships (e.g., friendship, common interest, kinship). Facebook, Google+, LinkedIn, Twitter and Weibo [17, 22] are some examples of social networks. Within these networks, a user f_i usually can create a personal profile, add other users as friends, endorse their skills/expertise, exchange messages among friends. These social networks may consist of thousands or millions of users; each user f_i can have different number of friends. Among them, some are more important (or influential, prominent, and/or active in a wide range of domains) than others [2, 12, 13, 19, 23]. Recognizing these diverse friends can provide valuable information for various real-life applications when analyzing and mining high volumes of valuable social network data.

Over the past few years, several data mining techniques [11, 16, 21] have been developed to help users extract implicit, previously unknown, and potentially useful information about the important friends. Recent works on social network mining include the discovery of strong friends [5] and significant friends [18] based on the degree of one-to-one interactions (e.g., based on the number of postings to a friend's wall).

However, in some situations, it is also important to discover users who (i) are influential in the social networks, (ii) have high level of expertise in some domains, and/or (iii) have diverse interest in multiple domains. In other words, users may want to find important friends based on their influence, prominence, and/or diversity. For instance, some users may be narrowly interested in one specific domain (e.g., computers). Other users may be interested in a wide range of domains (e.g., computers, music, sports),

but their expertise level may vary from one domain to another (e.g., a user f_i may be a computer expert but only a beginner in music).

In this paper, we propose a tree-based mining algorithm to find from social networks for those diverse friends, who are highly influential across multiple social network domains. To this end, one of our *key contributions* is an efficient tree-based algorithm called **DF-growth** for mining diverse friends from social networks. DF-growth takes into account multiple properties (e.g., influence, prominence, and/or diversity) of friends in the networks. Another *key contribution* is a prefix-tree based structure called **DF-tree** for capturing the social network data in a memory-efficient manner. Once the DF-tree is constructed, DF-growth computes the diversity of users based on both their influence and prominence to mine diverse groups of friends.

The remainder of this paper is organized as follows. The next section introduces the notion of diverse friends. Section 3 presents our DF-growth algorithm, which mines diverse friends from our DF-tree. Experimental results are reported in Section 4. Conclusions and future work are given in Section 5.

2. DIVERSE FRIENDS IN SOCIAL NETWORKS

This section presents the concept of diverse friends in social networks. Let us consider a social network on three different domains (domains D_1, D_2, D_3) and seven individuals—Antonios, Barbara, Georgios, Dimitrios, Evgenios, Zoe, and Hebe—with prominence values in each domain, as shown in Table 1. Each *domain* represents a sub-category (e.g., sports, arts, education) of interest. The **prominence value** of an individual reveals his level of expertise (e.g., importance, weight, value, reputation, belief, position, status, or significance) in a domain. In other words, the prominence value indicates how important, valued, significant, or well-positioned the individual is in each domain. The prominence value can be measured by using a common scale, which could be (i) specified by users or (ii) automatically calculated based on some user-centric parameters (e.g., connectivity, centrality, expertise in the domain, years of membership in the domain, degree of involvement in activities in the domain, numbers of involved activities in the domain). In this paper, the prominence value is normalized into the range (0, 1]. As the same individual may have different levels of expertise in different domains, his corresponding prominence value may vary from one domain to another. For instance, prominence value $Prom_{D_1}$(Antonios) of Antonios in domain D_1 is 0.45, which is different from $Prom_{D_2}$(Antonios) = 0.60. Moreover, $Prom_{D_1}$(Antonios) is higher than $Prom_{D_1}$(Georgios) = 0.20, implying that Antonios is more influential than Georgios in domain D_1.

Similar to the existing settings of a social network [5, 11, 18], let $F = \{f_1, f_2, \ldots, f_m\}$ be a set of social entities/friends in a social network. An *interest-group list* $L \subseteq F$ is a list of individuals who are connected as friends due to some common interests. Let $G = \{f_1, f_2, \ldots, f_k\} \subseteq F$ be a **group of friends** (i.e., friend group) with k friends. Then, $Size(G) = k$, which represents the number of individuals in G. A *friend network* $F_{SN} = \{L_1, L_2, \ldots, L_n\}$ is the set of all n interest-group lists in the entire social network. These lists belong to some domains, and each domain contains at least one list. The set of lists in a particular domain D is called a *domain database* (denoted as F_D). Here, we assume that there exists an interest-group list in every domain. The *projected list* F_D^G of G in F_D is the set of lists in F_D that contains group G. The frequency $Freq_D(G)$ of G in F_D indicates the number of lists L_j's in F_D^G, and the frequencies of G in multiple domains are represented as $Freq_{D_{1,2,\ldots,d}}(G) = \langle Freq_{D_1}(G), Freq_{D_2}(G), \ldots, Freq_{D_d}(G)\rangle$.

Table 1: Prominence of friends

Friend (f_i)	Prominence $Prom(f_i)$		
	Domain D_1	Domain D_2	Domain D_3
Antonios	0.45	0.60	0.50
Barbara	0.90	0.70	0.30
Georgios	0.20	0.60	0.70
Dimitrios	0.30	0.50	0.40
Evgenios	0.50	0.40	0.45
Zoe	0.42	0.24	0.70
Hebe	0.57	0.10	0.20

Table 2: Lists of interest groups in F_{SN}

Domain	Interest-group list L_j
D_1	L_1 = {Antonios, Barbara} L_2 = {Antonios, Barbara, Dimitrios} L_3 = {Georgios, Dimitrios}
D_2	L_4 = {Barbara, Georgios, Dimitrios} L_5 = {Barbara, Georgios, Evgenios} L_6 = {Barbara, Hebe} L_7 = {Georgios, Evgenios}
D_3	L_8 = {Antonios, Georgios} L_9 = {Antonios, Georgios, Evgenios} L_{10}= {Antonios, Zoe}

EXAMPLE 2.1. Consider F_{SN} shown in Table 2, which consists of n=10 interest-group lists $L_1, ..., L_{10}$ for m=7 social individuals/friends in Table 1. Each row in the table represents the list of an interest group. These 10 interest groups are distributed into d=3 domains D_1, D_2 and D_3. For instance, $F_{D_1} = \{L_1, L_2, L_3\}$. For group G = {Georgios, Evgenios}, its $Size(G)$=2. As its projected lists on the 3 domains are $F_{D_1}^G$=∅, $F_{D_2}^G$={L_6, L_7} and $F_{D_3}^G$={L_8}, its frequencies $Freq_{D_{1,2,3}}(G) = \langle 0, 2, 1\rangle$. □

DEFINITION 2.1. *The* **prominence value $Prom_D(G)$** *of a friend group G in a single domain D is defined as the average of all prominence values for all the friends in G:*

$$Prom_D(G) = \frac{\sum_{i=1}^{Size(G)} Prom_D(f_i)}{Size(G)}. \quad (1)$$

Then, prominence values $Prom_{D_{1,2,\ldots,d}}(G)$ of a friend group G in multiple domains are represented as $Prom_{D_{1,2,\ldots,d}}(G)$ = $\langle Prom_{D_1}(G), Prom_{D_2}(G), \ldots, Prom_{D_d}(G)\rangle$. □

EXAMPLE 2.2. Revisit F_{SN} in Table 2. The prominence value of friend group G = {Georgios, Evgenios} in D_1 = $\frac{Prom_{D_1}(\text{Georgios})+Prom_{D_1}(\text{Evgenios})}{Size(G)}$ = $\frac{0.20+0.50}{2}$ = 0.35. We apply similar computation on the other two domains D_2 and D_3 to get $Prom_{D_{1,2,3}}(G) = \langle 0.35, \frac{0.60+0.40}{2}, \frac{0.70+0.45}{2}\rangle$ = $\langle 0.35, 0.5, 0.575\rangle$. □

DEFINITION 2.2. *The* **influence $Inf_D(G)$** *of a group G of social entities/friends in a domain D in F_D is defined as the product of the prominence value of G in the domain D and its frequency in the domain database F_D, i.e.,*

$$Inf_D(G) = Prom_D(G) \times Freq_D(G). \quad (2)$$

The influence $Inf_{D_{1,2,\ldots,d}}(G)$ of G in multiple domains is then represented as $Inf_{D_{1,2,\ldots,d}}(G) = \langle Inf_{D_1}(G), Inf_{D_2}(G), \ldots, Inf_{D_d}(G)\rangle$. □

EXAMPLE 2.3. Continue with Example 2.2. Recall that $Prom_{D_{1,2,3}}(G) = \langle 0.35, 0.5, 0.575\rangle$. Recall from Example 2.1 that $Freq_{D_{1,2,3}}(G) = \langle 0, 2, 1\rangle$. Then, the overall influence of G in all 3 domains can be calculated as $Inf_{D_{1,2,3}}(G) = \langle 0.35 \times 0, 0.5 \times 2, 0.575 \times 1\rangle = \langle 0, 1, 0.575\rangle$. □

DEFINITION 2.3. *The* **diversity $Div(G)$** *of a group G of social entities/friends among all d domains in F_{SN} is defined as the average of all the influence values of G in all domains in the social network:*

$$Div(G) = \frac{\sum_{j=1}^{d} Inf_{D_j}(G)}{d}. \quad \square \quad (3)$$

EXAMPLE 2.4. Continue with Example 2.3. Recall that $Inf_{D_{1,2,3}}(G) = \langle 0, 1, 0.575\rangle$. Then, the diversity of G in these d=3 domains in F_{SN} is $Div(G)$=$\frac{0+1+0.575}{3}$=0.525. □

Here, a group G of friends in a social network F_{SN} is considered **diverse** if its diversity value $Div(G) \geq$ user-specified minimum threshold *minDiv*, which can be expressed as an absolute (non-negative real) number or a relative percentage (with respect to the size of F_{SN}). Given F_{SN} and *minDiv*, the research problem of ***mining diverse friends from social networks*** is to find every group G of friends having $Div(G) \geq minDiv$.

EXAMPLE 2.5. Let group G = {Georgios, Evgenios}. Recall from Example 2.4 that diversity $Div(G) = 0.525$. Given (i) F_{SN} in Table 2 and (ii) the user-specified $minDiv$=0.5, G is *diverse* because $Div(G)$=0.525 $\geq$ 0.5=$minDiv$.

However, group G' = {Evgenios}, such that $G' \subseteq G$ is *not* diverse because $Div(G') = \frac{(0.5\times0)+(0.4\times2)+(0.45\times1)}{3} = \frac{0+0.8+0.45}{3} = 0.417 < minDiv$. □

Note that, when mining frequent patterns, the frequency/support measure [1, 10] satisfies the downward closure property (i.e., all supersets of an infrequent patterns are infrequent). This helps reduce the search/solution space by pruning infrequent patterns, which in turn speeds up the mining process.

However, mining diverse friends is different from mining frequent patterns. As observed from Example 2.5 that group G' = {Evgenios} is not diverse but its super-group G = {Georgios, Evgenios} is diverse. In other words, diversity does *not* satisfy the downward closure property (i.e., if a group is not diverse, then *not* all of its super-groups are guaranteed to be diverse).

3. MINING DIVERSE FRIENDS

Given that diversity does *not* satisfy the downward closure property, we cannot prune those groups that are not diverse. Hence, the mining of diverse friends can be challenging. To handle this challenge, for each domain D, we identify the **(global) maximum prominence value $GMProm_D$** among all friends. Then, for each friend f_i, we calculate an upper bound of the influence value $Inf^U_D(f_i)$ by multiplying $GMProm_D$ (instead of the actual $Prom_D(f_i)$) with the corresponding frequency $Freq_D(f_i)$. The upper bound of diversity value $Div^U(f_i)$ can then be computed by using $Inf^U_D(f_i)$.

LEMMA 3.1. *Let G be a group of friends in F_{SN} such that a friend $f_i \in G$. If $Div^U(f_i) < minDiv$, then $Div(G)$ must also be less than $minDiv$.* □

EXAMPLE 3.1. Let us revisit F_{SN} in Table 2. Global maximum prominence values are $GMProm_{D_1}$=0.90, $GMProm_{D_2}$ = 0.70, and $GMProm_{D_3}$=0.70. Recall from Example 2.5 that $Freq_{D_{1,2,3}}(\{\text{Evgenios}\}) = \langle 0, 2, 1\rangle$. Then, we can compute $Div^U(\{\text{Evgenios}\}) = \frac{(0.90\times0)+(0.70\times2)+(0.70\times1)}{3} = 0.7 \geq minDiv$. So, we do not prune {Evgenios} to avoid missing its super-group {Georgios, Evgenios}, which is diverse. Similarly, $Div^U(\{\text{Zoe}\}) = \frac{(0.90\times0)+(0.70\times0)+(0.70\times1)}{3}$=0.23 < *minDiv*. Due to Lemma 3.1, we prune Zoe as none of its super-groups can be diverse. □

3.1 Construction of a DF-tree Structure

Our proposed DF-growth algorithm takes (i) a friend network F_{SN} and (ii) a user-specified *minDiv* threshold as two input parameters to construct a DF-tree as follows. It first scans F_{SN} to calculate $Freq_{D_j}(f_i)$ for each friend f_i in each domain D_j. For each f_i, DF-growth then uses $GMProm_D$ to compute the upper bound of the diversity value $Div^U(f_i)$, which is used to prune groups of friends who are not potentially diverse. Every potentially diverse friend f_i, along with its $Freq_{D_{1,\ldots,d}}(f_i)$, is stored in the header table.

Then, DF-growth scans F_{SN} the second time to capture the important information about potentially diverse friends in a user-defined order in the DF-tree. Each tree node consists of (i) a friend name and (ii) its frequency counters for all d domains in the respective path. The basic construction process of a DF-tree is similar to that of the FP-tree [10]. A key difference is that, rather than using only a single frequency counter capturing either the maximum or average frequency for all domains (which may lead to loss of information), we use d frequency counters capturing the frequency for all d domains. See Example 3.2.

EXAMPLE 3.2. To construct a DF-tree for F_{SN} shown in Table 2 when $minDiv$=0.5, DF-growth scans F_{SN} to compute (i) $GMProm_{D_{1,2,3}} = \langle 0.9, 0.7, 0.7\rangle$ for all d=3 domains, (ii) frequencies of each of the 7 friends in d=3 domains (e.g., $Freq_{D_{1,2,3}}(\{\text{Antonios}\}) = \langle 2, 0, 3\rangle$), (iii) upper bound of diversity values of all 7 friends (e.g., $Div^U(\{\text{Antonios}\}) = \frac{(0.9\times2)+(0.7\times0)+(0.7\times3)}{3} = 1.3$ using $Inf^U_{D_{1,2,3}}(\{\text{Antonios}\})$). Based on Lemma 3.1, we safely remove Zoe and Hebe having $Div^U(\{\text{Zoe}\})$=0.23 and $Div^U(\{\text{Hebe}\})$=0.23 both below *minDiv* as their super-groups cannot be diverse. So, the header table includes only the remaining 5 friends—sorted in some order (e.g., lexicographical order of friend names)—with their $Freq_{D_{1,2,3}}(\{f_i\})$. To facilitate a fast tree traversal, like the FP-tree, the DF-tree also maintains horizontal node traversal pointers from the header table to nodes of the same f_i.

Our DF-growth algorithm then scans each $L_j \in F_{SN}$, removes any friend $f_i \in L_j$ having $Div^U(f_i) <$*minDiv*, sorts

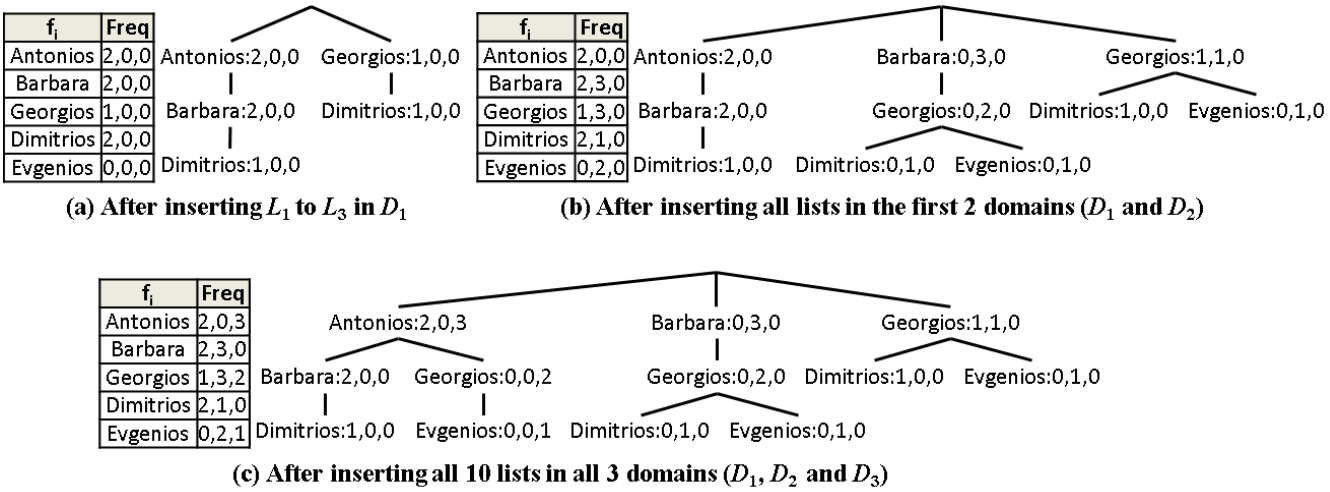

Figure 1: Construction of a DF-tree.

the remaining friends according to the order in the header table, and inserts the sorted list into the DF-tree. Each tree node captures (i) f_i representing the group G consisting of all friends from the root to f_i and (ii) its frequencies in each domain $Freq_{D_{1,2,3}}(G)$. For example, the rightmost node Evgenios:0,1,0 of the DF-tree in Figure 1(b) captures G={Georgios, Evgenios} and $Freq_{D_{1,2,3}}(G)$=$\langle 0,1,0\rangle$. Tree paths of common prefix (i.e., same friends) are shared, and their corresponding frequencies are added. See Figures 1(a), 1(b), and 1(c) for DF-trees after reading all interest-group lists in domain D_1, both D_1 and D_2, as well as the entire F_{SN}, respectively. □

With this tree construction process, the size of the DF-tree for F_{SN} with a given *minDiv* is observed to be bounded above by $\sum_{L_j \in F_{SN}} |L_j|$.

3.2 Mining of All Diverse Friend Groups

After constructing the DF-tree, our DF-growth algorithm recursively mines/discovers diverse friend groups by building projected and conditional trees in a fashion similar to that of FP-growth [10].

Recall that $Div(G)$ computed based on $Prom_D(G)$ does *not* satisfy the downward closure property. To facilitate pruning, we use $GMProm_D(f_i)$ to compute $Div^U(f_i)$, which then satisfies the downward closure property. However, if $Div^U(G)$ was computed as an upper bound to super-group G of f_i, then it may overestimate diversity of G and may lead to false positives. To reduce the number of false positives, DF-growth uses the **local maximum prominence value** $\boldsymbol{LMProm_D(G)} = \max_{f_i \in F_D^G}\{Prom_D(G)\}$ for the projected and conditional trees for G. See Lemma 3.2 and Example 3.3.

LEMMA 3.2. *The diversity value of a friend group G computed based on $LMProm_D(G)$ is a tighter upper bound than $Div^U(G)$ computed based on $GMProm_D$.* □

EXAMPLE 3.3. Let us continue Example 3.2. To mine potentially diverse friend groups from the DF-tree in Figure 1(b) using $minDiv = 0.5$, DF-growth first builds the

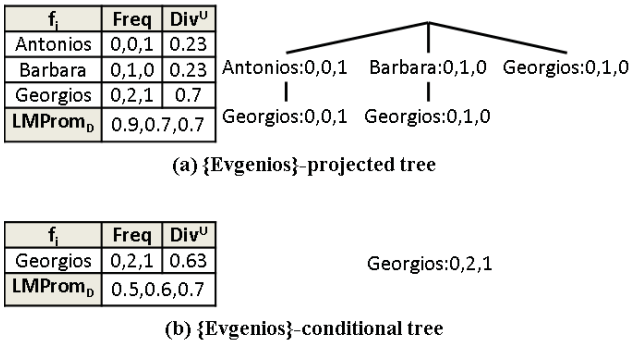

Figure 2: Tree-based mining of diverse friend groups.

{Evgenios}-projected tree—as shown in Figure 2(a)—by extracting the paths ⟨Antonios, Georgios, Evgenios⟩:0,0,1, ⟨Barbara, Georgios, Evgenios⟩:0,1,0 and ⟨Georgios, Evgenios⟩:0,1,0 from the DF-tree in Figure 1(b). For $F_{D_{1,2,3}}^{\text{Evgenios}}$ = {Antonios, Barbara, Georgios, Evgenios}, our DF-growth algorithm uses $LMProm_{D_{1,2,3}}(F_{D_{1,2,3}}^{\text{Evgenios}}) = \langle 0.9,\ 0.7, 0.7\rangle$ to compute the tightened $Div^U(G)$ such that the tightened $Div^U(\{\text{Antonios, Evgenios}\}) = \frac{(0.9\times 0)+(0.7\times 0)+(0.7\times 1)}{3}$=0.23 < *minsup*.

As $Div^U(\{\text{Antonios, Evgenios}\})$ and $Div^U(\{\text{Barbara, Evgenios}\})$ are both below *minsup*, DF-growth prunes Antonios and Barbara from the {Evgenios}-projected tree to get the {Evgenios}-conditional tree as shown in Figure 2(b). Due to pruning, our DF-growth algorithm recomputes (i) the local maximum prominence value $LMProm_{D_{1,2,3}}(F_{D_{1,2,3}}^{\text{Evgenios}})$ =$\langle 0.5, 0.6, 0.7\rangle$ and (ii) the tightened $Div^U(\{\text{Georgios, Evgenios}\}) = \frac{(0.5\times 0)+(0.6\times 2)+(0.7\times 1)}{3} = 0.63$ for the updated $F_{D_{1,2,3}}^{\text{Evgenios}}$ = {Georgios, Evgenios}. This completes the mining for {Evgenios}.

Next, DF-growth builds {Dimitrios}-, {Georgios}- and {Barbara}-projected trees as well as their conditional trees, from which potentially diverse friend groups can be mined. Finally, our DF-growth algorithm computes the true diversity value $Div(G)$ for each of these mined groups to check if it is truly diverse (i.e., to remove all false positives). □

Table 3: Dataset characteristics

Dataset	n=#transactions	m=#domain items	Max trans. length	Avg trans. length	Density
mushroom	8,124	119	23	23.0	Dense
T10I4D100K	100,000	870	29	10.1	Sparse
kosarak	990,002	41,270	2498	8.1	Sparse

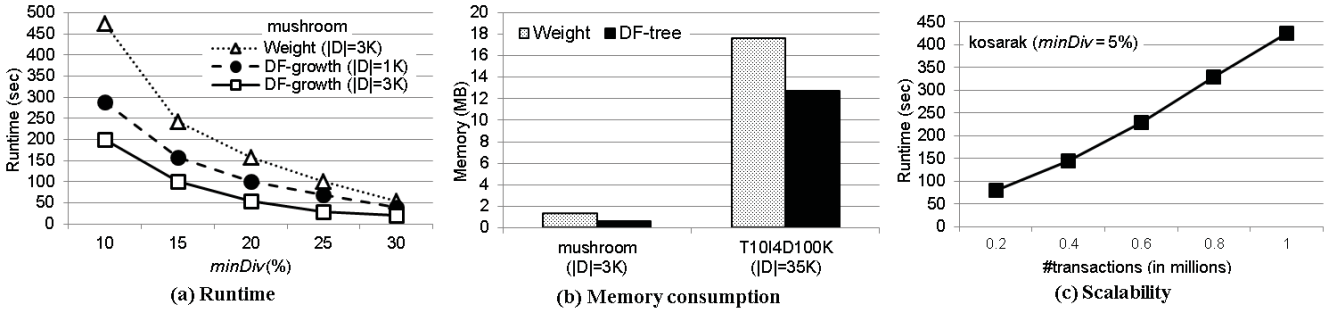

Figure 3: Experimental results.

3.3 Removal of Non-diverse Friend Groups

Our DF-growth algorithm makes good use of the global and local maximum prominence values of friend groups as upper bounds to diversity values of friend groups. Consequently, the algorithm discovers all truly diverse friend groups (i.e., *no* false negatives). However, it also discover some "potentially diverse" friend groups that are not truly diverse (i.e., some false positives). Hence, as its final step, our algorithm computes the true diversity values $Div(G)$ for each of these mined groups to check if it is truly diverse (i.e., to remove all false positives).

Example 3.4. Let us continue Example 3.3. After mining potentially diverse friend groups from {Evgenios}-, {Dimitrios}-, {Georgios}- and {Barbara}-projected trees as well as their conditional trees, our DF-growth algorithm computes the true diversity value $Div(G)$ for each of the mined groups to check if it is truly diverse (i.e., to remove all false positives). □

4. EXPERIMENTAL EVALUATION

To evaluate the effectiveness of our proposed DF-growth algorithm and its associated DF-tree structure, we compared them with a closely related *weighted* frequent pattern mining algorithm called Weight [24] (but it does not use different weights for individual items). As Weight was designed for frequent pattern mining (instead of social network mining), we apply those datasets commonly used in frequent pattern mining for a fair comparison: (i) IBM synthetic datasets (e.g., T10I4D100K) and (ii) real datasets (e.g., mushroom, kosarak) from the Frequent Itemset Mining Dataset Repository (`http://fimi.ua.ac.be/data`). See Table 3 for more detail. Items in transactions in these datasets are mapped into friends in interest-group lists. To reflect the concept of *domains*, we subdivided the datasets into several batches. Moreover, a random number in the range (0, 1] is generated as a prominence value for each friend in every domain.

All programs were written in C++ and run on the Windows XP operating system with a 2.13 GHz CPU and 1 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 case. We obtained consistent results for all of these datasets.

4.1 Runtime

First, we compared the runtime of DF-growth (which includes the construction of the DF-tree, the mining of potentially diverse friend groups from the DF-tree, and the removal of false positives) with that of Weight. Figure 3(a) shows the results for a dense dataset (mushroom), which were consistent with those for sparse datasets (e.g., T10I4D100K). Due to page limitation, we omit the results for sparse datasets. But, runtimes of both algorithms increased when mining larger datasets (social networks), more batches (domains), and/or with lower *minDiv* thresholds. Between the two algorithms, our tree-based DF-growth algorithm outperformed the Apriori-based Weight algorithm. Note that, although FP-growth [10] is also a tree-based algorithm, it was *not* design to capture weights. To avoid distraction, we omit experimental results on FP-growth and only show those on Weight (which captures weights).

4.2 Memory Consumption

Second, we evaluated the memory consumption. Figure 3(b) shows the amount of memory required by our DF-tree for capturing the content of social networks with the lowest *minDiv* threshold (i.e., without removing any friends who were not diverse). Although this simulated the worst-case scenario for our DF-tree, DF-tree was observed (i) to consume a reasonable amount of memory and (ii) to require less memory than Weight (because our DF-tree is compact due to the prefix sharing).

4.3 Scalability

Third, we tested the scalability of our DF-growth algorithm by varying the number of transactions (interest-group lists). We used the kosarak dataset as it is a huge sparse dataset with a large number of distinct items (individual users). We divided this dataset into five portions, and each por-

tion is subdivided into multiple batches (domains). We set *minDiv*=5% of each portion. Figure 3(c) shows that, when the size of the dataset increased, the runtime also increased proportionally implying that DF-growth is scalable.

4.4 Summary on Evaluation Results

Experimental results on (i) runtime, (ii) memory consumption (which reveals tree compactness) and (iii) scalability showed that our DF-growth algorithm is time- and space-efficient as well as scalable. As ongoing work, we plan to evaluate the *quality* (e.g., precision) of DF-growth in finding diverse friend groups. Moreover, for a fair comparison with Weight, we have used those datasets that are commonly used in frequent pattern mining. As ongoing work, we plan to evaluate DF-growth using real-life social network datasets.

5. CONCLUSIONS AND FUTURE WORK

In this paper, we (i) introduced a new notion of *diverse friends* for social networks, (ii) proposed a compact tree structure called *DF-tree* to capture important information from social networks, and (iii) designed a tree-based mining algorithm called *DF-growth* to find diverse (groups of) friends from social networks. Diversity of friends is measured based on their prominence, frequency and influence in different domains on the networks. Although diversity does not satisfy the downward closure property, we managed to address this issue by using the global and local maximum prominence values of users as upper bounds. Experimental results showed that (i) our DF-tree is compact and space-effective and (ii) our DF-growth algorithm is fast and scalable for both sparse and dense datasets. As ongoing work, we conduct more extensive experimental evaluation with various datasets (e.g., real-life social network datasets) and to measure other aspects (e.g., precision) of our DF-growth algorithm in finding diverse friends. We also plan to (i) design a more sophisticated way to measure influence and (ii) incorporate other computational metrics (e.g., popularity, significance, strength) with prominence into our discovery of useful information from social networks.

6. ACKNOWLEDGEMENTS

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

7. REFERENCES

[1] R. Agrawal & R. Srikant. Fast algorithms for mining association rules in large databases. In *Proc. VLDB 1994*, pp. 487–499.

[2] A. Anagnostopoulos, R. Kumar, & M. Mahdian. Influence and correlation in social networks. In *Proc. ACM KDD 2008*, pp. 7–15.

[3] D. Bianchini, V. de Antonellis, & M. Melchiori. A linked data perspective for collaboration in mashup development. In *Proc. DEXA Workshops 2013*, pp. 128–132.

[4] D. Bianchini, V. de Antonellis, & M. Melchiori. Exploiting social tagging in web API search. In *Proc. OTM 2013*, pp. 764–771.

[5] J.J. Cameron, C.K. Leung, & S.K. Tanbeer. Finding strong groups of friends among friends in social networks. In *Proc. SCA 2011*, pp. 824–831.

[6] A. Cuzzocrea & F. Folino. Community evolution detection in time-evolving information networks. In *Proc. EDBT/ICDT 2013 Workshops* (*LWDM*), pp. 93–96.

[7] A. Cuzzocrea, F. Folino, & C. Pizzuti. *DynamicNet*: an effective and efficient algorithm for supporting community evolution detection in time-evolving information networks. In *Proc. IDEAS 2013*, pp. 148–153.

[8] R. de Virgilio & A. Maccioni. Generation of reliable randomness via social phenomena. In *Proc. MEDI 2013*, pp. 65–77.

[9] A. Ferrara, L. Genta, & S. Montanelli. Linked data classification: a feature-based approach. In *Proc. EDBT/ICDT 2013 Workshops* (*LWDM*), pp. 75–82.

[10] J. Han, J. Pei, & Y. Yin. Mining frequent patterns without candidate generation. In *Proc. ACM SIGMOD 2000*, pp. 1–12.

[11] F. Jiang, C.K. Leung, & S.K. Tanbeer. Finding popular friends in social networks. In *Proc. SCA 2012*, pp. 501–508.

[12] K.Y. Kamath, J. Caverlee, Z. Cheng, & D.Z. Sui. Spatial influence vs. community influence: modeling the global spread of social media. In *Proc. ACM CIKM 2012*, pp. 962–971.

[13] W. Lee, C.K. Leung, J.J. Song, & C.S. Eom. A network-flow based influence propagation model for social networks. In *Proc. SCA 2012*, pp. 601–608.

[14] C.K. Leung & C.L. Carmichael. Exploring social networks: a frequent pattern visualization approach. In *Proc. IEEE SocialCom 2010*, pp. 419–424.

[15] N. Pernelle & F. Saïs. Classification rule learning for data linking. In *Proc. EDBT/ICDT 2012 Workshops* (*LWDM*), pp. 136–139.

[16] Y. Ruan, D. Fuhry, & S. Parthasarathy. Efficient community detection in large networks using content and links. In *Proc. ACM WWW 2013*, pp. 1089–1098.

[17] M. Schaal, J. O'Donovan, & B. Smyth. An analysis of topical proximity in the twitter social graph. In *Proc. SocInfo 2012*, pp. 232–245.

[18] S.K. Tanbeer, F. Jiang, C.K. Leung, R.K. MacKinnon, & I.J.M. Medina. Finding groups of friends who are significant across multiple domains in social networks. In *Pro. CASoN 2013*, pp. 21–26.

[19] S.K. Tanbeer, C.K. Leung, & J.J. Cameron. DIFSoN: discovering influential friends from social networks. In *Proc. CASoN 2012*, pp. 120–125.

[20] Z. Wu, W. Yin, J. Cao, G. Xu, & A. Cuzzocrea. Community detection in multi-relational social networks. In *Proc. WISE 2013*, pp. 43–56.

[21] T. Yang, P.M. Comar, & L. Xu. Community detection by popularity based models for authored networked data. *Proc. IEEE/ACM ASONAM 2013*, pp. 74–81.

[22] Q. Yuan, G. Cong, Z. Ma, A. Sun, & N. Magnenat-Thalmann. Who, where, when and what: discover spatio-temporal topics for twitter users. In *Proc. ACM KDD 2013*, pp. 605–613.

[23] C. Zhang, L. Shou, K. Chen, G. Chen, & Y. Bei. Evaluating geo-social influence in location-based social networks. In *Proc. ACM CIKM 2012*, pp. 1442–1451.

[24] S. Zhang, C. Zhang, & X. Yan. Post-mining: maintenance of association rules by weighting. *Information Systems*, **28**(7), pp. 691–707 (2003).