The Meme Ranking Problem: Maximizing Microblogging Virality

Francesco Bonchi, Carlos Castillo, Dino Ienco

Author-produced version of the article published in Journal of Intelligent Information Systems,40(2), 211-239.
The original publication is available at http://www.springerlink.com/content/0925-9902/?k=Ienco
DOI:10.1007/s10844-011-0181-4

Meme Ranking to Maximize Posts Virality in Microblogging Platforms

Francesco Bonchi[1], Carlos Castillo[1], and Dino Ienco[2]

[1] Yahoo! Research, Barcelona, Spain
E-mail: {bonchi,chato}@yahoo-inc.com
[2] Cemagref, UMR TETIS, Montpellier, France
E-mail: dino.ienco@teledetection.fr

Abstract. Microblogging is a modern communication paradigm in which users post bits of information, or *"memes"* as we call them, that are brief text updates or micromedia such as photos, video or audio clips. Once a user post a meme, it become visible to the user community. When a user finds a meme of another user interesting, she can eventually repost it, thus allowing memes to propagate virally trough the social network.
In this paper we introduce the *meme ranking problem*, as the problem of selecting which k memes (among the ones posted by 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, using parameters learnt from real logs of meme propagations. Our experimentation demonstrates the effectiveness and feasibility of these methods.

1 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 other users. Currently most major social networking platforms including Twitter, Facebook, Tumblr, LinkedIn, Yahoo! 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.

Author-produced version of the article published in Journal of Intelligent Information Systems,40(2), 211-239.
The original publication is available at http://www.springerlink.com/content/0925-9902/?k=Ienco
DOI:10.1007/s10844-011-0181-4

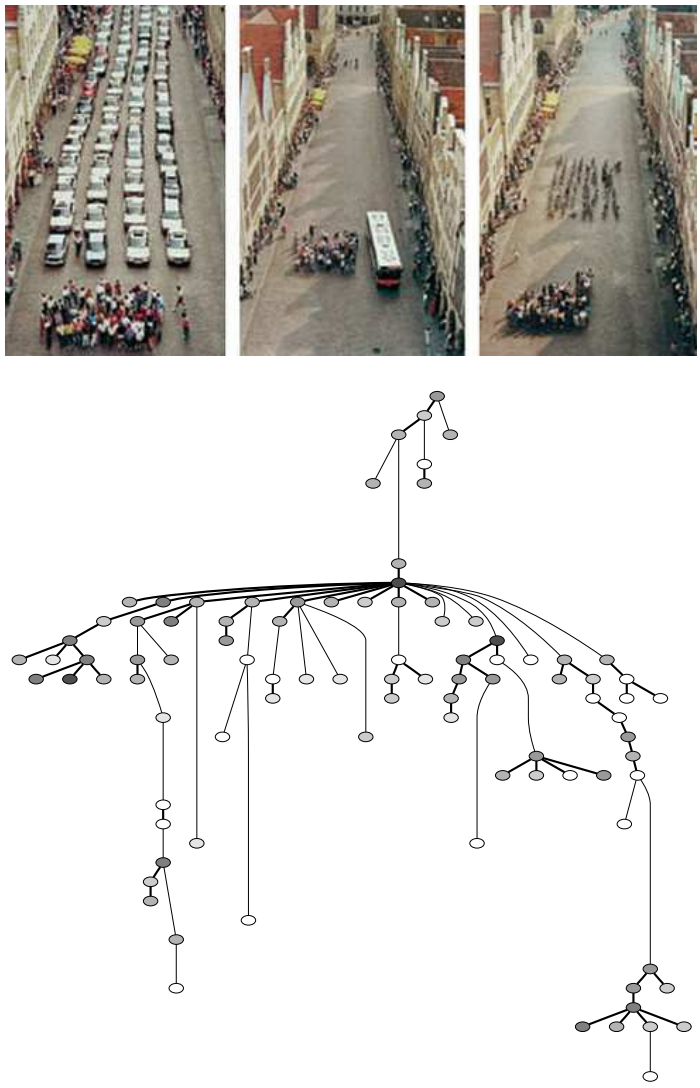

Fig. 1. Above: a meme about sustainable transportation. Below: depiction of the propagation history of this meme. Nodes are users with darker colors indicating nodes with more followers. Each edge indicates a repost, with the length of the edge proportional to the elapsed time; same-day reposts are short, thick lines. The entire history (top-to-bottom) is 60 days long.

Author-produced version of the article published in Journal of Intelligent Information Systems,40(2), 211-239.
The original publication is available at http://www.springerlink.com/content/0925-9902/?k=Ienco
DOI:10.1007/s10844-011-0181-4

Figure 1 shows one specific meme from the microblogging service Yahoo! Meme.[3] In this particular case, it is a photo posted by a user that shows the space required to transport a number of people by car, bus, or bicycle. We depict the propagation history[4] of this meme along time, similarly to [1]. Each node is a user: the darker a node is, the more followers the user has in the social graph. In this particular case, we can observe that the meme was not so popular until it was reposted by an influential user (very dark node), which then spawned a large number of reposts.

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.

Suppose a user receives $1 for each repost of her memes done by her followers, by the followers of her followers, and so on recursively. The objective of the meme ranking problem is to maximize users' profit. Stated a bit more formally, what we want to maximize is the size of the meme propagation trees.

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 $\#\mathbf{P}$-complete 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 [21]: 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 [21], 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.

[3] `http://meme.yahoo.com`

[4] Animations of several meme propagations are available at `http://barcelona.research.yahoo.net/memerank`

Author-produced version of the article published in Journal of Intelligent Information Systems,40(2), 211-239.
The original publication is available at http://www.springerlink.com/content/0925-9902/?k=Ienco
DOI:10.1007/s10844-011-0181-4

Paper contributions and organization

Summarizing the main contributions of the paper are as follows:

- In Section 3 we present an empirical analysis of meme propagations in Yahoo! Meme. The findings of this analysis are important to understand the mechanism of meme propagations, and to reproduce them in our propagation model and simulation framework.
- In Section 4 we first define the *meme propagation model*, then we introduce and deeply characterize the *meme ranking* problem. In particular, we show that computing the expected spread of a meme is #**P**-complete and we provide a theoretical bound on the number of samples needed to approximate it by means of Monte Carlo sampling. Our analysis highlights that, not only exact solutions are unfeasible, but also approximated solutions are prohibitive to be adopted in an on-line setting.
- In Section 5 we develop a set of computationally inexpensive heuristics, grounded on a careful analysis of how propagation occurs in a real social network.
- In Section 6 we present our simulation framework which implements the meme propagation model based on the actual Yahoo! Meme social graph, and on parameters learnt from real logs of meme propagations.
- In Section 7 we report the results of our experiments, showing that the proposed computationally inexpensive methods can increase a network's vitality.

Related works are discussed in the next section, and the last section of the paper outlines future research directions opened by this paper and presents concluding remarks.

2 Related Work

In recent years the whole area of analysis of social network systems has branched out in several sub-disciplines which focus on different aspects. These include the characterization and prediction of links, the detection of communities, and the study of influence propagation, among others. We focus on the latter outlining key results related to our work.

2.1 Empirical analysis of information propagation

In recent years, there has been tremendous interest in the phenomenon of influence exerted by users of an online social network on other users and in how it propagates in the network. The idea is that when a user sees their social contacts performing an action that user may decide to perform the action themselves. In truth, when a user performs an action, she may have any one of a number of

Author-produced version of the article published in Journal of Intelligent Information Systems,40(2), 211-239.
The original publication is available at http://www.springerlink.com/content/0925-9902/?k=Ienco
DOI:10.1007/s10844-011-0181-4

reasons for doing so: she may have heard of it outside of the online social network and may have decided it is worthwhile; the action may be very popular (e.g., watching a blockbuster film may be such an action); or she may be genuinely influenced by seeing her social contacts perform that action [13]. If there is genuine influence, it can be leveraged for a number of applications, the most famous among which is viral marketing. Other applications include personalized recommendations [32, 31] and feed ranking in social networks [30]. Besides, patterns of influence can be taken as a sign of user trust and exploited for computing trust propagation [17, 38, 14, 33] in large networks and in P2P systems.

Thanks to the richness of available data, one domain in which most of the analyses have been done is the blogging and microblogging domain. Gruhl *et al.* [16] characterize four categories of individuals based on their typical posting behavior within the life cycle of a topic, then they develop a model for information diffusion based on the theory of the spread of infectious diseases, capturing how a new topic spreads from blog to blog. They also devise an algorithm to learn the parameters of the model based on real data, and apply the algorithm to blog data, thus being able to identify particular individuals who are highly effective at contributing to the spread of infectious topics. In another work [3] it is shown that bloggers are more likely to join a group that many of their friends joined, especially if those friends belong to the same clique. Song *et. al* instead show that blogs are likely to link to content that other blogs have linked to [31]. In [2] instead the problem of how to identify influential bloggers is studied.

Adar and Adamic [1] describe how information propagates in the blogosphere, tracking the information "epidemics" of interesting blog postings that are referenced or copied by other blogs. The authors consider different features including structural information of the blog network, contents of the blogs, and temporal information to set up a classification scenario that predicts if two blogs are likely to be linked, and if one blog is likely to "infect" another blog with a post. The propagation information available in their case is incomplete, as many blogs rarely cite their sources. However there are ways of inferring the sources of a posting.

Leskovec et al. [23] 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.

Wu et al. [36] 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. Using a simulation model they

Author-produced version of the article published in Journal of Intelligent Information Systems,40(2), 211-239.
The original publication is available at http://www.springerlink.com/content/0925-9902/?k=Ienco
DOI:10.1007/s10844-011-0181-4

find that it is more easy that the transfer happens between two friends instead that two strangers. They also find that some users are more central in the process of information diffusion and more susceptible to obtain new content than normal users. They define this kind of user an early adopter. They also notice that information passed among contacts produce deeper propagation trees for non-popular (niche) assets.

Cha *et al.* [7] present a data analysis of how picture popularity is distributed across the Flickr social network, and characterize the role played by social links in information propagation. Their analysis provides empirical evidence that the social links are the dominant method of information propagation, accounting for more than 50% of the spread of favorite-marked pictures. Moreover, they show that information spreading is limited to individuals who are within close proximity of the uploaders, and that spreading takes a long time at each hop, oppositely to the common expectations about the quick and wide spread of word-of-mouth effect. [22] also shows that the photos users view in Flickr are often the ones they can observe their friends consuming too.

Crandall *et al.* [11] describe a framework for using data from large online communities to analyze the interactions between social influence and users similarity. Their empirical analysis over the social network of Wikipedia editors and LiveJournal users confirms that there exists a feedback effect between users' similarity and social influence, and that combining features based on social ties and similarity is more predictive of future behavior than either social influence or similarity features alone, showing that both social influence and ones own interests are drivers of future behavior and that they operate in relatively independent ways.

2.2 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 the following: given such a network, how 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.

Domingos and Richardson [12, 29] were the first to consider the propagation of influence and the problem of identification of influential users as an optimization problem: they developed a probabilistic model of interaction, and provided heuristics for choosing the influential users.

Later Kempe *et al.* [21] analyzed influence maximization as a problem of discrete optimization. In particular their work focuses on two fundamental propagation models: the *Linear Threshold Model* and the *Independent Cascade Model*. In both of 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

Author-produced version of the article published in Journal of Intelligent Information Systems,40(2), 211-239.
The original publication is available at http://www.springerlink.com/content/0925-9902/?k=Ienco
DOI:10.1007/s10844-011-0181-4

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 these propagation models, the influence maximization problem was shown to be NP-hard [21]. 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 these properties, approximation guarantees can be achieved through a simple greedy algorithm, whose key step is the *computation of the expected spread*. As we show later in Section 4.2, this is a complex computation, therefore they rely on running simulations of the propagation model for sufficiently many times to obtain an accurate estimate. While Kempe *et al.* only report empirical observations on how many simulations are sufficient to obtain a reasonable approximation, in Section 4.2 we provide a theoretical bound.

Following [21] a recent line of research [25, 9, 8, 10] has started developing methods for improving the efficiency of influence maximization algorithms.

Leskovec *et al.* [25] study the propagation problem by a different perspective namely *outbreak detection*: how to select nodes in a network in order to detect the spread of a virus as fast as possible? They present a general methodology for near optimal sensor placement in these and related problems. They also prove that the influence maximization problem of [21] is a special case of their more general problem definition. By exploiting submodularity they develop an efficient algorithm based on a "lazy-forward" optimization in selecting new seeds, achieving near optimal placements, while being 700 times faster than the simple greedy algorithm. Regardless this big improvement over the basic greedy algorithm, their method still face serious scalability problems as shown in [9]. In that paper, Chen *et al.* improve the efficiency of the greedy algorithm and propose new degree discount heuristics that produce influence spread close to that of the greedy algorithm but much more efficiently.

Tang *et al.* introduce the novel problem of topic-based social influence analysis [34]. They propose a Topical Affinity Propagation (TAP) approach to describe the problem using a graphical probabilistic model. They also deal with the efficiency problem by devising a distributed learning algorithm under the map-reduce paradigm. They also discuss the applicability of their approach to the problem of expert finding.

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. [20] 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.* [6] study different measures of user influence in Twitter. They perform an in-depth analysis on three different measures of a user influence, namely in-degree, re-tweets and mentions. The first is the number of people that follow the given user, the second is the number of times other users repost her content, and the last is the number of times other users mention her name. Among various findings, they empirically show that having a million followers is not necessarily an indication of influence.

Author-produced version of the article published in Journal of Intelligent Information Systems,40(2), 211-239.
The original publication is available at http://www.springerlink.com/content/0925-9902/?k=Ienco
DOI:10.1007/s10844-011-0181-4

3 Analysis of Propagations in Yahoo! Meme

In this section, we describe the Yahoo! Meme dataset and present key observations about how propagations occur. We describe our dataset pre-processing, the model we use to understand the data, some statistical properties of users, memes, and propagation histories, and a summary of empirical findings.

The dataset that we analyze 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 of the system.

3.1 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 1.

Table 1. 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 2 explains our notation.

The clustering coefficient of the (undirected) graph is 0.25, which is similar to what has been observed in other social networks such as Flickr (0.31) and LiveJournal (0.33) [27]. The fraction of pairs who are reciprocally connected is around 29%, which is relatively low compared to Flickr (62%), LiveJournal (74%), or Twitter (58%) [19], perhaps because this network is directly oriented

Author-produced version of the article published in Journal of Intelligent Information Systems,40(2), 211-239.
The original publication is available at http://www.springerlink.com/content/0925-9902/?k=Ienco
DOI:10.1007/s10844-011-0181-4

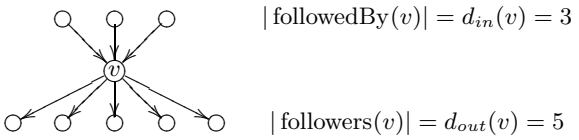

Fig. 2. Example of the *follows* relation and the notation we use in the rest of the paper. Arcs are always drawn in the direction of the possible propagation of memes.

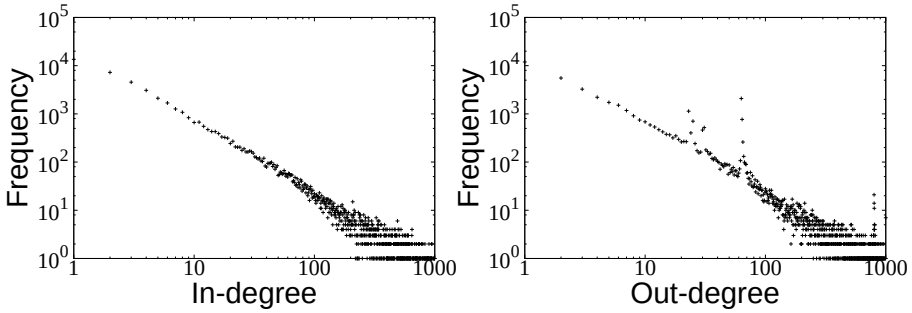

Fig. 3. Distributions of in-degree and out-degree in the social network.

to the propagation of interesting or amusing information, instead of having a more conversational nature.

With respect to the number of followers $d_{out}(\cdot)$, the distribution is unsurprisingly very skewed, with an average of 31 and a median of 5. In the case of the number of users being followed $d_{in}(\cdot)$, the distribution is also skewed with an average of 31 and a median of 3. The two distributions are shown in Figure 3.

3.2 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 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 4(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 (as the one in Figure 1). 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.

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 2.

Author-produced version of the article published in Journal of Intelligent Information Systems,40(2), 211-239.
The original publication is available at http://www.springerlink.com/content/0925-9902/?k=Ienco
DOI:10.1007/s10844-011-0181-4

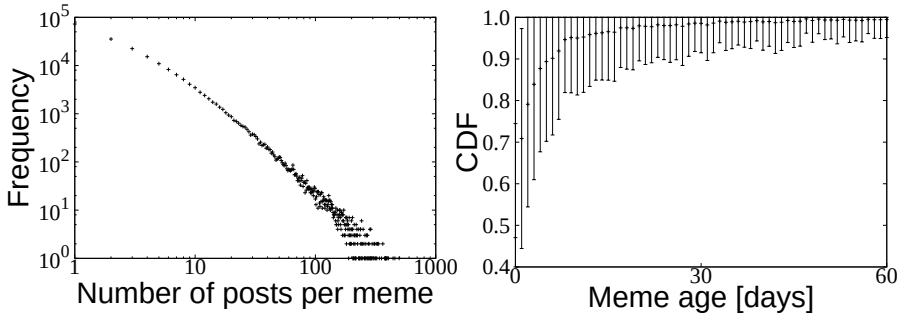

Fig. 4. Left: Repost distribution. Right: CDF of the fraction of reposts as a function of time.

Table 2. 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

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.

Temporal dependency. We observe that most reposts occur shortly after a meme is posted for the first time. Figure 4(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 5(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 re-post 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 5(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 Sec-

Author-produced version of the article published in Journal of Intelligent Information Systems,40(2), 211-239.
The original publication is available at http://www.springerlink.com/content/0925-9902/?k=Ienco
DOI:10.1007/s10844-011-0181-4

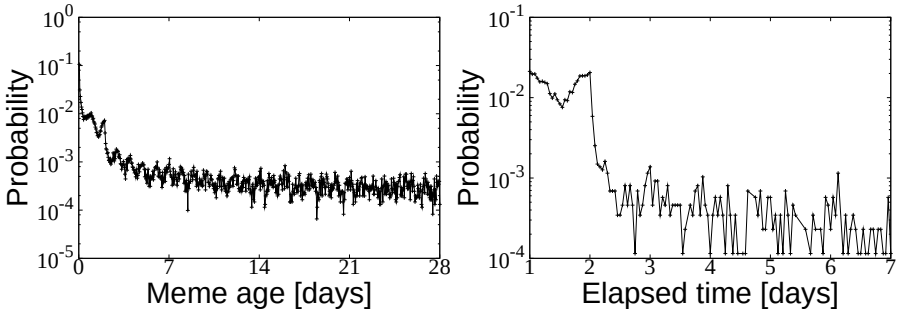

Fig. 5. Dependency of post probability with the age of the meme, and with the time passed since the previous post.

tion 6.1. 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 6 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.

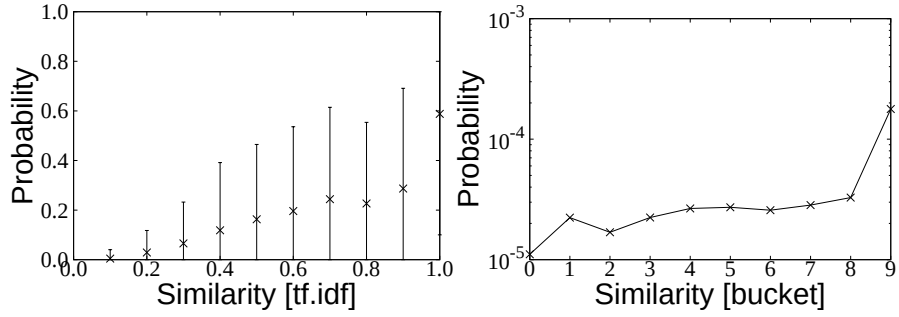

Fig. 6. Probability of a repost vs user-user similarity (left) and vs user-post similarity (right).

Other factors. We also tested how much the popularity (number of followers in the social graph) of the user starting a meme, affects the extent of the propagation of the meme: the two factors seem not to be correlated. Similarly, the branching factor at the root of a propagation tree (number of re-posts by

Author-produced version of the article published in Journal of Intelligent Information Systems,40(2), 211-239.
The original publication is available at http://www.springerlink.com/content/0925-9902/?k=Ienco
DOI:10.1007/s10844-011-0181-4

followers of the original poster) does not seem to be correlated with the size of the propagation tree.

3.3 Patterns of propagation

As we stated earlier, each meme propagation is a tree where the root is the user that started the meme and all the internal nodes are the users that reposted the meme. As we have a dataset of many propagations (a bag of trees) it is natural to analyze which common subtrees structure we can find frequently in our Yahoo! Meme propagations. Therefore we apply a frequent pattern mining approach [37] to our dataset, where the pattern extracted are subtrees.

The study of frequent patterns might uncover sub-structures in the network interaction. In [26] the authors also adopted frequent patterns to understand influence dynamic in a recommendation network. In their work patterns are directed subgraphs.

In our analysis we focus on patterns which are subtrees whose nodes are labelled w.r.t. two different aspects. The first one is related to the relative elapsed time of repost. This means that we label each node using the time it took to repost. We discretize the time in six bins as reported in Table 3. We name the resulting dataset *Relative Time*.

Table 3. Discretized time as node labels in *Relative Time* dataset.

class	elapsed time
0	the root node
H	no more than 1 hour
D	more than 1 hour but no more than 1 day
W	more than 1 day but no more than 1 week
M	more than 1 week but no more than 1 month
Y	more than 1 month

The second aspect that we want to analyze is the impact of the popularity of users, that is their number of followers, or in other terms, their out-degree in the social graph. Therefore we label each node with its number of followers. Also in this case we discretize in three bins: 0 represents nodes with less than 50 followers, 1 represents nodes with at least 50 but less than 100 followers and 2 represents users with at least 100 followers. We name this dataset *Followers*.

We run frequent subtrees mining algorithm [37] on these two datasets using a minimum support threshold of 70% for the *Relative time* dataset and 40% for the *Follower* dataset. In both datasets we use a very high threshold because we are interested in the very frequent propagation patterns. For the first dataset we use a threshold of 70% because with a lower threshold the algorithm crashes due to memory requirements.

Using these parameters we extract 555 frequent patterns from the *Relative Time* dataset and 56 frequent patterns from the *Follower* dataset. Example of obtained patterns are reported in Figure 7 and Figure 8 respectively.

Author-produced version of the article published in Journal of Intelligent Information Systems,40(2), 211-239.
The original publication is available at http://www.springerlink.com/content/0925-9902/?k=Ienco
DOI:10.1007/s10844-011-0181-4

The patterns reported in Figure 7 are a selection of representative frequent patterns among the 555 found in the *Relative Time* dataset. Each pattern is reported with its absolute support, that is the number of propagations that contain the pattern, among all the 1,100 propagations in our dataset. It is worth noting that many patterns can be contained in the same propagation: as an extreme example, a single propagation might actually contain all the patterns in Figure 7.

A first worth observation is that all the reported patterns have very high absolute support, all appearing in at least 800 of the 1,100 propagations. The second observation is that at the root node we can observe a mixture of quick reposts (the most) and few late reposts. In particular quick, early reposts create patterns of propagation both in depth and in width, thus confirming that they are the driving force behind the successful viral spread of information. This is confirmed also by our previous findings that the recency of a meme increases the propagation probability.

The patterns extracted from the *Follower* dataset (Figure 8) are mostly done of nodes of degree class 0. This is a consequence of the power law of the out-degree of the social network, that makes the vast majority of the nodes belong to class 0. It is interesting to observe that the popularity of users does not necessarily dictate the direction of the propagation of a meme. In Figure 8 indeed we can observe both a pattern of propagation from a class-7-degree node to a class-0-degree node, and a pattern of propagation in the reverse direction, i.e., a pattern of propagation from a class-0-degree node to a class-5-degree node.

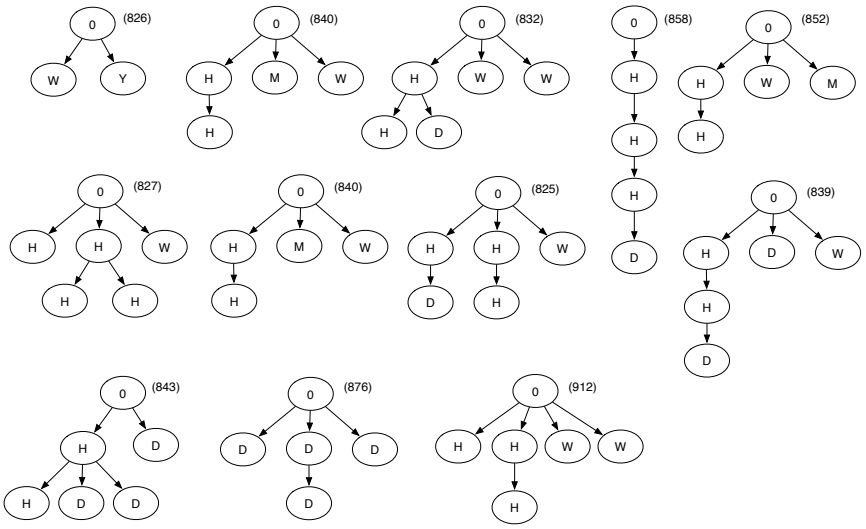

Fig. 7. Some representative patterns extracted from the *Relative Time* dataset, with their absolute support. Letters in nodes indicate elapsed time discretized as less than one Hour, Day, Week, Month, or Year.

Author-produced version of the article published in Journal of Intelligent Information Systems,40(2), 211-239.
The original publication is available at http://www.springerlink.com/content/0925-9902/?k=Ienco
DOI:10.1007/s10844-011-0181-4

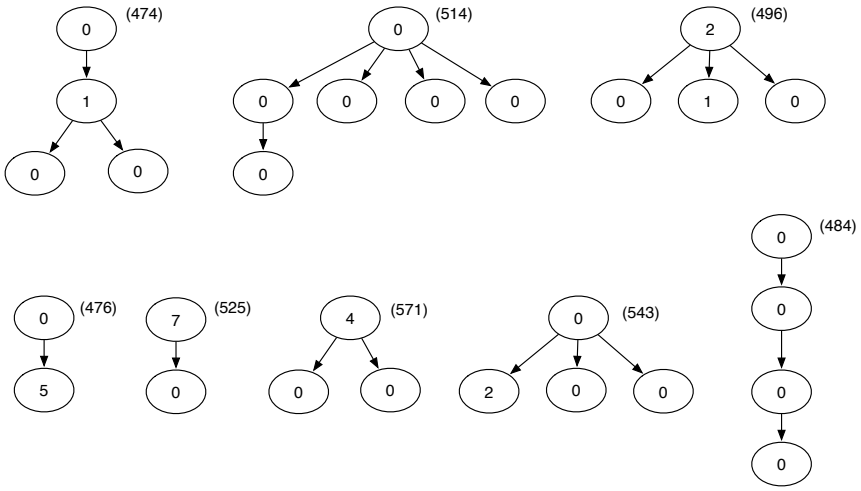

Fig. 8. Some representative patterns extracted from the *Follower* dataset, with their absolute support. Numbers in the nodes indicate the number of followers (discretized in ad-hoc bins).

3.4 Summary of findings

The findings of our preliminary inspections of Yahoo! Meme data, which drive the propagation modeling and simulation we discuss in this paper, are the following:

- 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.

Based on these finding in the next section we formally define a meme propagation model, which is a variant of the *Independent Cascade* model [21].

4 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.

Author-produced version of the article published in Journal of Intelligent Information Systems,40(2), 211-239.
The original publication is available at http://www.springerlink.com/content/0925-9902/?k=Ienco
DOI:10.1007/s10844-011-0181-4

Table 4. Notation used in the following sections.

$G = (V, E)$	the followed-follower social graph
$post(u, m, t)$	user u posted meme m at time t
$repost(v, u, m, t)$	user v reposted meme m at time t via u
$in(v, t)$	set of incoming memes for user v at time t
$cand(v, t) \subseteq in(v, t)$	the set of candidate memes for user v at time t
$\phi(v, t) \subseteq cand(v, t)$	the meme rank for user v at time t, i.e., the selected k memes
$\sigma(m, v)$	the expected spread of meme m from user v
$interest(m, v)$	a similarity value between a meme m and a user v
$influence(u, v)$	extent of influence exerted by user u on user v
Ω	super-user
num_clock	number of timestamps of the simulation process
$SUPER\_USER\_LIVE$	the number of timestamps Ω can propose new memes

4.1 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 $(u, v) \in E$ represents 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.,

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

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

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

Author-produced version of the article published in Journal of Intelligent Information Systems,40(2), 211-239.
The original publication is available at http://www.springerlink.com/content/0925-9902/?k=Ienco
DOI:10.1007/s10844-011-0181-4

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.

4.2 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 $k \in \mathbb{N}$, and assuming the meme propagation model described in Section 4.1, the Meme Ranking Problem requires to define a function $\phi : V \times T \rightarrow 2^M$ 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. As the propagation model is probabilistic, the function ϕ must maximize the expected number of reposts:

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

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. Consider a node w which is a follower of v. The meme ranking for v must take into account the likelihood that a meme m will be reposted by w if posted by v. But this likelihood strongly depends on the probability that m will be selected by the meme ranking for w once that v posted w. This probability in turn depends on what is posted by all the users that w follows, but also on the likelihood of all the other users following w of being interested in a given post, and *recursively on all the network*.

Author-produced version of the article published in Journal of Intelligent Information Systems,40(2), 211-239.
The original publication is available at http://www.springerlink.com/content/0925-9902/?k=Ienco
DOI:10.1007/s10844-011-0181-4

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 $m \in cand(w, t)$.

Definition 1 (Expected spread of a meme).
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). [5]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 derives from 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 $2^{|E|}$.

[5] This problem is in a sense the converse of the *Influence Maximization* problem, defined in [21] 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 are given a single user and we want to select k memes to propagate.

Author-produced version of the article published in Journal of Intelligent Information Systems,40(2), 211-239.
The original publication is available at http://www.springerlink.com/content/0925-9902/?k=Ienco
DOI:10.1007/s10844-011-0181-4

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 #**P**-complete problem [35]. It is very unlikely that there exists a polynomial time exact algorithm for a #**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. [28], 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, and compute the spread in these deterministic graphs. The average spread is an unbiased estimator of the expected spread of m from v.

In order to establish a bound on the number of sampled graphs (or possible worlds) needed to provide a good approximation of the expected spread $\sigma(m,v)$ we exploit Hoeffding Inequality [18].

Theorem 1 (Hoeffding Inequality).
Let $X_1, X_2, \ldots, X_r$ be independent and identically distributed random variables. Assume that X_i are almost surely bounded, that is $\forall i, Pr(X_i \in [a_i, b_i]) = 1$. Then for the sum of the variables $S = X_1 + ... + X_r$ we have

$$Pr(|S - E[S]| \geq \epsilon) \leq 2\exp(-\frac{2\epsilon^2}{\sum_{i=1}^{r}(b_i - a_i)^2}).$$

□

The next lemma, proving the approximation of the sampling, is a direct application of Hoeffding Inequality.

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*[6]*. 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 $E[\sigma_{X_i}(m,v)] = \sigma(m,v)$.*

[6] 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.

Author-produced version of the article published in Journal of Intelligent Information Systems,40(2), 211-239.
The original publication is available at http://www.springerlink.com/content/0925-9902/?k=Ienco
DOI:10.1007/s10844-011-0181-4

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* $\sigma(m,v)$*.*

4.3 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* [21], 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.

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 a 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. However, efficiently computing or approximating the top-k memes w.r.t. their expected spread, is an interesting open problem deserving further investigation.

5 Heuristic Methods

In this section we describe several heuristics for solving the meme ranking problem. In devising these heuristics, the main design goals are simplicity and computational feasibility, as the meme ranking needs to be executed for all the users and at each timestamp while they are logged into the microblogging system.

Let v be the user for which we are computing the meme rank. We introduce four 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; and methods combining

Author-produced version of the article published in Journal of Intelligent Information Systems,40(2), 211-239.
The original publication is available at http://www.springerlink.com/content/0925-9902/?k=Ienco
DOI:10.1007/s10844-011-0181-4

the best method from each of the previous three groups. These heuristics will be then evaluated in Section 7.

5.1 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. This method is justified by the temporal analysis we presented in Section 3.2, but more importantly, by the fact that it is the *de facto* standard in most microblogging platforms including Yahoo! Meme.

5.2 User-centered methods

The methods in the second group only focus on the user v to define the meme ranking function $\phi(v, t)$, without considering her followers.

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 4. 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 6.1.

In our simulation framework all the values of $interest(m, v)$, $influence(u, v)$ and $p(repost(v, u, m, t))$ are precomputed, for this reason the complexity of these approaches is the ranking step base on the different measure involved. Also in this case, given the number of average memes per user equals to q, the complexity is $O(q \log q)$.

5.3 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: note that in this case $influence(v, w)$ is not appropriate as it is independent from the meme.

Author-produced version of the article published in Journal of Intelligent Information Systems,40(2), 211-239.
The original publication is available at http://www.springerlink.com/content/0925-9902/?k=Ienco
DOI:10.1007/s10844-011-0181-4

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.* [6]: 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, lightweight heuristics.

To obtain the complexity for this group of heuristics we consider that the cost of both aggregation functions is linear in the number of followers n and the values of the functions $interest(m, v)$ and $p(repost(v, u, m, t))$ are precomputed. For each meme we perform n operations to obtain the aggregate value. If we always assume q as the average number of memes per user, we perform $O(qn)$ operations. When we have computed all the aggregation values for each meme, we need to sort them. This operation costs $O(q \log q)$ operations. For this reason the complexity of this group of heuristics is equal to $O(qn + q \log q)$.

6 Simulation framework

In this section we describe the simulation environment that we developed to compare different meme ranking strategies. One of the main features of this simulation framework is that it is based on real-world information, and the parameters are learnt from actual logs. This feature distinguishes our empirical evaluation from most previous works on influence propagation, where the parameters (e.g., the influence probabilities) of the propagation simulations are usually sampled at random from a uniform distribution.

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 4.1. In our experiments, all these pieces of input come from the Yahoo! Meme dataset described in Section 3.

In order to map from the real data to the propagation model (and thus to the simulator), we have to decide how to discretize time. 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

Author-produced version of the article published in Journal of Intelligent Information Systems,40(2), 211-239.
The original publication is available at http://www.springerlink.com/content/0925-9902/?k=Ienco
DOI:10.1007/s10844-011-0181-4

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.

Algorithm 1 $FeedRankingSim(k, num\_clock, Users, SUPER\_USER\_LIFE)$

```
1: repost_counter = 0
2: t = 1
3: while t ≤ num_clock do
4:    if t ≤ SUPER_USER_LIVE then
5:       for all v ∈ Users do
6:          for all m ∈ Ω do
7:             r = random Number (flips a coin)
8:             if prob(v, Ω, m, t) > r then
9:                addPost(u, post, clock)
10:   for all v ∈ Users do
11:      recSet = φ(v, t)
12:      for all m ∈ recSet do
13:         r = random Number (flips a coin)
14:         selected = false
15:         if (prob(v, u, m, t) > r) then
16:            repost_counter++
17:            selected = true
18:            addPost(u, m, t)
19:   t++
20: RETURN repost_counter
```

The general loop of the simulator is presented in algorithm 1. The procedure takes as parameters the number of top memes selected from the heuristic (k),the number of timestamps of simulation to perform (num_clock), the users set ($Users$), and the duration (in timestamps) of the period during which Ω can propose new memes ($SUPER\_USER\_LIFE$). For each timestamp t the simulator first shows to all the users the set of new memes currently posted by the super-user Ω, then for each user v 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 the associated repost probability (as described in Section 4.1).

6.1 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. We next describe how we learn these probabilities from a log of past meme propagations.

Interest We first discuss how we compute $interest(m, v)$ for each meme m and each user v. We represent m as a bag-of-words – considering all the words in its

Author-produced version of the article published in Journal of Intelligent Information Systems,40(2), 211-239.
The original publication is available at http://www.springerlink.com/content/0925-9902/?k=Ienco
DOI:10.1007/s10844-011-0181-4

text, or, in the case of images or multimedia files, all the tokens in the URL of the meme. We represent a user v as a concatenation of the bags-of-words of all the memes she has ever posted.

Next we compute the similarity between a user and a meme, or between two memes, using cosine similarity of their bags-of-words [4].

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.

Influence Following [15] 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 occurs, we consider this a successful case of influence. Each attempt, that is each meme posted by u, can be viewed as a Bernoulli trial.

The Maximum Likelihood Estimator (MLE) of success probability is the ratio of number of successful attempts over the total number of trials. Hence, influence probability of v on u is estimated as:

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

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 5(right) and following [15], 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:

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

The distributions of the learnt parameters $p^m_{u,v}$ and τ_v are reported in Figure 9. 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

Author-produced version of the article published in Journal of Intelligent Information Systems,40(2), 211-239.
The original publication is available at http://www.springerlink.com/content/0925-9902/?k=Ienco
DOI:10.1007/s10844-011-0181-4

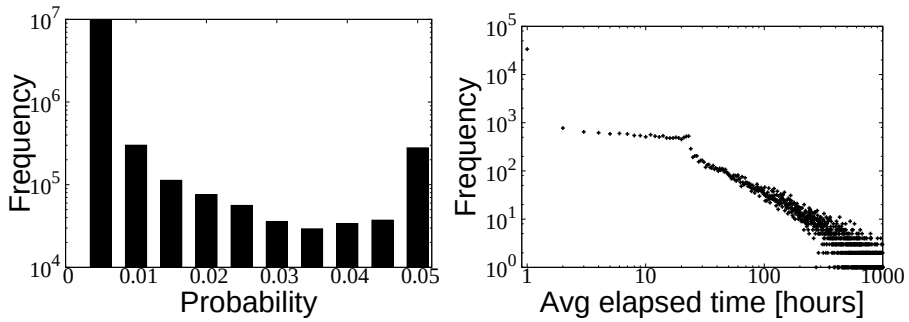

Fig. 9. Left: distribution of $p^m_{u,v}$. Right: distribution of τ_v.

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.

7 Meme ranking experiments

This section presents our experimental results. They were obtained by running simulations of the behavior of the network during one simulated month (720 hours). We "seed" 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. As described in Section 3 the social network contains 57K users. At each timestamp, the meme rank presents up to $k = 5$ posts to each user.

We tested all the strategies described above, and executed 50 independent runs of the simulator for each strategy. It should be noted that at the present time we can not compare with any previously proposed method, as we are the first to introduce and study the meme ranking problem. Moreover comparison with simulation-based estimation of memes spread is also unfeasible as we reported in Section 4.3.

The experimental results in terms of total network activity (number of reposts), averaged across all the runs of each experiment, is shown in Table 5.

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).

Author-produced version of the article published in Journal of Intelligent Information Systems,40(2), 211-239.
The original publication is available at http://www.springerlink.com/content/0925-9902/?k=Ienco
DOI:10.1007/s10844-011-0181-4

Table 5. 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

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 5 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*.[7]

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.

Figure 10 provides more details about how network activity evolves over time under the different heuristics. Overall, these experiments demonstrate that the particular set of memes selected and shown to users every time they log in, has a strong influence in the level of activity of the entire system.

Efficiency. Our straightforward implementation is quite efficient. We ran it on a quad-processor Intel Xeon 3GHz with 16GB of RAM. The most expensive

[7] `http://en.wikipedia.org/wiki/Borda_count`

Author-produced version of the article published in Journal of Intelligent Information Systems,40(2), 211-239.
The original publication is available at http://www.springerlink.com/content/0925-9902/?k=Ienco
DOI:10.1007/s10844-011-0181-4

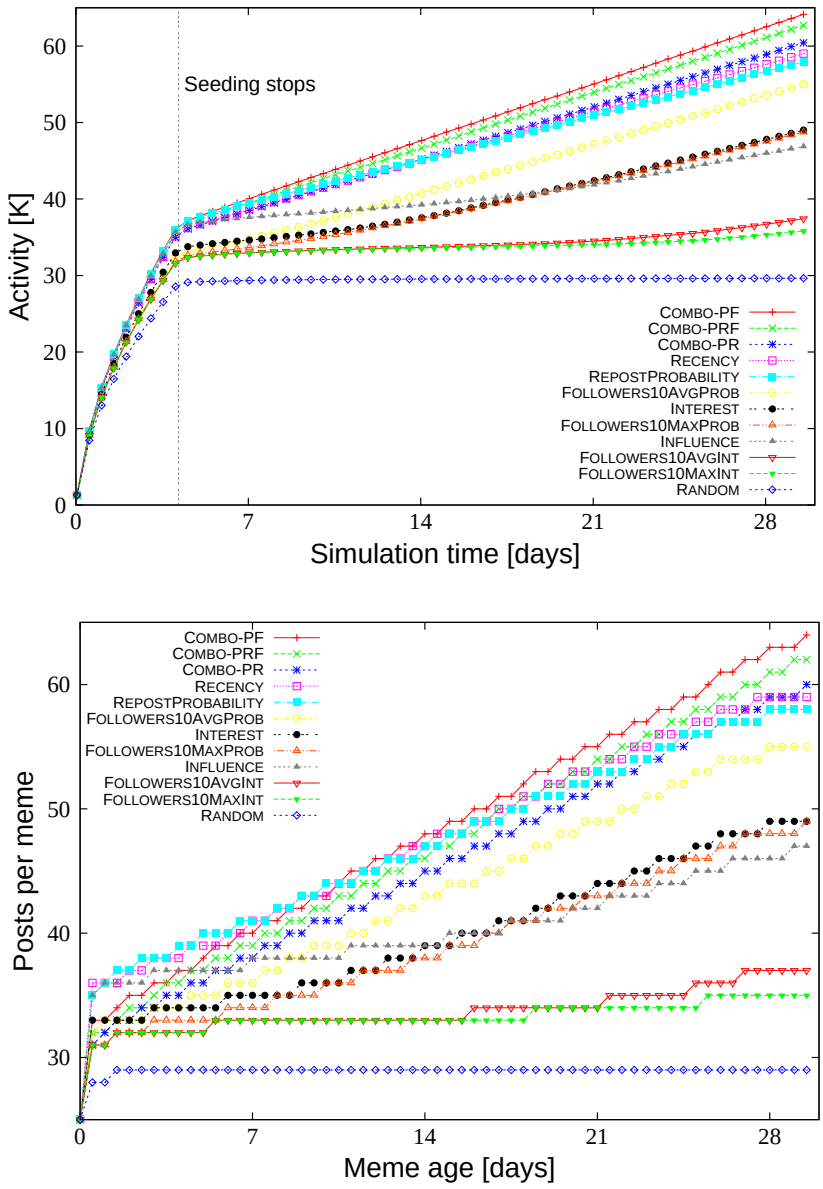

Fig. 10. Results of the meme ranking simulation. Top: network activity over time. Bottom: average reposts per meme vs. meme age.

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 5.

Author-produced version of the article published in Journal of Intelligent Information Systems,40(2), 211-239.
The original publication is available at http://www.springerlink.com/content/0925-9902/?k=Ienco
DOI:10.1007/s10844-011-0181-4

The running times are also indicative of the time computational requirements of the different 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.

Finally, since this method is quite efficient, a natural question to ask is to which extent we can improve the performance by paying a little bit larger price in terms of computation time. This is what we investigate next.

Varying sample size. We focus on the followers-based heuristics assessing how the size of the sample of followers selected to perform the computation affects the meme rank performance and its run time. Results for FOLLOWERS10AVGPROB are reported in Figure 11. As expected, sampling more neighbors makes the strategy perform better at the expense of a longer running time. The number of neighbors to sample is a parameter to be fine-tuned according to the desired trade-off and the target response time of the system to which the meme ranking solution is applied.

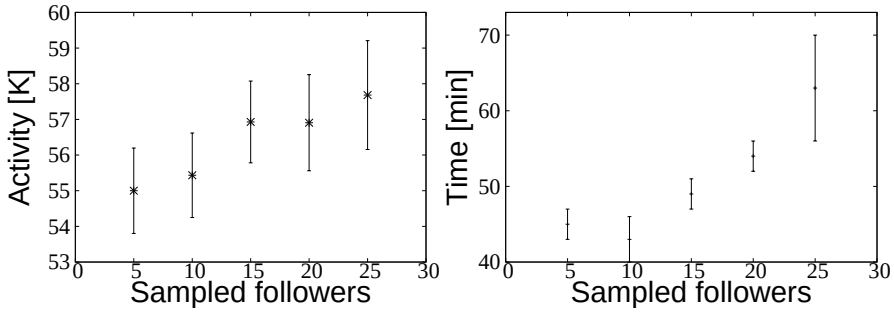

Fig. 11. Results of followers-based heuristics when varying the number of followers sampled. Left: in terms of network activity. Right: in terms of wallclock time of the simulation.

8 Conclusions

In this paper we introduced 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 proposed a meme propagation model, which is valid in general, beyond the specific objective function that we tackled.

Author-produced version of the article published in Journal of Intelligent Information Systems,40(2), 211-239.
The original publication is available at http://www.springerlink.com/content/0925-9902/?k=Ienco
DOI:10.1007/s10844-011-0181-4

The proposed model is 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[8] than other platforms that have a more conversational design. However, we are already working to reproduce our analysis in other platforms, such as, e.g., Twitter.

In the theoretical part of our investigation, we deeply characterized the computational hardness of the meme ranking problem and we showed that, even for a simpler formulation, we need to compute the expected spread of a meme from a user, that we proved being #**P**-complete. We discussed theoretical bounds for sampling, thus showing that even approximate solutions for this problem are prohibitively costly for our on-line recommendation context. Therefore, we turned our attention to efficient yet effective heuristic methods. In particular, we devised 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 compared our methods to two baselines, namely random selection, and a recency-based heuristic. The latter is the *de facto* standard for most microblogging platforms including Yahoo! Meme. The results of our experiments confirmed 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.

The main limitation of our work is the simulation-based evaluation, which is by its nature a simplification of the real process. For instance, we assume many things are constant during the simulation such as the users' connections, the rate of arrival of external posts, etc. The only possible way to overcome these limitations is to conduct a real-world human-based assessment.

As online social networks continue becoming not only larger, but also more densely connected [24], 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.

Reproducibility: our simulator and instructions to access publicly-available meme data through YQL, are available at `http://barcelona.research.yahoo.net/memerank`

Acknowledgments: the authors wish to acknowledge the Yahoo! Meme team for their help, Ulf Brefeld and Aris Gionis for their suggestions. This research is partially supported by the Spanish Centre for the Development of Industrial Technology under the CENIT program, project CEN-20101037, "Social Media" (www.cenitsocialmedia.es).

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

Author-produced version of the article published in Journal of Intelligent Information Systems,40(2), 211-239.
The original publication is available at http://www.springerlink.com/content/0925-9902/?k=Ienco
DOI:10.1007/s10844-011-0181-4

References

1. E. Adar and L. A. Adamic. Tracking information epidemics in blogspace. In *Web Intelligence 2005*.
2. N. Agarwal, H. Liu, L. Tang, and P. S. Yu. Identifying the influential bloggers in a community. In *Proceedings of the First International Conference on Web Search and Web Data Mining, (WSDM'08)*, 2008.
3. L. Backstrom, D. P. Huttenlocher, J. M. Kleinberg, and X. Lan. Group formation in large social networks: membership, growth, and evolution. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD'06)*, 2006.
4. R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. Addison Wesley, May 1999.
5. E. Bakshy, B. Karrer, and L. A. Adamic. Social influence and the diffusion of user-created content. In *EC '09: Proceedings of the tenth ACM conference on Electronic commerce*, pages 325–334, New York, NY, USA, 2009. ACM.
6. M. Cha, H. Haddadi, F. Benevenuto, and K. P. Gummadi. Measuring User Influence in Twitter: The Million Follower Fallacy. In *In Proceedings of the 4th International AAAI Conference on Weblogs and Social Media (ICWSM)*.
7. M. Cha, A. Mislove, and P. K. Gummadi. A measurement-driven analysis of information propagation in the flickr social network. In *Proceedings of the 18th International Conference on World Wide Web (WWW'09)*, 2009.
8. W. Chen, C. Wang, and Y. Wang. Scalable influence maximization for prevalent viral marketing in large-scale social networks. In *Proc. of the 16th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD'10)*.
9. W. Chen, Y. Wang, and S. Yang. Efficient influence maximization in social networks. In *Proc. of the 15th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD'09)*, 2009.
10. W. Chen, Y. Yuan, and L. Zhang. Scalable influence maximization in social networks under the linear threshold model. In *Proceedings of the 10th IEEE International Conference on Data Mining (ICDM'2010)*.
11. D. J. Crandall, D. Cosley, D. P. Huttenlocher, J. M. Kleinberg, and S. Suri. Feedback effects between similarity and social influence in online communities. In *Proc. of the 14th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD'08)*, 2008.
12. P. Domingos and M. Richardson. Mining the network value of customers. In *Proc. of the Seventh ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD'01)*.
13. N. E. Friedkin. *A Structural Theory of Social Influence*. Cambridge University Press, 1998.
14. J. Golbeck and J. Hendler. Inferring binary trust relationships in web-based social networks. *ACM Trans. Internet Technol.*, 6(4):497–529, 2006.
15. A. Goyal, F. Bonchi, and L. Lakshmanan. Learning influence probabilities in social networks. In *Proceedings of the Third ACM International Conference on Web Search and Data Mining (WSDM 2010)*.
16. D. Gruhl, R. V. Guha, D. Liben-Nowell, and A. Tomkins. Information diffusion through blogspace. In *Proceedings of the 13th international conference on World Wide Web, (WWW'04)*, 2004.
17. R. Guha, R. Kumar, P. Raghavan, and A. Tomkins. Propagation of trust and distrust. In *Proceedings of the 13th international conference on World Wide Web (WWW'04)*, 2004.

18. W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, March 1963.
19. A. Java, X. Song, T. Finin, and B. Tseng. Why we twitter: understanding microblogging usage and communities. In *Proceedings of the 9th WebKDD and 1st SNA-KDD 2007 workshop on Web mining and social network analysis, 2007*.
20. J. J. Jianshu Weng, Ee-Peng Lim and Q. He. Twitterrank: Finding topic-sensitive influential twitterers. In *WSDM*, page 10, 2010.
21. D. Kempe, J. M. Kleinberg, and É. Tardos. Maximizing the spread of influence through a social network. In *Proc. of the Ninth ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD'03)*.
22. K. Lerman and L. Jones. Social browsing on flickr. *CoRR*, abs/cs/0612047, 2006.
23. J. Leskovec, L. Backstrom, and J. Kleinberg. Meme-tracking and the dynamics of the news cycle. In *Proc. of the 15th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD'09)*.
24. J. Leskovec, J. Kleinberg, and C. Faloutsos. Graphs over time: densification laws, shrinking diameters and possible explanations. New York, NY, USA.
25. J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. VanBriesen, and N. S. Glance. Cost-effective outbreak detection in networks. In *Proc. of the 13th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD'07)*, 2007.
26. J. Leskovec, A. Singh, and J. M. Kleinberg. Patterns of influence in a recommendation network. In *Proceedings of the 10th Pacific-Asia Conference on Knowledge Discovery and Data Mining,(PAKDD'06)*, 2006.
27. A. Mislove, M. Marcon, K. P. Gummadi, P. Druschel, and B. Bhattacharjee. Measurement and analysis of online social networks. In *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement (IMC'07)*.
28. M. Mitzenmacher and E. Upfal. *Probability and Computing : Randomized Algorithms and Probabilistic Analysis*. Cambridge University Press, 2005.
29. M. Richardson and P. Domingos. Mining knowledge-sharing sites for viral marketing. In *Proc. of the Eighth ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD'02)*.
30. J. J. Samper, P. A. Castillo, L. Araujo, and J. J. M. Guervós. Nectarss, an rss feed ranking system that implicitly learns user preferences. *CoRR*, abs/cs/0610019, 2006.
31. X. Song, Y. Chi, K. Hino, and B. L. Tseng. Information flow modeling based on diffusion rate for prediction and ranking. In *Proceedings of the 16th international conference on World Wide Web (WWW'07)*, 2007.
32. X. Song, B. L. Tseng, C.-Y. Lin, and M.-T. Sun. Personalized recommendation driven by information flow. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval (SIGIR'06)*, 2006.
33. M. Taherian, M. Amini, and R. Jalili. Trust inference in web-based social networks using resistive networks. In *Proceedings of the 2008 Third International Conference on Internet and Web Applications and Services (ICIW'08)*, 2008.
34. J. Tang, J. Sun, C. Wang, and Z. Yang. Social influence analysis in large-scale networks. In *Proc. of the 15th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD'09)*, 2009.
35. L. G. Valiant. The complexity of enumeration and reliability problems. *SIAM Journal on Computing*, 8(3):410–421, 1979.
36. 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.

37. M. J. Zaki. Efficiently mining frequent trees in a forest: Algorithms and applications. *IEEE Transactions on Knowledge and Data Engineering*, 17(8):1021–1035, Aug 2005. special issue on Mining Biological Data.
38. C.-N. Ziegler and G. Lausen. Propagation models for trust and distrust in social networks. *Information Systems Frontiers*, 7(4-5):337–358, 2005.