2015 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology

A Sliding Window-Based Algorithm for Detecting Leaders from Social Network Action Streams

Quazi Marufur Rahman*, Anna Fariha*, Amit Mandal*, Chowdhury Farhan Ahmed*, and Carson K. Leung†
*Department of Computer Science and Engineering, University of Dhaka, Bangladesh
Email: maruf.csdu@gmail.com, anna@cse.univdhaka.edu, amitducse17@gmail.com, farhan@cse.univdhaka.edu
†Department of Computer Science, University of Manitoba, Winnipeg, MB, Canada
Email: kleung@cs.umanitoba.ca

***Abstract*—Influential users or leaders in a social network play important roles in viral marketing by spreading news quickly to a large number of people. Hence, various organizations aim to discover these leaders as campaign targets for advertisement so as to maximize customer reachability. Existing approaches detect leaders from a static social network. However, as social networks are evolving, detecting leaders from dynamic streams of social network data is in demand. In this paper, we propose a sliding window-based leader detection (SWLD) algorithm for discovering leaders from streams of user actions in social networks. Experimental results show that SWLD is accurate, requires short runtime and a small amount of memory space.**

***Keywords*-Web intelligence; social network mining; data stream mining**

I. Introduction

Social network (SN) services (e.g., Facebook, Google+, Twitter) are the platforms to build social community among users who share common interests or real-life connections (e.g., friendships). These services allow users to interact with each other as friends. These users and their connections can be captured in an undirected graph $G = (V, E)$—where (i) V captures users and (ii) E captures their connections—in which Big social network analytics and mining [7], [9], [10], [13] can be applied for finding useful patterns.

When a user u shares some interesting information in a social network, some of his friends (say, users v_1 and v_2) may then share the same information (e.g., v_1 and v_2 may retweet u's tweets). In this action, the user (e.g., u) who influences other users in the network can be considered as a *leader*. Finding these leaders from the social network is in demand because they play a vital role in real-life marketing campaigns by distributing news in the network within a short time span. Common techniques to find these leaders include *influence path mining*, which also mines how influences propagate in a social network. Specifically, for users u and v connected in a social network, if a user v performs an action a after u performing A, then u is considered to be influential as he influences v to perform a. In such a case, a directed edge $u \xrightarrow{a} v$ denotes the the influence path from u to v for that particular action a. A user is considered to be a *leader* in a network if he influences more than a user-specified number of users.

A recent algorithm [12] for leader discovery uses a *static* social network graph, together with a *static* user action log. However, as social networks keep evolving, it is more realistic to use a *dynamic* graph. Hence, our key contribution of this paper is our proposal of a new algorithm called **SWLD**, which uses a **s**liding **w**indow for **l**eader **d**etection from dynamic streams of user actions.

The remainder of this paper is organized as follows. The next section presents related works. Section III describes our sliding window-based leader detection algorithm. Experimental results and conclusions are given in Sections IV and V, respectively.

II. Related Works

Regarding works related to our proposal of a *sliding window-based* algorithm for leader detection from *dynamic streams of user actions in a social network*, Matsumura and Sasaki [14] discovered leadership behaviors from *human influence networks*. Song et al. [6] proposed the InfluenceRank algorithm to detect opinion leaders from *blogs*; they also found the importance of the propagated information spread through the network. Goyal et al. [2] used a sliding window-based algorithm to discover leaders from *community actions*. Bodendorf and Kaiser [1] used a *text mining approach* to detect leaders based on the shared text data in the network. Esslimani et al. [4] detected reliable leaders from *behavioral networks* by considering the high connectivity and influence propagation potential of users in the network. Lee et al. [8] proposed a network-flow based influence propagation model for static social networks. Braun et al. [3] proposed a tree-based algorithm for mining *diverse users* from a social network. Leung et al. [11] interactively mined influential friends from social networks that are *updated incrementally* (than from dynamic streams of user actions). Duan et al. [15] applied a user *clustering and sentiment analysis technique* to identify opinion leaders. Fariha et al. [5] focused on mining frequent interaction *patterns* (than users). Recently, the Apriori Probabilistic Path Mining (APPM) algorithm [12] was proposed to mine a static social network graph and

978-1-4673-9618-9/15 $31.00 © 2015 IEEE
DOI 10.1109/WI-IAT.2015.24

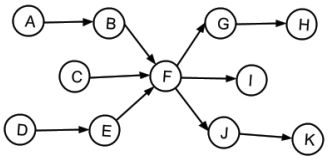


Figure 1. A propagation graph of 11 users who perform the same action in a social network.

Table I
LENGTH-2 & LENGTH-3 INFLUENTIAL PATHS INVOLVING USER F

2-length paths	3-length paths
1. $A \rightarrow B \rightarrow F$	1. $A \rightarrow B \rightarrow F \rightarrow G$
2. $B \rightarrow F \rightarrow G$	2. $A \rightarrow B \rightarrow F \rightarrow I$
3. $B \rightarrow F \rightarrow I$	3. $A \rightarrow B \rightarrow F \rightarrow J$
4. $B \rightarrow F \rightarrow J$	4. $B \rightarrow F \rightarrow G \rightarrow H$
5. $C \rightarrow F \rightarrow G$	5. $B \rightarrow F \rightarrow J \rightarrow K$
6. $C \rightarrow F \rightarrow I$	6. $C \rightarrow F \rightarrow G \rightarrow H$
7. $C \rightarrow F \rightarrow J$	7. $C \rightarrow F \rightarrow J \rightarrow K$
8. $D \rightarrow E \rightarrow F$	8. $D \rightarrow E \rightarrow F \rightarrow G$
9. $E \rightarrow F \rightarrow G$	9. $D \rightarrow E \rightarrow F \rightarrow I$
10. $E \rightarrow F \rightarrow I$	10. $D \rightarrow E \rightarrow F \rightarrow J$
11. $E \rightarrow F \rightarrow J$	11. $E \rightarrow F \rightarrow G \rightarrow H$
12. $F \rightarrow G \rightarrow H$	12. $E \rightarrow F \rightarrow J \rightarrow K$
13. $F \rightarrow J \rightarrow K$	

a dynamic user action log to detect leaders. However, these aforementioned related works do not use sliding window nor do they handle dynamic streams of user actions in the social network.

III. Our Sliding Window-Based Leader Detection (SWLD) Algorithm

Our sliding window-based leader detection (SWLD) algorithm discovers leaders from (i) a social network represented in an undirected graph $G = (V, E)$ and (ii) an action log capturing streams of user actions chronologically in the form $\langle u, a, t\rangle$ (which represents that a user u performs an action a at time t). Based on the contents of this action log, a *propagation graph* can be formed. Specifically, for a directed edge $(u_1 \rightarrow u_2) \in E$ from user u_1 to u_2 in the social network represented in $G = (V, E)$, any pair $\langle u_1, a, t_1\rangle$ and $\langle u_2, a, t_2\rangle$ from the action log such that $t1 < t_2$ represent that u_1 performs an action a at time t_1 and u_2 performs the same action a at a later time t_2. By considering all possible pairs of the same action, we obtain a propagation graph. A length-n *influence path* can be formed by selecting a connected path involving (n+1) vertices in the propagation graph. Then, a ***leader*** u is a length-n influential user who (i) has led $L_n(u) \geq \theta$ out of $N_n(u)$ propagations and (ii) has had a leading degree count $D_n(u) = \frac{L_n(u)}{N_n(u)} \geq \delta > 0$, where (i) $L_n(u)$ is the frequency of length-n influence paths such that u starts the propagation, (ii) $N_n(u)$ is the frequency of length-n influence paths such that involving u, and (iii) θ and δ are user-specified thresholds.

Example 1: $F \rightarrow G \rightarrow H$ is one of the 13 2-length influence paths involving 3 vertices in Figure 1. See Table I. Among all 11 users, user F involves in all length-2 influence paths and leads $L_2(F)$=2 of them (namely, $F \rightarrow G \rightarrow H$ and $F \rightarrow J \rightarrow K$). So, $D_2(F)$=$\frac{2}{13}$. Similarly, $A \rightarrow B \rightarrow F \rightarrow G$ is one of the 12 length-3 influence paths involving 4 vertices. User B involves in five length-3 influence paths and leads $L_3(B)$=2 of them (namely, $B \rightarrow F \rightarrow G \rightarrow H$ and $B \rightarrow F \rightarrow J \rightarrow K$). So, $D_3(B)$=$\frac{2}{5}$. ■

Our SWLD algorithm consists of two phases. In the first phase, regardless of the size of the stream of the action log, SWLD captures contents of the action log in a sliding window of size π. Hence, SWLD ensures there are at least M user actions to be kept in the sliding window, where M is a user-specified threshold. As SWLD uses a sliding window, it requires much smaller memory space than that for the entire action log.

In the second phase, SWLD calculates the leading degree count $D_n(u)$. To speed up the mining process, SWLD avoids enumerating all influence paths by scanning the propagation graph once to detect both forward neighbors ***FN***(u) & backward neighbors ***BN***(u) and compute their associated forward result ***FR***[u][n] & backward result ***BR***[u][n] for each user u. Here, for a directed edge $u \rightarrow v$ in the propagation graph, (i) v is a *forward neighbor* of u (i.e., $v \in FN(u)$) and (ii) u is a *backward neighbor* of v (i.e., $v \in BN(u)$). The forward result ***FR***[u][n] is a 2-dimensional structure capturing $L_n(u)$, i.e., frequency of length-n influence paths such that user u starts the propagation paths; the backward result ***BR***[u][n] is a similar 2-dimensional structure except that it captures the frequency of length-n influence paths such that user u ends the propagation paths. Here, ***FR***[u][1] = $|FN(u)|$ and ***BR***[u][1] = $|BN(u)|$.

Example 2: In Figure 1, $FN(F) = \{G, I, J\}$ and $BN(F) = \{B, C, E\}$. Hence, $L_1(F)$ = ***FR***[F][1] = 3 and ***BR***[F][1] = 3. Similarly, as $BN(I) = \{F\}$, ***BR***[I][1] = 1. ■

After computing ***FR***[u][1] and ***BR***[u][1] without generating and storing any influence paths, the remaining entries in these two structures can be computed by summing the frequencies of length-$(n-1)$ paths as follows:

$$L_n(u) = FR[u][n] = \sum_{v_i \in FN(u)} FR[v_i][n-1]; \quad (1)$$

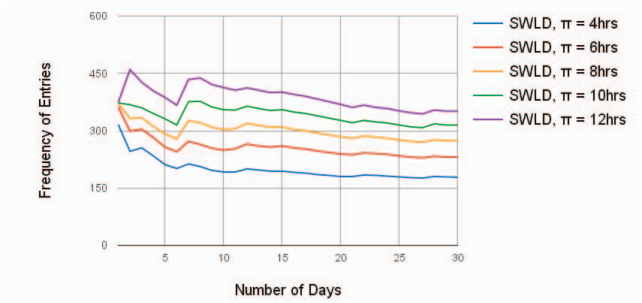
$$BR[u][n] = \sum_{v_i \in BN(u)} BR[v_i][n-1]. \quad (2)$$

Afterwards, $N_n(u)$ can then be computed as follows:

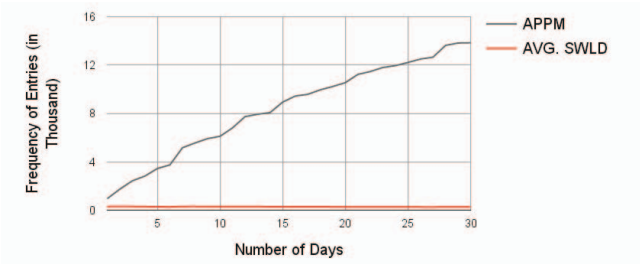
$$N_n(u) = FR[u][n] + BR[u][n] + \sum_{i=1}^{n-1} (BR[u][i] \times FR[u][n-i]). \quad (3)$$

Example 3: Continue with Example 2. As $FN(F) = \{G, I, J\}$, (i) ***FR***[G][1] = 1, (ii) ***FR***[I][1] = 0, and (iii) ***FR***[J][1] = 1. So, $L_2(F) = FR[F][2]$ = 1+0+1 = 2. To a further extent, $L_3(B) = FR[B][3] = FR[F][2]$ = 2 because $FN(B) = \{F\}$.

Similarly, (i) $FR[F][3]$ = 0, (ii) $BR[F][1]$ = 3, (iii) $BR[F][2]$ = 2, and (iv) $BR[F][3]$ = 0. So,



(a) Frequency of actions for SWLD with 5 different sliding window sizes.



(b) Action frequency for APPM vs. average action frequency for SWLD.

Figure 2. Frequency of actions for APPM vs. SWLD.

$N_3(F) = FR[F][3] + BR[F][3]$ + ($BR[F][1]\times FR[F][2]$ + $BR[F][2]\times FR[F][1]$) = 0 + 0 + (3×2 + 2×3) = 12. ■

Finally, our SWLD algorithm computes the leading degree count $D_n(u) = \frac{L_n(u)}{N_n(u)}$ for each available user u in the current sliding window based on the information stored in $FR[u][n]$ and $BR[F][n]$. After taking an average of the leading degree counts over all potential length n, users with a high leading degree count is considered to be influential, i.e., to be a leader. These leaders are detected and return as an output of our SWLD algorithm.

IV. Experimental Results

To evaluate our SWLD algorithm implemented in C++, we compared ours with the existing APPM algorithm [12] using a movie recommendation dataset named MovieLens[1]—which contains 100000 ratings by 943 users on 1682 movie collected in a period of seven months from the MovieLens website users—as an action log. Each movie rating in this dataset is of the form ⟨user ID, movie ID, rating, timestamp⟩, where timestamps are UNIX seconds since 1/1/1970 UTC. Each user is assumed to be able see other user actions (of performing a movie rating). Experiments were run on an Intel Core i5 machine with 3.20 GHz CPU, 8 GB of RAM, and 64-bit Windows 7 OS. We varied the size π of the sliding window from 4 hours to 6, 8, 10 and 12 hours.

Figure 2(a) shows the frequency of actions for each sliding window size π—where $\pi \in \{4, 6, 8, 10, 12 \text{ hours}\}$—for

[1]http://grouplens.org/datasets/movielens/

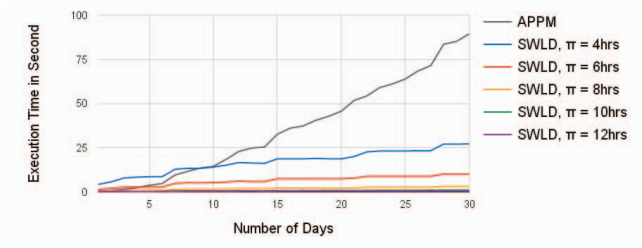
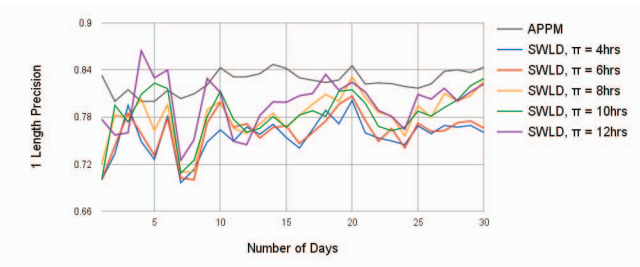


Figure 3. Execution time of APPM vs. SWLD.

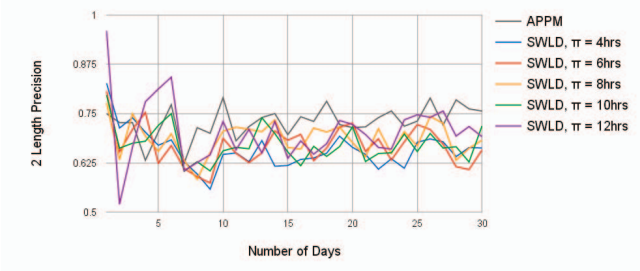
first 30 days of the MovieLens data containing 179624 interactions among 157 users. We observed that frequency of entries (i.e., actions) in a sliding window for SWLD was proportional to the window size. The larger the sliding window size, the higher was the frequency of actions in the window. As shown in Figure 2(b), the frequency of actions for SWLD was observed to be smaller than that for APPM because APPM operated on the entire dataset whereas SWLD operated on fewer π hours of data at any given time. Hence, SWLD required much less memory space than APPM.

Besides memory consumption, we also compared the execution time of APPM with that of SWLD using the five aforementioned different window sizes. The result shown in Figure 3 reveals that SWLD performed faster than APPM because APPM operated on the entire dataset whereas SWLD operated on fewer π hours of data. Moreover, SWLD effectively relies on $FR[u][n]$ and $BR[u][n]$ to compute the influence counts without explicitly enumerating all influence paths as in APPM. Hence, the execution time of SWLD was shorter than that of APPM. As the window size increased, we chose 2, 3, 4, 5 and 6 as minimum entries (or actions) for the window sizes π=4, 6, 8, 10 and 12 hours, respectively. During the execution of π=12 hours, SWLD pruned out all windows having less than 6 actions. Consequently, SWLD with π=12 hours operated on fewer windows due to pruning. It required shorter execution time than SWLD with π=4 hours.

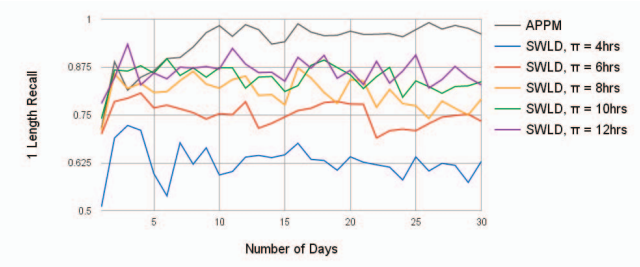
In addition to memory consumption and execution time, we also measured the accuracy of our SWLD algorithm. Specifically, we measured the *precision* (i.e., fraction of retrieved users being influential) and *recall* (i.e., fraction of influential users have been retrieved). Figure 4(a) shows the precision of SWLD with five different window sizes when using an influence probability threshold 2.7% in detecting length-1 influential users. The results show that SWLD with π=12 hours led to higher precision than that with π=4 hours when detecting length-1 influential users (i.e., length-1 leaders) because the larger the window size, the higher was the number of influential users captured in the same window. Similar comments apply to the detection of length-2 influential users (i.e., length-2 leaders). See Figure 4(b). Moreover, as shown in Figure 4(c), SWLD with π=12 hours led to higher recall than that with π=4 hours



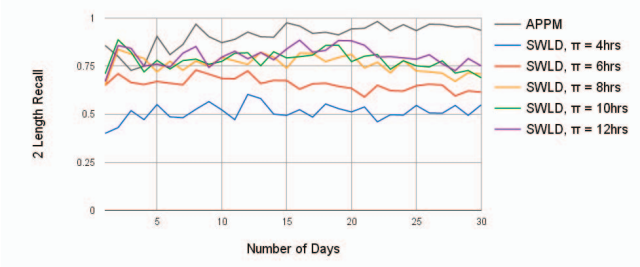
(a) Precision for length-1 influential users.



(b) Precision for length-2 influential users.



(c) Recall for length-1 influential users.



(d) Recall for length-2 influential users.

Figure 4. Precision and recall.

when detecting length-1 influential users. Figure 4(d) shows the recall for detecting length-2 influential users. The results show that the recall for SWLD with π=12 hours was almost the twice that for SWLD with π=4 hours.

V. Conclusions

In this paper, we proposed an algorithm called SWLD, which uses a sliding window for leader detection. Specifically, SWLD detects the most influential users, who can initiate and participate in lots of influence paths, from dynamic streams of actions performed by users in a social network. The algorithm avoids the expensive enumeration of all possible influence paths by using a dynamic programming approach to compute frequency counts of length-n influential users based on those of length-$(n-1)$ influential users and capturing the results in two 2-dimensional structures. Experimental results show that SWLD is accurate, requires short runtime and a small amount of memory space.

Acknowledgment

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

References

[1] F. Bodendorf & C. Kaiser, "Detecting opinion leaders and trends in online social networks," in *Proc. ICDS 2009*, pp. 65–68.

[2] F. Bonchi, L.V.S. Lakshmanan, & A. Goyal, "Discovering leaders from community actions," in *Proc. ACM CIKM 2008*, pp. 499–508.

[3] P. Braun, A. Cuzzocrea, C.K. Leung, R.K. MacKinnon, & S.K. Tanbeer, "A tree-based algorithm for mining diverse social entities," *Procedia Computer Science*, **35**: 223–232, 2014.

[4] A. Brun, A. Boyer, & I. Esslimani, "Detecting leaders in behavioral networks," in *Proc. ASONAM 2010*, pp. 281–285.

[5] A. Fariha, C.F. Ahmed, C.K. Leung, M. Samiullah, S. Pervin, & L. Cao, "A new framework for mining frequent interaction patterns from meeting databases," *Engineering Applications of Artificial Intelligence*, **45**: 103–118, 2015.

[6] K. Hino, B. Tseng, X. Song, & Y. Chi, "Identifying opinion leaders in the blogosphere," in *ACM CIKM 2007*, pp. 971–974.

[7] C. Hou, X. Yuan, C. Chen, & D. Wu, "Exploiting social media for stock market prediction with factorization machine," in *Proc. IEEE/WIC/ACM WI-IAT 2014, vol. 1*, pp. 142–149.

[8] W. Lee, C.K. Leung, J.J. Song, & C.S.-H. Eom, "A network-flow based influence propagation model for social networks," in *Proc. CGC (SCA) 2012*, pp. 601–608.

[9] C.K. Leung, "Big data mining and analytics," in *Encyclopedia of business analytics and optimization*, pp. 328–337, 2014.

[10] C.K. Leung & F. Jiang, "Big data analytics of social networks for the discovery of 'following' patterns," in *Proc. DaWaK 2015*, pp. 123–135.

[11] C.K. Leung, S.K. Tanbeer, & J.J. Cameron, "Interactive discovery of influential friends from social networks," *Social Network Analysis and Mining*, **4**(1): art. 154, 2014.

[12] Z.-L. Lin, A.L.P. Chen, M.F-. Tsai, & C.-W. Tzeng, "Discovering leaders from social network by action cascade," *Social Network Analysis and Mining*, **4**(1):art. 165, 2014.

[13] Y. Long, V.O.K. Li, & G. Niu, "Temporal behavior of social network users in information diffusion," in *Proc. IEEE/WIC/ACM WI-IAT 2014, vol. 2*, pp. 150–157.

[14] N. Matsumura & Y. Sasak, "Understanding leadership behavior in human influence network," in *Proc. IEEE/WIC/ACM WI-IAT 2006*, pp. 95–102.

[15] J. Zeng, B. Luo, & J. Duan, "Identification of opinion leaders based on user clustering and sentiment analysis," in *Proc. IEEE/WIC/ACM WI-IAT 2014, vol. 2*, pp. 377–383.