MyStream: an in browser stream processing personalization service to follow events from Twitter

Antoine Boutet, Frederique Laforest, Stephane Frenot, Damien Reimert

MyStream: an in browser stream processing personalization service to follow events from Twitter

Antoine Boutet
and Frederique Laforest
CNRS, Laboratoire Hubert Curien
Saint-Etienne, France
Email: antoine.boutet@univ-st-etienne.fr
Email: frederique.laforest@univ-st-etienne.fr

Stephane Frenot
and Damien Reimert
INSA-Lyon, CITI-INRIA
F-69621, Villeurbanne, France
Email: stephane.frenot@insa-lyon.fr
Email: damien.reimert@inria.fr

***Abstract*—Social media have become an essential tool to collect timely information on news and events. Analyzing social streams in real-time for personalization and recommendation purpose have become important topics in the data management community. In this paper, we propose MYSTREAM, a personalization service to follow events from Twitter. To improve the scalability of the service, MYSTREAM adopts an in-browser and hybrid architecture. MYSTREAM leverages the device of users to perform the computational operations on the users' browser, while the server only provides all the necessary material to perform these tasks. In addition, MYSTREAM adopts a stream-based processing approach to identify the relevant contents in a real-time manner. Moreover, the recommendation engine of MYSTREAM is highly modular. Users can build a personalized dashboard by assembling the recommendation modules they prefer to follow the considered event. We implemented and evaluated MYSTREAM using real trace from Twitter. We show that MYSTREAM is effective to follow an event from Twitter, particularly the live recommendation module quickly identifies the most valuable contents over time. In a system perspective, we show that the cost running MYSTREAM on the client device remains minimal.**

I. INTRODUCTION

Twitter has been massively adopted by millions users. This microblogging service has become an essential tool for the information dissemination and to collect timely information on news and events. Indeed, every event around the world is commented in real time by some community of users including journalists [1], [2], political figures [3] or official institutions. Due to the large volume of tweets, users need proper tools such as recommendation systems to quickly identify the most valuable contents among the information stream [4].

However, traditional recommendation systems that analyze data and periodically update models cannot follow the fast shift of users' interests and the highly dynamic nature of events and topics on Twitter. This limitation appeals for real-time recommendation systems [5] which pose great challenges. [6] and [7] have recently addressed this challenge by proposing solutions that analyze social streams to provide users with real-time topic recommendations. StreamRec [8], in turn, exploits explicit feedback from users (i.e. rating on items) to propose a recommendation system which can be performed in real-time. Nonetheless, all these solutions rely on centralized and powerful servers or cluster of nodes to support the cost of the recommendation operations, and possibly raise scalability issues when the amount of data increases. In this work, we explore another direction by leveraging the browser of users to compute real-time recommendation at the edge of the network on user machines.

In this paper, we present MYSTREAM, a mobile application to follow events from Twitter. MYSTREAM leverages the power of Twitter to have fresh and real time information related to an event or a topic identified through its hashtag [9]. To help users to discover relevant contents through the associated stream of tweets, MYSTREAM implements a modular recommendation engine. In addition, to capture the highly dynamic nature of exchanges on Twitter, MYSTREAM uses stream-based processing algorithms to identify the most valuable contents in real-time. Finally, to provide a user-centric service while tackling the scalability issue inherent to recommendation systems, MYSTREAM performs all computations dedicated to recommendations in the browser of users [10]. More precisely, the server forwards the stream of tweets related to the desired event to the users' machine, while on the client side, MYSTREAM relies on web technologies based on Javascript to perform local computations totally transparently from the user, before to present the data on the user interface.

The recommendation engine of MYSTREAM is highly modular. Each module provides a specific filtering scheme that processes the stream of tweets to highlight a specific content, from the computation of the most popular tweets or pictures to more complex personalized recommendation operations which highlight valuable tweets over time. Moreover, users assemble these modules to build a personalized dashboard to follow events. Users can also create their own recommendation modules to capture specific contents from the data stream. Finally, users can promote contents to create a personal journal on the event, and possibly push their report to a community.

In this paper, we present the capabilities of MYSTREAM to effectively help users to follow events from Twitter. This work conveys the feasibility of a real-time recommendation system exploiting the browser of users to perform the recommendation process. We implemented MYSTREAM and extensively evaluated our solution using real traces from the Twitter related to the 2014 New York City Marathon. We show that MYSTREAM quickly identifies valuable contents in real-time from the data stream. Moreover, from a system perspective, we show that MYSTREAM remains lightweight on the clients' smartphone.

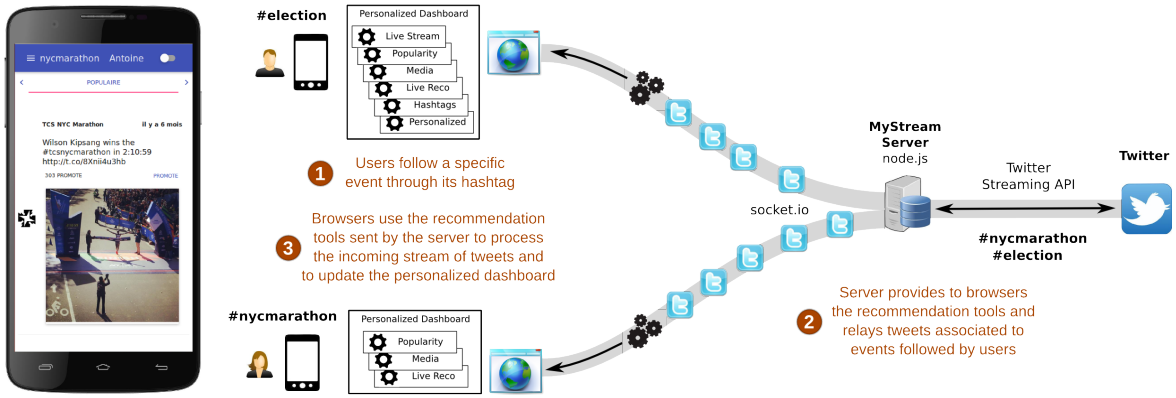


Fig. 1: MYSTREAM leverages the browser of users to process in real-time on the user's machine the stream of tweets associated to the followed event, and to update accordingly their personalized dashboard.

II. MYSTREAM

MYSTREAM provides a personalized service to follow events from Twitter. In this microblogging service, tweets adopt the hashtag convention [9] to associate a message to a specific event or subject. The stream of tweets associated to an event is leveraged as a feed of fresh and real-time information and directly processed by MYSTREAM. MYSTREAM relies on a hybrid architecture, a modular recommendation engine, and a personalized dashboard as explained below.

A. Hybrid architecture

Traditional recommendation systems are resource greedy. To provide a scalable system, MYSTREAM relies on a hybrid architecture [10]. In such an approach, the server decentralizes to client machines all the computation tasks while it manages a global orchestration and storage. In MYSTREAM, computations related to the recommendation engine are performed by the browser on the client machine while the server collects the stream of tweets from Twitter associated to the events followed by users, and relays them to the associated users' machine. The architecture of MYSTREAM is depicted in Figure 1.

Firstly, upon the first request from the user, the MYSTREAM server sends to the user's browser all required material related to the computation of recommendations. More precisely, the returned HTML page contains the Javascript pieces of code which deal with both the data exchanges with the server and the computation of each recommendation module. Once the HTML page is loaded, all exchanges with the server are performed asynchronously through Javascript data exchanges. When the user expresses to the server the event she wants to follow (i.e. hashtag), the MYSTREAM server uses the Streaming API of Twitter to collect the associated stream of tweets. This Streaming API [11] provides samples of the public stream of tweets containing the desired hashtag. Then, the server relays this stream to the browser of the associated user.

Finally, the browser processes the incoming stream of tweets. On the client side, MYSTREAM relies on Javascript. This language follows an event-driven programming model where the asynchronous data exchanges and the processing are totally transparent to users. Upon the reception of a tweet, the browser computes the process associated to the user recommendation settings (i.e. selected modules) and then updates accordingly each considered recommendation module.

B. Modular recommendation engine

To cope with the large volume of tweets, MYSTREAM relies on a highly modular recommendation engine to present to users relevant and interesting contents relative to the event they follow. Each module provides a filtering scheme which highlights a specific content.

To capture the highly dynamic nature of exchanges on Twitter and to provide a real-time service, the modules of MYSTREAM process directly the incoming stream of tweets. Specifically, each module has a specific event processing unit, dedicated to a specific content in the incoming tweets. For each incoming tweet, a dispatcher selects in a module registry the corresponding modules and calls their processing unit with a reference to the tweet. Each module can store their processing results in their own context. Moreover, modules can have dependencies. More precisely, some modules can process the stream of tweets outcoming from another module. For instance, the modules "Media" which presents the popular media processes only the tweets coming from the module which filters and identifies the popular tweets.

Each module associates a score to each considered content to reflect its relevance. This score is computed when the content is unknown, or updated otherwise. Modules are organized through separate tabs in the graphical user interface. The tab of a module is organized as a ranked list where the contents with the highest scores are displayed first. This ranked list has a limited size (typical value is 100) and contents with smallest scores (i.e less relevant) are automatically removed.

MYSTREAM provides different recommendation modules including: 1) **Live Stream:** the received stream of tweets, 2) **Popularity:** the most popular tweets, 3) **Media:** the media (i.e. image) of the most popular tweets, 4) **Hashtags:** the additional hashtags associated to the followed event, 5) **Live Recommendation:** the most valuable tweets over time. Moreover, users can define their own recommendation modules to highlight specific content. All these modules are detailed below.

(a) Popularity (b) Media (c) Hashtags (d) Live Reco. (e) Personal report (f) Personalized module

Fig. 2: MYSTREAM allows users to build a personalized dashboard with different recommendation modules.

*1) **Live Stream**:* This module displays the received stream of tweets associated to the event (i.e. hashtag) followed by the user. Consequently, similar to the Twitter timeline, tweets in this module are organized according to their publication date, the newest tweets being displayed first.

*2) **Popularity**:* To capture the popularity of tweets, MYSTREAM leverages the retweet convention (i.e. action of forwarding an interesting tweet to its followers). A retweet embeds the original tweet which includes the global number of retweets this message has collected so far. As a consequence, we can collect both the original tweet that has raised interest from users and its global level of popularity directly from the stream of tweets sent by Twitter. This behavior has also the advantage to overcome the limitation of the sampling produced by Streaming API used by the MYSTREAM server to collect messages associated to events from Twitter. Indeed, the level of popularity associated to a tweet is global and does not depend on the number of retweets received by our server. Consequently, with enough time, the list of the popular tweets closely approximates the real one (i.e. without sampling). After the reception of a retweet, the list of popular tweets displayed by this module is updated by sorting this list according to the number of retweets.

*3) **Media**:* This module only displays media embedded in tweets. To highlight the most valuable media, the popularity of their associated tweet is leveraged (i.e. number of retweets). As a consequence, media associated to highly popular tweets will be then displayed first by this module. This modules depends on the module "popularity" and it processes only tweets coming from this module.

*4) **Hashtags**:* A tweet can be associated to different events or subjects by using different hashtags. This module displays all associated hashtags to the main one followed by the users and allows them to discover associated subtopics and possibly change the hashtag they want to follow. Hashtags are displayed through a tag cloud where the size of each hashtag depends on the number of its occurrence in the stream.

*5) **Live Recommendation**:* Contents identified by only considering the popularity level through the retweets (i.e. popularity module) lack of freshness as reported in our evaluation Section III-C. To overcome this limitation, this live recommendation module performs operations which highlight valuable tweets over time. The temporal analysis of retweet (i.e. popularity) in Twitter follows a burst of activity just after the publication of the original tweet and then a long tail [12]. MYSTREAM leverages this temporal pattern to capture the burst of activity and to quickly identify the relevant contents over time.

To achieve that, the underlying algorithm of this module attributes a score to each tweet based on both criteria, its popularity and its freshness. While the former exploits the number of retweets, the latter captures the delay between the publication of the tweet in Twitter and its retweets. This module uses a sliding window [13], and refresh the score to each content after a certain time window of size w. The following equation computes the score attached to tweet t at the current date d:

$$score(t,d) = \sum_{RT(t,d)\in[d0,d0+w]} \frac{RT(t,d) \times w}{\Delta T}$$

where $RT(t,d)$ is the number of retweets that t has collected so far, $d0$ the starting date of the sliding window, and $\Delta T = d - d_t$ which reflects the freshness of t where d_t is the date when t has been created.

Each element of the sum follows the pattern of $\frac{1}{x}$. As a consequence, each element of the sum (i.e. each retweet in the considered sliding window) increases if $\Delta T < w$. It means when the retweet has happened before the considered time window w. In contrast, this score is minored if the retweet happens after the time window w. In addition, this score is weighted by the number of retweets t has obtained so far. Lastly, the score associated to t is cumulative and sums the score computed to each retweet received in the considered time window w. The starting date of the time window is different for each tweet and is initialized at the reception of its first retweet. When the time window is over for one tweet, its score is reinitialized at 0. The size of the time window defines the temporal granularity to follow the event, more the time window is small, more the event is detailed. The impact of the time window size is evaluated in Section III-C.

6) Personalized recommendation module: Users can create their own recommendation modules to highlight specific contents. To achieve that, users can define policies to select the desired content from the stream of tweets, and how to sort them. We illustrate this feature by creating a personalized filtering module which captures the official results from the stream of tweets associated to the 2014 New York City Marathon as explained in Section III-B.

C. Personal dashboard and report

Each user can decide to use or not a specific recommendation module to follow an event. For instance, in Figure 1 Bob uses the five modules presented above and one personalized module (e.g. prediction of voting results) to follow the elections, while Alice only exploits the modules "popularity", "media" and "live recommendation" to follow the New York City Marathon. By assembling preferred recommendation modules, users can build a personalized dashboard to follow the event according to their tastes. Moreover, users can promote contents to create a personal report on the followed event. All promoted contents are included in the personal report and users are able to arrange these contents themselves to produce a report which reflects the event. This report is stored on the MYSTREAM server. In addition, users can possibly push their report to a community.

D. Implementation

MYSTREAM is a mobile application and consequently adapted to smartphones or mobile device interfaces. MYSTREAM is a web application built with AngularJS and CoffeeScript. All exchanges between the browser of users and the server are performed through Socket.IO. The server, in turn, relies also on Javascript and uses Node.js as framework. Figures 2d-2f depict different modules during the 2014 New York City Marathon, each module holding in a separate tab.

III. EVALUATION

In this section, we evaluate the ability of MYSTREAM to effectively help users to follow events from Twitter. To achieve that, we use real traces from the Twitter activity related to the 2014 New York City Marathon. Fist, we present the considered dataset to evaluate MYSTREAM. Second, we give an example of personalized recommendation module to consult the official results of the marathon which transit in the stream of tweets. Then, we assess the capability of MYSTREAM to get valuable contents in real-time, and finally we evaluate the cost of operating MYSTREAM on the client machine.

A. Dataset

To assess MYSTREAM, we use real traces by using the Twitter activity related to the 2014 New York City Marathon. This marathon was run on Sunday, November 2, 2014 and gathered different categories (e.g. professional women, wheelchair division) with separate start times. We collected tweets associated to the hashtag *#nycmarathon* using the Twitter streaming API between the 29th of October and 18th of November. The resulting dataset gathers more than $41,000$ users for almost $100,000$ tweets. The collected messages include about $4,700$ (4.9%) mentions (i.e. direct messages to another user), $9,000$ (10.7%) retweets (i.e. forward messages to its followers), and $11,000$ hashtags (i.e. tags used to define topics). The average rate of tweets per second over this whole period is around 0.05. Considering only the day of the event (i.e. November 2), this average rate is equal to 0.73 (i.e. around $63,000$ tweets). This rate increases up to 3.29 during the hour where the Twitter activity related to the marathon is highest.

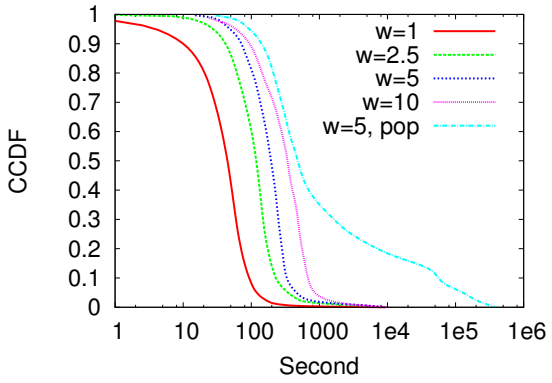


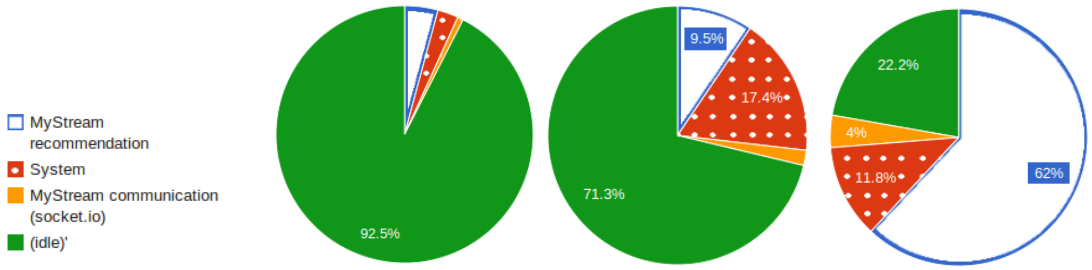
Fig. 3: MYSTREAM quickly identifies valuable tweets.

B. Personalized recommendation module

Users can create they own recommendation modules to highlight specific contents. Here we illustrate this feature by creating a module which lists the official results of the marathon. Official results of the marathon were available from the website of the organizer once the concurrent finished the race. From this website, users were able to published their performance on Twitter. These tweets look like: `"Valeria Straneo crossed 40K/24.9mi. Time 02:21:28, pace 05:43 min/mile, est. finish 11:39am http://t.co/qS9xe9NJ8L #tcsnycmarathon"`. Our trace contains $10,180$ tweets reporting results. We build a personalized module which captures and lists all these official results transiting in the stream of tweets (Figure 2f).

C. Live Recommendation

To evaluate the capability of MYSTREAM to quickly evolve fresh contents through the stream of tweets, we measure the latency between the publication of relevant tweets in Twitter and their identification and their display in the live recommendation module. As described in Section II-B5, this module leverages a time window to increase or decrease the score associated to each tweet, and to give more weight to fresh tweets. Figure 3 shows the Complementary Cumulative Distribution Function (CCDF) of the latency to display relevant tweets for different values of time windows (w). Here we consider only the top 5 tweets provided by the live recommendation module. Results show that the latency mainly depends on the time windows, the larger one, the longer latency. Indeed, when the time window increases, the most relevant tweets are selected through this whole time slot. As a consequence, the latency to display them also increases. However, regardless the value of w, we show that MYSTREAM timely identifies valuable contents. With a time window of 2.5 minutes, 50% of the contents age before 2 minutes while with a time window of 10 minutes, 50% remains under 3 minutes. Finally, we compare our solution with an approach which only takes into account the popularity. More precisely, in this candidate approach (called pop in Figure 3) the score associated to a content is the sum of the number of retweets this tweet has received under the time window. Results show that MYSTREAM drastically reduces the latency compared to an approach considering only the popularity to draw a time line of valuable contents.



(a) Legende (b) 10 tweets/sec. (c) 20 tweets/sec. (d) 50 tweets/sec.

Fig. 4: The average CPU usage related to MYSTREAM depends on the size of the stream but remains low for streams with less than 25 tweets per second.

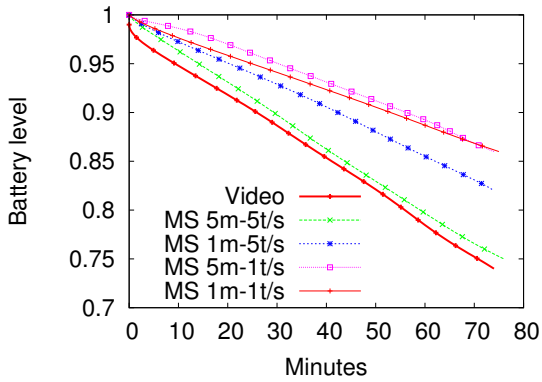


Fig. 5: Energy consumption related to MYSTREAM remains less important than watching a video stream.

D. Impact on smartphone

We now evaluate the cost of operating MYSTREAM on the client machine. Running MYSTREAM should not significantly impact the performance of a user's machine. Performance varies according to the size of the stream, the larger one, the more items processed on the user's device. Moreover, performance changes also according to the number of modules used by the user to follow the event. As the volume of tweets depends on the event and the considered moment, here we control the number of tweets sent to the device. Figure 4 depicts for three different sizes of stream the average CPU usage of the smartphone for each action performed during 5 minutes, namely the recommendation process (MYSTREAM using all the five modules presented in Section II-B), the system (Android), the communication (Socket.IO), and the idle time. Results show that for 10 and 25 tweets per second, the CPU usage dedicated to MYSTREAM is negligible compared to the idle time. However, when the stream becomes larger (i.e. 50 tweets per second) most of the CPU usage is dedicated to the recommendation process of MYSTREAM. As described in Section III-A, the pick of activity in Twitter during the New York City Marathon generated a stream of 3.29 tweets per second in average during one hour. As a consequence, for similar events MYSTREAM has a minimal impact on the CPU usage of user's smartphone most of the time.

Energy is the most crucial aspect in smartphone [14]. The energy consumption related to an application is crucial for the lifetime of the battery's smartphone and its adoption by users [15]. To assess the impact of MYSTREAM on the user's smartphone, we measured the battery consumption related to MYSTREAM. Figure 5 depicts the evolution of the battery level while MYSTREAM is running on the smartphone for two different sizes of stream (both 1 and 5 tweets per second). For this experiment, we used a Nexus 4 and a Wifi network access. The experiment is performed on a time slot of 75 minutes and we considered two different sets of dashboard configuration, one using all the five recommendation modules presented in Section II-B (called $5m$ in the figure), and the other using only the live stream module (called $1m$ in the figure). We also compared the battery consumption of MYSTREAM with the consumption spent when the user watches a video stream on its device. Firstly, results show that the impact of MYSTREAM using all the modules closely correspond to watching a video stream on the smartphone. Secondly, we show that the battery consumption depends on the modules selected by users in their personal dashboard, increasing the number of selected modules increases the battery consumption. For instance, for a stream of 5 tweets per second, using only one module can save the battery's lifetime around 8% compared to using five modules. Finally, we show that the size of the stream also impacts the battery consumption. Reducing the stream by 5, increases by 10 the battery lifetime. Interesting enough, the gap between using all recommendation modules or only one for a stream of 1 tweet per second, is drastically reduced compared to a stream of 5 tweets per second. Considering the volume of tweet generated during the New York City Marathon (i.e. in average 0.73 tweet per second the day of the marathon and up to 3.29 at the pick of activity), the battery consumption of MYSTREAM remains low with the respect to playing a video stream on its smartphone.

IV. RELATED WORKS

With several millions tweets generated every day on Twitter, users of this social platform can be easily overwhelmed by this large volume of information. To help users to discover the most valuable information, several recommendation systems have been proposed [4]. Due to the information discovery nature of Twitter, users can be interested in findings various informations such as new relevant friends [16], followers [17], hashtags [7], tweets or retweets [18].

Recommendation systems are resource greedy. Most recommenders are centralized. To cope with the high cost of recommendation system most approaches exploit elastic platforms to massively distribute the recommendation tasks on a large number of nodes [19]. At the opposite side, other solutions proposed to fully decentralize the recommendation process by providing recommenders which are user-centric at the architecture level [20]. Hybrid architectures have been introduced as an alternative to fully centralized or decentralized solutions. For instance, [21] exploits the centralized nature of Twitter and uses a peer-to-peer approach to compute locally recommendations of Twitter friends on the user device, and to exchange recommendations between peers. More recently, Hyrec [10] has proposed and evaluated a user-based collaborative filtering solution with a hybrid architecture where all the computational tasks are performed in the users' browser.

Twitter has also become a useful tool to collect timely information on news events [1]. Traditional state-of-the-art recommenders such as collaborative filtering schemes relay on an offline model-building phase that build a model storing meaningful correlations between users and items. They are not built for a real-time recommendation process. To improve the system's ability to evolve quickly, recent work have proposed recommenders adopting a stream-based approach [5], [8]. However, when they are applied to Twitter, these systems only recommend topics (i.e. hashtag) that match user interests at a specific moment [7], [6] and not specific information such as valuable tweets during an event as MYSTREAM does. In addition, these solutions are fully centralized and require costly elastic infrastructure to hope to scale without problem. As far as we know, MYSTREAM is the first personalized recommendation solution to follow specific event in real-time from Twitter using a hybrid architecture.

V. CONCLUSIONS AND DISCUSSIONS

In this paper, we report the design and evaluation of MYSTREAM, a personalized service to follow event from Twitter. MYSTREAM relies on a hybrid architecture, a modular recommendation engine, and a personal dashboard to provide a highly scalable and customizable mobile application. All the computation related to the filtering schemes of the data are performed locally on the users' device. In addition, to timely identify relevant contents through the stream of tweets, the recommendation engine of MYSTREAM uses a stream-based processing approach. Users leverage existing recommendation modules or create their own modules to build and to assemble their personalized dashboards.

To demonstrate the capabilities of MYSTREAM to effectively help users to follow events from Twitter, we developed and evaluated our solution using real traces from Twitter. We show that MYSTREAM is effective to help users to follow an event from Twitter. Particularly, the module in charge of the live recommendation quickly identifies the relevant content and produces a relevant time line.

MYSTREAM conveys the feasibility of a mobile application which uses the smartphone of users to process in real-time a stream of information. Although our solution provides effective results, leveraging more advanced stream processing algorithms and filtering schemes with a hybrid architecture are interesting directions for future work. In addition, exploring new and advanced web technologies such as browser to browser communication capability can also open new perspectives. Lastly, in MYSTREAM, the stream of tweets is collected from Twitter by the server. This behavior has the advantage to allow the server to implement adaptive mechanisms between the clients and the server to limit the size of the stream for instance. However, collecting the stream related to an event directly from the users' device is also attractive to reduce server task to minimum.

REFERENCES

[1] D. L. Lasorsa, S. C. Lewis, and A. E. Holton, "Normalazing twitter: journalism practice in an emerging communication space," *Journalism Studies*, vol. 13, no. 1, pp. 19–36, 2012.

[2] S. Schifferes, N. Newman, N. Thurman, D. Corney, A. Goker, and C. Martin, "Identifying and verifying news through social media: Developing a user-centred tool for professional journalists," *Digital Journalism*, vol. 2, no. 3, pp. 406–418, 2014.

[3] A. Boutet, H. Kim, and E. Yoneki, "What's in twitter, I know what parties are popular and who you are supporting now!" *Social Netw. Analys. Mining*, vol. 3, no. 4, pp. 1379–1391, 2013.

[4] S. M. Kywe, E.-P. Lim, and F. Zhu, "A survey of recommender systems in twitter," in *SocInfo*, 2012.

[5] Y. Huang, B. Cui, W. Zhang, J. Jiang, and Y. Xu, "Tencentrec: Real-time stream recommendation in practice," in *SIGMOD*, 2015.

[6] E. Diaz-Aviles, L. Drumond, L. Schmidt-Thieme, and W. Nejdl, "Real-time top-n recommendation in social streams," in *RecSys*, 2012.

[7] C. Chen, Y. Hongzhi, Y. Junjie, and C. Bin, "Terec: A temporal recommender system over tweet stream." *PVLDB*, vol. 6, no. 12, pp. 1254–1257, 2013.

[8] B. Chandramouli, J. J. Levandoski, A. Eldawy, and M. F. Mokbel, "Streamrec: a real-time recommender system." in *SIGMOD*, 2011.

[9] F. Kooti, C. Meeyoung, P. G. Krishna, and M. Winter, "The emergence of conventions in online social networks," in *ICWSM*, 2012.

[10] A. Boutet, D. Frey, R. Guerraoui, A.-M. Kermarrec, and R. Patra, "Hyrec: Leveraging browsers for scalable recommenders," in *Middleware*, 2014.

[11] F. Morstatter, J. Pfeffer, H. Liu, and K. M. Carley, "Is the sample good enough? comparing data from twitter's streaming API with twitter's firehose," *CoRR*, vol. abs/1306.5204, 2013.

[12] H. Kwak, C. Lee, H. Park, and S. Moon, "What is twitter, a social network or a news media?" in *WWW*, 2010, pp. 591–600.

[13] J. Vinagre and A. M. Jorge, "Forgetting mechanisms for incremental collaborative filtering," in *III International Workshop on Web and Text Intelligence*, 2010.

[14] A. Pathak, Y. C. Hu, and M. Zhang, "Where is the energy spent inside my app?: fine grained energy accounting on smartphones with eprof," in *EuroSys*, 2012, pp. 29–42.

[15] D. Ferreira, A. K. Dey, and V. Kostakos, "Understanding human-smartphone concerns: a study of battery life," in *Pervasive computing*. Springer, 2011, pp. 19–33.

[16] J. Hannon, M. Bennett, and B. Smyth, "Recommending twitter users to follow using content and collaborative filtering approaches," in *RecSys*, 2010, pp. 199–206.

[17] P. Nasirifard and C. Hayes, "Tadvise: A twitter assistant based on twitter lists," in *Social Informatics*, ser. LNCS, 2011, vol. 6984, pp. 153–160.

[18] I. Uysal and W. B. Croft, "User oriented tweet ranking: A filtering approach to microblogs," ser. CIKM, 2011.

[19] A. S. Das, M. Datar, A. Garg, and S. Rajaram, "Google news personalization: scalable online collaborative filtering," in *WWW*, 2007.

[20] A. Boutet, D. Frey, R. Guerraoui, A. Jegou, and A.-M. Kermarrec, "Whatsup: A decentralized instant news recommender," in *IPDPS*, 2013, pp. 741–752.

[21] S. Frénot and S. Grumbach, "An in-browser microblog ranking engine," in *ECDM-NoCoDA*, ser. LNCS, vol. 7518, 2012, pp. 78–88.