Exploiting User Feedback for Online Filtering in Event-based Systems*

Fabio Petroni
DIAG, Sapienza University of Rome, Italy
petroni@dis.uniroma1.it

Leonardo Querzoni
DIAG, Sapienza University of Rome, Italy
querzoni@dis.uniroma1.it

Roberto Beraldi
DIAG, Sapienza University of Rome, Italy
beraldi@dis.uniroma1.it

Mario Paolucci
Institute of Cognitive Sciences and Technologies, CNR, Italy
mario.paolucci@istc.cnr.it

ABSTRACT

Modern large-scale internet applications, like the ubiquitous social networks, represent today a fundamental source of information for millions of users. The larger is the user base, the more difficult it is to control the quality of data that is spread from producers to consumers. This can easily hamper the usability of such systems as the amount of low quality data received by consumers grows uncontrolled. In this paper we propose a novel solution to automatically filter new data injected in event-based systems with the aim of delivering to consumers only content they are actually interested in. Filtering is executed at run-time by first profiling both producers and consumers, and then matching their profiles as new data is produced.

CCS Concepts

•**Software and its engineering** → **Publish-subscribe / event-based architectures;** •**Human-centered computing** → Collaborative filtering;

Keywords

Event-Based Systems; Publish/Subscribe; Event Filtering.

*This work has been partially founded by the TENACE PRIN Project (n. 20103P34XC) and by the RoMA Smart-Cities project, both funded by the Italian Ministry of Education, University and Research.

SAC 2016,April 04-08, 2016, Pisa, Italy

DOI: http://dx.doi.org/10.1145/2851613.2851763

1. INTRODUCTION

In the recent years the world of global information has been dramatically changed by the widespread participation of people to social networks, whose user base continues to grow at an astonishing pace (e.g. Twitter's monthly active users have grown 19% during the last year peaking at 304M monthly active users in Q2 2015 – Statista 2015). In such systems users freely exchange data in several forms (status updates, tweets, links, comments, etc.) and select data they are willing to receive using coarse grained selection methods (friend/follower lists, tag selection, free text searches). The social network then provides each user with a personalized data stream whose content takes into account user selection criteria. Users are sometimes given the possibility to provide feedback on data they receive (e.g. "I like" on Facebook or re-tweets on Twitter), but this feedback is commonly taken into account only to increase the amount of data injected on the stream or to suggest new connections and selections to users. This coarseness in the available selection criteria, mixed with the current growth rate of data injected in these systems, starts to limit their effectiveness in conveying useful information to users: as user streams grow to unmanageable sizes they tend to contain more and more information whose perceived quality is low from the user standpoint.

By looking at the literature two main approaches can be identified to attack this problem. The first is based on *reputation systems*, that is online mechanisms that aggregate feedback from users' past experiences, to enable more informed decisions of other users in the future. While these systems have proven to be fundamental to profile users in specific applications (e.g. sellers on the eBay platform), their applicability in social networks where users can show different behaviors depending on the data they produce is still characterized by some open problems. The second approach is represented by *collaborative filtering* solutions, where users feedback expressed on a data set are leveraged to suggest specific data items to other users. The limitation of the latter approach is that it must work on a stable data set making impractical its application to social networks where new information is continuously injected and streamed in quasi-real-time to its recipients.

In this paper we propose a novel online filtering solution for

event-based systems (social networks can be considered as complex, large-scale instances of this group). Our solution leverages feedback expressed by users on received events to profile data sources. Profiling is based on an reputation metric calculated as a result of a collaborative process performed by receivers for the same data source. The profiling process takes into account that a source can produce events with different quality levels on different topics. New events injected in the system are filtered online depending both on the profile associated with their sources and on profiles built for characterizing the minimum quality thresholds of each potential destination. We evaluated the performance of our solution through a simulation-based study by applying it in the context of publish/subscribe systems. Reported results show that our solution is able to effectively profile both source and destination users and to automatically filter out a large percentage of new events that would be negatively evaluated by their recipients with a low percentage of false positives.

The rest of this paper is organized as follows: Section 2 presents related works; Section 3 defines the system model and states the problem; Section 4 presents our solution that is then evaluated in Section 5; Section 6 concludes the paper.

2. RELATED WORK

The interest in reputation shown by economy and game theory was quickly followed by a surge of attention in ICT. As the number of online users and transaction increased, transcending geographical limitations and personal acquaintance, traditional one-to-one word of mouth proved insufficient. Very soon, the first systems supporting reputation appeared online, proving themselves essential for trust maintenance and partner selection, and at the same time showing their vulnerabilities under specific attacks [4], pointing out the best statistics for reputation estimation and using simulation to show the importance of cognitive aspects. More recently, surveys on applications for reputation systems have began to appear [7, 9]. Finally, as the field consolidates in the most recent years, simplified recipes [5] have been proposed. Reputation in large scale dynamic system has been studied in EigenTrust [10].

Collaborative Filtering (CF) is a thriving subfield of machine learning, and several surveys expose the achievements in this fields [12]. CF solutions in the literature are often divided in two groups: *memory-based* and *model-based* [2]. Memory-based methods are used in a lot of real-world systems because of their simple design and implementation. However, they impose several scalability limitations that make their use impractical when dealing with large amounts of data. Model-based approaches have been investigated to overcome the shortcomings of memory-based algorithms. The most successful Model-based techniques are by far those based on low-dimensional factor models, as the Netflix Prize (www.netflixprize.com) established, in particular those based on *matrix factorization* (MF) [8]. The most popular MF solutions are *Alternating Least Squares* (**ALS**) and *Stochastic Gradient Descent* (**SGD**). None of the solutions in the literature, to the best of our knowledge, has been applied in a event-based messaging scenario.

3. SYSTEM MODEL AND PROBLEM

We consider a system where a set of users can produce or consume information. Users that exchange information adopt an *event-based publish/subscribe* communication model [1]. Without loss of generality, we assume that a user is either a *publisher*, i.e. a producer of events, or a *subscriber*, i.e. an event consumer. Following the publish/subscribe paradigm, each event is constituted by a set of n values, called *content-based address* (CBA), that characterize it as a point in a n-dimensional *event space*, and a *payload* without a precise structure. Each dimension of the event space is an *attribute* characterized by a continuous or discrete type (numbers, strings, enumerations, etc.) and a finite range of admitted values. A publisher can publish events in the whole event space. A subscriber can select the events he wants to receive by issuing a *subscription* that defines a subset of the event space, and thus restricts the CBAs accepted by the subscriber. If an event is characterized by a CBA included in the set defined by a subscription we say that the event *matches* the subscription. Interactions between publishers and subscribers are decoupled by an *event notification service* (ENS) that receives events injected by publishers and notifies them to all and only the subscribers whose subscriptions are matched.

From an information quality point of view, here we assume that a publisher is characterized by a different level of expertise in different CBAs. This models the fact that a user can be an expert on a specific topic and only have a superficial knowledge on a different one. The expertise of a publisher on each specific CBA has an impact on the quality of events he publishes on that CBA. We also assume that the expertise of a publisher does not change over time. This simplifying assumption intuitively models the fact that we expect the usage of our system to last for a timeframe smaller than the time needed for a user to sensibly vary his expertise on a topic. Each Subscriber is notified about events from the ENS and can express a vote (OK/KO) on each event depending on its personal evaluation of the event quality. As for the publishers, we assume that a subscriber applies a consistent judgment approach when voting for events (i.e. if an event is voted OK by a subscriber, the same event cannot be voted KO by the same subscriber at a different point in time), and, furthermore, that this judgment approach is the same regardless of the subscriber's subscription. The ENS doesn't know the publishers' expertise, nor it knows the judgment approaches applied by subscribers, but we assume it only receives votes expressed on notified events.

The problem we want to solve can be expressed as follows: *how can we minimize the number of notified events that are voted KO by the receiving subscribers ?*

An ideal solution to this problem consists in an ENS able to notify to subscribers all and only the events that will be voted as OK. Given the assumptions that characterize our model, this goal can be reached by reasoning on complete knowledge, i.e. letting the ENS collect votes for each triple $< publisher, CBA, subscriber >$ and then deterministically discard those notification characterized by a triple that previously received a KO vote. However, this solution is clearly poor from a scalability point of view as the number of triples can quickly explode also in relatively small systems,

and consequently the amount of information that the ENS should maintain would quickly become unmanageable. In the next section we introduce a solution that is able to provide performance that approach those of the ideal solution at a fraction of its cost (i.e. number of votes that it must collect and amount of information it needs to maintain).

4. ONLINE FILTERING SOLUTION

Our filtering solution is constituted by an algorithm that runs within the ENS throughout the whole system lifetime. The algorithm is structured in two phases: a *learning* phase (LP) and a *working* phase (WP). The aim of the learning phase is to collect votes and combine them to build profiles. Publisher and subscriber profiles are then used in the working phase to perform filtering on events that match subscriptions. The publisher profiling process consists in assigning to each of them a reputation score for each CBA where they issued events. Subscribers are profiled using votes they express to find a reputation threshold (RT) for each of them. Intuitively, reputation scores assigned to publishers should capture their expertise, while reputation thresholds should capture subscribers quality thresholds.

During the learning phase the algorithm gathers event votes from multiple subscribers, and combines these feedback to compute a publisher reputation score (REP) for the CBA. The REP value for a specific $< publisher, CBA >$ pair changes over time due to votes coming from different subscribers that at runtime decide to subscribe/unsubscribe that CBA. As more and more votes are collected the REP value approaches a asymptotic value that represents the reputation score obtainable by collecting votes from all the subscribers. Limiting the LP duration our system faces a trade off between the quality of its estimation of the ideal REP value and the amount of information (i.e. votes) it must collect.

Every time a publisher publishes a new event in the system the current estimation of its REP value for the target CBA, together with an uncertainty measure called GAP, is piggybacked with the event. During the working phase, before notifying the event to a target subscriber, the algorithm checks if the REP value associated to the event is larger than the RT value of the subscriber. In this case the event is notified, otherwise the event is dropped. Whenever some information is missing (e.g. an event without a REP value, or a non profiled subscriber without a reputation threshold) the event is notified.

Ideally, our solution should create publisher and subscriber profiles such that the REP value associated to an event is greater than the RT value of all and only the subscribers that would vote OK for it.

4.1 Reputation Score

The Beta Reputation [6] provides a mathematical method for computing reputation scores on the basis of binary evaluations (the OK/KO votes). It is based on the Beta distribution, a continuous family of probability functions indexed by two parameters α and β. Given a number of received votes $OK + KO$, the unknown relative frequency of OK votes a producer will receive in the future has a probability distribution expressed by a Beta function with parameters α and β set to the number of OK and KO incremented by one respectively, as in the following:

$$\alpha = OK + 1 \;\; and \;\; \beta = KO + 1 \tag{1}$$

The expected value (mean) of the Beta distribution is the most natural way to estimate the reputation score REP:

$$Rep(OK, KO) = \frac{\alpha}{\alpha + \beta} = \frac{OK + 1}{OK + KO + 2} \tag{2}$$

REP will then have values in the range $(0, 1)$, where value 0.5 represents a neutral rating. We can interpret this expectation (reputation score) as the most likely value for the relative frequency of OK votes that the publisher will obtain in the future.

The standard error of the Beta distribution provides a measure of the inaccuracy of the reputation estimator:

$$se(OK, KO) = \frac{1}{OK + KO + 2}\sqrt{\frac{(OK + 1)(KO + 1)}{(OK + KO)(OK + KO + 3)}} \tag{3}$$

This measure is inversely proportional to the number of votes $(OK + KO)$. Intuitively, the more representative is the subset of voters, the more accurate will be the reputation estimation. The expected value of the Beta distribution and his standard error form a basis for producer profiling, and are the building blocks of our filtering algorithm.

4.2 Filtering algorithm

Producers Profiling — The algorithm associates to each pair $< publisher, CBA >$ two values: a reputation score (REP) and an inaccuracy estimation of this score (GAP). The idea behind this latter score is to define an uncertainty range around the reputation estimation: the boundaries of this interval are defined by adding and subtracting the GAP value from the reputation score. REP is computed using the expected value of the Beta distribution (Eq. 2) and his standard error (Eq. 3) constitutes the GAP. The system updates these values during the learning phase each time a publisher publishes a new event on a CBA, and new votes are collected.

Consumers Profiling — Subscribers are profiled using an explorative mechanism. The goal of the algorithm is to identify a RT value for each subscriber using the votes it expresses. When a subscriber emits a new vote v on a notified event with REP r and GAP g, the algorithm stores a triple $< v, r, g >$ associated to the subscriber. All triples collected for a subscriber at runtime are sorted by placing them on a $[0, 1]$ scale according to a *worst case estimation* approach: given a triple $t =< v, r, g >$ if $v = KO$ the position of the triple is $r - g$, otherwise it is $r + g$. The subscriber's RT value is then chosen by picking the smallest value τ that minimizes the number of positive votes with a position $x_{OK} \leq \tau$ on the scale and the number of negative votes with position $x_{KO} > \tau$. The *worst case estimation* approach prevents inaccurate reputation estimations from corrupting the RT. Indeed, without this mechanism negative votes with over-estimate reputation scores would lead

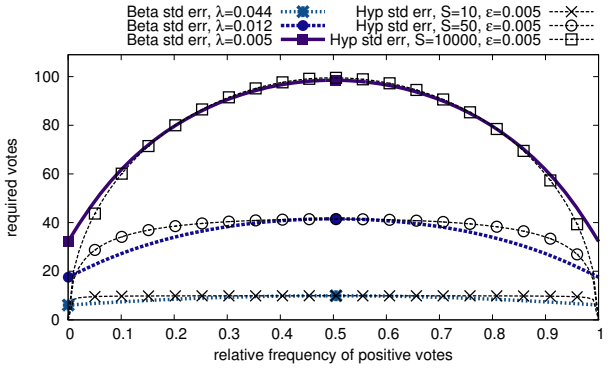


Figure 1: Number of votes required to obtain a specific standard error given a ratio of positive votes.

to over-strict RTs (positive votes with under-estimate reputation would lead to over-permissive RTs respectively).

Profiles Match — The algorithm drops all notifications of events whose attached REP value is lower or equal than the RT value on the target subscribers. The initial values for publishers REP and GAP is ∞ and 0 for subscribers RTs. In this way the first event issued by a publisher on a specific CBA is always delivered to all the subscribers subscribed to that CBA. The filtering mechanism is active from the second event on.

Learning Phase Stop Condition — During the learning phase votes are collected to improve the quality of publisher profiles. The more votes are collected for a pair $< publisher, CBA >$, the more the REP value will converge toward its final value and the GAP value will converge to 0. This convergence process is unlikely to be linear: the first few votes will strongly drive REP toward its target value strongly reducing the GAP, while further votes will marginally refine this result. Obviously, the system incurs some form of overhead for vote collection.For these reasons, the stopping condition for the learning phase represents a crucial point of our algorithm: the goal is to build high quality profiles to provide highly performant filtering with a small number of votes. When the learning phase stops for a pair $< publisher, CBA >$ the algorithm moves to the working phase where the REP value is kept constant (and set to the last value it has during the LP) and the GAP value is cleared and set to 0.

The adoption of naive conditions to decide when the LP should be stopped could lead to low quality profiles (or a large number of collected votes). Given that the GAP represents the inaccuracy of our REP estimation it make sense to consider a condition on its value to stop the LP: when the GAP is smaller than a given threshold the LP can be stopped. Figure 1 depicts, with a solid curve, the amount of votes that the algorithm should collect to have the Beta distribution *se* smaller than a desired value λ ($\lambda = 0.005$ in this specific case).

Our solution is to scale the Beta curve on the basis of the expected population $\overline{S}$ such that the maximum of the Beta *se* curve matches the maximum of the corresponding Hypergeometric *se* curve (represented as dashed curves with empty dots in Figure 1). After scaling the curve, the corresponding λ value can be computed (In the figure, for example, $\epsilon = 0.005$ and $\overline{S} = 50$ for the Hypergeometric leads to $\lambda = 0.012$), and this will constitute the threshold on the GAP value that will decide the LP conclusion. This strategy provides a practical solution to adapt the GAP threshold to various populations and strongly reduces the possibility that some LPs will never end.

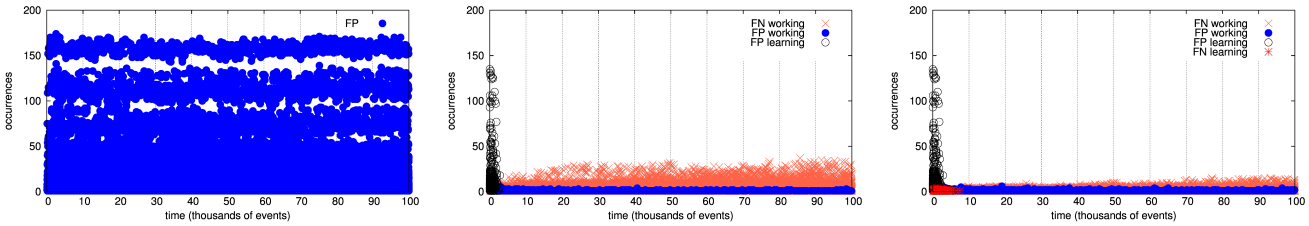
5. EXPERIMENTAL EVALUATION

In this section we report on the experimental evaluation we conducted on our solution. The goal of this evaluation was to assess how much our solution is effective in filtering undesired events and how sensitive it is to different characteristics of the input load.

Performance Metrics — In general, by looking at the final effect of our filtering solution applied to a specific notification we can identify four possible cases: (i) an event notified to a subscriber that votes OK represents a *true positive* outcome (**TP**), (ii) an event notified to a subscriber that votes KO represents a *false positive* outcome (**FP**), (iii) an event that is filtered for a subscriber that would vote OK on its notification represents a *false negative* outcome (**FN**) and, finally, (iv) an event that is filtered for a subscriber that would vote KO on its notification represents a *true negative* outcome (**TN**). To measure the performance of a *binary classifier* the concepts of *sensitivity* and *specificity* are often adopted. If we apply their definitions to our system we have that sensitivity (or True Positive Rate (**TPR**)) is the fraction of subscribers that are notified about an event (TP) among all those that would express an OK vote on its notification (TP + FN). It measures the proportion of actual positives which are correctly identified as such. Specificity (or True Negative Rate (**TNR**)) is the fractions of subscribers that are not notified of an event (TN) among all those that would express a KO vote on its notification (TN + FP). It measures the proportion of negatives which are correctly identified as such Furthermore, we consider the *Matthews correlation coefficient* (**MCC**) that measures the quality of binary classifications. It returns a value between -1 and $+1$ where $+1$ represents a perfect prediction, 0 no better than random prediction and -1 indicates total disagreement between prediction and observation. Finally, we also consider the *knowledge size* (**K**), i.e. the average ratio, among all subscribers in the system, between the number of pairs $< publisher, CBA >$ for which a subscriber s expressed at least a vote and the global number of pairs for which s received an event[1]. Therefore, a good filtering solution should work keeping K as small as possible.

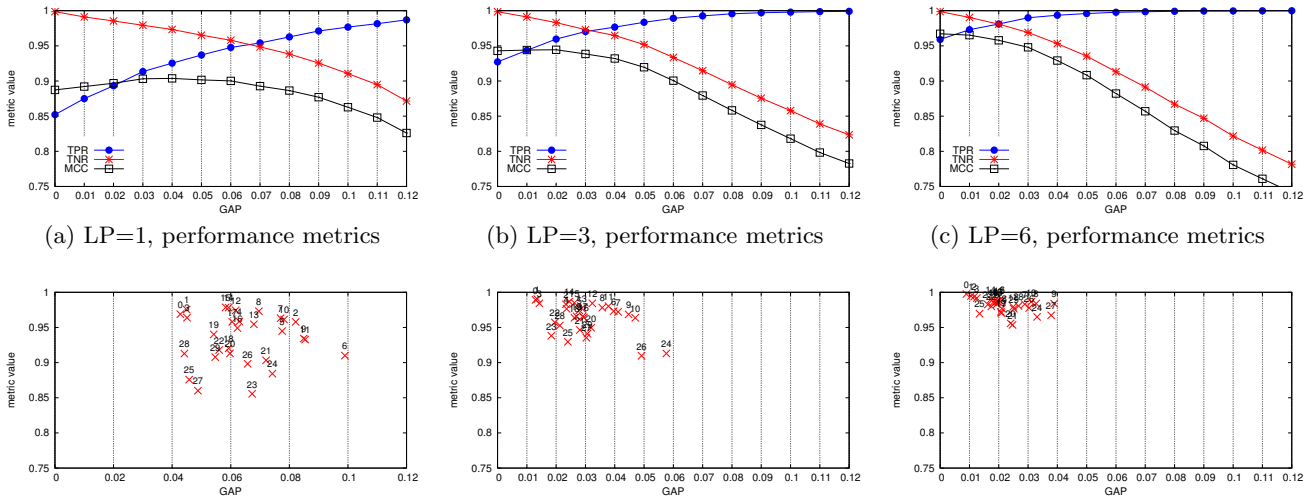
Simulation Model — To perform our evaluation we integrated our filtering solution within the SIENA event notification service [3] and implemented it with the OMNeT++ discrete event simulation environment. We consider a simplified data model where the event space is constituted by a single integer attribute with values ranging from 1 to 30. This simplifying assumption does not impact the generality of the results reported in the remainder of this section

[1]The knowledge size K could be considered as an abstract measure of the overhead generated by our solution.



(a) No filtering (b) $LP = 1$ (c) $LP = 6$

Figure 2: General filter behavior during the learning and working phases.



(a) LP=1, performance metrics (b) LP=3, performance metrics (c) LP=6, performance metrics

(d) LP=1, intersection points (e) LP=3, intersection points (f) LP=6, intersection points

Figure 3: Tradeoff between performance metrics varying LP and GAP.

as the considered metrics only depend on the distribution and amount of events published on CBAs. The behavior of publishers and subscribers was modeled taking into account existing studies on the dynamics of publish/subscribe applications and of churn in peer-to-peer networks [11]. We considered a fixed set of 200 subscribers. Every subscriber expresses 3 subscriptions each targeting a single CBA. A subscription lasts for a limited amount of time, defined by a Weibull distribution (shape $k = 0.5$ and scale $\lambda = 5$), after which it is replaced with a new one. Intuitively most of the subscriptions will last for a short timeframe (minutes) while some subscriptions will be characterized by a long duration (days or weeks). Subscription allocation over CBAs follows a Pareto distribution (scale $x_m = 1$ and shape $a = 1$). The popularity of CBAs decrease proportionally to the CBA id (i.e., CBA with id 1 is the most popular, with id 30 the least). Publishers publish events at a rate characterized by a Poisson distribution ($\lambda = 5$) on CBAs that are chosen uniformly at random in the event space. We considered only 18 publishers in our tests as the metric of interests do not depend from this parameter. Given that our tests are driven by the injection of new events, we limited the duration of each test run to the 100.000th event injected in the system. The publishers' expertise on the available CBAs was modeled using the Beta distribution, randomly varying its parameters ($\alpha, \beta \in [2, 20]$) to assign a different function to every publisher. In this way it is possible to mimic an application scenario where different publishers sport widely different knowledge levels on different topics. Subscriber quality thresholds were modeled as a Normal distribution (mean $\mu = 0.6$ and standard deviation $\sigma = 0.15$). In our simulations we do the simplifying assumption that events published by a publisher on a CBA are characterized by a quality level that corresponds to the expertise level of that publisher on that CBA. Every time a subscriber with quality threshold t is notified about an event with quality q such that $q \leq t$ it votes KO, otherwise it votes OK.

Results: General filter behavior — Figure 2 shows the evolution of the false positive FP metric as new events are injected for three different settings: no filtering applied to events (Figure 2a), filtering with LP limited to the first event produced in each CBA by each publisher ($LP = 1$ in Figure 2b), and filtering with LP limited to the first 6 events ($LP = 6$ in Figure 2c). In the last two graphs the learning and working phases can be easily recognized. Figure 2a shows how the number of false positives is large if no filtering is applied. Graphs in Figures 2b and 2c show a sharp decrease of the number of false positives during the learning phase. At steady-state (working phase) the FP metric is stable with

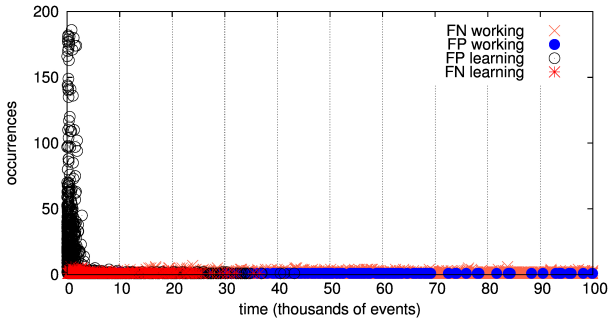


Figure 4: Online filtering solution.

a low number of occurrences. It is worth noting that the introduction of the filter can create false negatives (reported as red crosses in the graphs) whose amount can be controlled by increasing the learning phase duration.

Results: Learning Phase Duration vs. Knowledge — Figure 3 reports the behavior of the filter varying the GAP size (0.00-0.12) and the learning phase duration (1,3 and 6 events). The graphs on the top show the evolution of the TPR, TNR and MCC metrics varying the GAP size. The curves clearly show how the GAP size controls the tradeoff between TPR and TNR: when the GAP is set to 0 the filter is very effective in avoiding the notification of undesired events, but this comes at the cost of a lot of events that are discarded even if they would be rated OK by their target subscribers. As the GAP grows more events fall in its uncertainty window and are thus notified; as a consequence, the TPR metric tends to 1 while the TNR quickly drops to low values. The graphs on the bottom of the Figure depict all the intersection points between TNR and TPR calculated separately for each CBA (the number attached to each point is the CBA identifier), and show how the choice of the "best" GAP size when $LP = 1$ is quite difficult as no size fits all. This is a consequence of the complex dynamics that link event production, publishers expertise, and subscription distribution among the different available CBAs. The MCC metric confirms this as its maximum is obtained with lower values of GAP with respect to the intersection of the average TPR and TNR curves. When we increase the learning phase duration to 3 and 6 events the TPR and TNR curves both tend to 1 and their intersection points moves left toward lower values of the GAP.

These results clarify a fundamental point: given a learning phase duration, the GAP should be sized independently and differently for each CBA as their characteristics may vary, especially when the learning phase is short. On the other side, increasing the LP duration seems to improve performance. This, however, comes at the cost of a larger knowledge size. Keeping LP small while attaining good performance is a desirable result as the knowledge K, needed by the filter to build profiles, grows with it (plot omitted due to space constraints).

Results: Full system evaluation — Figure 4 wraps up all the results reported above by showing the evolution of the FP and FN metrics as events are injected in the system. By comparing this graph with those previously shown in Figure 2b and 2c, the improved results brought by the GAP adaptivity that drives the learning phase durations (separate evaluation for this mechanism has been omitted due to space constraints) are evident. The amount of FPs and FNs during the working phase can be considered residual.

6. CONCLUSIONS

In this paper, we introduced a novel solution for online filtering of information in event-based system. Our solution targets applications where consumers express binary preferences (OK/KO votes) on data they receive to capture their personal perception of the data quality. It collects these preferences to create producer/consumer profiles that are then used to filter out events that consumers would vote negatively. Its effectiveness has been shown through an extensive evaluation where the performance variability with respect to different application load scenarios has been analyzed.

7. REFERENCES

[1] R. Baldoni, L. Querzoni, S. Tarkoma, and A. Virgillito. Distributed Event Routing in Publish/Subscribe Communication Systems. In *MiNEMA State-of-the-Art Book*. Springer, 2009.

[2] J. S. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*, 1998.

[3] A. Carzaniga, D. Rosenblum, and A. Wolf. Design and evaluation of a wide-area notification service. *ACM Transactions on Computer Systems*, 2001.

[4] C. Dellarocas. Immunizing online reputation reporting systems against unfair ratings and discriminatory behavior. In *EC '00: Proceedings of the 2nd ACM conference on Electronic commerce*, 2000.

[5] C. N. Dellarocas. Designing Reputation Systems for the Social Web. *Social Science Research Network Working Paper Series*, 2010.

[6] R. Ismail and A. Josang. The beta reputation system. In *Proceedings of the 15th Bled Conference on Electronic Commerce*, 2002.

[7] A. Jøsang, R. Ismail, and C. Boyd. A survey of trust and reputation systems for online service provision. *Decision support systems*, 43(2):618–644, 2007.

[8] Y. Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th international conference on Knowledge discovery and data mining*, 2008.

[9] M. Paolucci, S. Picascia, and S. Marmo. Electronic Reputation Systems. In *Handbook of Research on Web 2.0, 3.0, and X.0*. IGI Global, 2010.

[10] M. S. S. Kamvar and H. Garcia-Molina. The eigentrust algorithm for reputation management in p2p networks. In *World Wide Web Conf. (WWW)*, 2003.

[11] D. Stutzbach and R. Rejaie. Understanding Churn in Peer-to-Peer Networks. *Proceedings of the 6th conference on Internet measurement*, 2006.

[12] X. Su and T. M. Khoshgoftaar. A survey of collaborative filtering techniques. *Advances in Artificial Intelligence*, 2009.