Faderank: an incremental algorithm for ranking Twitter users

Massimo Bartoletti[1], Stefano Lande[1], and Alessandro Massa[1,2]

[1] Università degli Studi di Cagliari, Italy
[2] Xorovo.com

Abstract. User reputation is a crucial indicator in social networks, where it is exploited to promote authoritative content and to marginalize spammers. To be accurate, reputation must be updated periodically, taking into account the whole historical data of user activity. In big social networks like Twitter and Facebook, these updates would require to process a huge amount of historical data, and therefore pose serious performance issues. We address these issues in the context of Twitter, by studying a technique which can update user reputation in constant time. This is obtained by using an arbitrary ranking algorithm to compute user reputation in the most recent time window, and by combining it with a summary of historical data. Experimental evaluation on large datasets show that our technique improves the performance of existing ranking algorithms, at the cost of a negligible degradation of their precision.

Keywords: reputation, social networks, performance

1 Introduction

The global growth of electronic communication facilities like peer-to-peer and social networks brought out two major problems: how to filter out low quality information, and how to enforce safe interactions. In many contexts, these issues are not completely disjoint: e.g., interacting safely in peer-to-peer networks may correspond to only download trusted contents or files. A possible way to address these issues is to associate each object (peer, user or resource) with an index, usually called *reputation*, which reflects the opinion the network has towards such object. The underlying assumption is that, by analysing the past interaction history of an object, one can predict the quality of its future interactions [10,22].

In the specific context of social networks, like e.g. Twitter and Facebook, reputation has several applications: for instance, it has been exploited to rank users [21,1], to marginalize spammers [26] and dishonest services [5], to distributed moderation among users [14,13], to maximize information spread in viral marketing strategies [11], and to refine search results [18].

Designing effective reputation systems for social networks is not an easy task, for two main reasons. First, they have to deal with the impressive amount of data generated by social networks: for instance, Twitter counts $\sim$10M daily active users and ∼500M tweets per day [2], while Facebook counts ∼900M daily active users, ∼44M comments and ∼4.5B likes per day [3]. Second, reputation system must protect themselves from misbehaving users who try to undermine the ranking mechanism in order to obtain unwarranted service or to prevent honest participants from obtaining legitimate service [8]. Although some defence techniques against these attacks have been proposed over the years [17,23,29,30,6], currently there is no reputation system which is (either provably or empirically) resilient to all kinds of attacks.

To make the situation even more complex, the problem of efficiency and that of security are strictly related. On the one hand, if a system tries to improve efficiency by reducing the frequency of reputation updates, an adversary could easily build a positive reputation in a first period of time, and then exploit it to carry on attacks in the time window where reputation is not updated. On the other hand, frequently recomputing reputation gives rise to a performance problem: ideally, for each update we have to process all the historical data, besides the new data. A possible approach to mitigate this issue is to truncate data older than a certain time: for instance, Klout — a popular reputation aggregator — only considers the last 90 days of user interaction [21]. However, this mechanism can be subject to *whitewashing attacks* where an adversary abuses the system for a while, and then simply waits some time before rebuilding a fresh reputation.

Contributions. In this paper we propose and evaluate a technique to reduce the overhead of keeping updated the reputation of Twitter users. Instead of naïvely truncating historical data, our technique aggregates it in constant time and space, by adapting the *fading memories* technique of [23]. The actual reputation of a user is computed by taking into account its recent (raw) behaviour, its behaviour in the aggregated history, and the gradient of behaviour change. In this way, we reach two goals. First, since the amount of data to be processed at each update is (on average) constant, the average execution time of an update is constant as well. Second, since the past interaction history is taken into account (although in aggregated form), we mitigate *whitewashing attacks* like the ones outlined above. A further feature of our technique is that it is parametric with respect to the reputation algorithm used to process raw data. Overall, one can choose the algorithm which offers the required defences on raw reputation, and calibrate the weights of the raw/aggregated/gradient components to obtain similar properties on the optimized algorithm.

We validate our technique, called *Faderank*, using two raw reputation algorithms: TURank [28], and a variant of PageRank [20] suited to rank Twitter users. In our experiments we use three real datasets, obtained by crawling Twitter for several weeks. Our datasets contain tweets, retweets, and follow relations of ∼10K users, spanning over a period of eleven months. Assuming a monthly reputation update, we compare the completion time of FadeRank, TURank and PageRank, showing that Faderank is a constant-time algorithm, while the computation time of the raw algorithms grows linearly on the size of the input. To evaluate the precision of Faderank we use the *Kendall τ rank distance* [12]. More precisely, for each dataset, for each iteration, and for each raw algorithm

(TURank and PageRank), we measure two distances: the distance between the ranking obtained by Faderank and the raw algorithm, and the distance between the latter and its *forgetful* version, where the history is truncated every month. Our experiments show that, in all datasets, there is a little degradation of the precision of Faderank w.r.t. the raw algorithms, but Faderank is still more precise than their forgetful versions. Further, we show that, compared with the forgetful algorithms, Faderank is more resilient to whitewashing attacks.

The sources of our FadeRank tool, as well as the experimental data used for its validation, are available online at `tcs.unica.it/software/faderank`.

2 Related work

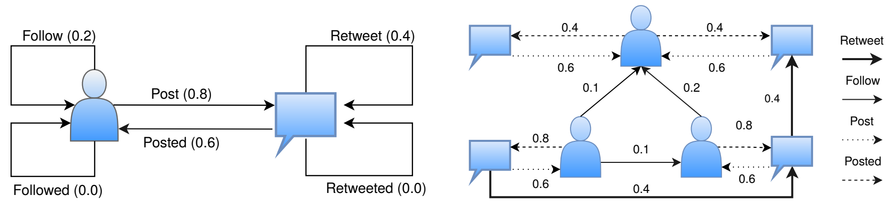
In this section we briefly survey the literature on reputation systems, with special emphasis on those used for ranking users of social networks.

Pagerank. Many reputation systems are based on PageRank [20], an algorithm originally introduced by Google to rank web pages. PageRank models the web as a directed graph (V, E), where V is the set of web pages, and E is the set of hyperlinks (i.e., references) from a page to another. The reputation of a web page is proportional to the reputation of the web pages that reference it. Being based on a single object-object relation, the PageRank model does not precisely capture the rich topology of social networks. In Twitter, for instance, users can follow other users, and send “tweets” which can be “retwitted” by other users. Designing a reputation system which flattens this structure to a single user-user relation may affect the precision of the results; hence, subsequent works have refined the PageRank model to take into account more complex structures.

ObjectRank. ObjectRank [4] generalises the PageRank model by considering different kinds of edges and nodes. Each node gives a part of its reputation to the nodes linked to it. The exact amount of this reputation is determined by (i) the weight on the edge which links the two nodes, and (ii) the reputation of the source node. To account for the fact that different kind of relations may affect the reputation in different ways, ObjectRank allows to associate a different weight to each kind of edge.

To apply ObjectRank to a new domain, one has to instantiate an *authority transfer schema graph* (V_S, E_S, w_S), where V_S is the set of *node kinds*, E_S is the set of *edge kinds*, and $w_S : E_S \to \mathbb{R}$ associates weights to edge kinds. From this schema graph and the dataset, ObjectRank constructs an *authority transfer graph* (V, E, w, k), which is used to compute the reputation. The component V is the set of nodes (the actual objects in the dataset), while E is the set of edges (associated to an edge kind by $k : E \to E_S$). The component $w : E \to \mathbb{R}$ associates a weight to each edge as follows:

$$w(e) = \frac{w_S(k(e))}{OutDeg_{k(e)}(u)} \tag{1}$$



(a) (b)

Fig. 1: In 1a, the user-tweet schema graph; in 1b, a user-tweet graph.

where $OutDeg_{e_S}(u) = \#\{e \in E \mid fst(e) = u \text{ and } k(e) = e_S\}$ is the number of edges of type e_S originating from the node u.

Similarly to PageRank, the reputation is a vector $\boldsymbol{r} \in \mathbb{R}^{|V|}$, defined as the fixed point of the following equation:

$$\boldsymbol{r} = d\,\boldsymbol{A}\,\boldsymbol{r} + \frac{(1-d)}{|V|}[1, ..., 1]^T \tag{2}$$

where d is a real constant (called *damping factor*), and $\boldsymbol{A} = (a_{uv})$ is the *transition matrix* where each element a_{uv} is given by

$$a_{uv} = \begin{cases} w((u,v)) & \text{if } (u,v) \in E \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

ObjectRank computes $\boldsymbol{r}$ through an iterative algorithm, which converges whenever the transition matrix $\boldsymbol{A}$ is irreducible and aperiodic [19]. The first condition is guaranteed by a suitable choice of the damping factor d, while the second one happens to be true for real-world datasets.

TURank. The reputation system proposed in [28], called TURank, instantiates the authority transfer schema graph of ObjectRank into a *user-tweet schema graph* $UTG_S = (V_S, E_S)$, displayed in Figure 1a. The set of nodes V_S comprises just two elements (*user* and *tweet*). The set of edges E_S renders the fact that Twitter users can: (i) dispatch *tweets*, (ii) *follow* users (i.e., when a user A follows B, she will receive all tweets posted by B), and (iii) *retweet* the messages they receive (i.e., if B retweets a message of A, this message will be received by all the followers of B). The weights associated to edges reflect the following assumptions: (i) authoritative users tend to follow other authoritative users; (ii) tweets retweeted by many authoritative users are likely to be interesting; (iii) users who post many interesting tweets are likely to be authoritative.

The authority transfer graph instantiated from the schema and the dataset is called *user-tweet graph*, a minimalistic example of which is displayed in Figure 1b. TURank analyses this graph, using Equation (2) to compute the reputation of Twitter users. Note that, as anticipated in Section 1, to update the reputation one must considering both historical data and new data to reconstruct the user-tweet graph and analyse it. We discuss in Section 3 how our proposal reduces the computational overhead of this operation.

Other adapations of Pagerank. Several other papers propose reputation systems for social networks by taking inspiration from PageRank and ObjectRank. Weng *et al.* [27] propose an algorithm also takes into account the topic similarity between the users (i.e., two users are similar to the extent they tweet on similar topics). More precisely, the algorithm in [27] can associate to each user many reputation values, i.e. one for each topic. To this aim, the PageRank model is modified in [27] so to have a different transition probability between each node (i.e. the values of the transition matrix), depending on the topic similarity between them. Haveliwala *et al.* [7] propose a similar algorithm to improve PageRank using topics, but they use a different "teleportation" vector (the $[1, ..., 1]^T$ vector in Equation (2)) for each topic, instead of changing the transition matrix.

Time-sensitive algorithms. Mariani *et al.* [16] investigate the problem of how temporal aspects affect reputation systems based on PageRank, reaching the conclusion that not considering these aspects undermines their accuracy. Hu *et al.* [9] address this issue in the field of social networks, by adapting PageRank to take into account three time factors: the age of an edge, the frequency with which edges are created, and the topic similarity of the nodes linked by new edges in a certain amount of time, under the assumption that trustworthy users focus their activity in a certain topic for a period of time.

Algorithms based on other techniques Many works propose reputation systems for social networks which do *not* employ the link-structure analysis with the PageRank model, and exploit instead machine learning techniques. Uysal *et al.* [24] compute a reputation for the incoming tweets of a user. The reputation of a tweet is proportional to the probability that the recipient will retweet it, and it is computed using a decision tree. To do that, they propose several features, like e.g. the number of followers of the author, the number of retweets, and the contents of the tweet itself. Wang [26] proposes a similar approach, using a Bayesian classifier to detect spam tweets. Differently from [24], the algorithm in [26] uses features obtained from a graph similar to the user-tweet graph of TURank. Vosecky *et al.* [25] propose a filter model that uses a SVM classifier to discard low quality tweets, and a rank model that uses Rank SVM to order tweet by reputation. They use two sets of features: content-based (like e.g. punctuation, spelling, grammatical indicators), and link-based, that utilizes the implicit relations between tweets, hyper-links, and users. Ma *et al.* [15] associates a reputation to tweets, rather than users. The reputation of a tweet is a measure of its popularity, and it is computed using a *sigmoid function*, taking into account the number of retweets, the number of possible views (i.e., number of user that can see the tweet because they follow the author or other users that retweeted it), and a model of the temporal dynamics of a tweet.

3 FadeRank

In this section we illustrate our technique, which is obtained by suitably combining three basic ingredients:

- an *arbitrary ranking algorithm*, used to compute the *raw reputation* from the data collected in a time interval;
- the *dependable trust model*, used to compute the *aggregated reputation* from the raw reputation and the historical data;
- the *fading memories* technique, used to aggregate the historical data in constant time and space.

Overall, we call FadeRank the combination of these three ingredients. Before introducing in Section 3.3 our algorithm, we present in Sections 3.1 and 3.2 the dependable trust model and the fading memories technique.

3.1 Dependable trust model

We exploit the dependable trust model of [23] to compute the *aggregated reputation* of users, taking as input the raw reputation obtained by an arbitrary ranking algorithm. The aggregated reputation of a given user at time interval $i \geq 0$, denoted by $AR[i]$, is the weighted sum of three components:

$$AR[i] \; = \; \alpha \cdot R[i] \; + \; \beta \cdot H[i] \; + \; \gamma(i) \cdot D[i] \tag{4}$$

where:

- $R[i]$ is the raw reputation of the user at time interval i;
- $H[i]$, defined by Equation (5) below, aggregates in a single value the *history* of the user reputation over the time intervals $0, \ldots, i-1$;
- $D[i]$, defined by Equation (6) below, represents the change of reputation of the user in the last time interval.

More precisely, the value $H[i]$ is computed as follows:

$$H[i] \;=\; \sum_{k=1}^{n} R[i-k] \cdot \frac{w_k}{\sum_{j=1}^{n} w_j} \qquad \text{where } n = \begin{cases} maxH & \text{if } i \geqslant maxH \\ i & \text{otherwise} \end{cases} \tag{5}$$

and where $maxH$ is the number of past raw reputation values stored by the system, and w_k is a weight. Some possible choices for w_k (taken from [23]) and their effects are displayed in Figure 2.

The value $D[i]$ is computed as the difference between the aggregated history and the current raw reputation as follows:

$$D[i] = R[i] - H[i] \tag{6}$$

The weights α, β, and γ in Equation (4) can be tuned to change the response of the reputation system to attacks. A larger value of α gives more weight to the

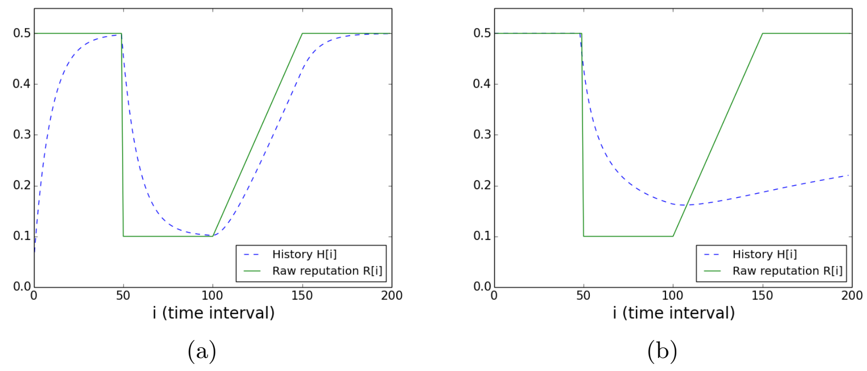


Fig. 2: The figures show a simulation of a whitewashing attack. We compare the trend of the history values $H[i]$ (solid lines) to the raw reputation values $R[i]$ (dashed lines), i.e. the behaviour of the attacker, for two different choices of w_k. Figure 2a shows an *optimistic* choice, i.e. $w_k = \rho^{k-1}$ (with $\rho < 1$), for which Equation (5) becomes an exponentially weighted sum. Instead, Figure 2b shows a *pessimistic* choice, i.e. $w_k = \frac{1}{R[i-k]}$, which yields a harmonic mean.

recent behaviour, while a larger value of β gives more weight to the past history (to address e.g., whitewashing attacks [10]).

The weight $\gamma(i)$ at time interval i is given by:

$$\gamma(i) = \begin{cases} \gamma_1 & \text{if } D[i] \geqslant 0 \\ \gamma_2 & \text{otherwise} \end{cases} \tag{7}$$

where γ_1 and γ_2 are two constants. A possible choice of these constants, as suggested by [23], could be $\gamma_1 < \beta < \gamma_2$. In the first case ($D[i] \geq 0$) we reward by a factor γ_1 an amelioration of the user behaviour, while in the second one ($D[i] < 0$) we penalise by a factor γ_2 its deterioration.

3.2 Fading memories

When computing the history value $H[i]$ in Equation (5), we assume that the system stores the raw reputation values for a user for the past *maxH* intervals. If, to account for the distant past, we were storing a large number *maxH* of values, we would cause a large memory footprint, as well as an increase of the time needed to compute $H[i]$ upon each update. On the other hand, a small value of *maxH* would make the malicious behaviour of a user forgotten after *maxH* intervals, so paving the way to whitewashing attacks.

To cope with this issue, we exploit the *fading memories* technique of [23], which allows to compute a bounded digest of the *whole* past history, and so to compute the value $H[i]$ in constant time and space. To do that, we store only the most recent raw reputation values exactly, while we aggregate (*fade*)

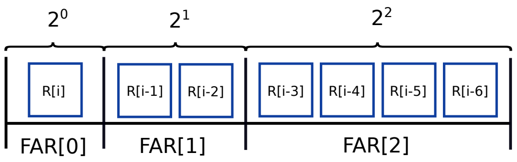


Fig. 3: Fading memories with $b = 2$ and $m = 3$, which aggregate $b^m - 1 = 7$ past raw reputation values into m values. The faded values $FAR[i]$ (for $i \in 0..2$) are obtained by aggregating $b^0 = 1$, $b^1 = 2$, and $b^2 = 4$ past raw reputation values.

the older values, with an accuracy that decreases proportionally to their age. More precisely, fixed two strictly positive integer constants b and m, the fading memories aggregate into m values the past reputation values: the i-th value is the digest of b^i reputation values. The 0-th fading memory is a digest of the b^0 most recent reputation value (i.e., just $R[i]$), the 1-th is a digest of b^1 values (i.e., $R[i-1], \ldots, R[i-b]$), and so on. Since $\sum_{i=0}^{m-1} b^i = b^m - 1$, the m fading memories actually represent a digest of the last $b^m - 1$ reputation values. Figure 3 shows an example of fading memories with $b = 2$ and $m = 3$. The memories for the recent past aggregates a smaller number of raw reputations than the one for the old past, thus making the former more precise.

To update the fading memories at the stroke of a new time interval we use Equation (8) below, where we denote with $FAR^t[i]$ (for $0 \leq i \leq m-1$) the i-th faded past reputation value at interval t of a given user:

$$FAR^{t+1}[i] \quad = \quad \frac{FAR^t[i] \cdot (b^i - 1) \; + \; FAR^t[i-1]}{b^i} \tag{8}$$

3.3 The FadeRank algorithm

Our FadeRank algorithm is illustrated in Algorithm 1. The main routine is FadeRank (lines 1–7), which takes as input *newData*, the user-tweet graph corresponding to the interactions in the most recent time interval. This graph is then passed to a ranking algorithm (line 2), which computes the raw reputations on *newData* (i.e., the ranking algorithm considers *newData* as the whole history of interactions). The call to ComputeReputation (line 4) updates the reputation of a user, exploiting the dependable trust model described in Section 3.1.

The function ComputeReputation takes as input the current fading memories vector *far*, and the *score* for the interval computed before. At line 9 we aggregate the *far* values into a single value to compute *history*, according to Equation (5). The *score* and *history* values are then used to compute the change of reputation *diff* (line 10) according to Equation (6). At line 11 we compute the weight γ as in Equation (7), and then at line 12 we compute the new reputation of the user, exploiting Equation (4). The last step is the update of *far* (line 5), performed by the function UpdateFAR, that employs Equation (8) (line 16).

Algorithm 1 FadeRank

```
1: procedure FADERANK($newData$)
2:     $rawScores \leftarrow$ REPUTATIONALGORITHM($newData$)
3:     for $i$=1 to $|users|$ do
4:         $users_i.score \leftarrow$ COMPUTEREPUTATION($users_i.far$, $rawScores_i$)
5:         $users_i.far \leftarrow$ UPDATEFAR($users_i.far$, $rawScores_i$)
6:     end for
7: end procedure

8: function COMPUTEREPUTATION($far$, $score$)
9:     $history \leftarrow \sum_{i=1}^{|far|} far_i \cdot \frac{w_i}{\sum_{k=1}^{|far|} w_k}$
10:    $diff \leftarrow score - history$
11:    $\gamma \leftarrow$ if $diff \geqslant 0$ then $\gamma_1$ else $\gamma_2$
12:    return $\alpha \cdot score + \beta \cdot history + \gamma \cdot diff$
13: end function

14: function UPDATEFAR($far$, $score$)
15:    for $i$=1 to m do
16:        $newFAR_i \leftarrow \frac{far_i \cdot (b^i - 1) + far_{i-1}}{b^i}$
17:    end for
18:    return $newFAR$
19: end function
```

4 Validation

In this section we validate FadeRank in terms of its performance and precision with respect to two raw ranking algorithms, i.e. TURank [28] and a variant of PageRank [20] tailored to Twitter[3]. Additionally, we compare the precision of both instances of FadeRank with the *forgetful* versions of the raw ranking algorithms, which truncate the history every month.

Hereafter, we shall denote with:

- FadeRank$_{\mathbf{T}}$: the instance of FadeRank which uses TURank as source of raw reputation;
- FadeRank$_{\mathbf{P}}$: the instance of FadeRank which uses our variant of PageRank;
- Forget$_{\mathbf{T}}$: the forgetful version of TURank;
- Forget$_{\mathbf{P}}$: the forgetful version of PageRank.

[3] Note that we cannot use PageRank *as is* because of limitations of Twitter APIs, which do not allow to obtain temporal information about the "follow" relation. To circumvent this limitation, our variant of PageRank operates on the "tweet" and "retweet" relations, by assigning to each user the sum of the score of its tweets.

Dataset	**#Users**	**#Tweet**	**#Follow**	**#Retweet**
D1	∼11K	∼14M	∼15M	∼5M
D2	∼12K	∼12M	∼15M	∼4M
D3	∼12K	∼11M	∼11M	∼3M

Table 1: Datasets details.

Datasets. To the purpose of validation we have constructed three datasets, obtained by a custom crawler which downloads data (i.e., the tweet, retweet and follow relations) exploiting the Twitter APIs. Table 1 shows the details of the datasets; all of them cover a time-span of 11 months. The datasets D1 and D3 contain data of Italian users (the former has a relevant portion of influential users; the latter contains mostly normal users), while the dataset D2 contains data of American users.

In the rest of this section we present the validation results only for dataset D1; the analysis of the other datasets leads to very similar results, so to save space we make it available online at `tcs.unica.it/software/faderank`.

Partitioning the datasets in time intervals. We are interested in evaluating and relating the performance and the precision of *incremental* algorithms (i.e. FadeRank$_{\mathbf{T}}$, FadeRank$_{\mathbf{P}}$, Forget$_{\mathbf{T}}$, and Forget$_{\mathbf{P}}$) with respect to *non-incremental* ones (i.e., TURank and PageRank). To this purpose, we partition each dataset in 11 time intervals, each one comprising data spanning over 30 days. Then, we execute the algorithms to update user reputation, with the following criteria:

- incremental algorithms: for each time interval, process only the data contained in such interval;
- non-incremental algorithms: for each time interval i, process the data from the first to the i-th time interval.

Choice of the parameters. For the purpose of the validation, we have to fix actual values for several parameters:

- the weights α, β, γ_1, γ_2 of the function AR in Equations (4) and (7);
- the weight function w_k in Equation (5);
- the values for the parameters b and m of the fading memories (Equation (8)).

The choice of the parameters for the dependable trust model aims at equalizing the scale and behaviour of FadeRank and of the raw reputation algorithm (TURank and PageRank). To this purpose, we choose $w_k = \rho^{k-1}$ as in Figure 2a (with $\rho = 0.9$), and we compute the weights α, β, γ_1, γ_2 as follows:

1. we start by executing FadeRank$_{\mathbf{T}}$ (resp. FadeRank$_{\mathbf{P}}$) on dataset D1, simulating an update of the user reputation in 30-days intervals;
2. for each update, we save the raw reputation (denoting with R_t^n the raw reputation of user n at interval t) and the *history values* (denoting with H_t^n the history value of user n at interval t);

Algorithm	α	β	γ_1	γ_2	w_k	ρ	b	m
FadeRank$_\mathbf{T}$	0.3	0.9	0.1	0.1	ρ^{k-1}	0.9	2	3
FadeRank$_\mathbf{P}$	0.3	1.2	0.1	0.1	ρ^{k-1}	0.9	2	3

Table 2: Choice of the parameters of FadeRank$_\mathbf{T}$ and FadeRank$_\mathbf{P}$.

3. we execute TURank (resp. PageRank) with the data from interval 0 to t, so to compute the reputation of each user (denoting with T_t^n the reputation of user n at the interval t);
4. we solve the over-determined linear system obtained with the equations in the form $\alpha \cdot R_t^n + \beta \cdot H_t^n = T_t^n$, using the minimum least squares method;
5. finally, we use the solution of the linear system as initial values of α and β, which we fine-tune to mimic the behaviour of TURank (resp. PageRank); using the same criteria we choose the parameters γ_1 and γ_2[4].

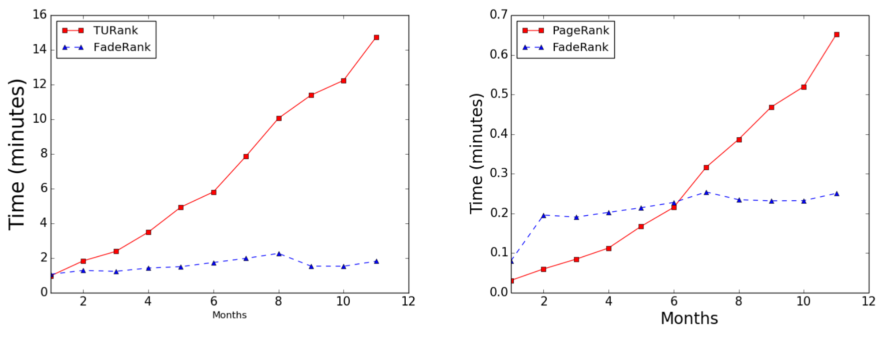
For the fading memories parameters we choose $b = 2$ and $m = 3$, so to have a small number (i.e., $2^3 - 1 = 7$) of values to store. Note that, by increasing the number of fading memories (i.e., choosing bigger values for b and m), the FadeRank algorithm would take account for the past more precisely. Table 2 summarizes the choice of parameters used in the validation.

Performance analysis. Figure 4 shows the execution time of FadeRank$_\mathbf{T}$ *vs.* TURank (4a), and of FadeRank$_\mathbf{P}$ *vs.* PageRank (4b). As expected, the experimental results show that the execution time of the non-incremental algorithms grows linearly with time (because the amount of data to be analysed grows at each update), while the execution time of FadeRank remains more or less constant. Note that the execution time of FadeRank cannot be exactly constant, because the size of the partition of the dataset is not constant (e.g., the monthly number of tweets may vary). The execution time of the forgetful versions of the algorithms (not shown in the figure) are very close to that of FadeRank.

Precision analysis. To evaluate the precision of FadeRank, we consider *rankings*, i.e. list of Twitter users sorted by their reputation (computed each month). More precisely, we compare the ranking of FadeRank$_\mathbf{T}$ (resp. FadeRank$_\mathbf{P}$) with the ones obtained by TURank and Forget$_\mathbf{T}$ (resp. PageRank and Forget$_\mathbf{P}$). To compare two rankings we use the Kendall's τ [12], similarly to [27]. The Kendall's τ is a value in the range $[-1; +1]$ which measures the correlation between two rankings: in particular, $\tau = +1$ indicates that the two rankings perfectly agree (i.e. they are the same), while $\tau = -1$ indicates perfect disagreement (i.e., a ranking is the inverse of the other), and $\tau = 0$ denotes no correlation.

Figure 5 shows the results of our experiments on dataset D1. We see that, despite its constant-time execution, FadeRank$_\mathbf{T}$ loses a small amount of precision

[4] Note that we cannot compute the γ-weights by solving the linear system, because the value D_t^n is a linear combination of the other two (i.e., $D_t^n = R_t^n - H_t^n$).



(a) (b)

Fig. 4: In 4a, the execution time of FadeRank$_{\mathbf{T}}$ (dashed line) and TURank (solid line) for dataset D1. In 4b, the same for FadeRank$_{\mathbf{P}}$ and PageRank.

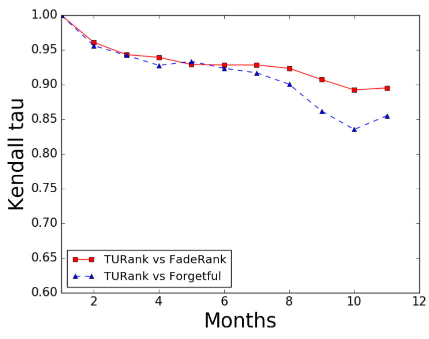
with respect to TURank, but it is still more precise than Forget$_{\mathbf{T}}$ on the long term. The comparison of FadeRank$_{\mathbf{P}}$ and PageRank shows similar results: in this case, the gain of precision of FadeRank with respect to the forgetful algorithm is evident from the starting time intervals.

To further compare the precision of FadeRank with that of the forgetful algorithms, we have experimented on an artificial dataset which represents a whitewashing attack scenario. The dataset, containing 500 users and 400K tweets spanning over 11 months, contains 30% of spammers, who — after being ranked low in the past — begin to share high-quality content in the attempt to whitewash their reputation. Since TURank and FadeRank process the whole past behaviour of users, they can address this attacks, by letting the reputation of spammers increase slowly. Conversely, the forgetful algorithms discard the past history every month, hence they are susceptible to such whitewashing attacks.

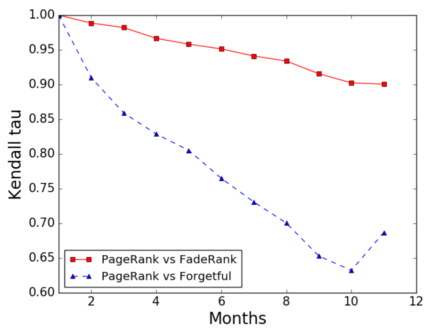
The results of our experiments, reported in Figure 6, show that the hybrid approach adopted by FadeRank, which only records a bounded digest of the past history (namely, $b^m - 1 = 7$ fading memories), is enough to address the whitewashing attack. More precisely, the diagram in Figure 6a shows the expected drop of precision for Forget$_{\mathbf{T}}$ with respect to TURank starting at the seventh month, while the precision of FadeRank remains within $\tau > 0.9$.

5 Conclusions

We have proposed an incremental reputation algorithm for Twitter, which updates the user reputation in (roughly) constant time. In this way we address a performance issue of reputation algorithms, which typically either consider the whole historical data at every update (so suffering from a huge computational overhead), or just discard the past (so being subject to whitewashing attacks).



(a)

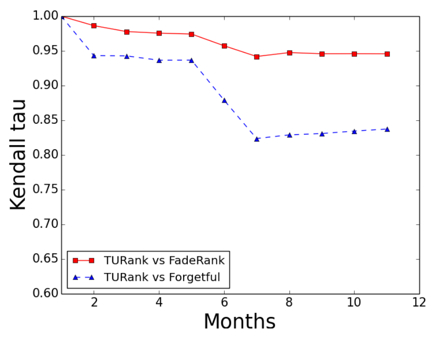


(b)

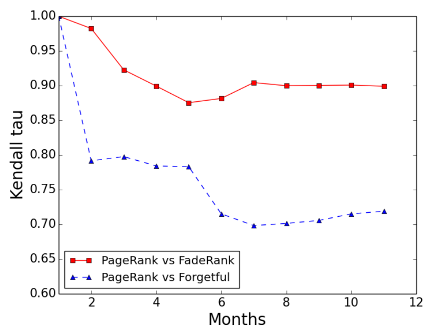
Fig. 5: In 5, the Kendall'sτ correlation between the rankings of FadeRank$_{\mathbf{T}}$ *vs.* TURank (solid line), and Forget$_{\mathbf{T}}$ *vs.* TURank, on dataset D1. In 5b, the same for FadeRank$_{\mathbf{P}}$, Forget$_{\mathbf{P}}$ and Pagerank.

Our algorithm, named FadeRank, exploits the technique introduced in [23] to summarize the past history in a bounded number of values — the *fading memories*. FadeRank combines the fading memories with the raw reputation computed in the most recent time interval, obtained (in constant time) by an arbitrary reputation algorithm. The actual behaviour of FadeRank depends on the weights associated to these components. A dominant weight on the raw component makes the reputation adapt very quickly to the most recent behaviour: a consequence of this choice is that spammers, i.e. malicious users that frequently tweet uninteresting or misleading content, could adopt a *strategic oscillation behaviour*, leading to a whitewashing attack [10]. A user who adopts this behaviour oscillates between building reputation, behaving non-maliciously, and then "milking" reputation, behaving maliciously. Instead, a dominant weight on the historical component makes the reputation adapt more slowly to the recent behaviour, so offering a defence against the above-mentioned attack.

We have validated FadeRank by comparing its performance and precision with those of two standard ranking algorithms, i.e. TURank [28] and a variant of PageRank [20] tailored to rank Twitter users. To perform the validation we have developed a crawler, through which we have downloaded from Twitter three large datasets of tweet/retweet/follow data, spanning over a period of 11 months. Our experiments show that, although the execution time of FadeRank is nearly constant at each reputation update (while, as expected, the execution time of TURank and PageRank grows linearly at each update), the loss in precision with respect to the raw algorithms is very limited. Further, FadeRank outperforms in precision the *forgetful* versions of the raw reputation algorithms, which naïvely truncate the past history. In particular, our experiments show that these algorithms suffer from a huge loss of precision in the presence of whitewashing attacks, while the ranking obtained by FadeRank is still quite close to that of the raw algorithms.



(a)



(b)

Fig. 6: In 6a, the precision of FadeRank$_{\mathbf{T}}$ and Forget$_{\mathbf{T}}$ for the whitewashing dataset. In 6b, the same for FadeRank$_{\mathbf{P}}$ and Forget$_{\mathbf{P}}$.

Acknowledgments. This work has been partially supported by Aut. Reg. of Sardinia P.I.A. 2013 "NOMAD", and by EU COST Action IC1201 "Behavioural Types for Reliable Large-Scale Software Systems" (BETTY).

References

1. Lithium Technologies acquires Klout. http://www.lithium.com/company/news-room/press-releases/2014/lithium-technologies-acquires-klout, accessed: 2016-04-09
2. Twitter usage statistics. Internetlivestats.com http://www.internetlivestats.com/twitter-statistics, accessed: 2016-04-09
3. The top 20 valuable Facebook statistics. Zephoria.com https://zephoria.com/top-15-valuable-facebook-statistics (december 2015), accessed: 2016-04-09
4. Balmin, A., Hristidis, V., Papakonstantinou, Y.: Objectrank: Authority-based keyword search in databases. In: VLDB. vol. 30, pp. 564–575 (2004)
5. Bartoletti, M., Cimoli, T., Murgia, M., Podda, A.S., Pompianu, L.: A contract-oriented middleware. In: FACS. LNCS, vol. 9539, pp. 86–104. Springer (2015)
6. Dimitriou, T., Karame, G., Christou, I.T.: SuperTrust: a secure and efficient framework for handling trust in super-peer networks. In: ACM PODC. pp. 374–375 (2007), http://doi.acm.org/10.1145/1281100.1281180
7. Haveliwala, T.H.: Topic-sensitive pagerank. In: WWW. pp. 517–526. ACM (2002)
8. Hoffman, K.J., Zage, D., Nita-Rotaru, C.: A survey of attack and defense techniques for reputation systems. ACM Comput. Surv. 42(1) (2009)
9. Hu, W., Zou, H., Gong, Z.: Temporal pagerank on social networks. In: WISE. LNCS, vol. 9418, pp. 262–276. Springer (2015)
10. Jøsang, A., Ismail, R., Boyd, C.: A survey of trust and reputation systems for online service provision. Decision support systems 43(2), 618–644 (2007)
11. Kempe, D., Kleinberg, J., Tardos, É.: Maximizing the spread of influence through a social network. In: ACM SIGKDD. pp. 137–146. ACM (2003)
12. Kendall, M.G.: A new measure of rank correlation. Biometrika 30(1/2), 81–93 (1938)

13. Lampe, C., Johnston, E.W., Resnick, P.: Follow the reader: filtering comments on slashdot. In: CHI. pp. 1253–1262 (2007)
14. Lampe, C., Resnick, P.: Slash(dot) and burn: distributed moderation in a large online conversation space. In: CHI. pp. 543–550 (2004)
15. Ma, H., Qian, W., Xia, F., He, X., Xu, J., Zhou, A.: Towards modeling popularity of microblogs. Frontiers of Computer Science 7(2), 171–184 (2013)
16. Mariani, M.S., Medo, M., Zhang, Y.C.: Ranking nodes in growing networks: When pagerank fails. Scientific reports 5 (2015)
17. Michiardi, P., Molva, R.: Core: a collaborative reputation mechanism to enforce node cooperation in mobile ad hoc networks. In: Communications and Multimedia Security. IFIP Conference Proceedings, vol. 228, pp. 107–121. Kluwer (2002)
18. Mislove, A., Gummadi, K.P., Druschel, P.: Exploiting social networks for internet search. In: HotNets. p. 79 (2006)
19. Motwani, R., Raghavan, P.: Randomized algorithms. Chapman & Hall/CRC (2010)
20. Page, L., Brin, S., Motwani, R., Winograd, T.: The pagerank citation ranking: bringing order to the web. Tech. Rep. 1999-66, Stanford InfoLab (1999)
21. Rao, A., Spasojevic, N., Li, Z., DSouza, T.: Klout score: Measuring influence across multiple social networks. In: Big Data. pp. 2282–2289. IEEE (2015)
22. Resnick, P., Kuwabara, K., Zeckhauser, R., Friedman, E.: Reputation systems. Communications of the ACM 43(12), 45–48 (2000)
23. Srivatsa, M., Xiong, L., Liu, L.: TrustGuard: Countering vulnerabilities in reputation management for decentralized overlay networks. In: WWW. pp. 422–431. ACM (2005)
24. Uysal, I., Croft, W.B.: User oriented tweet ranking: a filtering approach to microblogs. In: ACM CIKM. pp. 2261–2264. ACM (2011)
25. Vosecky, J., Leung, K.W.T., Ng, W.: Searching for quality microblog posts: Filtering and ranking based on content analysis and implicit links. In: Database Systems for Advanced Applications. pp. 397–413. Springer (2012)
26. Wang, A.H.: Don't follow me: Spam detection in Twitter. In: SECRYPT. pp. 1–10. IEEE (2010)
27. Weng, J., Lim, E.P., Jiang, J., He, Q.: TwitterRank: finding topic-sensitive influential twitterers. In: ACM WSDM. pp. 261–270. ACM (2010)
28. Yamaguchi, Y., Takahashi, T., Amagasa, T., Kitagawa, H.: TURank: Twitter user ranking based on user-tweet graph analysis. In: WISE. LNCS, vol. 6488, pp. 240–253. Springer (2010)
29. Yu, H., Gibbons, P.B., Kaminsky, M., Xiao, F.: SybilLimit: A near-optimal social network defense against sybil attacks. In: IEEE S&P. pp. 3–17 (2008)
30. Yu, H., Kaminsky, M., Gibbons, P.B., Flaxman, A.D.: SybilGuard: defending against sybil attacks via social networks. IEEE/ACM Trans. Netw. 16(3), 576–589 (2008)