Revisit Behavior in Social Media: The Phoenix-R Model and Discoveries

Flavio Figueiredo[1], Jussara Almeida[1], Yasuko Matsubara[2], Bruno Ribeiro[2], and Christos Faloutsos[2]

[1] Universidade Federal de Minas Gerais
[2] Carnegie Mellon University

Abstract. How many listens will an artist receive on a online radio? How about plays on a YouTube video? How many of these visits are new or returning users? Modeling and mining popularity dynamics of social activity has important implications for researchers, content creators and providers. We here investigate and model the effect of revisits on popularity. A revisit can be defined as a repeated play (song or video) or a re-tweet of the same hashtag over time. Using four datasets of social activity, with up to tens of millions media objects (e.g., YouTube videos, Twitter hashtags or LastFM artists), we show the effect of revisits in the popularity evolution of such objects. Secondly, we propose the PHOENIX-R model which captures the popularity dynamics of individual objects. PHOENIX-R has the desired properties of being: (1) parsimonious, being based on the minimum description length principle, and achieving lower root mean squared error than state-of-the-art baselines; (2) applicable, the model is effective for predicting future popularity values of time series.

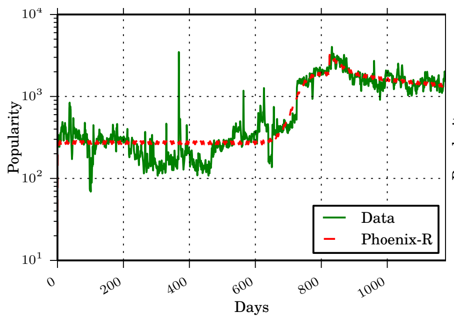
1 Introduction

How do we quantify the popularity of a piece of content in social media applications? Should we consider only the audience (unique visitors) or include revisits as well? Can the revisit activity be explored to create more realistic popularity evolution models? These are important questions in the study of social media popularity. In this paper, we take the first step towards answering them based on four large traces of user activity collected from different social media applications: Twitter[3], LastFM[4], and YouTube[5].

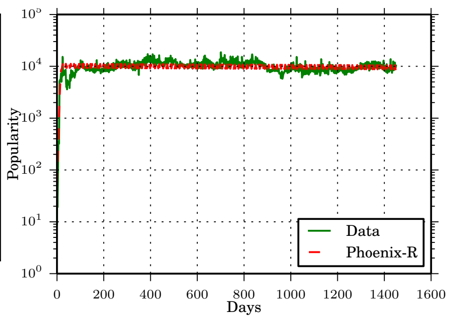
Understanding the popularity dynamics of online content is both a challenging task, due to the vast amount and variability of content available, and it can also provide invaluable insights into the behaviors of human consumption [6] and into more effective engineering strategies for online services. A large body of previous work has investigated the popularity dynamics of social media content, focusing mostly on modeling and predicting the *total number of accesses* a piece of content receives [5, 6, 9, 17, 21].

However, a key aspect that has not been explored by most previous work is the effect of revisits on content. The distinction between audience (unique users), revisits

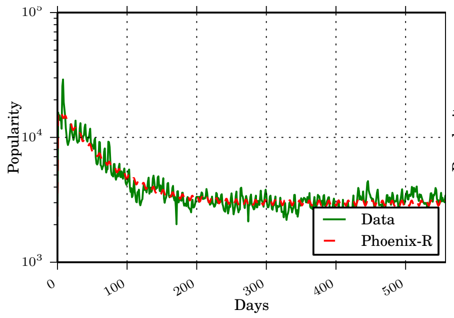
[3] `http://twitter.com`
[4] `http://lastfm.com`
[5] `http://youtube.com`



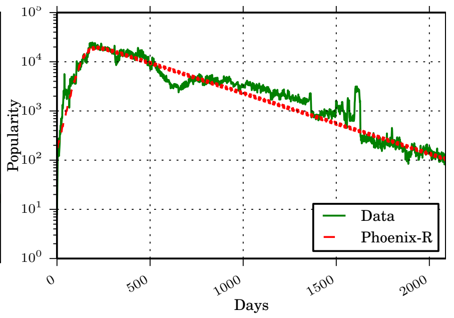
(a) Rock Song (multiple cascades)



(b) Flashdance (80's movie) clip (revisits)



(c) Korean Music Video (single cascade)



(d) User Dancing Video (single cascade)

Fig. 1: Different YouTube videos as captured by the PHOENIX-R model.

(returning users), and popularity (the sum of the previous two) can have large implications for different stakeholders of these applications - from content providers to content producers - as well as for internal and external services that rely on social activity data. For example, marketing services should care most about the audience of a particular content, as opposed to its total popularity, as each access does not necessarily represent a new exposed individual. Even system level services, such as geographical sharding [8, 23], can be affected by such distinction, as a smaller audience served by one data center does not necessarily imply that a smaller volume of activity (and thus lower load) should be expected. As prior studies of content popularity in social media do not clearly distinguish between unique and returning visits, the literature still lacks fundamental knowledge about content popularity dynamics in this environment.

Goals: In this context, we aim at investigating and modeling the effect of revisits on content, thus complementing prior efforts by providing a new understanding to the field of social media popularity dynamics. Our main goals are as follows: (1) Characterizing the revisits phenomenon and show how it affects the evolution of popularity of different media objects on social media applications; (2) Introducing the PHOENIX-R model that captures the evolution of popularity of individual objects. Unlike state-of-the-art alternatives [20,21], the model explicit accounts for revisits. Also, we develop the model so that it can account for multiple cascades, or outbreaks, of interest in a given object.

Characterization Results: We show that when analyzing final popularity values (up to the time data was crawled), revisits account from 40% up to 96% of the total popularity of an object (on median), depending on the type of content. We also show even when looking at small time windows, say hourly time windows, revisits can be up to 14x more common than new users accessing the object.

PHOENIX-R Results: The PHOENIX-R model explicitly addresses revisits in social media behavior and is able to automatically identify multiple cascades [13] using only popularity time series data. This is in contrast to previous methods such as the SpikeM [18] approach, which models single cascades only, and the TemporalDynamics [21] models, are linear in nature. Figure 1 shows the different behaviors which can be captured by the Phoenix-R model. Notice how the model captures multiple cascades, causing a growth in the popularity of video (a), videos which have a plateau like popularity after the upload (b), and two different single cascade dynamics (c-d). The PHOENIX-R model is not only effective, it is also scalable. Fitting is done in linear time per time series, and no parameters are required.

Outline: Section 2 presents an overview of definitions and background. This is followed by Section 3 which presents our characterization. PHOENIX-R is described in Section 4, whereas it's applicability to different tasks is presented in Section 5. Related work is discussed on Section 6. Finally, we conclude the paper in Section 7.

2 Definitions and Background

We start this section by presenting some definitions used throughout this paper (Section 2.1). Next, we briefly discuss existing models that can be used to study the popularity dynamics of media content (Section 2.2).

2.1 Definitions

We define an **object** as a piece of media content stored on an application. Specifically, an object on YouTube is a video, whereas, on an online radio like LastFM, we consider (the webpage of) an artist as an object. We also define an object on Twitter as a hashtag or a *musictag*[6]. A **social activity** is the act of accessing - posting, re-posting, viewing or listening to - an object on a social media application. Although other social activities are possible on such applications, we focus on those that are more aligned with the main object type of each website. The **popularity** of an object is the aggregate behavior of social activities on that object. We here study popularity in terms of the most general activities in each application: number of views for YouTube videos, number of plays for LastFM artists, and number of tweets with a hashtag. The popularity of an object is composed of **audience** (or unique users) and, **revisits**, (or returning users), and the evolution of the popularity of an object over time defines a popularity **time series**.

2.2 Existing models of popularity dynamics

Epidemic models: Previous work on information propagation on online social networks (OSNs) has exploited epidemic models [12] to explain the dynamics of the propagation process. An epidemic model describes the transmission of a "disease" through

[6] Users informing their followers which artists they are listening to.

Table 1: Comparison of PHOENIX-R with other approaches

	Revisits	Non-Linear	Forecasting	Multi Cascade
SI [12]		✓		
SpikeM [18]		✓	✓	
TemporalDynamics [21]			✓	
PHOENIX-R	✓	✓	✓	✓

individuals. The simplest epidemic model is the Susceptible-Infected (SI) model. The SI model considers a fixed population divided into S susceptible individuals and I infected individuals. Starting with $S(0)$ susceptible individuals and $I(0)$ infected individuals, at each time step $\beta S(t-1)I(t-1)$ individuals get infected, and transition from the S state to the I state. The product $S(t-1)I(t-1)$ accounts for all the possible connections between individuals. The parameter β is the strength of the infectivity, or virus.

Cha *et. al* used an SI model to study how information (i.e., the "disease") disseminates through social links on Flickr [4], whereas Matsubara *et. al* [18] proposed an alternative model called SpikeM. SpikeM builds on an SI model by adding, among other things, a decaying power law infectivity per newly infected individual, which produces a behavior that is similar to the model proposed in [6]. The SpikeM model was used to captured the time series popularity for a single cascade. One of the reasons why the SI model is useful to represent online cascades of information propagation is that individuals usually do not delete their posts, tweets or favorite markings [4, 18]. Thus, once an individual is infected he/she remains infected forever (as captured by the SI model).

Temporal Dynamics Models: Other models that can be explored in the study of content popularity dynamics are auto-regressive models and state space models, such as the Holt-Winters model and its extensions [21]. However, these models are linear in nature, and thus cannot account for more complex temporal dynamics observed in online content [18]. Although, these models have been successful in predicting *normalized* query behavior in search engines [21], the descriptive power of such models is less attractive. For example, such models cannot distinguish between audience and revisits, as we do. Moreover, Holt-Winters based models are very general, that is, they are used to predict time series behavior, but will not take into account cascades or information dissemination. This implies that the interpretation of model parameters is difficult. Thus, from a descriptive point of view, these models are of little help to understand the actual process that drives popularity evolution.

Multiple Cascades: Very recently, the work of Hu *et. al* focused on the defining longevity of social impulses, or multiple cascades [13]. However, unlike our approach, the authors are not focused on modeling the long term popularity of objects.

Table 1 summarizes the key properties of the aforementioned models as well as of our new PHOENIX-R model. Note that no existing model explicitly captures the revisits by the same user of the same object. Yet, as we show in the next section, the analysis of both long and short term content popularity evolution in our datasets reveals that users often revisit the same object many times. In comparison these approaches, PHOENIX-R explicitly captures both revisits and multiple cascades, allows for non-linear solutions, and can be used for accurate forecasting.

3 Content Revisit Behavior in Social Media

We now analyze the revisit behavior in various social media applications. We first describe the datasets used in this analysis as well as in the evaluation of our model, and then discuss our main characterization findings.

3.1 Datasets

Our study is performed on four large social activity datasets:

- The Million Musical Tweets Dataset (MMTweet): consists of 1,086,808 tweets of users about artists they are listening to at the time [11] (musictags). We focus on the artist of each tag as an object, and a total of 25,060 objects (artists) appear in this dataset.
- The 2010 LastFM listening habits dataset (LastFM): consists of the whole listening habits (until May 5th 2009) of nearly 1,000 users, with over 19 million activities on 107,428 objects (artists) [3].
- The 476 million Twitter tweets corpus (Twitter): accounts for roughly 20% to 30% of the tweets from June 1 2009 to December 31 2009 [24], and includes over 50 million objects (hashtags) tweeted by 17 million users.
- The YouTube dataset: consists of the popularity time series of over 3 million YouTube videos. On the page of each video YouTube provides the full daily time series of popularity for each object.

3.2 Main findings

Our goal is to analyze how the popularity acquired by different objects, in the long and short runs, is divided into audience and revisits. In particular, we aim at assessing to which extent the number of revisits may be *larger* than the size of the audience, in which case popularity is largely a sum of repeated user activities. Since this property may vary depending on the type of content, we perform our characterization on the LastFM, MMTweet, and Twitter datasets. We leave the YouTube dataset out of this analysis since, unlike the other datasets, it does not contain individual social activities, but only popularity time series. We will make use of the YouTube dataset to fit and evaluate our PHOENIX-R model, in the next section.

We first analyze the distribution of the final values[7] of popularity, audience, and revisits across objects in each dataset. For illustration purposes, Figure 2 shows the complementary cumulative distribution function of the ratio of the number of revisits to the audience size for all datasets, computed for objects with popularity greater than 500. We filtered out very unpopular objects, which attract very little attention during the periods of our datasets (over 6 months each). Note that the probability of an object having more revisits than audience (ratio greater than 1) is large. Indeed, though rare, the ratio of revisits to audience size reaches 10^2 and even 10^3.

In order to better understand these findings across all datasets, Table 2 shows, for each dataset: (1) the median of the ratio of number of revisits to audience size, (2) the

[7] Values computed at the time the data was collected.

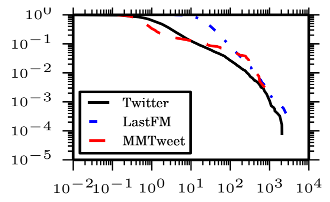


Fig. 2: Distributions of $\frac{\#Revisits}{Audience}$.

Table 2: Relationships between revisits, audience and popularity.

Dataset	Median $\frac{\#Revisits}{Audience}$	Median $\frac{\#Revisits}{Popularity}$	% objects with $\frac{\#Revisits}{Audience} > 1$
Twitter	1.70	0.62	66%
MMTweet	0.68	0.40	33%
LastFM	25.39	0.96	100%

Table 3: Quartiles of the ratio $\frac{\#Revisits}{Audience}$ for various time windows w.

Dataset	Time window (w)	25th percentile	Median	75th percentile
Twitter	1 hour	1.08	3.93	12
	1 day	1	2.5	6.28
	1 week	0.66	1.69	4.28
	1 month	0.56	1.44	3.75
MMTweet	1 hour	0.25	0.66	12.5
	1 day	0.55	0.83	1.26
	1 week	0.41	0.73	1.41
	1 month	0.31	0.56	1.17
LastFM	1 hour	20	21	25
	1 day	21	28	41
	1 week	20	30.5	55.25
	1 month	14	25	48

median of the ratio of number of revisits to total popularity; and (3) the percentage of objects where the revisits dominate the popularity (i.e., ratio of number of revisits to the audience size greater than 1). Note that revisits dominate popularity in 66% of the Twitter objects. Moreover, on median, 62% of the total popularity of these objects is composed of revisits, which account for 1.7 times more activities than the visits by new users (audience size). Again, for LastFM artists, revists are over 25 times more frequent than the visits by new users (on median), and the revisits dominate popularity in *all* objects. In contrast, the ratios of number of revisits to audience size and to total popularity are smaller for MMTweet objects, most likely because users do not tweet about artists they are listening to all the time, but rather only when they wish to share this activity with their followers. Yet, the revisits dominate popularity in 33% of the objects. These results provide evidence that, at least in the long run, revisits are much more common than new users for many objects in different applications. For microblogs, though less intense, this behavior is still non-negligible.

We further analyze the effect of revisits on popularity, focusing now in the short term, by zooming into smaller time windows w. Specifically, we analyze the distributions of the ratios of number of revisits to audience size computed for window sizes w equal to one hour, one day, one week, and one month. Table 3 shows the three distribution quartiles for the various window sizes and datasets considered. These quartiles were computed considering only window sizes during which the popularity acquired by the object exceeds 20. We adopted this threshold to avoid biases in time windows with very low popularity, focusing on the periods where the objects had a minimal attention (note that 20 is still small considering that each trace has millions of activities).

Focusing first on the LastFM dataset, we note that, regardless of the time window size, the number of revisits is at least one order of magnitude (14x) larger than the audience size for at least 75% of the analyzed windows (25^{th} percentile). In fact, the ratio between the two measures exceeds 55 for 25% of the windows (75^{th} percentile) on the weekly case. In contrast, in the MMTweet dataset, once again, the ratios are much smaller. Nevertheless, at least 25% of the of the windows we observe a burst of revisits in very short time, with the ratio exceeding 12 for the hourly cases. Once again, we suspect that these lower ratios may simply reflect that users do not tweet about every artist they list to. Thus, in general, we have strong evidence that, for music-related content, popularity is mostly governed by revisits, as opposed to new users (audience).

The same is observed, though with less intensity, in the Twitter dataset. Revisits are more common than new users in 50% of the time windows, for all sizes considered. Indeed, considering hourly time windows, popularity is dominated by revisits for 75% of the cases. While large ratios, such as those observed for LastFM, do not occur, the number of revisits can still be 12 times larger than the audience size during a single hour in 25% of the Twitter hourly windows.

Summary of findings: Our main conclusions so far are: (1) for most objects in the Twitter and LastFM datasets, popularity, measured both in the short (as short as 1 hour periods) and long runs, is mostly due to revisits than to audience size; and (2) revisits are less common on the MMTweet dataset, which we believe is due to data sparsity, but are still a significant component of the popularity acquired by a large fraction of the objects (in both long and short runs). These findings motivate the need for models that explicitly account for revisits in the popularity dynamics, which we discuss next.

4 The PHOENIX-R Model

In this section we introduce the proposed PHOENIX-R model(Section 4.1), show how we fit the model to a given popularity time series (Section 4.2). In the next section we present results on the efficacy of the model on our datasets when compared to state-of-the-art alternatives, and the applicability of the PHOENIX-R model.

Notation: We present vectors ($\mathbf{x}$) in bold. Sets are shown in non-bold calligraphy letters ($\mathcal{X}$), and variables are represented by lower case letters or Greek symbols (x, β). Moreover, $\mathbf{x}(i)$ means data index i (from 1), and $\mathbf{x}(: i)$ means sub-vector up to i.

4.1 Deriving the Model

We built the PHOENIX-R model is built based on the 'Susceptible-Infected-Recovery' (SIR) compartments, allowing for revisits and multiple cascades. Specifically, the PHOENIX-R model is built to capture the following behavior towards a given object:

- We assume a fixed population of individuals, where each individual can be in one of three states: susceptible, infected and recovered.
- At any given time s_i, an external shock i causes initial interest in the object. The shock can be any event that draws attention to the object, such as a video being uploaded to YouTube, a news event about a certain subject, or even a search engine indexing a certain subject for the same time (thus making an object easier to be found). We assume that the initial shock s_1 is always caused by one individual.

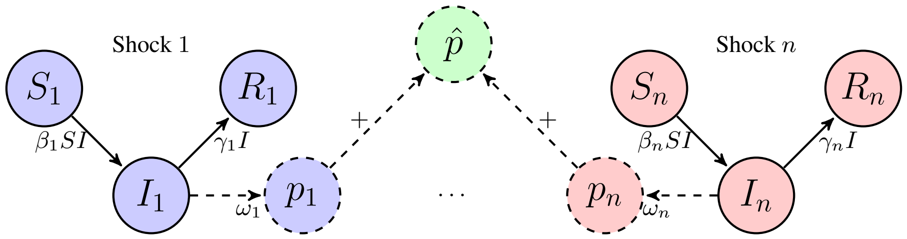


Fig. 3: Individual shocks that when added up account for the PHOENIX-R model.

– New individuals discover the object by being infected by the first one. Moreover, after discovery, these "newly infected" can also infect other individuals, thus contributing to the propagation.
– Infected individuals may access (watch, play or tweet) the object. It is important to note that being infected does not necessarily imply in an access. For example, people may talk about a trending video before actually watching it. Each infected individual accesses the object following a Poisson process with rate ω ($\omega > 0$)[8].
– After some time, individuals lose interest in the object, which, in the model, is captured by a recovery rate γ.
– Multiple external shocks may occur for a single object.

Figure 3 presents the PHOENIX-R model. In the figure, three compartments are shown for each shock S_i, I_i, and R_i, which represent the number of susceptible, infected and recovered individuals for shock i, respectively. Variable p_i, associated with shock i, measures the popularity acquired by the object due this shock. The total popularity of the object, i.e., the sum of the values of p_i for all shocks, is denoted by $\hat{p}$. We first present the model for a single shock, and then generalize the solution for multiple shocks. Also, we drop the subscripts while discussing a single shock. We present the model assuming discrete time, referring to each time tick as a time window.

Each shock begins with a given susceptible population ($S(0)$) and one infected individual ($I(0) = 1$). The total population is fixed and given by ($N = S(0) + 1$). The R compartment captures the individuals that lost interest in the object. Similarly the SI model, βSI susceptible individuals become infected in each time window. Moreover, γI individuals loose interest in (i.e., recover from) the object in each window. Revisits to the object are captured by the rate ω. Thus ω is the expected number of accesses of an individual up to time t, the probability of the individual accessing the object k times during a time interval of τ windows is given by $P(v(t+\tau) - v(t) = k) = \frac{(\omega\tau)^k e^{-\omega\tau}}{k!}$.

We assume that the shock starts at time zero, thus focusing the dynamics *after* the shock. Under this assumption, the equations that govern a single shock are:

$$S(t) = S(t-1) - \beta S(t-1) I(t-1) \tag{1}$$
$$I(t) = I(t-1) + \beta S(t-1) I(t-1) - \gamma I(t-1) \tag{2}$$
$$R(t) = R(t-1) + \gamma I(t-1) \tag{3}$$
$$p(t) = \omega I(t). \tag{4}$$

[8] Both [1, 14] show the poissonian behavior of mutiple visits from the same user.

The equation $p(t) = \omega I(t)$ accounts for the expected number of times infected individuals access the object, thus capturing the popularity of the object at time t due to the shock. We can also define the expected audience size of the object at time t due to the shock, $a(t)$, as: $a(t) = (1 - e^{-\frac{\omega}{\gamma}})\beta S(t-1)I(t-1)$. Each newly infected individual ($\beta S(t-1)I(t-1)$) will stay infected for γ^{-1} windows (see [12]). The probability of generating at least one access while the individual is infected is: $1 - P(v(t + \gamma^{-1}) - v(t) = 0) = 1 - e^{-\frac{\omega}{\gamma}}$. Thus, we here capture the individuals which where infected at some time and generated at least one access.

The PHOENIX-R model is thus defined as the sum of the popularity values due to multiple shocks. We discuss how to determine the number of shocks in the next section. Given a set of shocks $\mathcal{S}$, where shock i starts at given time s_i, the popularity $\hat{p}$ is:

$$\hat{p}(t) = \sum_{i, s_i \in \mathcal{S}} p_i(t - s_i)\mathbb{1}[t > s_i] \tag{5}$$

where $\mathbb{1}[t > s_i]$ is an indicator function that takes value of 1 when $t > s_i$, and 0 otherwise. Audience, size $\hat{a}(t)$ can be similarly defined. Also, both in the single shock and in the PHOENIX-R models, the number of revisits at time t, $\hat{r}(t)$, can be computed as $\hat{r}(t) = \hat{p}(t) - \hat{a}(t)$. The overall population that can be infected is defined by $N = \sum_i N_i = \sum_i S(0)_i + 1$.

Note that we assume that the population of different shocks do not interact, that is, an infected individual from shock s_i does not interact with a susceptible one from shock s_j, where $i \neq j$. While this may not hold for some objects (e.g., people may hear about the same content from two different populations), it may be a good approximation for objects that become popular in large scale (e.g., objects that are propagated world wide). It also allows us to have different β_i, γ_i, and ω_i values for each population. Intuitively, the use of different parameters for each shock captures the notion that some objects may be more (or less) interesting for different populations. For example, samba songs may attract more interest from people in Brazil.

Adding a period: In some cases, the popularity of an object may be affected by periodical factors. For example, songs may get more plays on weekends. We add a period to the PHOENIX-R model by making ω fluctuate in a periodic manner. That is:

$$\omega_i(t) = \omega_i * (1 - \frac{m}{2} * (sin(\frac{2\pi(t+h)}{e}) + 1)). \tag{6}$$

e is the period, and sin is a sine function. For example, for daily series we may set $e = 7$ if more interest is expected on weekends. Since an object may have been uploaded on a Wednesday, we use the shift h parameter to correct the sine wave to peak on weekends. The amplitude m captures oscillation in visits. The same period parameters are applied to every shock model. This approach is similar to the one adopted in [18].

The final PHOENIX-R model will has 5 parameters to be estimated from the data *for each* shock, namely, $S(0)_i$, β_i, γ_i, ω_i, s_i; plus the m and h period parameters. The last two do not change for individual shocks. We decided to fix e in our experiments to 7 days, when using daily time windows, and $e = 24$ hours when using hourly series.

Algorithm 1 Fitting the PHOENIX-R model. Only the time series is required as input.

```
1: function FitPhoenixR($\mathbf{t}$)
2:     $\epsilon = 0.05$
3:     $\mathbf{s} \leftarrow \{\}$
4:     $\mathbf{p}, \mathbf{s}' \leftarrow FindPeaks(\mathbf{t})$
5:     $\mathbf{s}[1] = 0$
6:     $\mathbf{s} \leftarrow append(\mathbf{s}')$
7:     $\mathcal{P} \leftarrow \{\}$
8:     $min\_cost \leftarrow \infty$
9:     for $i \leftarrow 1$ to $|s|$ do
10:        $\mathcal{F} \leftarrow LM(\mathbf{t}, \mathbf{s}(:i))$
11:        $\mathbf{m} \leftarrow PhoenixR(\mathcal{F})$
12:        $mdl\_cost \leftarrow Cost(\mathbf{m}, \mathbf{t}, \mathcal{F})$
13:        if $mdl\_cost < min\_cost$ then
14:            $min\_cost \leftarrow mdl\_cost$
15:            $\mathcal{P} \leftarrow \mathcal{F}$
16:        end if
17:        if $mdl\_cost > min\_cost * (1 + \epsilon)$ then
18:            break
19:        end if
20:    end for
21:    return $\mathcal{P}$
22: end function
```

4.2 Fitting the Model

We now discuss how to fit the PHOENIX-R parameters to real world data. Our goal is to produce a model that delivers a good trade-off between parsimony (i.e., small number of parameters) and accuracy. To that end, three issues must be addressed: (1) the identification of the start time of each individual shock; (2) an estimation of the cost of the model associated with multiple shocks; and, (3) the fitting algorithm itself. Note that one key component of the fitting algorithm is model selection: it is responsible for determining the number of shocks that will compose the PHOENIX-R model, choosing a value based on the cost estimate and model accuracy.

Finding the start times s_i of the shocks: Intuitively, we expect each shock to correspond to a peak in the time series. Indeed, previous work has looked at the dynamics of single shock cascades, finding a single prominent peak in each cascade [2, 18]. With this in mind, instead of searching for s_i directly, we initially attempt to find peaks. We can achieve both tasks using a continuous wavelet transform based peak finding algorithm [7]. We chose the algorithm since it has the following key desirable properties. Firstly, it can find peaks regardless of the "volume" (or popularity in the present context) in the time windows surrounding the peaks. It does so by only considering peaks with a high signal to noise ratio in the series, that is, peaks that can be distinguished the time series signal around the candidate peak. Secondly, the algorithm is fast, with complexity in the order of the length, n, of the time series ($O(n)$). Lastly and more importantly, using the algorithm we can estimate both the peaks and the start times of the shocks that caused each peak. We shall refer to the algorithm as $FindPeaks$.

As stated $FindPeaks$ makes use of a continuous wavelet transform to find the peaks of the time series. Specifically, we apply the Mexican Hat Wavelet[9] for this task. The Mexican Hat Wavelet is parametrized by a half-width l. We use half-widths (l) of

[9] https://en.wikipedia.org/wiki/Mexican_hat_wavelet

values $\{1, 2, 4, 8, 16, 32, 64, 128, 256\}$ to find the peaks. Thus, for the peak identified at position k_i, with wavelet determined by the parameter l_i, we define the start point of the shock s_i as: $s_i = k_i - l_i$. We found that using the algorithm with the default parameters presented in [7], combined with our MDL fitting approach (see below), proved accurate in modeling the popularity of objects[10].

Estimating the cost of the model with multiple shocks: we estimate the cost of a model with $|\mathcal{S}|$ shocks based on the minimum description length (MDL) principle [10, 19], which is largely used for problems of model selection. To apply the MDL principle, we need a coding scheme that can be used to compress both the model parameters and the likelihood of the data given the model. We here provide a new intuitive coding scheme, based on the MDL principle, for describing the PHOENIX-R model with $|\mathcal{S}|$ shocks, assuming a popularity time series of n elements (time windows). As a general approach, we code natural numbers using the $\log^*$ function (universal code length for integers)[11] [10], and fix the cost of floating point numbers at $c_f = 64$ bits.

For each shock i, the complexity of the description of the set of parameters associated with i consists of the following terms: $\log^*(n)$ for the s_i parameter (since the start time of i can be at any point in the time series); $\log^*(S_i(0))$ for the initial susceptible population; and $3{*}c_f$ for β_i, γ_i, and ω_i. We note that an additional cost of $\log^*(7){+}2{*}c_f$ is incurred if a period is added to the model. However, we ignore this component here since it is fixed for all models. Therefore, it does not affect model selection. The cost associated with the set of parameters $\mathcal{P}$ of all $|\mathcal{S}|$ shocks is:

$$Cost(\mathcal{P}) = |\mathcal{S}| \times (\log^*(n) + \log^*(S_i(0)) + 3 * c_f) + \log^* |\mathcal{S}|. \tag{7}$$

Given the full parameter set $\mathcal{P}$, we can encode the data using Huffman coding, i.e., a number of bits is assigned to each value which is the logarithm of the inverse of the probability of the values (here, we use a Gaussian distribution as suggested in [19] for the cases when not using probabilistic models.).

Thus, the cost associated with coding of the time series given the parameters is:

$$Cost(\mathbf{t} \mid \mathcal{P}) = -\sum_{i=1}^{n} \log(p_{gaussian}(\mathbf{t}(i) - \mathbf{m}(i); \mu, \sigma)). \tag{8}$$

where $\mathbf{t}$ is the time series data and $\mathbf{m}$ is the time series produced by the model (i.e., $\mathbf{t}(i) - \mathbf{m}(i)$ is the error of the model at time window i.) Here, $p_{gaussian}$ is the probability density function of a Gaussian distribution with mean μ and standard deviation σ estimated from the model errors. We do not include the costs of encoding μ and σ because, once again, they are constant for all models. The total cost is:

$$Cost(\mathbf{t}; \mathcal{P}) = \log^* n + Cost(\mathcal{P}) + Cost(\mathbf{t} \mid \mathcal{P}). \tag{9}$$

This accounts for the parameters cost, the likelihood cost, and the cost of the data size.

Fitting algorithm: The model fitting approach is summarized in Algorithm 1. The algorithm receives as input a popularity time series $\mathbf{t}$. It first identifies candidate shocks

[10] We used the open source implementation available with SciPy (`http://scipy.org`)

[11] $\log^*(x) = 1 + \log^*(\log x)$ if $x > 1$. $\log^*(x) = 1$ otherwise. We use base-2 logarithms.

using the $FindPeaks$ method, which returns the peaks $\mathbf{p}$ and the start times $\mathbf{s}'$ of the corresponding shocks in decreasing order of peak volume (line 3). To account for the upload of the object, we include one other shock starting at time $s_1 = 0$, in case a shock was not identified in this position. Each s_i is stored in vector $\mathbf{s}$, ordered by the volume of the each identified peak (with the exception of $s_1 = 0$ which is always in the first position) (lines 4 and 5). We then fit the PHOENIX-R model using the Levenberg-Marquardt (LM) algorithm adding one shock at a time, in the order they appear in $\mathbf{s}$ (loop in line 9), that is, in decreasing order of peak volume (after the initial shock). Intuitively, shocks that lead to larger peaks account for more variance in the data. For each new shock added, we evaluate the MDL cost (line 12). We keep adding new shocks as long as the MDL cost decreases (line 13) or provided that an increase of at most ϵ over the best model is observed[12] (line 17). We set the Levenberg-Marquardt algorithm to evaluate the mean squared errors of the model and adopt a threshold ϵ equal to 5%. We also note that we initialize each parameter randomly (uniform from 0 to 1), except for $S_i(0)$ values. For the first shock we do test multiple initial values: $S_1(0) = 10^3$, 10^4, 10^5, and 10^6. The other $S_i(0)$ values are initialized to the corresponding peak volume.

5 Experiments

In this section we discuss the experimental evaluation of the PHOENIX-R model. Initially, we present results on the efficacy of the model on our datasets when compared to state-of-the-art alternatives (Section 5.1) Next, we show results for the applicability of the model for popularity prediction (Section 5.2).

5.1 Is PHOENIX-R Better than Alternatives?

We compare PHOENIX-R with two state-of-the-art alternatives: the TemporalDynamics [21], used to model query popularity; and the SpikeM model [18], which captures single cascades. We compare these models in terms of time complexity, accuracy, estimated by the root mean squared errors (RMSE), and cost-benefit. For the latter, we use the Bayesian Information Criterion (BIC) [21], which captures the tradeoff between cost (number of parameters) and accuracy of the model.

In terms of time complexity, we note that the PHOENIX-R model scales linearly with the length of the time series n. This is shown in Figure 4, which presents the number of seconds (y-axis) required to fit a time series with a given number of time windows (x-axis). TemporalDynamics also has linear time complexity [21]. In contrast, the equations that govern the SpikeM model requires quadratic ($O(n^2)$) runtime on the time series length, making it much less scalable to large datasets.

In terms of accuracy, we make an effort to compare PHOENIX-R with the alternatives in fair settings, with datasets with similar characteristics from those used in the original papers. In particular, when comparing with TemporalDynamics, we run the models proposed in [21] selecting the best one (i.e., the one with smallest root mean

[12] MDL based costs will decrease with some variance and then increase again. The ϵ threshold is a guard against local minima due to small fluctuations.

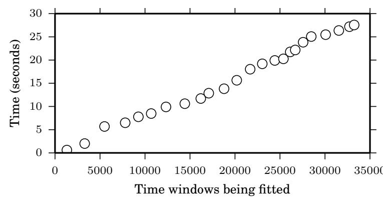


Fig. 4: Scalability of PHOENIX-R

Table 4: Comparison of PHOENIX-R with TemporalDynamics [21] and SpikeM [18]: Average RMSE values (with 95% confidence intervals in parentheses). Statistically significant results (including ties) are shown in bold.

	PHOENIX-R vs. TemporalDynamics (daily series)		PHOENIX-R vs. SpikeM (hourly series)	
	RMSE PHOENIX-R	RMSE TemporalDynamics	RMSE PHOENIX-R	RMSE SpikeM
MMTweet	**2.93** (± 0.23)	4.18 (± 0.49)	-	-
LastFM	**7.09** (± 0.23)	8.31 (± 0.32)	-	-
Twitter	**72.05** (± 6.08)	194.79 (± 20.49)	**10.83** (± 1.61)	**9.77** (± 2.24)
YouTube	**280.58** (± 29.29)	3429.19 (± 577.76)	-	-

squared error) for each time series. Moreover, we use long term daily time series (over 30 days), with a total popularity of at least 1,000[13]. We compare PHOENIX-R and TemporalDynamics under these settings in our four datasets, including YouTube.

When comparing with SpikeM, we use Twitter hourly time series trimmed to 128 time windows around the largest peak (most popular hour). We focus on the 500 most popular of these times series for comparison. We chose this approach since this is the same dataset explored by the authors. Moreover, we focus on a smaller time scale because the SpikeM was proposed for single cascades only.

Table 4 shows the average RMSE (along with corresponding 95% confidence intervals) computed over the considered time series for all models. Best results of each comparison (including statistical ties) are shown in bold. Note that PHOENIX-R has statistically lower RMSE than TemporalDynamics in all datasets. These improvements come particularly from the non-linear nature of PHOENIX-R , which better fits the long term popularity dynamics of most objects. The difference between the models is more striking for the YouTube dataset, where most time series cover long periods (over 4 years in some cases). The linear nature of TemporalDynamics largely affects its performance in those cases, as many objects do not experience a linear popularity evolution over such longer periods of time. As result, PHOENIX-R produces reductions on average RMSE of over one order of magnitude. In contrast, the gap between both models is smaller in the LastFM dataset, where the fraction of objects (artists) for which a linear fit is reasonable is larger. Yet, PHOENIX-R produces results that are still statistically better, with a reduction on average RMSE of 15%.

When comparing with SpikeM, the PHOENIX-R model produces results that are statistically tied. We consider this result very positive, given that this comparison favors SpikeM: the time series cover only 128 hours, and thus there is no much room for

[13] Similar results were achieved using other thresholds.

Table 5: Comparing Phoenix-R with TemporalDynamics [21] for prediction. The values on the table are RMSE. Statistically significant results are in bold

		5%			25%			50%		
		1	7	30	1	7	30	1	7	30
MMTweet	PhoenixR	**11.61**	**12.78**	**15.15**	**8.67**	**6.74**	**8.82**	**4.08**	**6.87**	**13.58**
	TempDynamics	**17.07**	**17.41**	**16.52**	**9.63**	**10.78**	**14.46**	**25.19**	**23.08**	**30.39**
Twitter	PhoenixR	**53.68**	**60.78**	**215.76**	**132.21**	**135.15**	**210.30**	**75.58**	**229.59**	**254.93**
	TempDynamics	**104.45**	**129.36**	**255.69**	643.39	643.83	786.50	420.74	587.86	598.75
LastFM	PhoenixR	**2.37**	**3.97**	**5.71**	**8.60**	**12.06**	**14.66**	**11.34**	**15.03**	**15.43**
	TempDynamics	6.47	7.03	8.00	11.15	14.62	17.86	14.91	18.15	18.80
YouTube	PhoenixR	**91.62**	**106.38**	**138.88**	**83.76**	**113.14**	**147.04**	**127.53**	**97.97**	**115.97**
	TempDynamics	3560.65	3631.09	3661.81	5091.82	5107.82	5143.70	4136.14	4139.73	4169.26

improvements from capturing multiple cascades, one key feature of PHOENIX-R . Yet, we note that our model is more general and suitable to modeling popularity dynamics in the longer run, besides being much more scalable, as discussed above.

As a final comparison, we evaluate the cost-benefit of the models using BIC, as suggested by [21]. We found that we beat TemporalDynamics in terms of BIC on at least 80% of the objects in all datasets but LastFM. For LastFM objects, the reasonable linear evolution of popularity of many objects, makes the cost-benefit of TemporalDynamics superior. Yet, PHOENIX-R is still the preferred option in 30% of the objects in this dataset. Compared to SpikeM we also find that again, statistically equal BIC scores are achieved for both models.

5.2 Predicting Popularity with PHOENIX-R

We here assess the efficacy of PHOENIX-R for predicting the popularity of objects a few time windows into the future, comparing it against TemporalDynamics[14]. To that end, we train the PHOENIX-R and TemporalDynamics models for each time series using 5%, 25%, and 50% of the initial daily time windows. We then use the δ time windows following the training period as validation set to learn model parameters. In each setting, we train 10 models for each time series, selecting the best one on the validation period. We then use the selected model to estimate the popularity of the object δ windows after the validation (test period). We experiment with δ equal to 1, 7 and 30 windows.

Table 5 shows the average RMSE of both models on the test period. Confidence intervals are omitted for the sake of clarity, but the best results (and statistical ties) in each setting are shown in bold. PHOENIX-R produces more accurate predictions than TemporalDynamics in practically all scenarios and datasets. Again, the improvements are quite striking for the YouTube dataset, mainly because the time series cover long periods (over 4 years in some cases). While the linear TemporalDynamics model fits reasonably well the popularity dynamics of some objects, it performs very poorly on others, thus leading to high variability in the results. In contrast, PHOENIX-R is much more robust, producing more accurate predictions for most objects, and thus being more suitable for modeling and predicting long periods of social activity.

[14] We do not use SpikeM for this task, as it is suitable for tail forecasting only (i.e., predicting after the peak)

6 Related Work

Popularity prediction of social media has gained a lot of attention recently, with many efforts focused on linear methods to achieve this task [20–22]. However, not all of these methods are useful for modeling *individual* time series. For example, linear regression based methods [20, 22] can be used for prediction but are not explanatory of individual time series. Moreover, as we showed in our experiments, there is strong evidence that linear methods are less suitable for modeling popularity dynamics than non-linear ones, particularly for long term dynamics. This comes from the non-linear behavior of social cascades [18]. Li *et. al.* [17] proposed a non-linear popularity prediction model. However they focused on modeling the video propagation through links on a single online social network, and not on general time series data, as we do here. Moreover, unlike PHOENIX-R , their method does not account for revisits or multiple cascades.

Recent work has also focused on modeling the dynamics of news evolution [18], or posts on news aggregators [2, 15, 16]. These prior efforts do not explicitly account for revisits nor multiple cascades, as we do. For example, the authors either assume unique visits only [18], or focus on applications that do not allow revisits (e.g., once a user likes a news posted on a application, she/he cannot like it a second time) [15]. In other cases, the models do not distinguish between a first visit by a user and a revisit [2].

Very recently, Anderson *et. al.* [1] analyzed revisits in social media applications. However, unlike we do here, the authors were not focused on modeling the evolution of popularity of individual objects, but rather the aggregate and user behavior.

7 Conclusions

In this paper we presented the PHOENIX-R model for social media popularity time series. Before introducing the model, we showed the effect of revisits on the popularity of objects on large social activity datasets. Next, we defined the model, showing it's applicability for predicting future popularity values of individual objects. Our main findings are:

- **Discoveries:** We explicitly show the effect of revisits in social media popularity.
- **Explanatory model:** We define the PHOENIX-R model, which explicitly accounts for revisits and multiple cascades. Both factors are not captured by state-of-the art alternatives.
- **Scalable and Parsimonious:** Our fitting approach make's use of the MDL principle to achieve a parsimonious description of the data. We also show that fitting the model is scalable (linear time).
- **Effectiveness** of model: We showed the effectiveness of the model not only when describing popularity time series, but also when predicting future popularity values for individual objects. Gains can be up to one order of magnitude larger than baseline approaches, depending on the dataset.

References

1. A. Anderson, R. Kumar, A. Tomkins, and S. Vassilvitski. Dynamics of Repeat Consumption. In *Proc. WWW*, 2014.

2. C. Bauckhage, K. Kersting, and F. Hadiji. Mathematical Models of Fads Explain the Temporal Dynamics of Internet Memes. In *Proc. ICWSM*, 2013.
3. O. Celma. *Music Recommendation and Discovery in the Long Tail*. Springer, 1 edition, 2010.
4. M. Cha, A. Mislove, B. Adams, and K. P. Gummadi. Characterizing social cascades in flickr. In *Proc. WOSN*, Aug. 2008.
5. M. Cha, A. Mislove, and K. P. Gummadi. A Measurement-Driven Analysis of Information Propagation in the Flickr Social Network. In *Proc. WWW*, 2009.
6. R. Crane and D. Sornette. Robust Dynamic Classes Revealed by Measuring the Response Function of a Social System. *Proceedings of the National Academy of Sciences*, 105(41):15649–53, Oct. 2008.
7. P. Du, W. A. Kibbe, and S. M. Lin. Improved peak detection in mass spectrum by incorporating continuous wavelet transform-based pattern matching. *Bioinformatics*, 22(17):2059–2065, 2006.
8. Q. Duong, S. Goel, J. Hofman, and S. Vassilvitskii. Sharding social networks. In *Proc. WSDM*, Feb. 2013.
9. F. Figueiredo, F. Benevenuto, and J. Almeida. The Tube Over Time: Characterizing Popularity Growth of YouTube Videos. In *Proc. WSDM*, 2011.
10. M. H. Hansen and B. Yu. Model Selection and the Principle of Minimum Description Length, 2001.
11. D. Hauger, M. Schedl, A. Kosir, and M. Tkalci. The Million Musical Tweets Dataset: What Can we Learn from Microblogs. In *Proc. ISMIR*, 2013.
12. H. W. Hethcote. The Mathematics of Infectious Diseases. *SIAM Review*, 42(4):599–653, 2000.
13. Q. Hu, G. Wang, and P. S. Yu. Deriving Latent Social Impulses to Determine Longevous Videos. In *Proc. WWW*, 2014.
14. C. Huang, J. Li, and K. W. Ross. Can Internet Video-on-Demand be Protable? In *Proc. SIGCOMM*, 2007.
15. H. Lakkaraju, J. McAuley, and J. Leskovec. What's in a Name? Understanding the Interplay between Titles, Content, and Communities in Social Media. In *Proc. ICWSM*, 2013.
16. K. Lerman and T. Hogg. Using a Model of Social Dynamics to Predict Popularity of News. In *Proc. WWW*, 2010.
17. H. Li, X. Ma, F. Wang, J. Liu, and K. Xu. On popularity prediction of videos shared in online social networks. In *Proc. CIKM*, 2013.
18. Y. Matsubara, Y. Sakurai, B. A. Prakash, L. Li, and C. Faloutsos. Rise and Fall Patterns of Information Diffusion. In *Proc. KDD*, 2012.
19. V. Nannen. A Short Introduction to Model Selection, Kolmogorov Complexity and Minimum Description Length (MDL). *Complexity*, (Mdl):20, 2010.
20. H. Pinto, J. Almeida, and M. Gonçalves. Using Early View Patterns to Predict the Popularity of YouTube Videos. In *Proc. WSDM*, 2013.
21. K. Radinsky, K. Svore, S. Dumais, J. Teevan, A. Bocharov, and E. Horvitz. Behavioral Dynamics on the Web: Learning, Modeling, and Prediction. *ACM Transactions on Information Systems*, 32(3):1–37, 2013.
22. G. Szabo and B. A. Huberman. Predicting the Popularity of Online Content. *Communications of the ACM*, 53(8):80–88, 2010.
23. A. Vakali, M. Giatsoglou, and S. Antaris. Social networking trends and dynamics detection via a cloud-based framework design. In *Proc. WWW*, Apr. 2012.
24. J. Yang and J. Leskovec. Patterns of temporal variation in online media. In *Proc. WSDM*, 2011.