River: A Real-time Influence Monitoring System on Social Media Streams

Mo Sha[†], Yuchen Li[‡], Yanhao Wang[†], Wentian Guo[†], Kian-Lee Tan[†]

[†]*School of Computing, National University of Singapore, Singapore*
{sham,yanhao90,wentian,tankl}@comp.nus.edu.sg

[‡]*School of Information Systems, Singapore Management University, Singapore*
yuchenli@smu.edu.sg

***Abstract*—Social networks generate a massive amount of interaction data among users in the form of streams. To facilitate social network users to consume the continuously generated stream and identify preferred viral social contents, we present a real-time monitoring system called River to track a small set of influential social contents from high-speed streams in this demo. River has four novel features which distinguish itself from existing social monitoring systems: (1) River extracts a set of contents which collectively have the most significant influence coverage while reducing the influence overlaps; (2) River is *topic-based* and monitors the contents which are relevant to users' preferences; (3) River is *location-aware*, i.e., it enables user influence query on the contents falling into the region of interests; and (4) River employs a novel sparse influential checkpoint (SIC) index to support efficient updates against the streaming rates of real-world social networks in real-time.**

***Keywords*—Social network analysis, influence maximization, Twitter, location-based service.**

I. Introduction

The last few decades have witnessed the booming of online social networks (OSNs) where hundreds of millions of people interact with each other and produce an unprecedented amount of content. The prevalence of OSNs has prompted much interest in the study of *information diffusion*, as a piece of information could quickly become pervasive through the "word-of-mouth" propagation among friends in the network. Such a diffusion phenomenon has been shown to be powerful in many applications, such as viral marketing [1], [2], network monitoring [3], and recommendation systems [4]. As such, there have been extensive studies on social influence due to its immense value in real-world scenarios. See a recent survey [5] on social influence for a more comprehensive discussion.

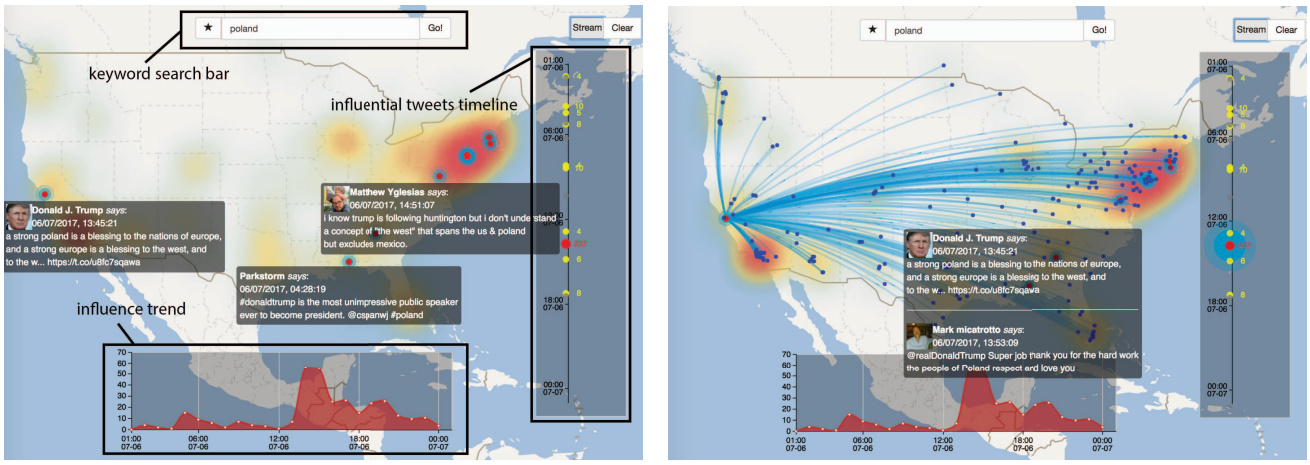
Most existing studies focus on analyzing static social interaction data, e.g., retweets on Twitter and shares on Facebook, and build models to understand social influence [6], [7]. In reality, social influences are highly dynamic and the interactions between users can be altered drastically by breaking news and trending topics [8]. Thus, the influence model built on static data can quickly become outdated. Although there are some efforts on dynamic influence analytics, e.g., the dynamic influence maximization (IM) problem [9]–[13] that tracks a set of k influential users with the largest influence from an evolving network, the performance is still unsatisfactory to meet the demand for large-scale applications. For example, the state-of-the-art dynamic IM solutions can only process a few hundreds of updates per second [9]–[11], which is far lower than the update rates of real-world social streams, e.g., about 7500 tweets are generated on Twitter per second.

In this demo, we present River: a real-time influence monitoring system on dynamic social streams. River tracks influential social elements (e.g., tweets/blogs) against social streams updated at high rates. River is equipped with the following unique features, which distinguish itself from existing dynamic social influence analytic solutions.

- River is based on the influence maximization problem (IM) over social streams [8], [14]. By taking into account the influence overlaps among different social elements, it extracts a set of contents collectively achieving the largest influence coverage on social audiences. Such requirement makes the influence monitoring problem to be NP-hard.
- River is topic-aware and query-specific. Users can input keywords/hashtags as queries to the system. River will extract the topical information of queries and proceed to track influential elements on the corresponding topics.
- River is location-aware. Many OSNs are equipped with location-based services. For example, geo-tagged tweets can be posted from GPS-enabled devices. River can leverage such spatial information to capture region-constrained influences. When a user specifies a region of interest, River will track the elements which are the most influential over the social audiences in the region.
- River employs a novel sparse influential checkpoint (SIC) index proposed in our previous work [8] to enable real-time stream processing. By exploiting the submodularity of the influence function, the results returned by SIC are guaranteed to be $(\frac{1}{4}-\varepsilon)$-approximate to the optimal ones where $\varepsilon \in (0,1)$ is a tunable parameter to achieve the trade-off between efficiency and quality. In the Twitter application, River can perform 30K updates per second, which is much higher than the update rate of the Twitter stream, i.e., 7.5K updates per second.

The rest of the paper is organized as follows. The related work is discussed in Section II. Section III presents a general working scenario of River. Subsequently, Section IV intro-

2375-9259/18/$31.00 ©2018 IEEE
DOI 10.1109/ICDMW.2018.00203



(a) Web interface of River

(b) Interactive influential tweet exploration

Fig. 1: In 6th July 2017, Donald J. Trump visited Poland and gave a speech at 1:19 PM, Warsaw (4:19 AM, PST). He posted a tweet to mention this event at 1:45 PM.

duces the streaming model as well as the monitoring problem of River. Section V describes River's processing framework and the SIC index. Section VI presents the demonstration details and Section VII concludes the whole paper.

II. Related Work

Social media users often rely on keyword search to explore rich contents that are continuously generated on social media platform [15]–[17]. However, keyword search does not consider the topical information as well as the social influence, which leads to suboptimal search results. There have been some existing studies on tracking topic-aware influencers in social streams [18]. [18] computes the influence scores of users based on information flows and proposes a method to track top-k influential users on a specific topic. Several commercial websites such as Lithium [1] and Keyhole [2] also provide services for social influence analytics. Lithium provides a social impact ranking to measure the influence scores of users. It collectively considers the number of tweets, followers, active audience, mentions, and replies, as well as the overall sentiment to compute the influence score. Keyhole tracks the trends for hashtags, accounts, keywords, and mentions in real-time and provides query interfaces and visualization tools to show the various statistics of a specific topic, i.e., top posts, influencers, location distribution, sentiment, and etc. However, the above solutions ignore the influence overlaps among users and may lead to the redundancy issue. In addition, all of them except Keyhole are location-unaware. Keyhole only provides the location distribution of users concerning a fixed topic, but it cannot handle ad-hoc region queries.

To the best of our knowledge, River is the first real-time social influence monitoring system which simultaneously supports the aforementioned features that are vital for real-world applications. For example, a senate candidate could leverage River to track sets of tweets that influence the most people in the state of California on different political issues, such as "Tax Reform", "Refugee Ban", and "Healthcare Policy". Accessing such real-time information would help the candidate identify sudden public opinion swings and assist her in planning for subsequent public speeches or tweets.

[1]https://www.lithium.com
[2]http://keyhole.co

III. System Overview

The working scenarios of River are illustrated in Figure 1. Figure 1a is the web interface of River and Figure 1b is a view of River's user interaction. In Figure 1a, users can input a set of query keywords to specify their tracking preference, e.g., "Poland" showed in the search bar. Users can also restrict the region of interest. In this example, we are interested in all the tweets posted in the United States. Subsequently, River starts to stream in new tweets and track influential ones. There are three main components in the web interface that visualize the results to end users. First, a timeline on the right of the interface shows when the influential tweets (marked as colored nodes), which are related to "Poland", are posted. The color represents a tweet's degree of influence impact, and Trump's tweet (marked as red) achieves the largest influence among all tweets during this period. Meanwhile, these influential tweets are also illustrated as nodes in the map which can be hovered over to show the full contents. The timeline slides when a batch of new tweets are streamed into the system and the influential tweets are updated in real-time. Second, the heatmap demonstrates the degree of influence achieved by the extracted tweets in different regions. Third, a time series on the bottom show the influence trend of the tracked tweets to demonstrate the variation of the public opinion on the user query keywords. Figure 1b illustrates that users can click on a certain node which attracts their interest in terms of the influence impact, and the tweets which are influenced (i.e.,

replies/retweets) will be shown in the map. A child node can be expanded recursively if possible and all nodes are interactive to show the contents of the tweet chain.

In this demo, we showcase River's efficiency in handling massive updates of the twitter stream. In particular, we allow users to specify the rate of twitter stream updates. We keep collecting a sample stream of incoming tweets from Twitter streaming API[3] and users can fast forward the stream by specifying different timeline shift (minute/hour/day) of the updates. River employs a novel SIC index to selectively maintain $O(\frac{\log N}{\varepsilon})$ checkpoints for a sliding window of N tweets, and each of which tracks the candidate solution w.r.t. different starting timestamps of the sliding window. As a consequence, this allows the system to always utilize the first non-expired checkpoint to extract the influential tweets w.r.t. the up-to-date sliding window, while still guarantees the approximation ratio.

IV. Monitoring Problem

Social Element. A social element e is defined as a tuple $e = \langle u, t, doc, loc, par \rangle$ where $e.u$ is the user who performs/posts e, $e.t$ is the timestamp, $e.doc$ is the textual content of e represented by a bag of l words $\{m_1, \ldots, m_l\}$ drawn from a vocabulary, and $e.loc$ is the location where the social element is posted. In addition, $e.par$ represents the element which e responds to (e.g., retweets/replies) and captures the influence from $e.par$ to e. In reality, "tweet" and "retweet" on Twitter are typical social elements, which are associated with timestamps and contents. A tweet/retweet will have a geo-tag if the user posts it from a GPS-enabled device and allows for position sharing.

Social Stream. A social stream comprises a sequence of n social elements indexed by $1, \ldots, n$ and ordered by their timestamps. To capture the temporal information, we adopt the sliding window model [19]. Given a window of length N, a sliding window W_t contains social elements whose indices are between $t - N + 1$ and t.

Topic-Aware Relevance. We model the content of each social element as a set of weighted topics. We treat the topic model, e.g., LDA, as a black box and any topic extractions or mining techniques can be adopted to transform the social content into a latent space Z and the output is a weighted vector H_e with $|Z|$ dimensions. Such topical distribution can be assumed to remain stable in a period of time, and the model can be retrained whenever necessary.

Given a set of query keywords, i.e., q, we can measure the topic-aware relevance between the user query and a social element. By treating the query keywords as a document, q can also be projected into a $|Z|$ dimensional topical vector. We follow several previous works, e.g., [15], [16], to measure the relevance between the social element e and the query q.

$$\phi(e, q) = \sum_{z \in Z} \mathbf{rel}(e.doc, z) \cdot \mathbf{rel}(q, z) \tag{1}$$

[3]https://developer.twitter.com/en/docs/tweets/sample-realtime/overview/GET_statuse_sample

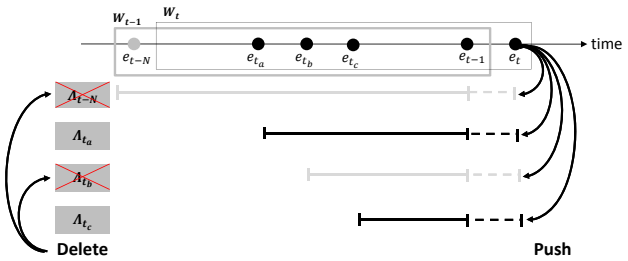


Fig. 2: Processing framework of the SIC index.

where $\mathbf{rel}(e.doc, z)$ and $\mathbf{rel}(q, z)$ represent the relevance of the topic z against the social element and the query keywords respectively.

Region-constrained Social Influence. We follow [8] and use social interaction data to quantify social influence. We say a social element e influences a user u in W_t w.r.t. a region R, denoted by $(e \rightsquigarrow u)_t^R$, if there exists a social element e' posted by user u such that $e' \in W_t$ and $e'.loc \in R$, and e' is *directly* or *indirectly* influenced by e. Given a user specified region R and a set of social elements S, we measure the influence of S over the current sliding window W_t by defining the influence set of S w.r.t. R.

Definition 1: (Influence Set) The influence set of a social element set S w.r.t. the query region R at time t, denoted as $I_t^R(S)$, is the set of users influenced by at least one of the social elements in S. Equivalently, $I_t^R(S) = \{u | \exists e \in S \text{ s.t. } (e \rightsquigarrow u)_t^R\}$.

Intuitively, the *influence set* of S denotes the set of users who recently post a social element under the impact of S. With the aforementioned setup, we are now ready to define the problem.

Influential Social Element Set Selection. Given the user keyword query q and a region of interest R, the goal of River is to maintain a set of social elements which are topic-wise relevant to the query while achieving the largest influence w.r.t. the up-to-date window W_t. We define the ranking function $f_t^{R,q}(S)$ to quantify the utility of a social element set S:

$$f_t^{R,q}(S) = \frac{\lambda}{n} \cdot I_t^R(S) + (1 - \lambda) \cdot \sum_{e \in S} \phi(e, q) \tag{2}$$

where n is a factor to normalize $I_t^R(S)$ into the same scale as $\phi(e, q)$ and λ is a tunable parameter to balance between topic relevance and social influence. Then, we formally define the influential social element selection problem.

Definition 2: Given a set of keywords q, a region R, and the result size k, the influential social element selection problem aims to extract the optimal set S_t^* that maximizes $f_t^{R,q}(\cdot)$ at any time t, given at most k elements can be selected, i.e., $S_t^* = \arg\max_{S:|S| \leq k} f_t^{R,q}(S)$.

V. River Processing Framework

It can be shown that it is NP-hard to extract the influential elements for each sliding window W_t, since it is a general version of the problem defined in [8][4]. Although

[4]When λ=1, the problem is equivalent to the one defined in [8].

Algorithm 1: SIC MAINTENANCE

1 **Required:** SIC at time $t-1$: $\{\Lambda_{t_1}, \Lambda_{t_2}, \ldots, \Lambda_{t_s}\}$;
2 **while** *receiving element* e_t *at time* t **do**
3 Create $\Lambda_{t_{s+1}}$ where $t_{s+1} = t$;
4 **foreach** Λ_{t_i} **do**
5 Push e_t to Λ_{t_i};
6 **if** $t_1 \leq t - N$ **then**
7 Delete Λ_{t_1} from SIC;
8 **foreach** Λ_{t_i} **do**
9 $\Lambda^- \leftarrow \emptyset$;
10 **foreach** Λ_{t_j} *such that* $t_j > t_i$ **do**
11 **if** $\Lambda_{t_j} \geq (1-\varepsilon)\Lambda_{t_i} \wedge \Lambda_{t_{j+1}} \geq (1-\varepsilon)\Lambda_{t_i}$ **then**
12 $\Lambda^- \leftarrow \Lambda^- \cup \{\Lambda_{t_j}\}$;
13 **else**
14 $\Lambda_{t_i} \leftarrow \Lambda_{t_j}$;
15 **break**;
16 Delete the checkpoints in Λ^- from SIC;
17 Retrieve the result of Λ_{t_1} for the query at time t;

a greedy heuristic, which iteratively selects a element with the maximum gain in $f_t^{R,q}(\cdot)$, achieves $(1-\frac{1}{e})$-approximate to the optimal solution because $f_t^{R,q}(\cdot)$ is monotone and submodular[5] [20], the efficiency of the greedy heuristic is far from satisfactory to handle massive updates of social streams.

To process massive updates, River employs our novel SIC index [8] to dynamically extract the influential elements. As shown in Figure 2, SIC stores a number of checkpoints which starts at different timestamps of the sliding window. Each checkpoint maintains the influential element set for the corresponding interval, e.g., checkpoint Λ_{t_a} keeps the solution among social elements contained in the time interval $[t_a, t-1]$ at time $t-1$. In this way, the first non-expired checkpoint is used as the solution for the entire window. There are two steps for SIC to handle an incoming social element[6]: (1) A *push step* which updates each of the checkpoints with the new element; (2) A *delete step* which deletes both the expired checkpoint and the checkpoints which are unnecessary to store.

Push. Let us picture a sliding window at time $t-1$ which contains N active social elements[7] (Figure 2). Given an incoming social element e_t which falls into the user specified region R, the push step uses e_t to update each of the checkpoints in the window. As the checkpoints only need to handle insertion rather than deletion and our ranking function is monotone and submodular, we invoke existing append-only streaming algorithm for submodular optimization to update the influential element set that corresponds to different starting points. We note that each checkpoint takes $O(\frac{\log k}{\varepsilon})$ ranking function calls to update and maintains a $(\frac{1}{2}-\varepsilon)$-approximate solution against the append-only stream.

Delete. After updating all checkpoints with the new social element, we first delete the expired checkpoint as the window slides (Λ_{t-N} in Figure 2). To speed up the processing efficiency, SIC leverages the monotone and submodular property of the ranking function to delete checkpoints which can be approximated by nearby checkpoints. In particular, SIC keeps a sequence s of checkpoints $\{\Lambda_{t_1}, \Lambda_{t_2}, \ldots, \Lambda_{t_s}\}$. Intuitively, given any three consecutive checkpoints Λ_{t_a}, Λ_{t_b}, Λ_{t_c} kept by SIC where $t_a < t_b < t_c$ and a parameter $\varepsilon \in (0,1)$, as long as $(1-\varepsilon)\Lambda_{t_a}$ (we also use Λ_{t_a} to denote the utility value of the solution maintained in Λ_{t_a}) is less than those of Λ_{t_b} and Λ_{t_c}, we delete Λ_{t_b} as Λ_{t_c} is $(1-\varepsilon)$-approximate to Λ_{t_b}.

The pseudo code of the SIC maintenance is presented in Algorithm 1. We perform the push step in Lines 3-5 and the delete step in Lines 6-16. Note that the deletion procedure can be done with only one swipe over the checkpoints in SIC. After the procedure for SIC maintenance, the result of Λ_{t_1} is always retrieved for the monitoring query at any time. It has been shown in [8] that only $O(\frac{\log N}{\varepsilon})$ checkpoints need to be maintained. The overall processing complexity of SIC is thus $O(\frac{\log k \cdot \log N}{\varepsilon^2})$ function calls, which is significantly lower than the greedy algorithm that requires $O(k \cdot N)$ function calls. In addition, SIC maintains a $(\frac{1}{4}-\varepsilon)$-approximate solution according to the theoretical result of [8].

We compare our SIC algorithm with two state-of-the-art IM algorithms: (1) IMM [21] for IM in static social networks; and (2) UBI [10] for dynamic IM in evolving networks. The results for efficiency are shown in Figure 3. We can see SIC has

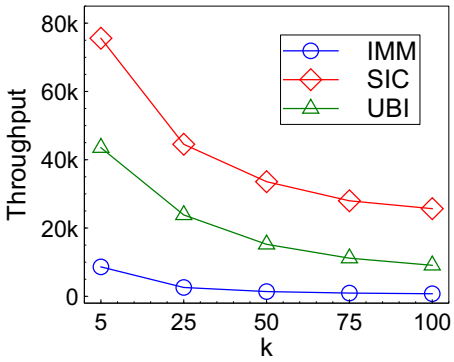


Fig. 3: A comparison of the efficiency of IM methods

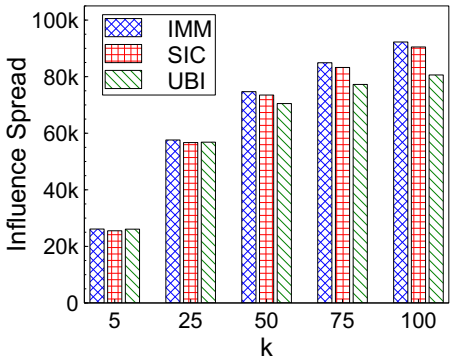
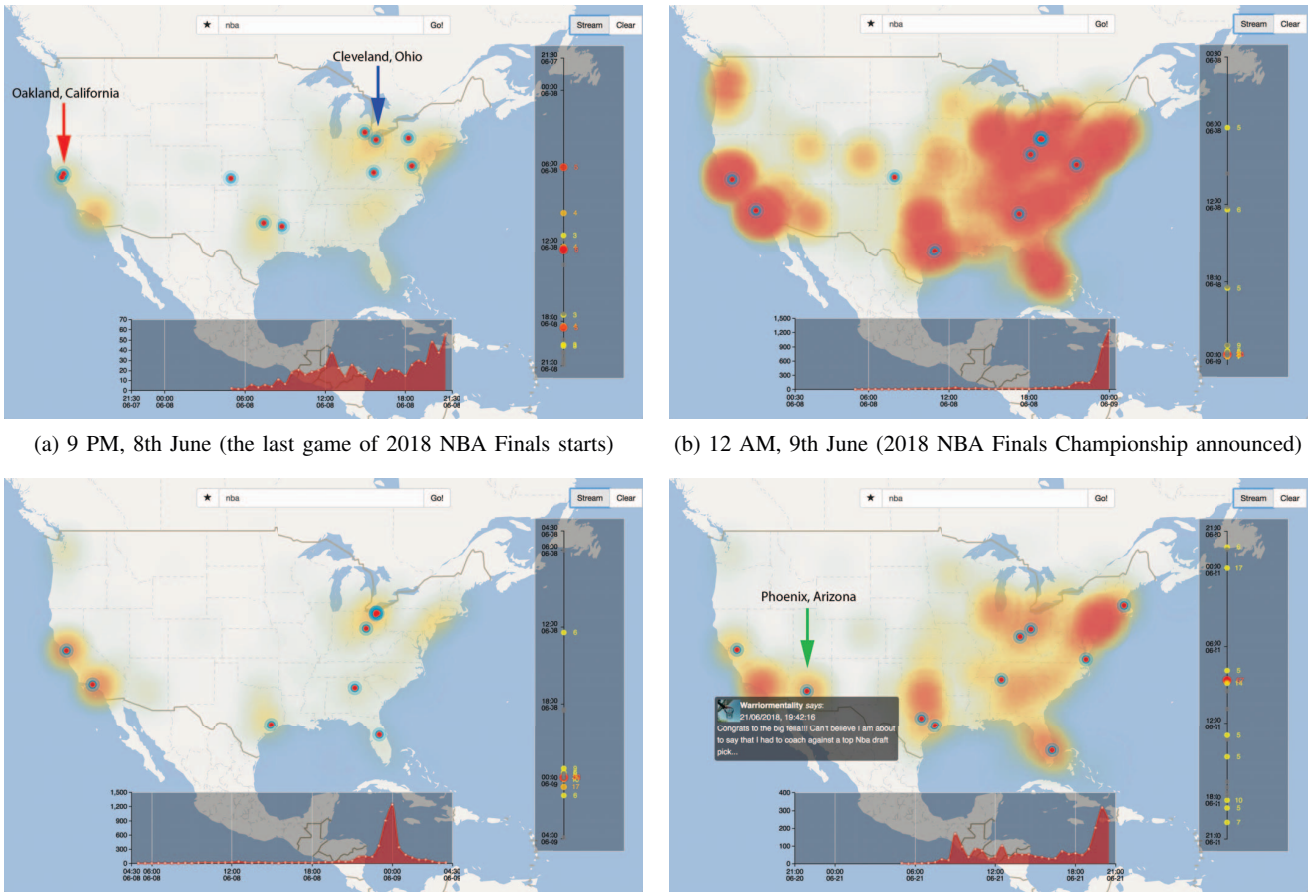


Fig. 4: A comparison of the result quality of IM methods

[5]A set function f is monotone iff $f(A) \leq f(B)$ for any $A \subseteq B$, and f is submodular iff $f(A \cup \{x\}) - f(A) \geq f(B \cup \{x\}) - f(B)$ for any $x \notin B$.

[6]Note that our processing framework can also handle a batch of updates. We present the scenario for single element update for simplicity.

[7]SIC can also handle varying length sliding windows, e.g., the time-based sliding window which maintains a number of elements in a fixed time interval.



(a) 9 PM, 8th June (the last game of 2018 NBA Finals starts)

(b) 12 AM, 9th June (2018 NBA Finals Championship announced)

(c) 4AM, 9th June (4 hours after the game ends)

(d) 8PM, 21st June (the 2018 NBA Top Draft Pick announced)

Fig. 5: The last game of NBA Finals between Golden State Warriors (home to Oakland, California, pointed by red arrow) and Cleveland Cavaliers (home to Cleveland, Ohio, pointed by blue arrow) started at 9PM, 8th June. Phoenix Suns (home to Phoenix, Arizona, pointed by green arrow) won the NBA Top Draft Pick at 8PM, 21st June. All times are in EDT.

much higher throughputs (i.e., the average number of elements processed per second) than UBI and IMM. It is able to process over 30K elements per second when $k = 50$, which can meet the requirement for processing real-world social streams. The results for influence spread (i.e., the average number of users influenced by the element sets returned by each method) are illustrated in Figure 4. The quality of the results returned by SIC is always close to IMM with varying k. The differences in influence spread are always less than 3%. Conversely, UBI can only return high-quality results when k is small but its result quality degrades when k increases.

VI. Demonstrations

We will demonstrate River to the conference audiences using the web interface that we have prototyped. We keep collecting the incoming tweets via the Twitter Streaming API over months, and the average incoming rate is about 20 tweets per second. Among all tweets, about 12% have the fine-grained geographic information (i.e., GPS coordinates), and the rest contains coarse-grained geographic information (i.e., rectangular bounding boxes). As of the date of publication, we have collected more than 100 million tweets. We use a server running Ubuntu 16.04 as the back-end, with four Intel E7-4820 1.9GHz processors and 128 GB memory.

The conference attendees will be able to experience three main scenarios with River, namely *query processing*, *interactive influential tweet exploration*, and *scalability demonstration*. Next, we will present the above scenarios in detail based on two examples: (i) Trump's visit to Poland (as shown in Figure 1) and (ii) 2018 NBA Finals and Draft Pick (as shown in Figure 5).

Scenario 1: Query Processing. Users can input a set of keywords, e.g., "Poland" or "NBA", and River will analyze the keywords to extract the latent topic interest of the users. Moreover, River's interface can support to circle an area in the map if users wish to constrain the region of the interests, e.g., within the United States. After users confirm their query (keywords and region), the system initializes and starts to stream

in tweets. As shown in Figure 1a, a sliding timeline is placed on the right of the interface and a set S of k influential tweets that are relevant to the user's query are always maintained and visualized. The collective influence score of S at time t, i.e., $I_t^R(S)$, is presented as a time-series trend chart placed on the bottom, which shows obvious influence fluctuations in some important moments. To demonstrate the impact region of S, a heatmap visualizes degree of the influence impact of S in different parts of the region.

Scenario 2: Interactive Influential Tweet Exploration. Users can stop the stream by pushing the "pause" button and closely examine the influential tweets extracted from the snapshot. As shown in Figure 1b, user can hover on one of the influential tweets in the timeline to access its full content and the influence region of the hovered tweet will be visualized using a heatmap. The user can also click on one of the influential tweet to explore in detail. Specifically, we visualize the retweets/replies triggered by the selected influential tweet on the map according to their geo-location and user can further click the retweets/replies on the map to see the full contents. This demonstration scenario allows the attendees to access the effectiveness of the extracted influential tweets on the spot. The attendees can then resume the stream to see the variation of the monitored influential tweets and impact regions.

Scenario 3: Scalability. We demonstrate the scalability of River in two ways. First, the attendees can increase the parameter k to track a larger number of influential tweets. Second, the attendees can fast forward the Twitter stream by choosing different timeline shifts (minute/hour/day). With an increasing timeline shift, every single update contains a more significant amount of incoming tweets, which stresses the River processing framework. Other than demonstrating the efficiency, the attendees can also visualize the influence fluctuation over a longer period of time with the fast forward feature. For instance, Figure 5 illustrates the evolving of influence distribution when "NBA" is chosen as the keyword and sliding in the timeline around June of 2018. Then, the influence distributions before, during and after the 2018 NBA Champion announce are shown in Figures 5a, 5b, and 5c respectively. Different from Figure 5b in which most areas have a large influence, Figures 5a and 5c emphasize that the locations of both teams are more influential, which indicates that the people in these locations pay attention to the game more continuously during the whole period. The same phenomenon can be observed in the 2018 NBA Draft Pick which is illustrated in Figure 5d. Specifically, we can see that Phoenix is enjoying an unusually large influence as it wins the first overall pick. To summarize, this example shows that River enhances the real-time, topic-aware and location-aware influence monitoring on social media streaming data.

VII. Conclusion

In this paper, we introduced our River system that aimed to provide a real-time monitoring service for influential contents tracking against high-speed social streams. We presented the problem context of River and demonstrated its novel features in a user-friendly interface, powerful processing framework over a novel index structure, and analysis efficiency. For future work, we plan to incorporate personalized topical influence in finding the relevant social contents [22].

Acknowledgment

We thank the anonymous reviewers for their insightful comments to improve the paper. Yuchen is supported by the Singapore MOE Tier 1 grant MSS18C001.

References

[1] P. M. Domingos and M. Richardson, "Mining the network value of customers," in *KDD*. ACM, 2001, pp. 57–66.

[2] Y. Li, D. Zhang, and K.-L. Tan, "Real-time targeted influence maximization for online advertisements," *Proc. VLDB Endow.*, vol. 8, no. 10, pp. 1070–1081, 2015.

[3] J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. M. VanBriesen, and N. S. Glance, "Cost-effective outbreak detection in networks," in *KDD*. ACM, 2007, pp. 420–429.

[4] M. Ye, X. Liu, and W.-C. Lee, "Exploring social influence for recommendation: a generative model approach," in *SIGIR*. ACM, 2012, pp. 671–680.

[5] Y. Li, J. Fan, Y. Wang, and K.-L. Tan, "Influence maximization on social graphs: A survey," *IEEE Trans. Knowl. Data Eng.*, 2018.

[6] A. Goyal, F. Bonchi, and L. V. S. Lakshmanan, "Learning influence probabilities in social networks," in *WSDM*. ACM, 2010, pp. 241–250.

[7] D. Kempe, J. Kleinberg, and E. Tardos, "Maximizing the spread of influence through a social network," in *KDD*. ACM, 2003, pp. 137–146.

[8] Y. Wang, Q. Fan, Y. Li, and K.-L. Tan, "Real-time influence maximization on dynamic social streams," *Proc. VLDB Endow.*, vol. 10, no. 7, pp. 805–816, 2017.

[9] N. Ohsaka, T. Akiba, Y. Yoshida, and K.-i. Kawarabayashi, "Dynamic influence analysis in evolving networks," *Proc. VLDB Endow.*, vol. 9, no. 12, pp. 1077–1088, 2016.

[10] X. Chen, G. Song, X. He, and K. Xie, "On influential nodes tracking in dynamic social networks," in *SDM*. SIAM, 2015, pp. 613–621.

[11] Y. Yang, Z. Wang, J. Pei, and E. Chen, "Tracking influential individuals in dynamic networks," *IEEE Trans. Knowl. Data Eng.*, vol. 29, no. 11, pp. 2615–2628, 2017.

[12] Y. Li, J. Fan, Y. Wang, and K.-L. Tan, "Influence maximization on social graphs: A survey," *IEEE Trans. Knowl. Data Eng.*, 2018.

[13] J. Fan, J. Qiu, Y. Li, Q. Meng, D. Zhang, G. Li, K.-L. Tan, and X. Du, "Octopus: An online topic-aware influence analysis system for social networks," in *ICDE*. IEEE, 2018, pp. 1569–1572.

[14] Y. Wang, Y. Li, J. Fan, and K.-L. Tan, "Location-aware influence maximization over dynamic social streams," *ACM Trans. Inf. Syst.*, vol. 36, no. 4, p. 43, 2018.

[15] Y. Li, D. Zhang, Z. Lan, and K.-L. Tan, "Context-aware advertisement recommendation for high-speed social news feeding," in *ICDE*. IEEE, 2016, pp. 505–516.

[16] D. Zhang, Y. Li, J. Fan, L. Gao, F. Shen, and H. T. Shen, "Processing long queries against short text: Top-k advertisement matching in news stream applications," *ACM Trans. Inf. Syst.*, vol. 35, no. 3, pp. 28:1–28:27, 2017.

[17] Y. Li, Z. Bao, G. Li, and K.-L. Tan, "Real time personalized search on social networks," in *ICDE*. IEEE, 2015, pp. 639–650.

[18] K. Subbian, C. C. Aggarwal, and J. Srivastava, "Querying and tracking influencers in social streams," in *WSDM*. ACM, 2016, pp. 493–502.

[19] Y. Wang, Y. Li, and K.-L. Tan, "A sliding-window framework for representative subset selection," in *ICDE*. IEEE, 2018, pp. 1268–1271.

[20] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher, "An analysis of approximations for maximizing submodular set functions - I," *Math. Program.*, vol. 14, no. 1, pp. 265–294, 1978.

[21] Y. Tang, Y. Shi, and X. Xiao, "Influence maximization in near-linear time: A martingale approach," in *SIGMOD*. ACM, 2015, pp. 1539–1554.

[22] Y. Li, J. Fan, D. Zhang, and K.-L. Tan, "Discovering your selling points: Personalized social influential tags exploration," in *SIGMOD*. ACM, 2017, pp. 619–634.