2010 IEEE International Conference on Data Mining Workshops

The Meme Ranking Problem: Maximizing Microblogging Virality

Dino Ienco*, Francesco Bonchi#, Carlos Castillo#

* *Computer Science Department, University of Turin, Italy*

Yahoo! Research, Barcelona, Spain

***Abstract*—Microblogging is a communication paradigm in which users post bits of information (brief text updates or micromedia such as photos, video or audio clips) that are visible by their communities. When a user finds a "meme" of another user interesting, she can eventually repost it, thus allowing memes to propagate virally trough a social network.**

In this paper we introduce the *meme ranking problem*, as the problem of selecting which k memes (among the ones posted their contacts) to show to users when they log into the system. The objective is to maximize the overall activity of the network, that is, the total number of reposts that occur. We deeply characterize the problem showing that not only exact solutions are unfeasible, but also approximated solutions are prohibitive to be adopted in an on-line setting. Therefore we devise a set of heuristics and we compare them trough an extensive simulation based on the real-world Yahoo! Meme social graph, and with parameters learnt from real logs of meme propagations. Our experimentation demonstrates the effectiveness and feasibility of these methods.

I. Introduction

Microblogging is a well-established social communication medium in which users share short snippets of text, images, sounds or videos (*memes* in this paper) with others users. Basically all major social networking platforms including Twitter, Facebook, Tumblr, LinkedIn, Meme, etc., offer microblogging features, although there are minor differences among them, e.g., in the types of meme that can be posted, and major differences in the way people provide feedback to each other (comments, votes, favorites, etc.) and in the way social connections are established (one-way or two-way, with users opting-in or opting-out to being *followed* by another user). However, the basic mechanics are the same for all of them: a user posts a meme, if other users like it, they repost it, and by a process of virality, a large number of users can be potentially reached by a particular meme.

In this paper we devise methods to select, for each user, a set of k memes to show her when she logs into the system. We call this task *meme ranking*.

In selecting the k memes to show to a user, the objective function that we adopt is not simply to maximize the number of these memes that the user is likely to repost, but instead to maximize the global *"virality"* of the selected memes. This means that we do not focus only on the immediate user's satisfaction, favoring the memes which are more likely to interest her. Instead, we also consider the likelihood of her followers of being in turn interested and possibly reposting a given meme, thus recursively propagating it. The rationale for this objective function is twofold. By the general perspective of the network, maximizing the virality of memes and thus the total number of reposts, means keeping high the total level of activity of the network, i.e., its vitality. From the user perspective instead, receiving many reposts might be gratifying, thus enhancing the user's sense of belonging to a community and her engagement with the microblogging network.

In this paper we formally introduce the *meme ranking problem*, which to the best of our knowledge has never been described before in the literature, and we deeply characterize it, highlighting its complexity. In particular, we show that computing the expected spread of a meme is $\#\mathbf{P}$-hard and we provide a theoretical bound on the number of samples needed to approximate it by means of Monte Carlo sampling. It is worth noting that the computation of the expected spread is also the base operation in all the literature on influence spread maximization [1]: hence our theoretical contribution goes beyond the present paper. The conclusion of our analysis is that while Monte Carlo sampling approximation can be afforded in the off-line context of viral marketing [1], it can not be applied in our on-line recommendation context. Therefore, we develop a set of computationally inexpensive heuristics, based on information learnt by analyzing past meme propagations.

II. Related work

Empirical analysis of information propagation. Adar and Adamic [2] describe how information propagates in the blogosphere, tracking the information "epidemics" of interesting blog postings that are referenced or copied by other blogs. Leskovec et al. [3] study the propagation of distinctive snippets of text (typically related to news events) in a corpus of news articles and blogs postings. They develop a scalable clustering algorithm to trace the sources of these fragments of text across the network. Using this algorithm, they are able to infer the structure of the propagation network and use it to determine, for instance, that with respect to new items, blogs lag behind news sources by a few hours.

978-0-7695-4257-7/10 $26.00 © 2010 IEEE
DOI 10.1109/ICDMW.2010.127

Wu et al. [4] study the propagation of information in e-mail networks, showing that the transmissibility of a piece of information decays with network distance. For instance, users that are close together in the organizational hierarchy of a large company, are more likely to share common interests and thus are more likely to "infect" each other with new information. Bakshy et al. [5] describe how information propagates in the on-line game Second Life, tracking the propagation of in-game "gestures" which are information assets that can be copied by other players.

Maximizing information propagation. Suppose we are given a social network together with the estimates of reciprocal influence between individuals in the network. Suppose we want to push a new product in the market. The problem of *influence maximization* is to select the set of initial users, up to a given number, so that they eventually influence the largest number of users in the social network. Kempe *et al.* [1] analyzed influence maximization under on two fundamental propagation models: the *Linear Threshold Model* and the *Independent Cascade Model*. In both these models, at a given timestamp, each node is either active (an adopter of the innovation, or a customer which already purchased the product) or inactive, and each node's tendency to become active increases monotonically as more of its neighbors become active. Time unfolds deterministically in discrete steps. As time unfolds, more and more neighbors of an inactive node u may become active, possibly making u become active, and u's decision may in turn trigger further activation of nodes to which u is connected. Under both propagation models, the influence maximization problem was shown to be NP-hard [1]. Kempe *et al.* however showed that the *influence spread* of a set of nodes is a function with the nice properties of being *monotone* and *submodular*, thanks to which approximation guarantees can be achieved through a simple greedy algorithm, whose key step is the *computation of the expected spread*. As we show later this is a complex computation, for which they run simulations of the propagation model for sufficiently many times to obtain an accurate estimate. While [1] only reports empirical observations on how many simulations are sufficient to obtain a good approximation, in Section IV-B we provide a theoretical bound.

How influential a user is, can also be considered as a domain-specific characteristic, in the sense that a user may be influential in certain topics and not influential in others. Weng et al. [6] study a subset of the Twitter network and compute from network-based and content-based features to measure how influential are users for each topic.

Cha *et al* [7] study different measures of user influence in Twitter. They perform an in-depth analysis on three different measures of influence, namely indegree, retweets and mentions, showing that having a million followers is not necessarily an indication of influence.

III. Empirical analysis of meme propagation

In this section, we describe the Yahoo! Meme dataset and present key observations about how the propagation occurs. The dataset contains all of the publicly-available information visible on the Web site (e.g., users, followed-follower relationship, posted memes, "*via*" links indicating meme reposts, etc.), at the end of November 2009, roughly corresponding to the first 8 months of operation.

Social network. We define a social directed graph based on the *followed-follower* relationship. That is a graph $G = (V, E)$, where V is the set of all the users and for a given pair of users, $u, v \in V$ we say that they are *connected* if: (i) v has added herself explicitly as a follower of u; or (ii) v has reposted a meme by u. In these cases, we draw an arc $(u, v) \in E$. In this representation, the direction of the arc goes from the followed to the follower as that is the direction in which memes eventually propagate.

We discarded from our sample users who were disconnected from the rest of the network according to the relationship described above. The result is a sample with the characteristics described in Table I.

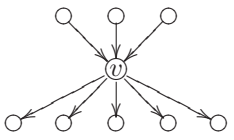
Table I
Summary of properties of our sample.

Number of nodes (users)	57K	
Number of connections:	1.48M	100%
Follows explicitly, but does not repost	1.14M	77%
Follows explicitly and reposts	233K	16%
Reposts, but does not follow explicitly	103K	7%

For the 16% of edges that are both following explicitly and reposted, in roughly 4/5 of the cases the users are first connected by an explicit relationship, and then a re-post occurs, in the remaining 1/5 of the cases, it is first a re-post and then a follow relationship.

In the rest of this paper, we make no distinction between a user v declaring explicitly to be a follower of another user u, or simply reposting a meme posted by u. In both cases, we say that v **follows** u. Figure 1 explains our notation.

Degree distribution is unsurprisingly skewed (Figure 2) with $d_{out}(\cdot)$ having an average of 31 and a median of 5 and $d_{in}(\cdot)$ an average of 31 and a median of 3.



$$|\,\text{followedBy}(v)| = d_{in}(v) = 3$$

$$|\,\text{followers}(v)| = d_{out}(v) = 5$$

Figure 1. Example of the *follows* relation and the notation we use in the rest of the paper. Arcs are always drawn in the direction of the possible propagation of memes.

Memes and propagation dynamics. The sample covers 948K memes, which belong to different types: short snippets of text, photos, audio, or videos. On average each meme

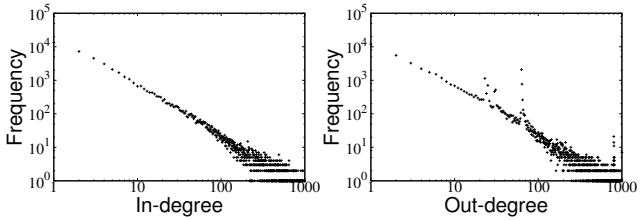


Figure 2. Left: distribution of in-degree. Right: distribution of out-degree in the social network.

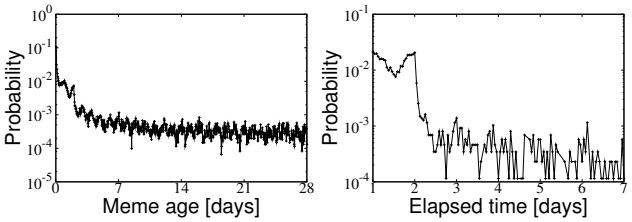


Figure 4. Dependency of post probability with the age of the meme, and with the time passed since the previous post.

generates 2.5 posts (the original post + 1.5 reposts). Not surprisingly, the number of reposts per meme seems to follow a power law distribution, as shown in Figure 3(left). The fraction of memes that are never reposted is 77%, and most memes have very few reposts. Since each user can only repost the same meme once, a meme propagation is a tree. It might happen that two or more users start a propagation of the same picture of piece of news independently and concurrently. However, we treat these cases as different memes.

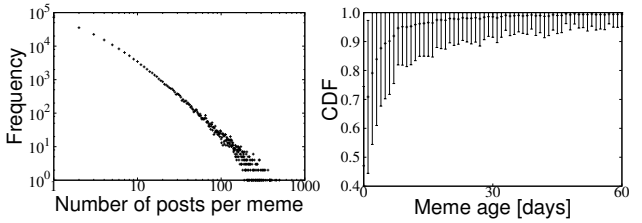


Figure 3. Left: Repost distribution. Right: CDF of the fraction of reposts as a function of time.

As we are interested in understanding highly viral memes, in the rest of this section we focus our analysis on a sample of popular memes posted in the last 3 months of our observation period, and having more than 25 reposts. Our sample contains 1,100 such memes. Statistics on size (total number of reposts), depth (length of longest repost chain), and branching factors (number of reposts per node) of the corresponding propagation trees are reported in Table II.

We can observe that the branching factor at the root is generally much larger than the branching factor at the other nodes (11.7 in average, against 2.4 of overall average). This may be due to the effect of time, as re-posts are more likely to occur shortly after the item is posted for the first time.

Table II
Statistic of the propagation trees of the sample of highly viral memes.

	min	max	avg	median
size	26	823	52.8	40
depth	2	33	8.5	8
branching factor	1	216	2.4	1
branching factor at the root	1	216	11.7	7

Temporal dependency. We observe that most reposts occur shortly after a meme is posted for the first time. Figure 3(right) shows that in most cases over 80% of the reposts of a meme are done in the first 10 days. Examining this fact closer, we find that there are two ways, not necessarily independent, in which repost probability depends from time.

First, as already said, the reposts probability depends on the *age* of a meme, this is the time passed since it was first posted by any user and the present: in Figure 4(left) we can observe an exponential decay in the repost probability during the first few days.

Second, if we consider the time between a particular repost by a user u (not necessarily the first one) and a repost from one of the followers of u, we observe that this is often in a quite narrow interval, as shown in Figure 4(right). This can be explained by constraints in the screen space of the user interface: after some time all memes are eventually moved to the second page, which is rarely visited.

Similarity dependency. We represent memes as bags of words (containing text, tokens in their URLs, etc.) and each user as the concatenation of all the memes she has ever posted or reposted. Next we compute similarity between users and between memes and users using cosine similarity as detailed in Section VI-A. In each case (user-user or user-meme), we discretize the similarity into bins containing the same number of pairs per bin.

Finally, for all the user-user pairs in each s, we compute the probability that a meme posted by one user is re-posted by the other user (when both are connected and their similarity is s); and in the same way, for all the user-meme pairs in each bin s, we compute the probability that a meme posted by one user is re-posted by one of her followers whose similarity with the meme is s.

The result in Figure 5 matches the intuition: the more similar a user is to one of her followers, the more likely that the follower will re-post a meme posted by the followed, and similarly the more similar a meme is to a user, the more likely she is to repost it.

Summary of findings:

- The number of reposts per meme is very skewed and follows a power law.
- A user v is more likely to repost a meme of user u, shortly after u's post. Repost probability also depends on the absolute age of the meme.
- Reposts are more likely between two users who have posted similar memes in the past.
- Reposting a meme m is more likely for users who have posted memes similar to m in the past.

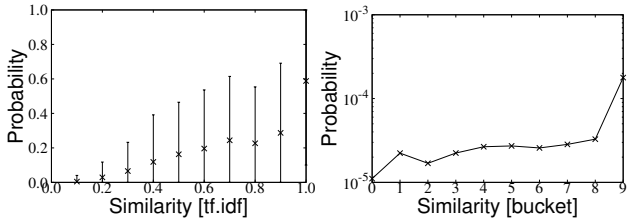


Figure 5. Probability of a repost vs user-user similarity (left) and vs user-post similarity (right).

IV. The meme ranking problem

In this section, we introduce the meme propagation model that we adopt, then we formally define and characterize the meme ranking problem.

A. Meme propagation model

We have a social network modeled as a followed-follower graph, i.e., a directed graph $G = (V, E)$, where an arc $(u, v) \in E$ represent the fact that user v is a follower of user u. We represent by M the set of all memes. In our meme propagation model time unfolds deterministically in discrete steps. If a user u posts a meme m at time t, whether a new post or a repost, we denote this event by the predicate $post(u, m, t)$. Similarly we use the predicate $repost(v, u, m, t)$ to denote that v reposted m "via" u.

We assume (for the moment) that we are able to define the probability of the event $repost(v, u, m, t)$. In other terms, denoting by T the temporal domain, we assume we have a computable function $p : M \times V \times V \times T \to [0, 1]$, that might be defined, for instance, by inference from historic data. We expect this probability to depend on factors such as the influence exerted by user u on her follower v, the topic-interestingness of meme m for v, and elapsed time.

Since in this paper our goal is to devise meme ranking strategies, we focus only two types of posts: initial posts and reposts that are due to the meme rank itself. In other terms we do not consider repost of memes which were not presented to the user by the meme ranking system: even if these are obviously possible in a real system, we are not interested in studying them here. Finally, we assume that a user can not post the same meme twice.

The meme rank is a function ϕ that at each timestamp t selects the top-k memes to show to v from a set of candidates memes $cand(v, t)$. We denote the set of selected memes as $\phi(v, t)$. The set of candidates memes is the union of all memes previously posted by users in V that are followed by v, i.e.,

$$in(v, t) = \{m \in M \,|\, post(u, m, t') \wedge (u, v) \in E \wedge t' < t\}.$$

From $in(v, t)$ we must subtract the memes previously presented to v, or previously posted by v. That is:

$$cand(v, t) = in(v, t) \setminus \{m \in M \,|\, post(v, m, t') \vee \phi(v, t')\}.$$

We have now all the ingredients to define the meme propagation model. At each timestamp t, a user v "decides" for each meme presented to her by the meme ranking function whether to reposts it or not. In our propagation model this decision is probabilistic, i.e., it is determined by flipping a coin of bias $p(repost(v, u, m, t))$.

Note that the same meme can enter the candidate set for v, "via" two different users that v follows. In this case the two instances are considered independently and both can be selected by the meme ranking. If both are selected, the user will flip a coin for both of them independently. The only constraint is that the user can only post a meme once.

We have described how reposts happen. We still have to say how we model initial posts. For sake of uniformity (and later also to implement our propagations simulator) we model initial posts by augmenting our graph G with a special node Ω, which is followed by all the users in V. Essentially Ω is a super-user, the *environment*, which feeds the system with new ideas that the other users can eventually adopt. Node Ω posts at each timestamp some new memes from an infinite queue: these memes are environmental inputs that the users in V might decide to post or not following the same mechanism of reposting described above.

B. Problem characterization and complexity

The meme ranking problem requires to select at each timestamp t and for each user v, the set of k memes whose propagation subtrees rooted in v are expected to be the largest (in number of nodes) among the memes in $cand(v, t)$.

Problem 1 (The Meme Ranking Problem): Given a followers graph $G = (V, E)$ and a $k \in \mathbb{N}$, the Meme Ranking Problem requires to define a function $\phi : V \times T \to 2^M$ that for each user v and a timestamp t, selects a set of memes $\phi(v, t) \subseteq cand(v, t)$, $|\phi(v, t)| = k$, to present to user v. The function must maximize the total number of reposts according to the meme propagation model previously described:

$$maximize \sum_{m \in M} |\{w \in V \,|\, \exists t \in T : post(w, m, t)\}|.$$

The complexity of the problem derives from the fact that the decisions made by the meme ranking function at one node are not independent from the decisions made by the same function at all the other nodes. Another level of hardness is brought in the picture by the probabilistic framework. Indeed any meme $m \in M$ induces on the social graph $G = (V, E)$ a different probabilistic graph $G_m = (V, E, p)$, where p is the function described above that associates to each arc $(u, v) \in E$ the probability that m will "travel" over the arc (i.e., $p(repost(v, u, m, t))$). Those probabilistic graphs are all different because different memes have different chances of being interesting to a given user.

Even removing all the dependencies previously described the problem remains hard. Consider the simpler, local version of the problem, in which we execute the meme rank for a single node v and we assume that whatever posted

by v will also be shown to all the other users. In other words, the meme rank only applies to our node v, while we assume that any other node w can see and repost any meme $m \in cand(w,t)$.

Given a meme $m \in M$, its corresponding probabilistic graph $G_m = (V, E, p)$, and a node $v \in V$, consider the probability space in which each sample point specifies one possible set of outcomes for all the coin flips on the arcs in E. Let X denote one sample point in this probability space. That is, X is a deterministic subgraph of G_m containing all the arcs for which the coin flip has given a positive outcome (m can travel on that arc). Given another node $w \in V, w \neq v$, let $path(v,w)$ be an indicator random variable that is 1 if there exists a directed path from v to w and 0 otherwise. Moreover let $path_X(v,w)$ denote the outcome of such variable in X.

We define the spread of m from v in X as the number of nodes reachable from v in X:

$$\sigma_X(m,v) = \sum_{w \in V} path_X(v,w).$$

We denote the expected spread of a meme m from a node v as $\sigma(m,v)$:

$$\sigma(m,v) = \sum_X \Pr[X] \cdot \sigma_X(m,v).$$

Problem 2 (Local Meme Ranking): Given a user v, a timestamp t, and $k \in \mathbb{N}$, the problem requires to compute $\phi(v,t) \subseteq cand(v,t)$, $|\phi(v,t)| = k$, such that: $\nexists\, m_1 \in cand(v,t) \setminus \phi(v,t)$, $m_2 \in \phi(v,t) : \sigma(m_1,v) > \sigma(m_2,v)$.

The hardness of this simpler problem[1] derives by the fact that computing the expected spread requires to sum over all *possible worlds* X (using the jargon of probabilistic/uncertain data management), and these worlds are $2^{|E|}$.

To further characterize the complexity of this computation, consider the following. By linearity of expectation we have that:

$$\sigma(m,v) = E[\sum_{w \in V} path(v,w)] = \sum_{w \in V} E[path(v,w)].$$

The expected value of an indicator random variable for an event is just the probability of that event. Thus to compute $\sigma(m,v)$ we can just sum the probability of each node w to be reachable from v:

$$\sigma(m,v) = \sum_{w \in V} \Pr[path(v,w) = 1]$$

The problem of computing the probability that two nodes are connected in a probabilistic graph is called *reliability problem*, that is known to be $\#\mathbf{P}$-complete problem [8]. It is very unlikely that there exists a polynomial time exact algorithm for a $\#\mathbf{P}$-complete problem as this would imply that $\mathbf{P} = \mathbf{NP}$, thus usually problems in this class are approximated by means of Monte Carlo sampling (see, e.g. [9], Chapter 10). In our context this would mean: for a meme m and a user v, sample r possible worlds X according to their probability distribution[2], and compute the spread in these deterministic graphs. The average spread is an unbiased estimator of the expected spread of m from v.

The next lemma, establishing a bound on the number of sampled graphs (or possible worlds) needed to provide a good approximation of the expected spread $\sigma(m,v)$, is a direct application of Hoeffding Inequality [10].

Lemma 1: Given a node $v \in V$, a meme $m \in M$, and its corresponding probabilistic graph $G_m = (V, E, p)$. Consider accuracy parameters ϵ and δ, and a number of samples r. Let X_i, $1 \leq i \leq r$, be a set of r graphs sampled according to their probability distribution. The random variables $\sigma_{X_i}(m,v)$ are independent identically distributed and they are bounded in the range $[0, |V| - 1]$. They also have the same expectation $E[\sigma_{X_i}(m,v)] = \sigma(m,v)$.

By selecting $r \geq \frac{(|V|-1)^2}{2\epsilon^2} \ln(\frac{2}{\delta})$, we have:

$$Pr(|\frac{1}{r}\sum_{i=1}^{r} \sigma_{X_i}(m,v) - \sigma(m,v)| \geq \epsilon) \leq \delta;$$

i.e., r samples provide an (ϵ, δ)-approximation for $\sigma(m,v)$.

Discussion. Notice that even though the number of samples is polynomial, and not exponential as the exact counting, due to the factor $(|V|-1)^2$ it is still prohibitively large. In their paper on the *influence maximization problem* [1], Kempe *et al.* need to compute the expected spread in the key step of the greedy algorithm. They find empirically that simulating the propagation $10K$ times brings a reasonably good approximation in a network with approximately $10K$ nodes and $50K$ arcs. This is still very costly, but acceptable in their context as: (1) they must perform it only $k \times |V|$ times (where k is the number of step of the greedy algorithm, that is also the desired size of the set of users to target in a direct marketing campaign); (2) their algorithm runs off-line, as it is in the context of a marketing decision making process, and not an on-line recommendation.

Instead in our context the meme rank must be performed on-line, at each timestamp t, for each user v of the network, and for each candidate meme $m \in cand(v,t)$, where $|cand(v,t)| \gg k$. This makes it prohibitive to adopt simulation based approximation as an actual meme ranking strategy.

[1]This problem is in a sense the converse of the *Influence Maximization* problem, defined in [1] in the context of *Viral Marketing*. In their problem it is given a single piece of information and the problem is that of identifying k users from which to start the propagations so to maximize the expected spread. Oppositely in our problem we're given a single user and we want to select k memes to propagate.

[2]The probability of a possible world X is given by

$$Pr[X] = \prod_{e \in E_X} p(e) \prod_{e \in E \setminus E_X} (1 - p(e)) \ ,$$

where $E_X \subseteq E$ denotes the set of arcs for which the coin flip has given a positive outcome in X.

We also considered to apply simulation only to a bounded extent. In concrete, we tried as meme rank strategy to select the top-k memes w.r.t. their spread up to distance 2 neighborhood of the node v. However, even this light simulation turned out to be computationally prohibitive, or better, unfeasible in a real-world on-line system.

Following these theoretical and empirical considerations, we can only rely on simple and efficient heuristics that can be adopted in a real-world on-line setting, as we do next.

V. Heuristic Methods

Let v be the user for which we are computing the meme rank. We introduce three groups of methods: baseline methods for comparison purposes; user-centered methods, i.e., heuristics geared only on v; followers-centered methods i.e., heuristics geared on the immediate followers of v.

Baseline methods. As baselines for the meme ranking function we consider the RANDOM and RECENCY heuristics.

The RANDOM heuristic simply chooses k memes at random (from the ones in the set of candidate memes for user v at time t). The RECENCY heuristic chooses the k most recent memes from the candidates (the *de facto* standard in most microblogging platforms including Twitter and Yahoo! Meme).

User-centered methods. The first method in this group is the INTEREST heuristic. This strategy chooses the k memes which are more likely to interest user v. In other terms, INTEREST selects the k memes which have the highest *topic similarity* with user v (that we denote $interest(m, v)$), without considering the influence exerted on v by those users that she follows and that posted the memes.

The second method, named INFLUENCE, instead considers as the score of the meme the influence exerted on v by the user which posted the meme, that we denote $influence(u, v)$, without considering topic similarity.

The method combining topic similarity and user influence is named REPOSTPROBABILITY. This is essentially $p(repost(v, u, m, t))$, the probability of reposting introduced in Section IV. This method considers both $influence(u, v)$ and $interest(m, v)$, and it is a function of time.

The details on how $interest(m, v)$, $influence(u, v)$ and $p(repost(v, u, m, t))$ are learnt from a log of historical propagation data will be provided in Section VI-A.

Followers-centered methods. The methods in the third group are based on the set of followers of our user v. The idea is that the meme ranking function should select k memes that are more likely to be reposted by the followers of v. For this purpose we adopt the measures of interest and probability described above (while $influence$ is not appropriate in this case, as it is independent from the meme).

In order to make the heuristics computationally lightweight, we select a fixed number n of followers of v and we aggregate the interest and probability measures over the set of selected followers. In our experiments, except where explicitly stated, we always sample at most $n = 10$, and we aggregate the score by taking the average and the maximum.

As methods to select the n followers to aggregate on, we tried different strategies: random selection, popularity or out-degree (i.e., select the top-n followers that have the largest number of followers), and spread (i.e., select the top-n followers that in the past have received the largest number of reposts). We empirically found that considering the top-n followers w.r.t. spread yields better performance than popularity. This confirms the finding on Twitter by Cha *et al.* [7]: the out-degree of a user is not the best metric of her future influence. Therefore, in what follows, we adopt the spread (computed off-line on a past log of propagations) as criterium to select the n "most important" followers, on which to compute our on-line, light-weight heuristics.

VI. Simulation framework

In this section we describe the simulation environment that we developed to compare different meme ranking strategies. Essentially our simulator is a software that takes in input (1) a social network graph, (2) a set of memes, (3) a function to compute the repost probability $p(repost(v, u, m, t))$. Then, it simulates meme propagations according to the propagation model described in Section IV-A. In our experiments, all these pieces of input come from the Yahoo! Meme dataset described in Section III. In order to map from the real data to the propagation model (and thus to the simulator), we assume an hourly granularity: that is each timestamp in our discrete time model corresponds to an hour. This also implicitly implies that we compute a new meme rank for each user every hour.

Other input parameters of the simulator are the number of memes which will feed the network during the simulation, the number of memes which will enter in the network at each new timestamp, and the total temporal extension of the simulation. Last but not least, the number of simulation rounds that we perform due to the intrinsic probabilistic nature of the propagation model.

The general loop of the simulator is as follows: for each timestamp t and for each user v, the simulator first shows to the user the set of new memes currently posted by the super-user Ω, then it computes the meme rank $\phi(v, t)$ for the given heuristic. For all the memes in $\phi(v, t)$ and the new memes coming from Ω, the user flips a coin to decide which to post and which not to post according to associated the repost probability (exactly as described in Section IV-A).

A. Learning the repost probabilities

An important input for our simulator are the repost probabilities, which we compute as a function of time, interest and influence, from a log of past meme propagations.

1) Interest: We compute $interest(m, v)$ for each meme m and each user v as follows. We represent m as a bag-of-words – considering all the words in its text, or, in the case of images or multimedia files, all the tokens in the URL of the meme. We represent user v as a concatenation of the bags-of-words of all the memes she has ever posted. Next the similarity between a user and a meme, or between two memes, is the cosine similarity of their bags-of-words.

Finally, we also consider an *extended representation* of a meme, in which we concatenate all the bags-of-words representing the users who first re-posted that meme. This extended representation turns out to be necessary in practice as most memes are images, and we do not attempt to process the images to extract visual features from them. Of course any other reasonable meme-representation can be adopted within our framework.

2) Influence: Following [11] we define the influence exerted by u on v by considering that each meme posted by u has a fixed probability of being reposted by v. When a repost occur, we consider this a successful case of influence. Each attempt, that is each meme posted by u, can be viewed as a Bernoulli trial. Hence, influence probability of v on u is estimated as:

$$influence(u, v) = \frac{|\{m \in M \mid \exists t \in T : repost(v, u, m, t)\}|}{|\{m \in M \mid \exists t \in T : post(u, m, t)\}|}$$

3) Repost probability: First, we learn a time independent repost probability from $interest(m, v)$ and $influence(u, v)$. This is done by means of *logistic regression* on a training dataset made of positive and negative instances (reposts happened or not) from Yahoo! Meme data. We denote this learnt probability $p^m_{u,v}$.

Then, inspired by Figure 4(right) and following [11], we incorporate time by means of a step function. That is, the repost probability $p(repost(v, u, m, t))$ remains equals to $p^m_{u,v}$ for an interval of time τ_v, then it drops to a non-null but very small ϵ (in all our experiments we used $\epsilon = 10^{-6}$).

The time interval length τ_v is defined as the ceiling of the average elapsed time, observed in the data, between a post by a user u that is followed by v, and its repost by v.

More precisely, let t_u the time in which u posted m, we define the time-dependent repost probability as follows:

$$p(repost(v, u, m, t)) = \begin{cases} \max(p^m_{u,v}, \epsilon) & \text{if } t - t_u \leq \tau_v; \\ \epsilon & \text{otherwise.} \end{cases}$$

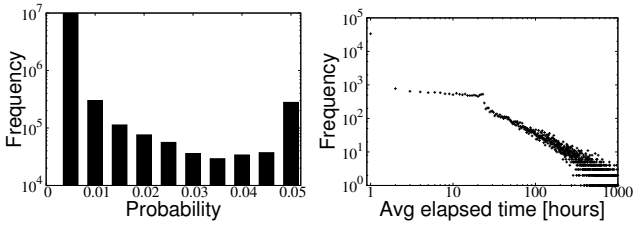


Figure 6. Left: distribution of $p^m_{u,v}$. Right: distribution of τ_v.

The distributions of the learnt parameters $p^m_{u,v}$ and τ_v are reported in Figure 6. We can observe that most of the users have a very short average elapsed time: more than half of the users have $\tau_v = 1$ hour.

In the case of the memes "posted" by the super-user Ω we compute the probabilities $p(repost(\Omega, v, m, t))$ in the same way as we do for all the other users. In this case the probability will represent the likelihood for the user v to start a new meme.

The only difference between the theoretical propagation model and the implementation in the simulator, is that for each meme m posted by Ω we make the user v that has the maximal $p(repost(\Omega, v, m, t))$ post the meme deterministically: this way we force all the input memes to enter in the network at least once. All the other users will flip the coin as usual. The effect is that all the memes will have at least one initiator, and sometimes (rarely) more than one.

VII. Meme ranking experiments

Next we report our experimental results obtained by running simulations of the behavior of the network during one simulated month (720 hours). We feed the network by making Ω offer 10 memes (real memes from Yahoo! Meme) per hour during the first 100 simulated hours, for a total of 1,000 memes. The social network contains 57K users as described in Section III. At each timestamp, the meme rank presents up to $k = 5$ posts to each user. We test all the strategies described above, making 50 independent runs of the simulator for each strategy. The results in terms of total network activity (number of reposts), averaged across all the runs of each experiment, are shown in Table III.

Table III
Final network activity for each meme ranking strategy, expressed in thousands of posts. Boldface indicate the best in each group. The last column is the wallclock time of one run of the simulator.

Method	Activity [K]	Time [H:M]
Random	29 ±1	0:03
Recency	**59** ±1	0:16
Interest	49 ±2	0:29
Influence	47 ±1	0:26
RepostProbability	**58** ±1	0:36
Followers10MaxProb	53 ±1	0:29
Followers10AvgProb	**55** ±1	0:36
Followers10MaxInt	47 ±1	0:26
Followers10AvgInt	48 ±1	0:26
Combo-PR	61 ±3	0:51
Combo-PF	**64** ±1	0:53
Combo-PRF	63 ±2	1:25

A first observation is that all methods perform much better than Random, with Recency generating more than the double of global activity. The good performance of Recency is consistent with the dependency of repost probability from time. The importance of time is also confirmed by the fact that RepostProbability performs much better than Interest and Influence (recall that RepostProbability is a combination of Interest and Influence to which we incorporate the time dependency).

The methods based on the followers of the user suffer from the fact that taken alone, they do not consider at all the probability that a meme will pass the first obstacle, that is, the likelihood that our user will post it when presented by the meme rank. This is obviously an important limit, but it can be overcome when combining with methods that keep in consideration such first and main hurdle. Finally, even in this group of methods, the ones that are time-dependent outperform those that are not, with average outperforming maximum as aggregation method.

Combined methods. Table III also contains results of heuristics obtained by combining the best methods from each group. Specifically: COMBO-PR aggregates REPOSTPROBABILITY and RECENCY, COMBO-PF aggregates REPOSTPROBABILITY and FOLLOWERS10AVGPROB, finally COMBO-PRF combines all the three methods. The aggregation of the ranks in each case is done by simple *Borda counting* (see http://en.wikipedia.org/wiki/Borda_count).

As expected all the combined methods work well, outperforming all the base heuristics. It is interesting to note that COMBO-PF performs slightly better than COMBO-PRF this can be explained by the fact that the temporal dependency is already considered by REPOSTPROBABILITY, thus making RECENCY superfluous.

Efficiency. Our straightforward implementation is quite efficient. We run it on a a quad-processor Intel Xeon 3GHz with 16GB of RAM. The most expensive heuristic to simulate is COMBO-PRF which takes about one hour and a half to simulate one month of network activity, as shown in the last column of Table III. Running time is also indicative of the computational requirements of the strategies in practice.

The conclusion we can draw from these experiments is that the best trade-off between efficacy and efficiency is achieved by COMBO-PF which essentially combines the probability of reposts of the given user with that of its immediate followers. Therefore, we can conclude that considering the followers of a user induces improvements in the performance achievable by only considering the user herself, or the recency of the meme.

VIII. CONCLUSIONS

We introduce the *Meme Ranking Problem*, as the problem of selecting k memes to show to a user when she logs into a microblogging system. The objective is to maximize the overall activity of the network, that is, the total number of reposts that occur. We propose a meme propagation model based on empirical observations drawn from an analysis of meme propagations in Yahoo! Meme. We choose Yahoo! Meme as the basis of our analysis, because this microblogging platform is more explicitly oriented towards creating information cascades[3] than other platforms that have a more conversational attitude.

[3]Yahoo! Meme slogan is CREATE-FOLLOW-REPOST.

We characterize the computational hardness of the meme ranking problem justifying the need for efficient yet effective heuristic methods. Hence we devise several heuristics based on estimating the probability of reposts. Such probability is learnt from a real log of past propagations and it takes into account the influence existing among users, the topical similarity, or interest, of a meme to a user, and the intrinsic decay of the repost probability along time. We compare our methods with the recency-based heuristic, which is the *de facto* standard for microblogging platforms such as Twitter and Yahoo! Meme. The results of our experiments confirm that (1) the proposed methods are feasible and can be adopted as an on-line meme ranking method, and (2) they induce a total level of network activity larger than what is achieved with the baseline methods.

As online social networks continue becoming not only larger, but also more densely connected, the problem of selecting what to show to users about their contacts' activities will become an even more pressing issue. Presumably, the meme ranking problem will increase in practical importance, and thus finding more effective solutions to the meme ranking problem will be part of our future investigations.

REFERENCES

[1] D. Kempe, J. M. Kleinberg, and É. Tardos. Maximizing the spread of influence through a social network. In *KDD'03*.

[2] E. Adar and L. A. Adamic. Tracking information epidemics in blogspace. In *Web Intelligence 2005*.

[3] J. Leskovec, L. Backstrom, and J. Kleinberg. Meme-tracking and the dynamics of the news cycle. In *KDD'09*.

[4] F. Wu, B. A. Huberman, L. A. Adamic, and J. R. Tyler. Information flow in social groups. *Physica A: Statistical and Theoretical Physics*, 337(1-2):327–335, June 2004.

[5] E. Bakshy, B. Karrer, and L. A. Adamic. Social influence and the diffusion of user-created content. In *EC'09*.

[6] J. J. Jianshu Weng, Ee-Peng Lim and Q. He. Twitterrank: Finding topic-sensitive influential twitterers. In *WSDM 2010*.

[7] M. Cha, H. Haddadi, F. Benevenuto and K. P. Gummadi. Measuring User Influence in Twitter: The Million Follower Fallacy. In *Proc. of the Fourth International AAAI Conference on Weblogs and Social Media (ICWSM '10)*.

[8] L. G. Valiant. The complexity of enumeration and reliability problems. *SIAM Journal on Computing*, 8(3):410–421, 1979.

[9] M. Mitzenmacher and E. Upfal. *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*. Cambridge University Press, 2005.

[10] W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, March 1963.

[11] A. Goyal, F. Bonchi, and L. Lakshmanan. Learning influence probabilities in social networks. In *Proc. of the 3rd ACM Int. Conf. on Web Search and Data Mining (WSDM 2010)*.