GuruMine: a Pattern Mining System for Discovering Leaders and Tribes

Amit Goyal[1] Francesco Bonchi[2] Laks V. S. Lakshmanan[1] Byung-Won On[1]

[1]*Computer Science Department, University of British Columbia*
Vancouver, BC, Canada
{goyal,bwon,laks}@cs.ubc.ca

[2]*Yahoo! Research Barcelona, Spain*
bonchi@yahoo-inc.com

***Abstract*— In this demo we introduce GuruMine, a pattern mining system for the discovery of *leaders*, i.e., influential users in social networks, and their *tribes*, i.e., a set of users usually influenced by the same leader over several actions.**

GuruMine is built upon a novel pattern mining framework for leaders discovery, that we introduced in [1]. In particular, we consider social networks where users perform actions. Actions may be as simple as tagging resources (urls) as in del.icio.us, rating songs as in Yahoo! Music, or movies as in Yahoo! Movies, or users buying gadgets such as cameras, handholds, etc. and blogging a review on the gadgets. The assumption is that actions performed by a user can be seen by their network friends. Users seeing their friends' actions are sometimes tempted to perform the actions themselves. On the basis of the propagation of such influence, in [1] we proposed various notions of leaders and developed algorithms for their efficient discovery.

GuruMine provides users with a friendly and intuitive graphical interface for selecting the actions of interest, and the kind of leaders to mine. The set of parameters driving the pattern discovery process can be iteratively refined, and the result is updated, without incurring a completely new computation whenever possible. Once a set of leaders has been extracted, GuruMine can easily validate them on a set of actions unseen during the pattern mining, by analyzing the portion of network reached by the influence of the selected leaders, on the unseen actions. GuruMine also offers various visualizations over social networks: the propagation of an action, the leaders, their tribes, and the interactions between different leaders and tribes. In this demo we will show: (i) how the pattern mining process can be driven towards the discovery of a good set of leaders, (ii) the ease of use of GuruMine system, and (iii) its outstanding performances on large real-world social networks and actions databases.

I. Introduction

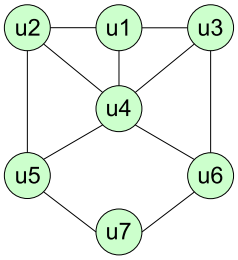
In this demo we introduce GuruMine, a pattern mining system for the discovery of influential users in social network. The basic assumption underlying GuruMine is the well known *word-of-mouth* effect, thanks to which actions, opinions, buying behaviors, innovations and so on, propagate in a social network. Seeing actions performed by their friends may make individuals perform those actions: we can think of this as a (potentially recursive) *influence* propagating from users to their network friends. If such influence patterns repeat with some statistical significance, that can be of interest to companies, for targeted advertising campaigns. This is known as *viral marketing* [2], [3], [4]: by targeting the most influential users in a social network we can activate a chain-reaction of influence driven by word-of-mouth, in such a way that with a very small marketing cost we can actually reach a very large portion of the network. Viral marketing models are based on probabilistic causation: there is a probability with which a user will perform an action if his neighbors have performed it. Thus such models require as input the social graph together with the edge-weights representing the probabilities of influence.

GuruMine is based on a different framework, that we defined in [1]. Given a social graph together with a log of user actions, we can determine the propagation of influence for performing each of the actions. We ask: how can we leverage this for determining who are the leaders in setting the trend for performing actions? And given the leaders, which other users form their tribes? In [1] we tackle these questions as a pattern discovery problem. We define various notions of leader, tribe leader, and their confident and genuine variants. We develop efficient algorithms for extracting leaders of various flavors from an input data consisting of a social graph and a table of user actions.

Based on this framework GuruMine is a useful system for an exploratory analysis of influence in social networks. GuruMine allows us:

- to discover the leaders and their tribes at a very local scale;
- to iteratively fine-tune parameters that control the definition of leaders according to the given application;
- to determine the number of leaders that we actually want to select;
- to validate the selected leaders on a set of actions unseen during the pattern mining, by analyzing the portion of network reached by the influence of the selected leaders, on the unseen actions;
- to visualize the propagation of an action, the leaders, their tribes, and the interactions between different leaders and their tribes.

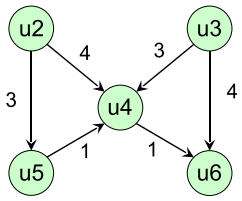
A key novel aspect of the demo is the ability to visualize and interact with various patterns entirely in terms of graphs.



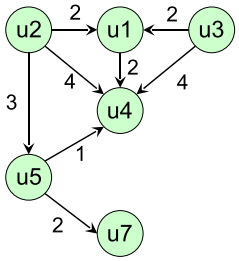
(a)

User	Action	Time
u3	b	13
u2	b	13
u2	a	15
u1	b	15
u3	a	16
u5	b	16
u4	b	17
u5	a	18
u7	b	18
u4	a	19
u6	a	20

(b)



(c)



(d)

Fig. 1. (a) Example social graph; (b) A log of actions; (c) Propagation of action a and (d) of action b.

II. A Pattern Mining Framework

In this section we provide the background on the pattern mining framework underlying GuruMine.

A *social graph* is an undirected graph $G = (V, E)$ where the nodes are users, and edges represent social ties between the users. The tie may be explicit in the form of declared friendship, or it may be derived on the basis of shared interests between users. An *action log* is a relation $Actions(User, Action, Time)$, which contains a tuple (u, a, t) indicating that user u performed action a at time t. We will assume that the projection of $Actions$ on the first column is contained in the set of nodes V of the social graph G. In other words, users in the $Actions$ table correspond to nodes of the graph. We let $\mathcal{A}$ denote the universe of actions. We say that an action $a \in \mathcal{A}$ *propagates* from user v_i to v_j iff $(v_i, v_j) \in E$ and $\exists (v_i, a, t_i), (v_j, a, t_j) \in Actions$ with $t_i < t_j$. Notice that there must be a social tie between v_i and v_j, both must have performed the action, one strictly before the other. This leads to a natural notion of a propagation graph. For each action a, we define a *propagation graph* $PG(a) = (V(a), E(a))$ as follows. $V(a) = \{v \mid \exists t : (v, a, t) \in Actions\}$; there is a directed edge $v_i \xrightarrow{\Delta t} v_j$ whenever a propagates from v_i to v_j, with $(v_i, a, t_i), (v_j, a, t_j) \in Actions$, where $\Delta t = t_j - t_i$. Figure 1(a)-(b) shows an example of a social graph and a log of actions performed by users. By linking the actions log in Figure 1(b) to the social graph in Figure 1(a), we can trace the propagation of influence for performing actions. The result, for actions a and b, is shown in Figure 1(c)-(d). Observe that the graphs in Figure 1(c)-(d) are directed and edges labelled even though the original social graph is undirected and edges unlabelled. Direction is dictated by the order in which actions were performed. If x performed action a before y and there is a social tie between x and y in Figure 1(a), then Figure 1(c) contains a directed edge from x to y; while the label on the edge is given by the time elapsed between the two actions. In other words, the propagation of an action is just a directed instance (a flow) of the undirected graph G, and the log of actions $Actions(User, Action, Time)$ can be seen as a collection of propagations. Influence can propagate transitively. Thus, a definition of leader w.r.t. setting the trend for performing an action should take this into account. To aid in the definition of leaders, we define the notion of an influence graph next. The elapsed time along a (directed) path in a propagation graph is the sum of edge labels along the path. E.g., in Figure 1(c), the elapsed time on the path $u2 \rightarrow u4 \rightarrow u6$ is $4 + 1 = 5$.

Given action a, a user u, and a maximum propagation time threshold π, we define the *influence graph* of the user u, denoted $Inf_\pi(u, a)$, as the subgraph of $PG(a)$ rooted at u, such that it consists of those nodes of $PG(a)$ which are reachable from u in $PG(a)$ and such that every path from u to any other node in $Inf_\pi(u, a)$ has an elapsed time at most π. The propagation time threshold π allows us to set limits on how long after an action is performed, we regard another user performing that action as influenced by the previous user. Given a threshold ψ, we say that user u acted as a *leader w.r.t. action* a whenever the size (in number of nodes) of $Inf_\pi(u, a)$ is at least ψ. The threshold ψ ensures sufficiently many users are influenced by the given user in the context of action a.

A user to be identified as a leader must act as such sufficiently often, i.e., for a number of actions larger than a given action threshold σ. This may be seen as the *minimum frequency* constraint in pattern discovery and association rule mining [5].

Definition 1 (Leaders): Given a set of actions $I \subseteq A$, and three thresholds π, ψ and σ, a user $v \in V$ is a *leader* iff:

$$\exists S \subseteq I, |S| \geq \sigma : \forall a \in S \,.\, size(Inf_\pi(v, a)) \geq \psi$$

In Figure 1, if we choose the time bound to be $\pi = 5$ units, number of users to be influenced by a leader to be $\psi = 3$, and number of actions in which this happens to be $\sigma = 2$, then user $u2$ is a leader w.r.t. both actions a and b, since users $u4, u5, u6$ are influenced by $u2$ in Figure 1(c), while $u1, u4, u5, u7$ are influenced by $u2$ in Figure 1(d). Notice that $u3$ is not a leader w.r.t. either action. If user $u5$ had performed b after $u4$, then $u3$ would be regarded a leader w.r.t. b, in addition to $u2$.

A stronger notion of leadership might be based on requiring that w.r.t. each of a class of actions of interest, the set of influenced users must be the same. To distinguish from the notion of leader illustrated above, we refer to this notion as *tribe leader*, meaning the user leads a fixed set of users (tribe) w.r.t. a set of actions. Clearly, tribe leaders are leaders but not vice versa.

Definition 2 (Tribe-leaders): Given a set of actions $I \subseteq A$,

and thresholds π, ψ and σ, a user $v \in V$ is a *tribe leader* iff:

$$\exists S \subseteq I, |S| \geq \sigma, \exists U \subset V, |U| \geq \psi : \forall a \in S \,.\, U \subseteq Inf_{\pi}(v, a).$$

In addition to using an absolute threshold on the number of actions in which a user acts as a leader, we could apply a "confidence threshold", similarly to the classical measure of confidence in association rules [5]. More precisely, for a user $v \in V$, let $P(v) = \{a \in \mathcal{A} \mid v \text{ performed } a\}$ and $L(v) = \{a \in \mathcal{A} \mid v \text{ is a leader w.r.t. } a\}$. Then the *leadership confidence* of v is the ratio $conf(v) = |L(v)|/|P(v)|$. Given a set of actions $I \subseteq \mathcal{A}$, and a confidence threshold $0 < \varphi \leq 1$, a user v is said to be a *confidence leader* if it is a leader and $conf(v) \geq \varphi$.

It may happen that one user acts as a leader according to the problems defined so far, but in concrete he is always a follower of the same leader. In some sense he benefits from the influence of a true leader, so that he also may seem a leader. To avoid this kind of "fake" leaders, we propose the following. Given the usual three thresholds π, ψ and σ, for a user v, let $gen(v)$ denote the ratio

$$\frac{|\{a \in L(v) \mid \nexists u \in V : u \text{ is leader for } a \wedge v \in Inf_{\pi}(u, a)\}|}{|L(v)|}$$

i.e., the fraction of actions led by v for which v's leadership is genuine, in that it is not a consequence of v being present in the influence graph of some other leader w.r.t. that action. We call $gen(v)$ the genuineness score of v. Given a set of actions $I \subseteq \mathcal{A}$, and a threshold $0 < \gamma \leq 1$, we define a leader v to be a *genuine leader* provided the genuineness score of v is above the threshold, i.e., $gen(v) \geq \gamma$.

Both confidence and genuineness constraints can be applied to tribe leaders as well. Note that when we focus on a single action a, the notions of leader for a and tribe leader for a coincide. Thus we use the exact same definitions. So a genuine tribe leader is a tribe leader whose genuineness score is above a threshold, and similarly for confidence. Our goal is to efficiently extract leaders and tribe leaders, possibly with each of the other criteria (confidence and genuineness) or even with both.

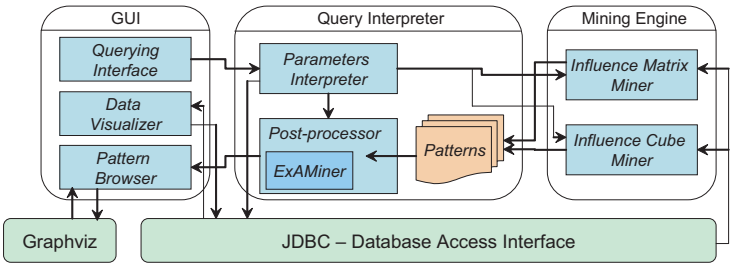


Fig. 2. GuruMine architecture.

In this section we provided the needed background about the pattern mining framework underlying GuruMine. However, it should be noted that it is not mandatory for GuruMine's user to provide all the thresholds: she can query our system asking, e.g., the top-k users w.r.t. average number of followers per action given π, or the top-k tribe leaders w.r.t. confidence, as will be discussed in the following section.

III. The GuruMine system overview

The GuruMine architecture (see Figure 2) is composed of three modules (GUI, query interpreter, mining engine), plus an underlying DBMS which interacts with all the three modules. For sake of compatibility, the coupling between the core mining system and the underlying DBMS, have been realized by means of JDBC [6], an API that provides cross-DBMS connectivity to a wide range of SQL databases. This means that, wherever the underlying data is stored, GuruMine can be easily connected to that database.

A. The Mining Engine

GuruMine mining engine is based on the algorithms we developed in [1]. Given a social graph and a table of user actions, our algorithms can discover leaders of various flavors by making one pass over the actions table. This makes our algorithms scalable to very large input graphs and action logs. By sliding a time window of width π chronologically backwards on the actions log table (see [1] for more details) we can compute an *influence matrix* $IM_{\pi}(U, A)$, where U is the number of users and A is the number of actions. The entry $IM_{\pi}(u, a)$ is the number of users/nodes, influenced by u w.r.t. action a within time π. This number includes u. So, a user u performed action a iff $IM_{\pi}(u, a) > 0$. Then leaders can be computed from the IM_{π} easily. When it comes to computing tribe leaders, influence matrix is inadequate: for tribe leaders, we need to check that a fixed set of $\geq \psi$ users were influenced by the leader on sufficiently many actions. To address this problem, we compute an *influence cube*. The influence cube is a $User \times Action \times User$ cube with cells containing boolean entries: $IC_{\pi}(u, a, v) = 1$ if user v was influenced by user u w.r.t. action a, w.r.t. an underlying time threshold π. Post-processing the influence matrix or the influence cube for producing the final result is done by the *Post-processor* module, within the query interpreter.

B. The Query Interpreter

The second module takes care of interpreting the given query, i.e., taking the various parameters, retrieving from the underlying DBMS the source data, and passing them to the correct module. In particular, it must inform the mining engine: (1) what to compute – the influence matrix (for standard leaders) or the influence cube (for tribe leaders), (2) the value of parameter π, and (3) which data is to be used. All the other parameters will be passed on to the post-processor which takes care of assembling the final result to and passes it on to the *Pattern Browser* module in the GUI. The post-processor computes leaders by a simple scan of the rows of the influence matrix. In the case of tribe leaders, they are extracted from the influence cube $IC_{\pi}(u, a, v)$ by means of frequent itemsets mining [5]. Notice that we are *not* interested in all frequent itemsets, but only in those ones whose size is ψ or more. For this purpose, we use ExAMiner [7], a special algorithm optimized for our needs.

C. Incremental Mining

Our system supports incremental analysis and analysis and exploration on two levels. All our mining algorithms have

been developed to support incremental mining. E.g., if more actions are added to the actions log, our algorithms can just go over the additional action tuples and update the mined patterns. In addition, the query interpreter of GuruMine also implements a caching system, so that some incremental queries can be answered without redoing the whole computation from scratch. Whenever an influence matrix or cube is computed, GuruMine also stores its metadata: i.e., the parameter π, and the data set from which it as been computed. If a new query arrives that is defined over the same data set and with the same value for parameter π but with the other parameters changed, the query interpreter will pass the request directly to the post-processor, that will be retrieve the previously computed matrix or cube without activating the mining engine, and compute the required patterns.

D. Graphical User Interface

One of the highlights of the system is that the patterns are visualized in the form of graphs. The user can explore these patterns directly by interacting with the graphs. We describe and illustrate the graphs and the types of interactions supported below.

1. Leaders Interaction Graph: In this graph, all the leaders will be displayed along with the interactions between them. Nodes will represent leaders and will be color-coded. The intensity of the color is propotional to number of followers, while the size of node (diameter of the circle) is propotional to confidence. This will give a feel for the extent of influence of a leader at a quick glance. Edges are directed and represent interaction between leaders in the form of flow of influence. Solid edges assert that there is a direct social tie between the leader nodes in the original social graph. Broken edges capture the fact that there is no direct social tie between the leader nodes. They are connected by a path where the intermediate nodes themselves are not leaders. Edges will also be color-coded. Stronger intensity of color of edge (u, v) implies node v is influenced by node u for more actions.

2. Leaders Influence Graph: This graph will represent the influence of one leader at a time, with leader as the root. All nodes except the root represent followers of the leader. Edges are directed and represent the flow of influence. Edges are color-coded with intensity of the color being proportional to number of actions for which the influence propagates (as defined above). Stronger the intensity of color means of edge $u, v)$, the more the actions for which v is influenced by u. The graph will be animated. That is, the flow of influence would be shown in real time.

3. Propagation Graph for an Action: This will show the interaction between several leaders and their followers for a particular action. The action can be chosen by the user. Nodes include leaders and followers for the chosen action. Leader nodes will be color coded, Followers nodes will be simply black. Intensity of the color is proportional to number of followers. Similarly size of the node represents confidence. We can also use colors to distinguish between tribe ;leaders and non-tribe leaders. Edges are directed and represent the flow of influence. There is no color-coding of edges in this case. This graph will be animated. That is, the flow of influence would be shown in real time.

Interacting with the Graphs: The user can ask for any of the three types of graphs from the GUI. There will be a zoom/scroll facility to manipulate displayed graphs which can help the user visualize it effectively depending on its size. Once a graph is displayed, she can change the values of the parameters. As mentioned earlier, the changed output graph will be computed incrementally and quickly returned to the user. Additionally, the user may navigate from one graph to another. Here are some examples. The user can double-click on a node (leader) in the leaders interaction graph and thus see the leader influence graph for the chosen leader. In the latter graph, she may choose to double-click on an edge and see what the actions are for which influence propagates from one node to another and then double-click on one of the displayed actions. This will take her to the propagation graph for the chosen action. She can also visualize the propagation graph for more than one action and study their commonalities and differences. The user can impose genuineness threshold as a constraint using a slider bar. As the threshold is adjusted existing nodes/edges may drop off or new ones may pop up.

IV. What will be demonstrated

Our demo will show GuruMine at work on a real-world social network and actions dataset. The demo will show how, thanks to the friendly and intuitive GUI and the efficient mining engine, the exploratory pattern mining process (i.e. human-guided, interactive, iterative, and visual) can be driven towards the discovery of a good set of leaders. The importance of the efficient exploratory mining together with incremental computing will be also highlighted. We will show how to visually browse through the leaders and their tribes, how to validate a set of leaders on new, unseen actions, and how to visualize the propagation of their influence.

References

[1] A. Goyal, F. Bonchi, and L. V. S. Lakshmanan, "Discovering leaders from community actions," Computer Science Department, University of British Columbia Vancouver, BC, Canada, Tech. Rep. 7, June 2008, Submitted for publication.

[2] P. Domingos and M. Richardson, "Mining the network value of customers," in *Proc. of the Seventh ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD'01)*.

[3] M. Richardson and P. Domingos, "Mining knowledge-sharing sites for viral marketing," in *Proc. of the Eighth ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD'02)*.

[4] D. Kempe, J. M. Kleinberg, and É. Tardos, "Maximizing the spread of influence through a social network," in *Proc. of the Ninth ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD'03)*.

[5] R. Agrawal, T. Imielinski, and A. N. Swami, "Mining association rules between sets of items in large databases," in *Proc. of the 1993 ACM SIGMOD Int. Conf. on Management of Data (SIGMOD'93)*.

[6] `http://java.sun.com/javase/technologies/database/`.

[7] F. Bonchi, F. Giannotti, A. Mazzanti, and D. Pedreschi, "ExAMiner: Optimized level-wise frequent pattern mining with monotone constraints," in *Proceedings of the Third IEEE International Conference on Data Mining (ICDM'03)*.