Scalable Social Analytics for Live Viral Event Prediction

Puneet Jain Duke University
Justin Manweiler and **Arup Acharya** IBM T. J. Watson Research
Romit Roy Choudhury UIUC

Abstract

Large-scale, predictive social analytics have proven effective. Over the last decade, research and industrial efforts have understood the potential value of inferences based on online behavior analysis, sentiment mining, influence analysis, epidemic spread, etc. The majority of these efforts, however, are not yet designed with real-time responsiveness as a first-order requirement. Typical systems perform a post-mortem analysis on volumes of historical data and validate their "predictions" against already-occurred events. We observe that in many applications, real-time predictions are critical and delays of hours (and even minutes) can reduce their utility. As examples: political campaigns could react very quickly to a scandal spreading on Facebook; content distribution networks (CDNs) could prefetch videos that are predicted to soon go viral; online advertisement campaigns can be corrected to enhance consumer reception.

This paper proposes *CrowdCast*, a cloud-based framework to enable real-time analysis and prediction from streaming social data. As an instantiation of this framework, we tune CrowdCast to observe Twitter tweets, and predict which YouTube videos are most likely to "go viral" in the near future. To this end, CrowdCast first applies online machine learning to map natural language tweets to a specific YouTube video. Then, tweets that indeed refer to videos are weighted by the perceived "influence" of the sender. Finally, the video's spread is predicted through a sociological model, derived from the emerging structure of the graph over which the video-related tweets are (still) spreading. Combining metrics of influence and live structure, CrowdCast outputs sets of candidate videos, identified as likely to become viral in the next few hours. We monitor Twitter for more than 30 days, and find that CrowdCast's real-time predictions demonstrate encouraging correlation with actual YouTube viewership in the near future.

Introduction

With enormous popularity and pervasive use, social networks have unexpectedly established a global infrastructure to monitor, understand, and infer human sentiment at unprecedented scale and fidelity. Both the research community and industry have recognized this value, developing tooling and algorithms to ingest and process social data at petabyte scales, simultaneously exploiting and driving the proliferation of "Big Data" analytics. However, the promise of Big (social) Data, the availability of extreme quantities of data, creates substantial practical challenges. In particular, sophisticated analysis at scale demands overwhelming quantities of computation to distill sparse social signals at a relatively weak signal-to-noise ratio. Consequently, most nontrivial work towards understanding social data has relied on an offline analysis — bulk data processing or machine learning enabling a "post mortem" on historical data to infer macro-scale (coarse) trends. Alternatively, research towards short-duration, micro-scale inference has successfully demonstrated *predictive power*, but typically only "predicting" past events — no longer relevant to real applications. By neglecting nontrivial scalability challenges, thus barring immediate live deployment, these predictive analytics cannot yet enable practical, high-fidelity social forecasting.

Today, industrial systems can make simple social insights in real time — e.g., tracking, but not predicting, trending content (Topsy 2014). Alternatively, the academic community has proposed sophisticated offline analyses with predictive power. In this paper, we seek to bridge the gap. We present *CrowdCast*, a *real-time framework* for high-fidelity predictive insights from social data. CrowdCast has been especially tuned for early inference of soon-to-become "viral" events. As deployed, CrowdCast (1) absorbs a live stream of Twitter tweets; (2) maps natural language tweets into specific topics; (3) weighs content sharing events a against a model of social "tastemakers;" (4) monitors the progress of a particular topic as an evolving, temporal graph structure of information dispersion; (5) tracks correspondence with sociological models of epidemic disease spread; and (6) reports correspondences as early indications of "virality."

Quantifying Virality

While we view CrowdCast's techniques as more general, we evaluate their efficacy in context of a specific social question: *"Which currently-unpopular videos on YouTube are likely to go viral in the near future?"* We choose this question for two reasons: *immediate applications* and *scientific verifiability*. Today, early identification of viral videos can gainfully be applied to network optimization through mobile data prefetching or in optimization of content delivery networks (CDNs). Such predictions are also invaluable in various business contexts. For example, with the rise of so-called "viral marketing," it is valuable to calibrate the success or failure of an active campaign, enabling adjustments in real time. Finally, an objective notion of "virality" can be quantified through as the rate of change of YouTube viewership, enabling empirical evaluation of CrowdCast's predictive power.

There is poor consensus of what constitutes "virality." For empirical study, we require an unambiguous metric. Accordingly, we define our prediction goal as follows: CrowdCast only considers YouTube videos that are currently unpopular (specifically, watched less than 1000 times) and will make a binary prediction: *viral* or *non-viral*. A *true positive* result will only be awarded if the video becomes extremely popular (specifically, being watched more than one half million times), *i.e., more than 500-fold increase in viewership*.

Like many inference platforms, CrowdCast is tunable, depending on the application's relative sensitivity to false positives/negatives. For mobile video prefetching, we would tune for minimal false positives, to reduce waste of scarce battery life and wireless bandwidth. For CDN optimization, substantial storage and network resources reduce the cost of false positives, thus we would tune for few false negatives to maximize system utility. As evaluated, CrowdCast accurately predicts about 50% of the videos which actually become viral, with about two false positives for every true positive, only predicting 0.12% of non-viral videos as viral.

Architecting for Real-time Analytics

It is challenging to adapt prior art in offline methods for virality prediction to the practical constraints of a real-time platform. CrowdCast's architectural design was refined across many iterations, so that we may identify existing and novel techniques feasible with tolerable computational cost.

CrowdCast seeks to apply the state-of-the-art in viral event detection, but uniquely, to do so in an online manner. Recent work has shown that viral information spread can be detected by: (1) constructing a tree with edges denoting how a possibly-viral topic spreads from person to person through social sharing events and (2) classifying only the *structure* of that tree as either "wide" or "deep." A deep tree structure, quantifiable as the average path length between any pair of nodes in the tree, indicates a high tendency for a topic to be re-shared. This re-sharing structure strongly correlates to an explosive, viral information dispersion. Unfortunately, known effective metrics (such as average path length) are computationally intractable to compute over large graphs in real time. However, prior work has shown that social network relationships form scale-free graphs.[1] We show that subgraphs of the Twitter social network *constructed by actual social sharing interactions* are also scale-free. Heuristics exist for low-complexity estimation of average path length on scale-free graphs. Through this insight, CrowdCast detects the characteristic viral dissemination structure scalable, in real time.

While it is difficult to track and predict the viral spread of a single video, it is entirely another matter to consider all the thousands of videos in active discussion at any instant. Thus, CrowdCast leverages the many optimizations of Big Data software tooling that enable horizontal scalability. Through a lightweight temporal analysis featuring online machine learning and real-time use of Big Table storage, CrowdCast continuously monitors live Twitter social sharing activity in the aggregate, flagging and tracking anomalous behaviors indicative of a viral spread. Ultimately, CrowdCast maintains rank order of the ≈ 500 videos it perceives as most likely to become viral in the next few hours.

CrowdCast judiciously applies Big Data software tooling in an on-line fashion, carefully *maintaining a tolerable computational delay on the critical path* (milliseconds). However, to be useful, CrowdCast's predictions must not only be timely, they must build upon, and ensure an analytic accuracy comparable to prior art. Accordingly, CrowdCast must still leverage the output of a (relatively) slow Map Reduce-style distillation on vast qualities of social data. However, so that this distillation may be applied on the critical path, CrowdCast periodically builds a real-time *analytic cache* through a large-scale graph traversal — making a tiny concession to data freshness for a massive efficiency gain.

Contributions

CrowdCast remains an active research project. We endeavor to improve prediction accuracy and generalize applicability across domains. Despite ample room for further research, *we believe CrowdCast today makes the following contributions:*

Twitter Activity as a Scale-Free Graph: We model Twitter activity as a graph structure and identify those graphs as scale-free. Through this property, we can efficiently estimate average path length. Ultimately, we enable real-time detection for the tell-tale structure of viral information dispersion.

Real-time Social Graph Analytics: We synergistically combine advanced distributed graph processing based on Google Pregel and low-latency Big Table data storage. By merging these techniques, we enable an analytic cache of social interpersonal influence, amenable to real-time queries.

End-to-end Real-time System Architecture: We define a social processing pipeline with a constant-time critical path. Thereby, we enable live analysis feasible at the full ingress rate of worldwide Twitter activity. In particular, this pipeline includes: (1) online machine learning to understand natural language context; (2) per-topic tracking and aggregation Twitter social activity; (3) and computation and storage requirements linear with the ingress rate of Twitter tweets.

We deployed CrowdCast as a complete prototype in a production cloud. 30 days of live predictions demonstrated sufficient correlation with future YouTube viewership to enable plausible applications of viral video prediction.

Related Work

CrowdCast extends or is complementary to a large body of related work in social analytics, especially those techniques: (1) based on theory of information diffusion, (2) understanding social influence, (3) predicting from past Twitter activity, and (4) enabling live social forecasting applications.

Sociological Models of Virality

Substantial prior work considers the nature of viral information spread. (Guerini and Strapparava 2011) advocates that the nature of content drives virality, rather than "influencers." This notion is partially contradicted by (Hoang and

[1] A graph where vertex degree distribution follows a power law.

Lim 2012), proposing the SIR/SIRS-based virality model, and emphasizing the importance of influencers. Recently, several researchers have hypothesized various factors contributing to virality of a video. A TED talk (Allocca 2012) explains how "tastemakers," "creatively participatory community," and "unexpectedness" contributes to virality on YouTube. (Broxton et al. 2010) analyzes two years of confidential YouTube datasets to understand how human nature and other factors contribute to viral video spread. The authors present correlation between the degree of social sharing and the video viewership, how video category correlates to social linking behaviors, and differentiate the lifespan of popular and viral videos. (Shamma et al. 2011) considers data from one day of Yahoo! Zinc sessions to establish correlation between sharing behavior and video viewership. (Berger and Milkman 2009) considers psychological/emotional causes underlying viral news articles, using three months of The New York Times. (Wallsten 2010) considers the famous 2008 "Yes We Can" election campaign for President Barack Obama, and how community contributed in making this video viral.

The model proposed by (Goel, Watts, and Goldstein 2012) influences our techniques. The authors understand a viral spread based on a underlying structure of a social sharing graph. Importantly, the work differentiates *popular* from *viral* spread. Popular stories display a star-like structure (similar to one-time broadcast, implying that a large number of individuals received information from same source) and then the spread quickly dissipates. Conversely, viral stories show a tree-like structure where many nodes participate in information spread through active re-sharing. While the authors are able to identify viral spread by structure, the analysis assumes a complete dataset, only available after the phenomenon has occurred. We extend this model across various dimensions: (1) a moving-window temporal analysis based on tweet creation times; (2) incorporating graph models of influence, in addition to structure; (3) making real-time predictions based on partial information; and (4) through computational optimizations to enable tractable live analysis.

Influence Analysis in Social Networks

The nature of interpersonal influence (who influence whom, and to what degree) has been studied extensively in social networks. Various metrics have been proposed to quantify how influence contributes to information spread, especially by understanding the importance of individuals in the network. Most popular have been measures of vertex centrality, such as degree, closeness, betweeness, and PageRank. However, these metrics attempt to quantify influence in a static network context, and might not fully apply with dynamic behavioral patterns in real networks. (Cha et al. 2010) considers the impact of Twitter in-degree, "retweets," and "mentions," contradicting prior hypotheses that high in-degree should imply higher influence. (Weng et al. 2010) extends PageRank to define "TwitterRank," an influence metric that accommodates link structure as well as topic similarity.

We blend techniques of static and dynamic graph influence analysis. CrowdCast periodically applies PageRank to a (relatively) static graph of social relationships. PageRank influence scores are then applied as weighting mechanism in a second dynamic graph, based on dynamic Twitter social activities (e.g., tweet, retweet, mention, reply). We believe this hybrid approach is especially promising, as it blends the advantages of global omnipotence in large-scale static graph analysis with live insights over smaller, dynamic graphs.

Future Predictions using Twitter

(Zaman et al. 2010) presents a technique to predict retweet count of an individual tweet, using the last few hours of tweets and retweet count data. (Dong et al. 2010) considers applications of Twitter to improve real-time web search results. (Sakaki, Okazaki, and Matsuo 2010) shows the possibility of using Twitter to aid earthquake prediction in Japan, outperforming existing earthquake notification systems such as JMA. (Mathioudakis and Koudas 2010), (Szabo and Huberman 2010), (Lerman and Hogg 2010) uses Twitter for making predictions, such as popularity of a digg.com article, YouTube videos, and trending topics.

These techniques are very much aligned with our goals for CrowdCast, however they uniformly share the same major caveat. Each of these works relies on an offline analysis that cannot be easily run with a bounded makespan, and thus they are not suitable for real-time prediction applications.

Motivating our design, (Morstatter, Liu, and Carley 2013) shows that Twitter's live 1% randomly-sampled public tweet stream contains sufficient data to extract global trends and patterns. In our context, viral events occur at massive scale, shared by tens or hundreds of thousand of users; even at low sampling rates, early warnings of viral events are detectable.

Applications of Live Viral Video Prediction

Prior work has identified various applications for predicting soon-to-be viewed videos. (Bao et al. 2013) proposes opportunistic video content prefetching for smart devices over WiFi, reducing the data cost on 3G/4G wireless for consumers and cellular providers. Related to our technique of predicting viral videos via Twitter, (Agarwal 2009) considers using Twitter-provided real-time lists of popular Internet topics as a basis to optimize content delivery networks (CDNs), enabling predictive content prefetching. Several industrial services provide real-time analysis of popular YouTube videos. Closest to our goals, Unruly Media claims a proprietary heuristic called "ShareRank" to provide a visual dashboard of popular videos from the last 24 hours (UnrulyMedia 2013). However, Unruly does not publish its algorithms: neither is it clear what (if any) predictions are performed, nor is there quantification of accuracy or timeliness.

Overview and System Design

We propose CrowdCast as a cloud software framework for *Crowd*sourced (social) fore*Cast*ing. CrowdCast accommodates the continuous, high-volume, real-time nature of online social networks, tapping a torrent of social data in real time to make live predictions of viral behaviors. While our techniques are more generally applicable, we focus our attention on Twitter for three reasons, (1) *high volume:* according to recent statistics, 500 million tweets per day with a

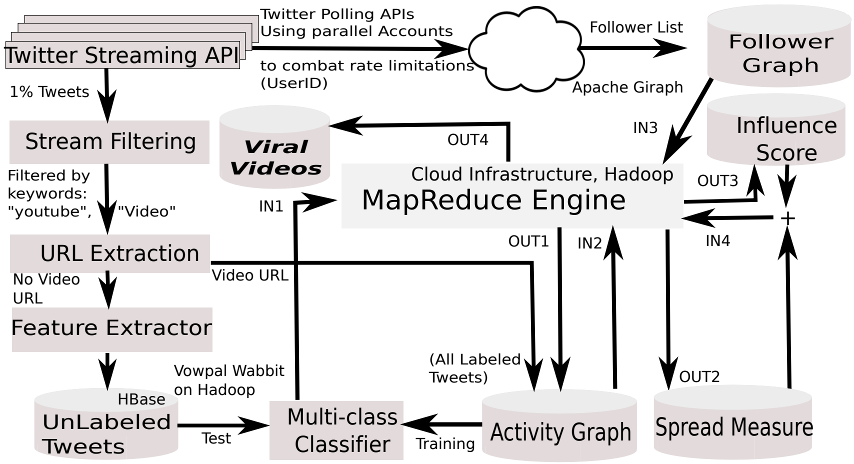

Figure 1: CrowdCast architecture:

Critical path: follow the flow of Twitter tweets down from the top-left corner, ultimately contributing to the activity graph. Unlabeled tweets pass through classification, mapping each to an appropriate *video_id*.

Graph processing: Following Tweets right through the top-right corner and below traces construction of the analytic cache, used in influence analysis.

record of *143,199 tweets in one second* (Twitter blog 2013); (2) *open, live access:* Twitter directly provides a free stream sampling 1% of live tweets over a TCP connection, and a 10% live sampling "Decahose" is available for pay (GNIP 2014);[2] (3) *live interactivity:* both by design and in practice, Twitter activity tends to center around what is happening *now*, encouraging a live, distributed conversation.

CrowdCast is engineered to process a live stream of Twitter tweets at 1%, 10%, or even 100% rates. We have deployed our prototype on several virtual machines in a production infrastructure-as-a-service (IaaS) cloud. As Twitter traffic is bursty, both with seasonality effects (e.g., time of day) and with exceptional events (e.g., political uprising), we design CrowdCast to accommodate load variability through elastic scalability — growing and shrinking the deployment to optimize for need and cost. To enable full-Twitter scalability with a reasonable cost (10s of virtual machines, not 1000s), however, it is essential that CrowdCast's performance degrades only linearly with load, necessitating a *constant-time per-tweet critical path*. To enable such a performant pipeline at scale, CrowdCast synergistically combines the state-of-the-art in Big Data software tooling.

In this section, we (1) provide background on the Big Data software tooling leveraged in the CrowdCast solution; (2) describe the Hadoop processing applied to a large scale (billions of edges) Twitter relationship graph to distill key influencers, known as "tastemakers;" (3) present a constant-time critical path processing pipeline applied to a live stream of Twitter tweets, and featuring online machine learning on natural language; (4) describe the construction of per-video Twitter activity graph, enabling quantification through metrics of influence and graph structure analysis, and informing predictions of viral events; and (5) overcome the computational challenge of live graph structure analysis, eliminating the final barrier to real-time deployment of CrowdCast.

Supporting Platforms

CrowdCast must leverage state-of-the-art software frameworks to enable production-level robustness at Twitter scale.

Apache Hadoop and HDFS Hadoop Map Reduce enables high-volume data-centric processing in a distributed cluster environment. The Hadoop filesystem (HDFS) provides high-throughput distributed storage, optimized for bulk IO. HDFS serves as a backing store to Apache HBase. Hadoop Map Reduce is a prerequisite for Apache Giraph.

Apache HBase HBase is an interactive (low-latency) key-value data store in the model of Google Big Table. HBase efficiently maintains a logical mapping of ordered keys to a set of values, with values subsequently ordered according to a system of temporal "versioning." Versioning support makes HBase amenable to CrowdCast's requirements for time-series analysis: for a particular key, HBase efficiently returns all relevant data in temporal order. HBase is also well-suited to managing large quantities of streaming data with deadlines. CrowdCast configures HBase to automatically expire (discard) sufficiently-old social data, once it is too old for a meaningful contribution to analytics.

Apache Giraph Giraph is a groundbreaking platform for large-scale, distributed graph analysis, inspired by Google Pregel (Malewicz et al. 2010). CrowdCast applies PageRank on a Twitter connectivity graph (follower-friend relationships) of ≈ 133 million vertices and ≈ 1.7 billion edges, generated from 3 months of continuous polling to Twitter's public APIs, and integration of academic datasets (Kwak et al. 2010; Li and others 2012). Giraph completes PageRank on the complete graph within 1.5 hours on a 10-node cluster.

Vowpal Wabbit Vowpal Wabbit is a data-driven machine learning toolkit, principally for regression and classification, and is highly optimized to run online in one pass. Impressively, Vowpal Wappit is IO, rather than computationally, bound. CrowdCast applies Vowpal Wappbit for a bag-of-words natural language classification to map tweets to a specific YouTube video. For example, during the well-publicized "Gangham Style" music video viral event, many tweets may imprecisely reference the video as "gangham," "g-style," or simply "dance vid." Online machine learning classification enables CrowdCast to translate this natural language text to a specific YouTube video content ID.

[2] 100% tweet access is available for purchase, but no provider currently distributes greater than a 10% sampling as a live stream.

Analytic Cache (each user)

CrowdCast leverages a form of influence analysis in predictions of viral spread. Specifically, CrowdCast takes a large Twitter follower-friend relationship graph as input, applies the PageRank algorithm, and outputs PageRank scores as a proxy for quantifying relative influence. To build our graph, we initialized an HBase table with all inferable relationships from two existing datasets (Kwak et al. 2010) and (Li and others 2012). Additional Twitter datasets could not be used due to anonymized user IDs (Cha et al. 2010). The resulting graph was still too sparse and disconnected for our use. Therefore, we supplemented this table by crawling Twitter public APIs for six months (at the rate limit specified by Twitter terms of service). We continue to enhance our graph.

Our graph currently contains 131 million users and 1.7 billion follower-friend relationships. While this is far from a true replication of the global Twitter relationship graph, it is sufficiently dense to be useful. However, it is not feasible to traverse this graph in real time. Instead, CrowdCast periodically leverages the Giraph graph processing framework to compute a PageRank score for each vertex (completing after several hours). CrowdCast stores each PageRank score in an HBase table, keyed by user ID. We refers to this table as an *analytic cache*, reflecting that it provides constant-time access to this periodic, long-running analytic process.

The values accessible from the analytic cache (in constant time) are the PageRank scores of individual users. We employ these values as vertex weights in the context of a Twitter "activity graph" (discussed later), tracking the Twitter social actions for a particular video. For a fixed aggregated activity graph weight, a few vertices with large weights reflects a different form of information dispersion than many vertices with low, similar weights. Namely, the former case reflects "celebrity" endorsement; the later is indicative of a peer-to-peer (or "viral") spread — exactly what we seek to detect.

Constant-time Critical Path (each tweet)

Globally, Twitter users take thousands of social actions per second, in various forms: "tweet" (original text content up to 140 characters), "retweet" (re-shared text content, increasing dissemination to the followers of the re-tweeting user), "mention" (tweet directed to or about a specific user, increasing the probability that the mentioned user will see the content in her Twitter "feed"), or "reply" (similar to a mention, but responding to the content of a particular tweet by the mentioned user). Every minute, CrowdCast processes thousands of Twitter social actions relating to YouTube videos. At any point in time, thousands of different, specific YouTube videos are in active discussion on Twitter. CrowdCast monitors this global discussion, tracking those videos displaying an early indication of viral spread, and outputs a rank order of those videos most likely to go viral soon.

Processing Streaming Tweets CrowdCast opens a TCP connection to Twitter's public API. Asynchronously, Twitter pushes randomly-sampled tweets with JSON-formatted metadata. For each, CrowdCast checks: (1) for original content (*is_retweet = false*); otherwise, if it is a retweet/reply and from whom (*retweet_status.user*, *in_reply_to_status_id*); (2) for others to whom the tweet is directed (who is mentioned?) (*entities*); and (3) for another user which actually wrote the tweet (*contributors*).[3]

Next, CrowdCast processes the natural language text of the tweet content. Often, tweets describing a YouTube video will contain a direct hyperlink to watch the video. When present, CrowdCast parses these URLs[4] to extract an alphanumeric code representing a specific YouTube content ID (*video_id*). In cases where an explicit URL has been omitted, we apply bag-of-words classification on the text content to infer a likely YouTube content ID (observed by CrowdCast in the recent past). We describe the process of automatic training and classification in the next subsection.

Third, CrowdCast consults the analytic cache to retrieve the influence score corresponding to the originator of the social action. This score characterizes the impact of the action in extending the reach of the video, both in terms of video viewership and in subsequent Twitter activity.

CrowdCast completes the tweet processing by recording one or more entries into an HBase *activity graph* table based on (1) the extracted JSON metadata, (2) the explicit or inferred *video_id*, and (3) the influence score corresponding to the source of the tweet or other social action. Each row in the table denotes an edge in a user-to-user social action graph for a particular video. The key of each row is the *video_id*. The value denotes the edge, represented as the four-tuple <*source, destination, type, weight*>: (1) *source*: the user ID of the corresponding to the Twitter account taking the social action; (2) *destination*: the user ID to whom the social action is directed (in the case of a regular tweet, a loopback to the source is used); (3) *type*: the kind of social action (tweet, retweet, reply, mention, or contribution); and (4) *weight*: the PageRank score as retrieved from the analytic cache table.

Inference from Tweet to Video CrowdCast applies two complementary processes to infer the relevant video for a particular natural language tweet, *automated training* and *classification*. For every video-related tweet, one of two conditions hold, the tweet is labeled or unlabeled: (1) *labeled tweets* contains the specific YouTube URL of the video, and directly provide the *video_id*, while (2) *unlabeled tweets* require that the relevant *video_id* must be inferred from the natural language content, selecting from the set of other videos in active discussion on YouTube. Each tweet, labeled or unlabeled, passes through Vowpal Wabbit, a streaming tool for machine learning classification. Labeled tweets contribute to online training of the classifier; unlabeled tweets are classified online. Vowpal Wabbit's multi-class, bag-of-words classification algorithm assumes a fixed number of to-be-inferred classes (smaller numbers of classes improve accuracy). Accordingly, CrowdCast continuously maintains a bounded-size set of the most tweeted *video_id* values within a moving time window. Labeled tweets (and thus inferred labels) are filtered to membership in this set. Figure 2 plots video classification accuracy through standard cross-validation on labeled tweets (after removing the URL).

[3] Twitter contributors are listed for official communication of a business or other entity but authored by a third-party account.

[4] Twitter's API expands shortened URLs to original full form.

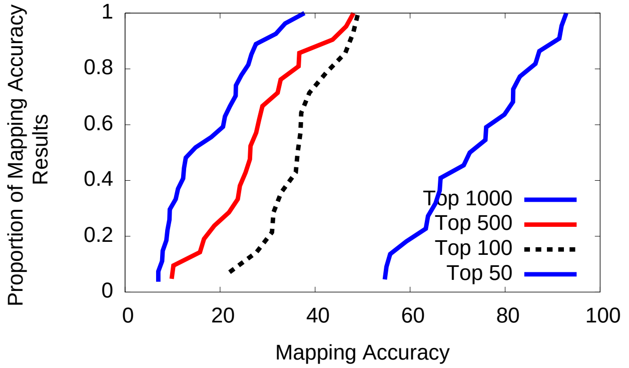

Figure 2: Classifier precision during cross-validation, mapping (de)labeled tweets to YouTube videos. Top X value refers to the maximum unique *video_id*s considered as classifier labels (videos of popularity rank $> X$ excluded).

Vowpal Wabbit is able to train and classify online, and is thus compatible with CrowdCast's constant-time critical path. Vowpal Wabbit's performance is IO, not computationally, bound, and is thus extremely quick per tweet. However, there are startup and shutdown costs to the process invocation. For example, upon termination after one or more training inputs, Vowpal Wabbit must persist its in-memory classification model. Consequently, for performance optimization, it is most sensible to run Vowpal Wabbit in batches. Accordingly, CrowdCast buffers incoming labeled and unlabeled tweets in HBase tables. Periodically (every few minutes), CrowdCast pulls and clears the unprocessed queue, initiating a Hadoop job sequence — first processing labeled tweets to update the classification model, then inferring labels for unlabeled tweets. With only limited batching, CrowdCast achieves total *training-plus-labeling* processing latency averaging 0.063 ms per tweet, throughput of *950K per minute*.

Twitter Activity Graph (each video)

HBase's support for *ordered keys* and *versioning* enables efficient retrieval of all Twitter activity relating to a particular video. With the table as constructed above, all activity relating to a particular video will be stored in a contiguous portion of a file written into HDFS, typically will be managed by a single HBase virtual machine, and during retrieval, will benefit from large cache blocks. Moreover, since each entry utilizes the same key (*video_id*) with multiple versions, HBase maintains entries about the same video in chronological order. Ultimately, a time series analysis may be conducted simply as a single-pass, sequential processing on a continuous chunk of an HDFS file. HDFS is specifically optimized to support such operations at high throughput.

The continuous process of per-tweet HBase insertions logically defines large-scale, continuously evolving graph structures for each video (Figure 3). For a particular video with a nontrivial quantity of social activity, the corresponding graph structure may have evolved in one of two different forms: *star-like* or *tree-like*. (1) In *star-like* graph structures, a single or small number of users tweet, and those tweets are received by may users (broadcast effect). However, the tweets are not retweeted or further shared, and the topic activity quickly subsides. (2) In *tree-like* graph structures, when users tweet about a topic, the content is retweeted, replied or mentioned frequently, by a substantial proportion of recipients. The topic disseminates across many users and propagates through multiple "generations" of sharing activity (recognizable as a tree structure with greater depth).

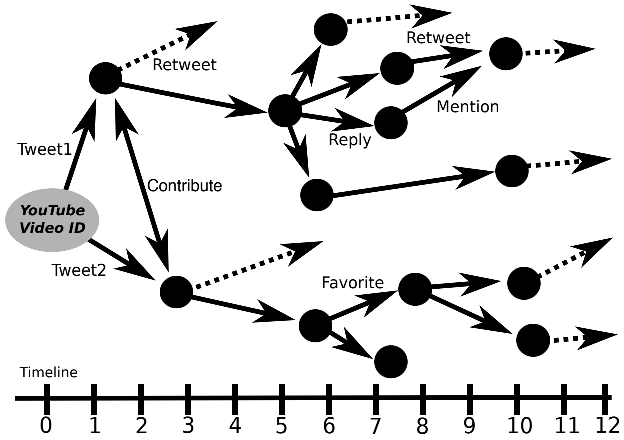

Figure 3: Twitter activity graph constructed for a single viral video. Note the tree-like structure expanding over time.

Note that through content-based addressing (*video_id* as HBase key value), CrowdCast constructs tweet activity graphs about the same video emerging from completely independent sources. We do not expect the arbitrary graph to be fully star-like or tree-like. Instead, CrowdCast quantifies each video's graph structure on a continuum, based on the average path length between all connected node pairs in the graph. Simply, the most tree-like structures (those that have the greatest average path length) appear most viral.

Computing Average Path Length CrowdCast quantifies a live Twitter activity graph as star-like or tree-like based on its all-pairs average path length. Longer average path lengths imply tree-like structure. Exact average path length may be computed through the $\Theta(|V|^3)$ Floyd Warshall algorithm or the $O(|V|^2 \log |V| + |V||E|)$ Johnson's algorithm for all-pairs shortest path, followed by a simple arithmetic mean.[5] However, neither technique is computationally feasible for processing our Twitter activity graphs live; makespan across all graphs (one for each of ≈ 500 videos) would be measured in hours. We describe next how CrowdCast applies a fast estimation heuristic, achieving a *three orders of magnitude speedup*, as shown by the log-scale CDF plot in Figure 4.

Average Path Length Heuristic No general average path length heuristics exists, but it can be effectively estimated for scale-free graphs. A scale-free graph is defined as having a degree distribution following a power law: $P(x) \propto x^{-\beta}$, where $P(x)$ is the fraction of nodes having degree x. Recent work in (Clauset, Shalizi, and Newman 2009) proposes a method to estimate power-law distribution parameters β and

[5] Let V and E denote the sets of vertices and edges, respectively.

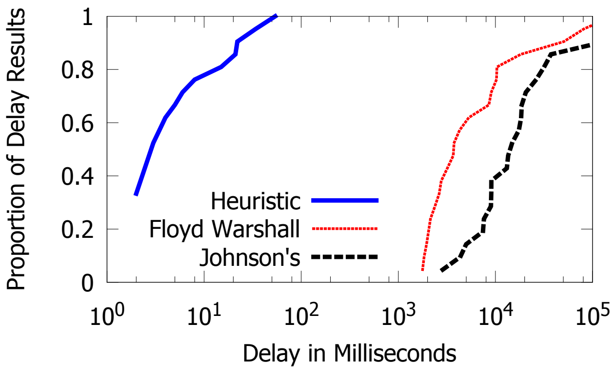

Figure 4: Comparing performance of CrowdCast's average path length heuristic versus the Johnson and Floyd Warshall algorithms for all-pairs shortest path. Empirically for our relatively-sparse (vertex dominated) graphs, we find that scalar overheads in Johnson's algorithm diminish runtime performance below Floyd Warshall, despite an asymptotic theoretic complexity advantage — $O(|V|^2 \log |V|)$ versus $\Theta(|V|^3)$ — $O(|V||E|)$ term is irrelevant in sparse graphs. Regardless, our linear $\theta(|V|)$ heuristic is three orders of magnitude faster than either, in both theory and practice.

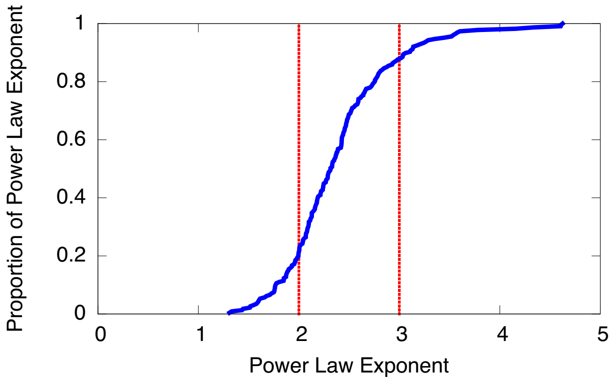

Figure 5: β distribution for activity graphs. For $2 < \beta < 3$ ($\approx 65\%$), average path length may be estimated accurately.

minimum value $x_{\min}$ from empirical data.[6] Accordingly, we leverage the authors' reference implementation to estimate β for a large set of activity graphs. Figure 5 shows the distribution of activity graph β values for all observed videos. For $\approx 65\%$ activity graphs, $2 < \beta < 3$. Average path length may be estimated when $\beta \geq 2$ (Albert and Barabási 2002):

$$d_{\text{avg}} = \begin{cases} \text{no formulation exists,} & \beta < 2 \\ \text{const,} & \beta = 2 \\ \dfrac{ln(ln(N))}{ln(\beta - 1)}, & 2 < \beta < 3 \\ \dfrac{ln(N)}{ln(ln(N))}, & \beta = 3 \\ ln(N), & \beta > 3 \end{cases}$$

As shown in Figure 6, we find empirically that in our Twitter activity graphs, the standard formulation for average path length estimation works well for $2 < \beta < 3$ and poorly for $\beta > 3$ (no formulation exists for $\beta < 2$). Therefore, we estimate average path length only for the dominant $2 < \beta < 3$ regime, roughly 65% of cases — for which estimation error is less than 25% in 95% of graphs. For all others, we apply the Floyd Warshall algorithm, with performance as shown in Figure 4 and complexity $\Theta(|V|^3)$.

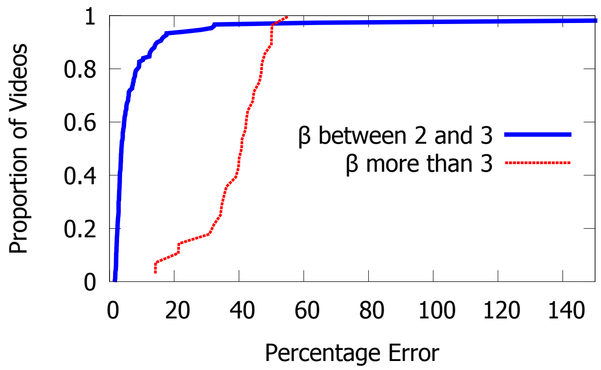

Figure 6: Average path length estimations using theory of scale-free graphs. When $2 < \beta < 3$, estimation proves sufficiently accurate, but we avoid the formulation for $\beta > 3$.

Evaluation

We validate CrowdCast's accuracy for a month using Twitter's freely-provided 1% tweet sample stream. We filter tweets to a list of video-related keywords or those containing a YouTube link directly.[7] Remaining tweets pass through our processing pipeline, as described above. Strictly for evaluation purposes, we continuously poll YouTube APIs to track viewership for videos in CrowdCast's consideration. Periodically, we sample and record CrowdCast's top 500 "predicted viral" videos. As we describe next, ground truth is only revealed later through "post-mortem" analysis.

Quantifying Virality In the YouTube context, CrowdCast attempts to determine which videos will become viral in the next hours. We consider a viral event to have occurred only when a video experiences an explosive increase in viewership over a short duration (less than 24 hours). Trivially, one could predict a video to become viral as its viewership approaches some threshold to become "viral enough." Instead, *we only consider predictions of a 500-fold increase in popularity.* Specifically, *CrowdCast makes predictions for videos with a total current viewership less than 1000, and predicts whether they will hit 500,000 views in the next 24 hours.*

Compared Techniques We evaluate CrowdCast's heuristic in conjunction with two alternative formations. **Node-**

[6] $x_{\min}$ denotes a lower bound for power-law distribution parameter x such that power-law behavior still holds.

[7] These tweets may be written in any language. We have successfully tested (but do not use for our evaluation) a language inference heuristic to filter to only English language tweets. Heuristic based on an empirical histogram of character frequency counts.

Count tracks the overall level of social activity about a particular video, ignoring the activity graph structure. **AverageDistance** considers the activity graph structure in isolation, but without quantifying the level of activity. **CrowdCast** combines these notions with PageRank scoring from the analytic cache. For a fixed activity level, a low aggregate PageRank implies that the activity represents a peer-to-peer (viral) information dissemination, rather than a celebrity endorsement. We use this scoring to complement the similar notion of **AverageDistance**. Aggregated inverted PageRank, however, is more unstable and performs poorly in isolation, thus is only shown through its contribution to **CrowdCast**.

NodeCount simple metric on tweet activity, ranks videos in descending order by vertex count in the activity graph.

AverageDistance ranks videos in descending order of average path length in the activity graph.

CrowdCast ranks videos in descending order by:

$$\frac{\textsc{NodeCount} \cdot \textsc{AverageDistance}}{\sum_{\forall i} p_i}$$

where p_i denotes the PageRank of activity graph vertex i.

Summary of Collected Data and Results[8]

initial video views (included in results)	$\leq$ 1,000
video views to be "viral" (o/w "non-viral")	$\geq$ 500,000
"viral" proportional viewership increase	$\geq$ 500x
duration of influence data collection	120 days
Twitter REST API rate limit (per acct.)	15 / min
# Twitter REST API calls	3.5M
# vertices in influence graph	131.3M
# edges in influence graph	1.69B
duration of tweet processing	30 days
mean rate of tweets (1% sample)	2466 / min
# tweets (all languages, filtered)	54M
% tweets in English	48.9%
# tweets related to videos	16.6M
# unique videos linked in tweets	3.2M
# videos with viewership < 1000	1.1M
# videos with viewership > 500K	1036
median time-to-viral (1K to 500K views)	366 hours
% viral	0.096%
overall CrowdCast precision	0.225
overall CrowdCast recall	0.537
overall CrowdCast F-measure	0.309
overall CrowdCast fallout (lower is better)	0.00149
respective standard deviations	0.053, 0.096, 0.054, 0.0095

Precision, Recall, Fallout, and F-Measure Let Y be be the set of all YouTube videos observed from the content of Twitter tweets. Let $V \subset Y$ be the set of YouTube videos that actually "go viral" (at least 500x increase in viewership). Let $P \subset Y$ be the set of YouTube videos predicted by CrowdCast to go viral. Then, $V \setminus P$ denotes CrowdCast's false negative predictions, $P \setminus V$ denotes CrowdCast's false positive predictions, and $Y \setminus V$ denotes the set of videos which did not go viral. *We evaluate CrowdCast's prediction efficacy by the following standard metrics of information retrieval:*

$$\textsc{Precision} = \frac{|V \cap P|}{|P|}$$

prop. of ***predicted viral*** *that are* ***viral***

$$\textsc{Recall} = \frac{|V \cap P|}{|V|}$$

prop. of ***viral*** *that are* ***predicted viral***

$$\textsc{Fallout} = \frac{|(Y \setminus V) \cap P|}{|(Y \setminus V)|}$$

prop. of ***non-viral*** *yet* ***predicted viral***

$$\textsc{F-Measure} = \frac{2 \cdot \textsc{Precision} \cdot \textsc{Recall}}{\textsc{Precision} + \textsc{Recall}}$$

harmonic mean of precision and recall

We sample CrowdCast's performance as 30 independent trials, one for each day (00:00 AM–11:59 PM) of one month. At midnight of each day, we sample CrowdCast's prediction of the (max) 500 videos it projects as most likely to become viral in the next 24 hour period. From these 500, we filter out any videos with more than 1000 YouTube views. 24 hours later, the subset of these videos with greater than 500,000 views are CrowdCast's *true positive* results and the remainders are *false positives* (reduce Precision). Any other videos meeting the both the 1000 and 500,000 requirement are considered *false negatives* (reduce Recall, F-Measure). Any remaining videos meeting the 1000 requirement are *true negative* results (improving Fallout). Figure 7 plots CDFs.

Figures 8(a,b) reflect the strong correspondence between the virality of videos and activity on Twitter. Figure 8(a) shows that viewership on YouTube changes rapidly, on small time scales (few hours). As evident, most of the videos do not become viral, receiving less than 100,000 views. Figure 8(b) shows the corresponding activity on Twitter for the same videos. Our result confirm previous studies Most videos ($\approx 70\%$) do not receive any social action (beyond a single initial tweet). Intuitively, the likeliness of receiving one or more social actions increases with longer durations. For videos receiving more than 10 social actions, our CDF curve flattens, reflecting that it may receive many future actions – becoming "viral-like."

Figure 9 compares the increase in YouTube viewership of predicted-viral videos to that of true-viral videos.

Figure 10 shows the distribution of *time-to-viral* taken by all videos receiving eventual viewership more than 500K. While 25% of videos become viral within one week, some become viral within 4 hours – confirming the importance of timeliness in virility prediction, such as CrowdCast enables.

Figure 11 plots Twitter users' self-reported timezones. Visually evident, EST, CST, and Central Europe are prominent, reflecting strong daily seasonality effects in worldwide Twitter usage. We design CrowdCast for elastic cloud deployment; substantial infrastructure cost savings may

[8] All figures approximate, subject to insignificant cloud downtown and maintenance interruptions.

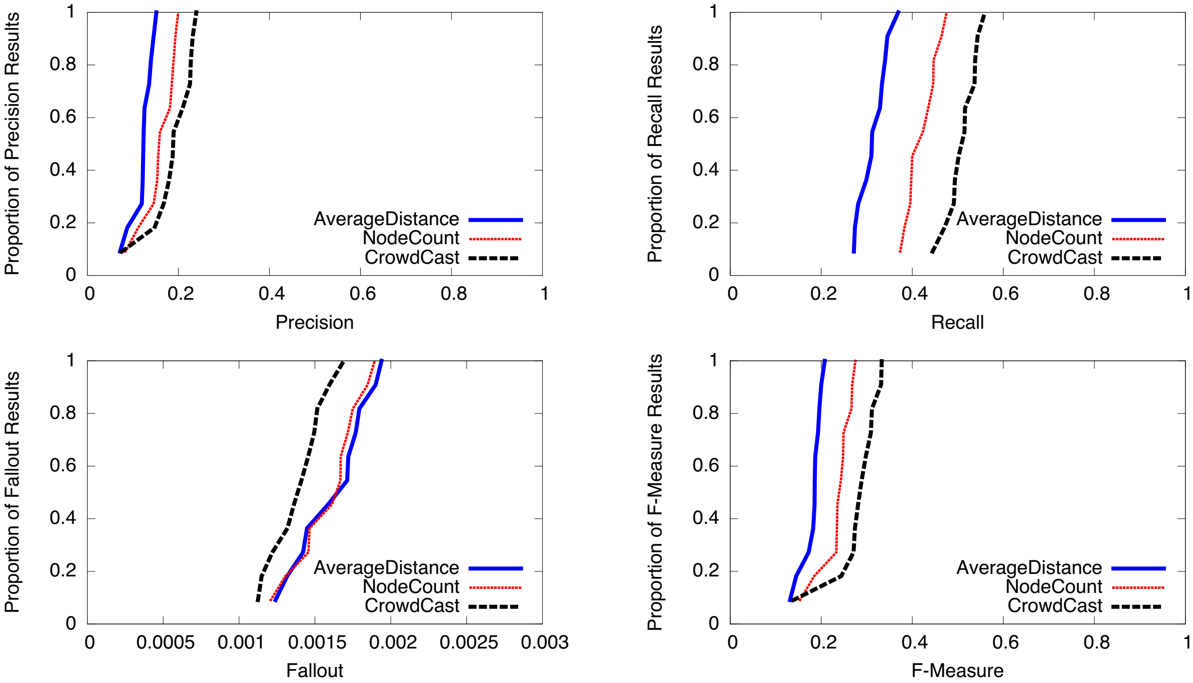

Figure 7: CDFs showing that CrowdCast predicts about 50% of the videos which actually become viral (*b:* RECALL), with about two false positives for every true positive (*a:* PRECISION), only predicting 0.12% of non-viral videos as viral (*c:* FALLOUT).

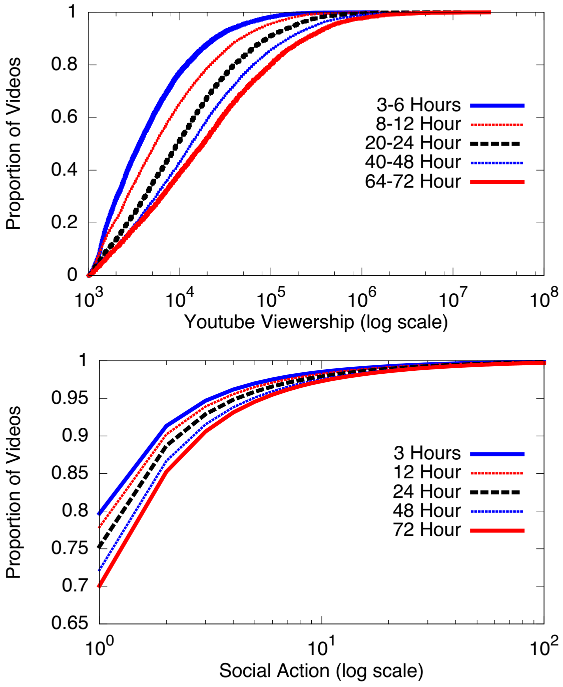

Figure 8: Distribution of video virality, plotted by time since first detected Tweet: (a) viewership of YouTube videos mentioned on Twitter; (b) Twitter social actions taken on videos.

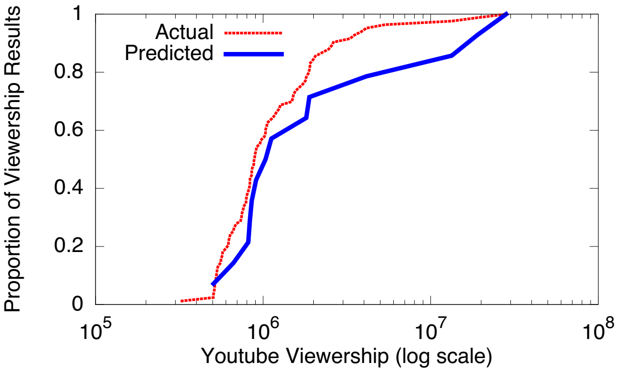

Figure 9: Viewership change, actual/predicted-viral videos.

be achieved by automatically provisioning/deprovisioning cloud virtual machine instances per need.

Conclusion

While voluminous prior art has demonstrated the predictive power of social forecasting, the promises of existing techniques are offset by substantial practical caveats. We preset CrowdCast, a software framework to bridge sophisticated social analysis with the utility of real-time systems. CrowdCast monitors a stream of Twitter activity, leverages large-scale graph analysis, invokes online machine learning, and applies low-complexity graph heuristics to output live prediction of soon-to-occur viral events. While we evaluate our techniques in the context of viral videos, we believe

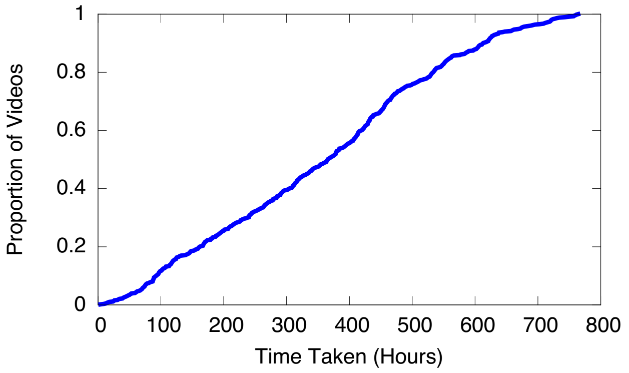

Figure 10: Plot showing distribution of video *time-to-viral* (no 24 hour time constraint; videos not eventually reaching our 500K viewership threshold for virality excluded).

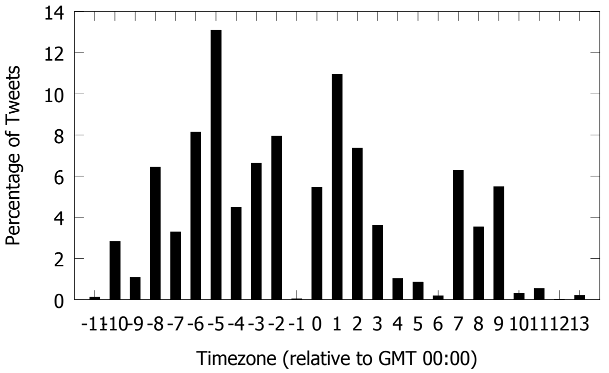

Figure 11: Twitter activity by self-reported timezone (user profile), for reference note: -5 (New York), +1/+2 (Europe).

that CrowdCast's system design can be applied more generally across domains. Thus, CrowdCast leaves open broad avenues for research in integration of complementary analytic techniques and in pursuit of alternative inference goals.

References

Agarwal, S. 2009. Social networks as internet barometers for optimizing content delivery networks. In *ANTS*. IEEE.

Albert, R., and Barabási, A.-L. 2002. Statistical mechanics of complex networks. *Reviews of modern physics* 74(1):47.

Allocca, K. 2012. Kevin allocca: Why videos go viral. http://www.ted.com/talks/kevin_allocca_why_videos_go_viral.html.

Bao, X.; Gowda, M.; Mahajan, R.; and Choudhury, R. R. 2013. The case for psychological computing. In *HotMobile*. ACM.

Berger, J. A., and Milkman, K. L. 2009. What makes online content viral? *Available at SSRN 1528077*.

Broxton, T.; Interian, Y.; Vaver, J.; and Wattenhofer, M. 2010. Catching a viral video. In *ICDMW*. IEEE.

Cha, M.; Haddadi, H.; Benevenuto, F.; and Gummadi, P. K. 2010. Measuring user influence in twitter: The million follower fallacy. *ICWSM* 10:10–17.

Clauset, A.; Shalizi, C. R.; and Newman, M. E. 2009. Power-law distributions in empirical data. *SIAM review* 51(4):661–703.

Dong, A.; Zhang, R.; Kolari, P.; Bai, J.; Diaz, F.; Chang, Y.; Zheng, Z.; and Zha, H. 2010. Time is of the essence: improving recency ranking using twitter data. In *WWW*. ACM.

GNIP. 2014. GNIP: The Social Media API. http://gnip.com/twitter/decahose/.

Goel, S.; Watts, D. J.; and Goldstein, D. G. 2012. The structure of online diffusion networks. In *EC*. ACM.

Guerini, M., and Strapparava, C. 2011. Exploring text virality in social networks. In *ICWSM*.

Hoang, T.-A., and Lim, E.-P. 2012. Virality and susceptibility in information diffusions. In *ICWSM*.

Kwak, H.; Lee, C.; Park, H.; and Moon, S. 2010. What is twitter, a social network or a news media? In *WWW*. ACM.

Lerman, K., and Hogg, T. 2010. Using a model of social dynamics to predict popularity of news. In *WWW*. ACM.

Li, R., et al. 2012. Towards social user profiling: unified and discriminative influence model for inferring home locations. In *SIGKDD*. ACM.

Malewicz, G.; Austern, M. H.; Bik, A. J.; Dehnert, J. C.; Horn, I.; Leiser, N.; and Czajkowski, G. 2010. Pregel: a system for large-scale graph processing. In *SIGMOD*. ACM.

Mathioudakis, M., and Koudas, N. 2010. Twittermonitor: trend detection over the twitter stream. In *SIGMOD*. ACM.

Morstatter, F.; Liu, H.; and Carley, K. M. 2013. Is the sample good enough? comparing data from twitter's streaming api with twitter's firehose.

Sakaki, T.; Okazaki, M.; and Matsuo, Y. 2010. Earthquake shakes twitter users: real-time event detection by social sensors. In *WWW*, 851–860. ACM.

Shamma, D. A.; Yew, J.; Kennedy, L.; and Churchill, E. F. 2011. Viral actions: Predicting video view counts using synchronous sharing behaviors. In *ICWSM*.

Szabo, G., and Huberman, B. A. 2010. Predicting the popularity of online content. *Comm. of the ACM* 53(8):80–88.

Topsy. 2014. Twitter search, monitoring, & analytics. http://topsy.com.

Twitter blog. 2013. Maximum tweer per second. https://blog.twitter.com/2013/new-tweets-per-second-record-and-how.

UnrulyMedia. 2013. Unruly sharerank. http://www.unrulymedia.com/socialvideolab.

Wallsten, K. 2010. "yes we can": How online viewership, blog discussion, campaign statements, and mainstream media coverage produced a viral video phenomenon. *Journal of Information Technology & Politics* 7(2-3):163–181.

Weng, J.; Lim, E.-P.; Jiang, J.; and He, Q. 2010. Twitterrank: finding topic-sensitive influential twitterers. In *WSDM*. ACM.

Zaman, T. R.; Herbrich, R.; Van Gael, J.; and Stern, D. 2010. Predicting information spreading in twitter. In *Workshop on Computational Social Science and the Wisdom of Crowds, NIPS*, volume 104, 17599–601. Citeseer.