What's Happening and What Happened: Searching the Social Web

Omar Alonso, Vasileios Kandylas, Serge-Eric Tremblay, Jake M. Hofman, Siddhartha Sen
Microsoft
omalonso,vakandyl,sergetr,jmh,sidsen@microsoft.com

ABSTRACT

Every day millions of users share links and post comments on different social networks. At scale, this behavior can be very useful for building a new type of search engine that exploits relevant links and their associated metadata in a temporal fashion. Our goal is to find links that are relevant on social networks as a mechanism to discover what people are talking about at a given point in time and make such information searchable and persistent. In other words, a continually updated archive of relevant content that is currently being shared, beyond the obvious trending news of the day. The techniques we use surface new and interesting content by mining social network posts that contain links, constructing diffusion trees from those links, and extracting related entities and other associated metadata. By looking at the size of the trees and their structure in combination with the conversation around each link and related topics, we designed and implemented a search engine that provides relevant fresh content and features a "wayback machine". We demonstrate the effectiveness of our approach by processing a dataset comprising millions of English language tweets generated over a one year period. Finally, we perform an offline evaluation of our techniques and conduct a use case study using an available data set of fake and real news links.

1 INTRODUCTION

With more than 400 million tweets being generated every day, Twitter has fast grown into one of the largest sources of real time information on the Internet. People tweet about a range of topics varying from personal thoughts to their opinion about ongoing events. The ease with which this information can be published and shared has been one of the primary reasons for the success of such microblogging services. On the flip side, the low barrier for information generation has led to the proliferation of uninteresting content and difficulty in finding relevant information. A lot of the generated data is either noise, duplicates, or information which is of low value for a global audience. In addition, the scale of this data makes it hard to not only index, aggregate and search, but also to efficiently mine it to extract interesting and meaningful insights.

Searching for relevant content in social networks is not a solved problem. The temporal signal is important for real-time search [7] and for re-finding posts [16] but not really exploited as a feature. Twitter and Facebook offer trending features that show a term or phrase that represents a very high activity due to the increased rate of people talking about the topic. By selecting a particular trend, relevant posts are returned, ranked by some engagement metric like number of likes, shares per link, retweets, or similar. Trending features are limited to extremely popular items and fail to surface other relevant, but perhaps less globally popular, content. Also, it is not possible to see what was trending in the past.

Tweets can contain links shared by other users in the form of re-tweets or favorites (likes) producing a cascading effect. That is, collaborative content creation and sharing by users in the platform. We can think of this sharing activity as large-scale human computation that aggregates and filters high-quality information. How can we take advantage of these fresh new links and create a new type of search engine? In contrast to current web search engines that are based on link citation for ranking websites, we propose a different approach that relies on a combination of temporal and sharing behavior making relevant information always fresh, while at the same time providing "wayback machine" functionality.

We present two use case scenarios that involve searching for now and then: 1) a user searches for the latest information on a topic and 2) a user searches for what was said in the past for a given topic. The first scenario may look similar to a popular news or trending scenario. This differs from traditional news in that the content is selected by aggregate user behavior instead of editors and includes more than just the most popular content at any given time. Regarding the second scenario, searching in the past is now limited to using re-finding techniques [16], trying the Internet Archive's wayback machine, or examining the notes and references of a Wikipedia page. We archive what the crowd noted as relevant at a particular point in time and make it easily accessible.

The research questions are: (R1) what are the main characteristics of a search engine that focuses on the social web? and (R2) how feasible would be to search on the past using our techniques?

Our work requires the design and implementation of a working system that indexes Twitter data over a period of time. The engineering effort is driven by our research on a number of specific components. That is, identification of high quality links based on a combination of viral and popular metrics—the quality and appeal of these links has been justified by prior work (see below); the extraction of useful metadata from links and topics; the use of trusted users as an endorsement signal; and the ability to create temporal snapshots that are later used for searching in the past. We also provide a user interface based on social cards (links, people, pivots) as an alternative to the common top-10 links provided by traditional search engines.

WebSci'17, June 25-28, 2017, Troy, NY, USA.

978-1-4503-4896-6/17/06…$15.00
DOI: http://dx.doi.org/10.1145/3091478.3091484

2 RELATED WORK

There is a large body of work that looks at the problem of identifying currently popular content [4, 22]. While the details vary across implementations, the basic idea behind all of this work is to look for items that are experiencing a large and sudden increase in popularity—likes, clicks, retweets, etc.—over a relatively short period of time. This tends to surface the most extremely popular content in any given time window, often revealing celebrity stories, top-line news stories featured across major media outlets, or tweets from popular, real-world events [5]. And while such stories are surely of interest to many users, at the same time, popularity-based trending algorithms miss a good deal of interesting and relevant content.

The work we present here leverages past results in [12] to identify trending content based not just on how popular it is, but by how it spreads throughout the population of a social network. In other words, we seek to identify content that looks "viral" based on how it is being shared. Interestingly, as shown in [12], the most viral content at any given time is often distinct from the most popular content being shared online. As we demonstrate here by identifying and exposing viral stories in near real-time, virality-based rankings tend to surface items that are more niche in nature while still of interest to a large audience.

It is worth noting that since our metric for virality is so different from popularity, even complex analytics or machine learning pipelines based on the latter will tend to yield different stories. This is because the "reward signal" [15] that drives these pipelines is fundamentally different. Algorithms for ranking web pages such as PageRank and HITS are based on the (slowly-varying) linkage of pages on the web. Our research ignores the "web graph" completely, looking instead at how links are shared *over time* through a *social network*. Page-ranking algorithms typically do not incorporate social effects, nor do they vary at the timescales of social sharing.

Augmenting search results with social data, known as social annotations, has been an area of recent product and research work by Google, Microsoft and Yahoo as part of their search engines [19], [18], [10], [14]. It is unclear how effective social annotation is for the social web. Other relevant research includes extracting links from tweets for discovering new information [20], presenting distinct search results compared to Google and Bing [1], and for identifying planned social events [21]. Twitter's Earlybird search engine architecture is presented in [7] but does not expand more on how Twitter uses link sharing information for search.

There is new research that taps into the different motivations and dynamics for link sharing content in social networks. From a non-information retrieval perspective, the study of social buttons and other counters as a metric for user engagement has been coined as the 'Like economy' by [11]. A separate study of the value of tweets as anchor text is presented in [17]. A large analysis on URL sharing behavior on Twitter is described in [8]. Looking at user activity as a human computation image classification task in Pinterest is described in [23].

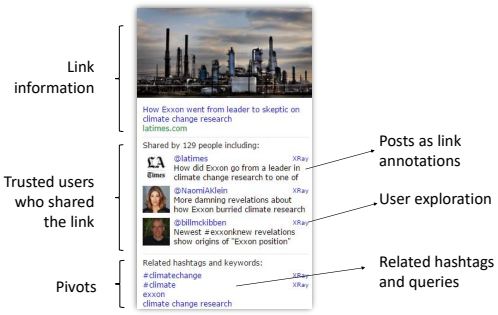

Figure 1: A social card contains three components: link information (e.g., image, title, etc.), a sample of trusted users who have share the link, and a number of pivots.

3 SYSTEM OVERVIEW & PRELIMINARIES

In contrast to traditional web search engines that show the top-10 most relevant links along with query-biased snippets, our system presents search results as social cards (Figure 1). A social card contains an image associated with a link, a page title and short description, the name(s) of high-profile users who have shared the link along with their related posts, "pivots", and the ability to examine user accounts in more detail. Pivots are a subset of the link annotations that allow users to query by related topics or hashtags. We can think of social cards as a variation of social annotations in traditional search engines. However, instead of augmenting links in the search engine result page, the results are more visual and the annotations more prominent, providing context. Related hashtags and queries are selected from a ranked list as explained later.

The functionality works as follows. In the search box, the user can type a query and see a list of social cards that are temporally aware as presented in Figure 2, ranked according to our link relevance technique. By clicking on the calendar and selecting a date, the user can search (or browse) to see what was relevant on that day, thus providing a archival functionality.

Before describing the main techniques, we introduce the following terminology. An *adopter* is a Twitter user who tweeted about a link. A *friend* is someone the Twitter user follows. A *diffusion tree* describes how a link has spread among a set of adopters from one friend to the next. The nodes are the adopters; an edge from user U to user V indicates that V learned about the link from U. A *forest* is the set of all trees for a given link and is used to compute the link popularity and virality as explained below.

4 BACK-END PIPELINE

This section contains a detailed description of our methodology and the back-end pipeline that we use to detect and index relevant links. We discuss the challenges that we faced especially in scaling this pipeline to work on large datasets and provide insights into our design choices. The processing pipeline includes the following basic steps, which are described in detail in the subsequent sections:

(1) Content selection: select tweets that contain links and satisfy simple user, content and time range criteria.

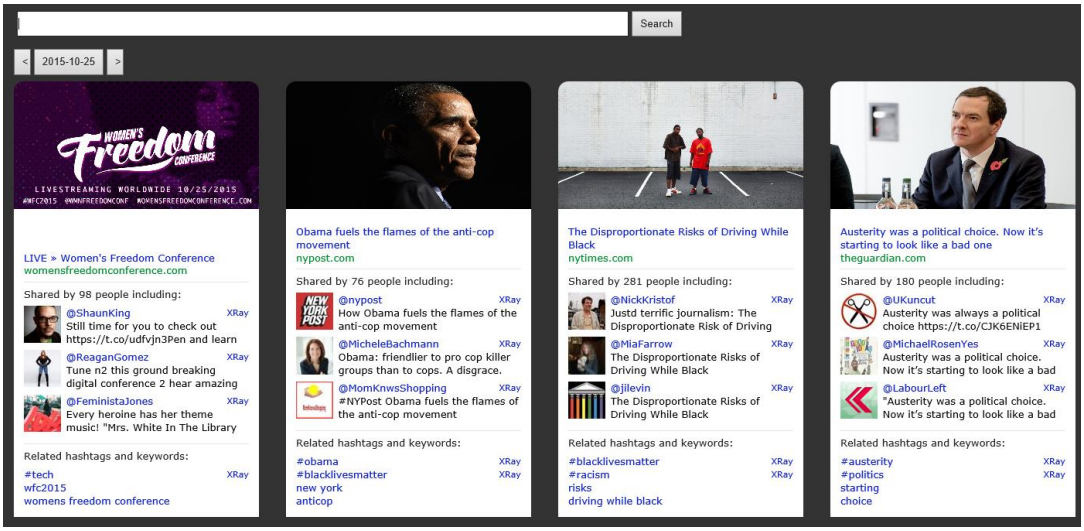

Figure 2: An example of social cards ranked by virality score. Note that user's posts augment the link by providing more context and/or opinions. Using the calendar button on the top left, it is also possible to go back in time and see links that were viral in the past.

(2) User selection: extract and normalize links and select those that have been shared by a minimum number of trusted users.
(3) Link selection: clean-up links, compute link virality and popularity, cluster similar links.
(4) Final cut: apply heuristic criteria to select good quality links.
(5) Annotations: generate metadata for the selected links from the associated tweets.
(6) Indexing and presentation.

4.1 Tweet Selection

As a first step, we select all the tweets that were posted in a chosen time range, including retweets. We have experimented with various ranges from one hour to one month, but for the rest of this paper we use a one day time period. For the tweets in the time range, we keep only those that contain links.

Other simple filters are applied at this stage so that only tweets of good quality are used and spam content is removed. The filtering is done by selecting only tweets whose language is detected as English and have a minimum computed quality score [3].

4.2 Link Pre-Processing and User Filtering

Once an initial set of tweets is selected, the links are extracted and normalized using a set of domain-specific rules to avoid spurious duplicates. For example, query parameters are removed that might make two links appear different when in fact they refer to the same story. Not all query parameters can be removed: the video ID in YouTube links, for example, is critical for distinguishing content. This stage also uses the MinHash clustering algorithm to identify near-duplicate documents at scale [6]. The algorithm works by computing a summary sketch of the content linked to by each URL, specifically the minimum value of a hash function applied to each token on the page. Several of these summary sketches are computed using different hash functions and are combined to form a "shingle print" for the page. As shown in [6], the probability that two pages have the same summary sketch is proportional to the fraction of overlap in their content, and so we can use the shingle print as an identifier for clusters of content later in the pipeline.

Within the time period considered, only one tweet per adopter per link is used; this ensures that the diffusion trees of a link that will be computed later will span disjoint subsets of Twitter users. Associated link information is also extracted from the available metadata (e.g., Twitter cards, Open Graph metadata, etc.), such as the link title, description, image and type. Links are also categorized with an existing system into Open Directory-like categories.

The main filtering that happens in this stage is user-based. First, only links from adopters who have a minimum number of followers are selected. A second filter selects only those links that have been shared by a minimum number of "trusted users". This minimum number depends on the time period under consideration; for a one-day interval, we require at least one trusted user among the set of adopters.

The concept of "trusted users" refers to users who reliably tweet high-quality content and share relevant links [13]. Trusted users are identified by using Twitter's verified users (a manually selected set of users that has been verified by Twitter staff) as a seed set which is expanded by crawling activity on the social network. Starting with the verified users, we detect other users who have had a conversation with them, initiated by the verified user. These users then become trusted and belong to the first "ring" of trusted users.

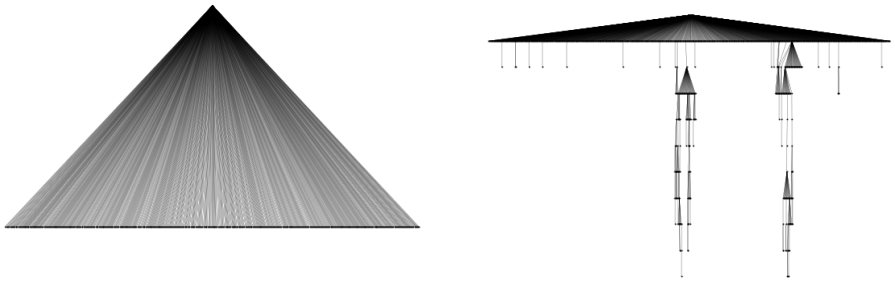

Figure 3: Two diffusion trees for two different links. Each node depicts an account and edges indicate that the child account adopted from its parent. The tree on the left shows "broadcast" diffusion where all adoptions are straight from the source, whereas the tree on the right shows multi-generational diffusion typical of viral content.

This process is repeated several times, gradually increasing the set of trusted users from the thousands of verified to tens of millions of trusted users. This filtering of users is necessary because of the high numbers of fake and spam accounts on Twitter. Because the set of trusted users is in the order of tens of millions, the filtering does not affect the links that are selected later significantly, since a viral link is very likely to have been shared by at least one trusted user. The main result of this filtering is that it limits the influence of fake or spam accounts in the construction of diffusion trees, as explained in the next section.

4.3 Link Selection

This stage focuses on the selection of the links that will be shown in the system. Link selection consists of finding links that are either popular or viral.

To select links we use an algorithm developed by Goel et al. [12]. The algorithm constructs a forest of diffusion trees for each link and computes metrics on this forest to determine the popularity and virality of the links. The algorithm works as follows:

- Aggregate tweets by link.
- For each adopter of a link, crawl his/her friend list to determine the most likely person he/she heard it from. Following Goel et al. [12], we assume this is the friend who most recently tweeted the link unless explicit retweet information is provided. (Other selection rules are possible, e.g. using frequency of communication within a recent time window, but we have yet to explore these alternatives.)
- Use the above to construct the diffusion trees for the link.
- Characterize the size of the trees—a measure of popularity—and the structure of the trees—a measure of virality. Currently, we use the average pairwise distance between nodes of the tree as a measure of virality and the number of nodes as a measure of popularity.

The forest constructed for each link provides a chronological and visual depiction of the diffusion of the story through the Twitter network. Figure 3 shows two example trees for two different links computed by our system, where each node represents an account on Twitter that has posted that link and an edge indicates that the child node adopted from its parent, either through a retweet or repost. The top-most node is the root of the tree, having introduced the story without any of its friends previously doing so. Each generation adopted after its parents, although time is not explicitly shown in the visualization. Both of these trees have approximately the same popularity in that they contain an almost equal number of nodes, but the structure of the two trees differs drastically. In particular, the tree on the left is characteristic of "broadcast" diffusion where a popular outlet with a large direct audience (e.g., the New York Times or CNN) posts content that is adopted by many of its followers but mostly ceases to spread further. The tree on the right, in contrast, depicts "viral" diffusion where the story is continually shared by small audiences over many generations.

In theory there are a number of metrics that can quantify the difference between such diffusion processes. As mentioned above, here we use the Weiner Index proposed by [12], which is simply the average pairwise distance between nodes in the diffusion tree:

$$v(T) = \frac{1}{n(n-1)} \sum_{i=1}^{n} \sum_{\substack{j=1 \\ j \neq i}}^{n} d_{ij}, \tag{1}$$

where d_{ij} indicates the length of the shortest path between nodes i and j. A low score of approximately 2 on this index represents a broadcast event, whereas a higher score represents more viral events.

4.3.1 Crawling the Twitter graph. Constructing the diffusion trees requires knowing the user from whom each adopter heard about a link. We query the Twitter API to obtain the friend list of each adopter, from which we select its parent in the tree (if one exists). Since Twitter imposes a rate limit on their API, issuing these queries on-demand (during the viral link computation) is prohibitively expensive, and thus we run a crawl in the background. The crawl prioritizes both trusted users as well as users with high tweet activity in the preceding month. During viral link detection, queries for the friend list simply return the latest results of the crawl.

4.3.2 Similar link clustering. After the diffusion trees have been computed, similar links are clustered using the link shingle prints computed in a previous stage. Other combinations of features are incorporated, such as a normalized link URL and link title similarity.

ID	Tweet
1	At least 3 dead, 7 injured in #Lafayette shooting, police say gunman 58-year-old 'lone white male' http://t.co/xUcLzsuKgq
2	58-yr-old 'lone white male' killed 2 people & wounded 9 during showing of 'Trainwreck' before turning gun on himself http://t.co/ecBQx9Rbye"
3	Mysterious drifter identified as Louisiana movie theater gunman http://t.co/VtbVW1eZ3J via #Foxnews
4	Police say 'drifter' killed 2, injured 9 in Lafayette, La. movie theater shooting http://t.co/l3DnganpmO http://t.co/YLhj2UP31r

Table 1: The social signature {`movie theater, Lafayette, gunman opens fire`} for the tweets presented in this example.

All normalized links in the same cluster with the same shingle print are deemed to refer to the same story, and so their diffusion trees are merged. Of the links that remain, only those with a minimum number of adopters (across the forest of all trees) are considered for inclusion, and their popularity and virality metrics are computed.

4.3.3 Link selection. At this point the links have already been pre-filtered and the virality and popularity scores have been computed. As found in [12], there is a surprisingly low correlation between popularity and virality, and so we choose links based on both measures. We select popular links that are above a certain threshold, dependent on the time period being used. Separately thresholding by a minimum virality score allows us to identify interesting content that has spread from one person to the next, even if it is not the most popular content being passed around.

Content diversity is often a key requirement of internal customers of our system, and simply presenting the overall most viral (or most popular) links does not guarantee a diverse set of stories. We have found, in general, that the viral structure of stories varies by topic, and thus we leverage the link categories provided during pre-processing to group links by category and select the most viral and popular links across categories.

4.4 Final Filtering

A final filtering stage ensures that the data shown to the user are not only of sufficient quality but are also well-presented to users. To this end we apply a set of empirically derived criteria:

- Links that match their domain are discarded, as these lead to the domain homepage and not to a specific document. Even though they may be popular, such links are not very useful.
- Tweets that are missing important metadata for the shared link (e.g. missing title) are also removed. These tweets are useful when computing the virality and popularity scores of the link, but they are not appropriate for display
- The language of the linked document is detected and non-English links are removed. This could have occurred at an earlier stage, but it is computationally cheaper to reduce the number of links in the previous stages, before scraping and performing language detection for the target page.

The output of this stage is the final set of links used in the system.

4.5 Metadata Generation and Link Annotation

Before any links can be shown to the user, they must be annotated with related information. This information consists of:

- The link title, description and image.
- Selected tweets that mention the link. These are selected after ranking the users by trusted ring number and number of followers. They are also filtered to remove retweets, @user responses, or tweets that contain profanity.
- Hashtags associated with the link, which can be used for re-querying for other links related to the hashtag.
- N-grams associated with the link, computed based on the social signature technique described as follows. Once the high quality links have been identified, we extract the chatter around them in the form of n-grams using a statistical model that identifies salient terms [2]. For computing the signatures, we use the text of the tweets that share the link (after clean-up). Table 1 shows four tweets that link to the same article and the social signature for that link. These can also be used for re-querying for other links related to the n-gram.

4.6 Indexing and Presentation

In the final stage we prepare the data for presentation. The system supports two types of use: (1) show the top viral or top popular links for a given date (overall or per category) as a home page, and (2) allow searching by hashtag or n-gram for relevant viral or popular links.

For the first use case we select the highest quality links relevant to the user's query. We choose links with titles, descriptions and good images (not broken or boilerplate images like the logo of the news site). We also select links that are documents (e.g. no links that are photos, vines, etc.), not redirects, and have "good" file path structure (e.g. file system path depth not too long or too short etc.).

For the second use case we do not apply any of these filters because we assume that a user conducting a search is be looking for an overly complete set of results. In both cases we rank the presented set of links by virality or popularity according to the user's choice. More advanced ranking functions that interleave viral and popular links can also be used.

To support search, we find all the hashtags and n-grams that are mentioned with a link and also the n-grams from the link's social signatures. The top hashtags and n-grams are shown underneath each link as part of the result card and clicking on them conducts a search for related links. We also build an index of links to support searching for any hashtag or n-gram that the user enters in the query box.

5 SYSTEM IMPLEMENTATION

Having presented the logic underlying our pipeline, in this section we provide details of the system that executes it. A distributed

cluster is well suited for processing the firehose of tweets and selecting those that match the filtering criteria. We have used the SCOPE [9] language to implement the pipeline. Implementing the viral link detection as a SCOPE job however requires a computationally expensive three-way join between three relations: (adopter, link), (adopter, friend), and (friend, link), in addition to the link processing described above. Therefore, in order to continuously detect viral links over a sliding time window, we need to reuse as much of the computation as possible between runs. We achieve this by decoupling both the link processing and the Twitter graph crawl from the construction of the diffusion trees. Link processing greatly reduces the number of tweets to consider and produces a dataset that can be repeatedly processed by the tree construction algorithm over different (overlapping) time windows. Running the graph crawl in the background avoids the rate-limited, synchronous calls to the Twitter API in the middle of tree construction. Together, this allows us to run the viral link detection every hour over a trailing 24-hour window.

An unfortunate consequence of rate-limited access to the Twitter API is that we cannot pinpoint our view of the Twitter graph to a narrow time interval. Crawling the entire graph takes months, and our current prioritization of users to crawl is not influenced by the tree construction process. Doing the latter is possible, but it would more than double the tree construction time because we first need to determine which trees are viral before we know which user links to refresh, which in turn requires us to reconstruct the trees. Such on-demand refreshing of user links is more viable during real-time construction of the viral trees (discussed below). In general, although we rely on the Twitter firehose, our current processing is not real-time: we wait for a given hour's data to stabilize, e.g. 30 minutes after the hour ends, before including it in a query. This means that our results can be replicated to some degree by relying on the publicly-accessible Twitter Search and Streaming APIs (https://dev.twitter.com/docs). The key difference is that the firehose gives us access to all the Twitter data, not just a sample of the data or the most recent data matching a search criteria.

Providing real-time detection of viral links is the subject of ongoing and future work. In particular, we have built a pipeline that consumes the Twitter firehose directly and incrementally constructs the diffusion trees of each link. This pipeline supports a sliding window similar to that of the SCOPE job, but the window is advanced at a much finer granularity: ancestors of the tree are constantly being phased out while new descendants are being added. The addition of a potential descendant creates an opportunity to refresh the corresponding user's friend list; whether such API calls can be managed at scale remains to be seen. Moreover, incremental processing restricts the amount of link filtering and pre-processing that can be done a priori, thus imposing different resource requirements and scalability challenges outside the scope of this paper.

6 RESULTS AND EVALUATION

Due to confidentiality we cannot disclose the user engagement, query log, or behavioral data we used in evaluating our system. That said, the core of our research is on techniques for manipulating large data sets efficiently and not so much on the end-user perspective so, instead, we present evaluation results as follows. We first report the daily numbers that our system process in every stage of the data processing pipeline. This gives the reader an idea of the computation that is required to run the system on a daily basis and the impact of each step. We then show an offline relevance evaluation task using a standard crowdsourcing approach. For the "wayback machine" feature, we use the US Elections 2016 event as comparison against a well-documented Wikipedia page. Finally, we demonstrate how our techniques help filter fake links using an available data set.

Phase	Tweets	Links	Users	Hashtags	Ngrams
Content selection	227M	54M	25M	-	-
User selection	90M	18M	17M	-	-
Link selection	-	15K	-	-	-
Final cut	-	1K	-	9K	3K

Table 2: Daily numbers produced by the four stages of the data processing pipeline. The "-" indicates that the value is computed at a later stage or because the value is not used in a later stage.

Please help us evaluate the relevance of an article (or link) and associated information (e.g., image, hashtags, and users). You'll be given a recent article that has been shared by a user in a social network like Twitter along with its related hashtag. Your task is to assess the quality of each item in the 4 question below.

Article: [Image] [Link]
Hashtag: "`#thehashtag`"
Shared by [User]

Q1. Do you think this article would be interesting to at least some users?
[] Yes [] Somewhat [] No
Q2. Do you think the image thumbnail is relevant to the article?
[] Yes [] Somewhat [] No [] Other (broken, too small, low quality)
Q3. Do you think the related hashtag is informative about the article?
[] Yes [] Somewhat [] No
Q4. Do you think the article is shared by a credible user?
[] Yes [] Somewhat [] No

Figure 4: Offline task relevance evaluation template.

6.1 Daily Numbers

For the 24-hour window of tweets that we describe in this paper and after all the filtering we apply, we produce a data set per day consisting of approximately 1,000 links. Table 2 shows more details, including the output of each stage. Some values are missing, either because that value is computed at a later stage (e.g. number of hashtags at the beginning), or because the value is not used in a later stage (e.g. number of tweets in the final stage).

Question	Yes	Somewhat	No
Q1	99%	1%	0%
Q2	80%	10%	8%
Q3	64%	22%	14%
Q4	70%	16%	14%

Table 3: Offline relevance evaluation results.

6.2 Offline Evaluation

We conducted an offline evaluation using an internal crowdsourcing tool[1] for collecting labels. The data set consisted of 100 items selected at random and each item was assessed by 5 human judges with the majority vote treated as final label. Results are presented in Table 3 and the evaluation template is presented in Figure 4. The link relevance results confirm that the link selection works well and the user endorsement numbers are also reliable. The identification of the pivots (i.e., related topics or hashtags) on the other hand is an area that needs improvement. For example, some of the pivots are a bit too generic (e.g., #news) or too brand specific (i.e., #cnn) and they do not seem to provide enough value to the user.

Evaluation results	
R@25	82%
R@25 (filtered)	92%
P@25	86%
R@50	93%
P@50	92%

Table 4: Evaluation metrics for the US Elections. R@25 indicates recall considering the top-25 hashtags comparing to the Wikipedia page. Similarly, P@50 indicates precision considering the top-50 hashtags.

6.3 Wayback Machine

As described earlier, our system contains a wayback machine feature that allows users to go back in time to search and discover. To evaluate the accuracy of this functionality, we extract the top-25 hashtags per day for 2016 (9K in total) and manually annotate the ones that refer to the US elections forming a list of political events in reverse chronological order. We then compare against the Wikipedia US election timeline page that contains an entry per event, used as baseline, and compute precision and recall:

$$\text{Precision (P)} = \frac{\#(\text{relevant entries retrieved})}{\#(\text{retrieved entries})}$$

$$\text{Recall (R)} = \frac{\#(\text{relevant entries retrieved})}{\#(\text{relevant entries})}$$

Overall, our system returns 82% of the entries compared to the baseline. However, if we discount third party events (e.g., Libertarian presidential debate, Libertarian National Convention, Green National Convention, etc.), the recall increases to 92%. If we increase coverage by taking the top-50 hashtags for 2016 (18K in total), the recall is 93%, including the previously discarded third party events. For precision, we use the top-50 hashtags and measure if any of those hashtags match the title of the Wikipedia page entry for the same specific time slot. As example, for 2/18/2016, more than one relevant hashtag are present in top-50 (e.g., SCPrimary, GOPTownHall, DemTownHall) related to the event ("Republican town halls are held in Greenville, South Carolina and Columbia, South Carolina"). The reported precision is 92%. and is computed as the ratio of matched entries over the retrieved entries from Wikipedia page (the ground truth). Table 4 shows a summary of the recall metrics. Due to space constraints, Table 5 presents a condensed version of the US elections timeline along with the top ranked political hashtag for the same date. In summary, the social tagging in aggregate works pretty well for enumerating the main topics for a given day, thus creating a representative crowd-based temporal annotation.

The wayback machine feature allows the user to zoom into a particular day by searching or browsing. Figure 5 shows an example for the day of the US Elections (November 8) and the day after (November 9). For Election day, the results are mostly about people voting and potential irregularities. The hashtags for that day describe the magnitude of the event with references to the main candidates by name or slogan. The day after the election shows a different picture: celebrities announcing moving to Canada (there is also an associated hashtag) and reactions to the new president. Hashtags like #notmypresident and #calexit (California exit from the US) echo the sentiment from a segment of the population after the election results.

6.4 Fake News Case Study

The proliferation of fake news on social networks has been the technology story of the 2016 US elections. A recent analysis by BuzzFeed[2] examined the engagement on Facebook of top fake and real election news stories from 19 major news outlets. The analysis found that the top fake election news stories generated more total engagement than the top election stories. We used the same set of top fake and top real news to study how our system handled this content. Our study consisted of feeding this data set through our data processing pipeline and reporting how and where each of the many steps contributed in filtering out the fake stories. Even though the dataset from BuzzFeed is limited, it can still indicate to what extent our system is susceptible to fake news.

The set of links under consideration was divided in 3 time buckets (February to April, May to July, and August to Election day). Each bucket has 20 fake and 20 real labeled news articles. Whereas the BuzzFeed study was done on links shared, commented or liked on Facebook, our pipeline operates on links shared on Twitter, so there are differences in engagement. Overall, we found that the total number of shares of the real links was higher than the fake links, with fake link shares exhibiting a big increase in the time bucket just before the election (Figure 6). This differs from the BuzzFeed findings, where the engagement of the fake was higher than the real for the August-Election bucket and where the engagement for the real stories was actually decreasing with time.

[1] http://research.microsoft.com/en-us/um/redmond/events/fs2012/presentations/rajesh_patel.pdf

[2] https://www.buzzfeed.com/craigsilverman/viral-fake-election-news-outperformed-real-news-on-facebook

Date	Hashtag	Entry slot from Wikipedia page
1/14	#GOPDebate	Sixth Republican debate is held in North Charleston, South Carolina
1/17	#DemDebate	Fourth Democratic debate is held in Charleston, South Carolina
1/25	#DemTownHall	A Democratic forum, a Town Hall event, is held in Des Moines, Iowa
1/28	#GOPDebate	Seventh Republican debate is held in Des Moines, Iowa
2/4	#DemDebate	Fifth Democratic debate is held in Durham, New Hampshire
2/6	#GOPDebate	Eighth Republican debate is held in Manchester, New Hampshire
2/11	#DemDebate	Sixth Democratic debate is held in Milwaukee, Wisconsin
2/13	#GOPDebate	Ninth Republican debate is held in Charleston, South Carolina
2/25	#GOPDebate	10th Republican debate is held in Houston, Texas
3/1	#SuperTuesday	Super Tuesday
3/3	#GOPDebate	Eleventh Republican debate is held in Detroit, Michigan
3/5	#SuperSaturday	Democratic and Republican primaries/caucuses
3/6	#DemDebate	Seventh Democratic debate is held in Flint, Michigan
3/8	#MichiganPrimary	Democratic and Republican primaries/caucuses
3/9	#DemDebate	Eighth Democratic debate is held in Miami, Florida
3/10	#GOPDebate	Twelfth Republican debate is held in Miami, Florida
4/14	#DemDebate	Ninth Democratic debate is held in Brooklyn, New York
4/26	#SuperTuesday	Democratic and Republican primaries/caucuses
5/20	#PrimaryDay	Third nationally televised Libertarian presidential debate
6/7	#PrimaryDay	Democratic and Republican primaries/caucuses
6/22	#LibTownHall	Libertarian presidential town hall hosted and aired by CNN
7/18-7/21	#RNCinCLE	Republican National Convention is held in Cleveland, Ohio
7/25-7/28	#DemsInPhilly	Democratic National Convention is held in Philadelphia, Pennsylvania
9/26	#debatenight	First presidential debate was held in Hempstead, New York
10/4	#VPDebate	Only vice presidential debate was held in Farmville, Virginia
10/9	#debate	Second presidential debate was held in St. Louis, Missouri.
10/19	#debatenight	The third presidential debate was held in Las Vegas, Nevada
11/8	#ElectionNight	US Election Day
11/19	#ElectoralCollege	The electors of the Electoral College meet and formally vote.

Table 5: Comparison of the top hashtags detected by our techniques versus slots in the Wikipedia page (https://en.wikipedia.org/wiki/United_States_presidential_election,_2016_timeline#2016**) for the main events in the US Elections 2016. The second column shows the highest ranked political hashtag (other semantically similar hashtags are available).**

To study how our different steps contributed to the filtering of the links, we ran our data pipeline on each temporal bucket and looked for the presence of the labeled data set at each step of the pipeline. Figure 7 shows charts comparing the number of real links and fake links over the four steps of data pipeline. Step 0 in Figure 7 corresponds to the input to the pipeline (i.e. the initial data set of 20 fake and 20 real links).

By looking at the first row, for the first bucket, all the fake links (and 9/20 of real) were filtered out. Step 3 (viral link computation) contributes the most in the reduction of the fake links. Almost all of the fake links are not viral enough to be selected, whereas most of the real news links are viral enough and get selected. The second bucket shows similar behavior. In the last bucket, as we get closer to the election, it becomes harder to filter out fake links based on virality alone, since a lot more people are sharing these types of links. Step 3 contributes but not as much as in the previous two cases; specifically it reduces the fake links to half. However, step 4 further reduces the fake links by applying other filtering on the link metadata and user metadata associated with the link.

Our system was not designed with the goal of identifying and removing fake links. Indeed, even though several fake news links are filtered out, some still remain. As shown in in the first row of figure 7, the bulk of the filtering is happening in step 3, the viral score computation. The removed links all had relatively low viral scores. A few fake links actually had high viral scores and were not removed. This is more prevalent in the last time bucket from August to Election day. In that bucket, a higher proportion of fake links had high viral scores. We cannot say with certainty why this happens and the data set is too small to know if this is a common phenomenon with all fake links. It seems plausible however that as the election day approaches, people become more engaged and share political articles more frequently, thus causing high viral scores for them. We can further improve the fake filtering by fine tuning some parameters. We can imagine many ways of optimizing the pipeline at each step by including additional conditions. For example, we tried adding the following:

- Content selection: > 200 shares per link
- User selection: > 1 verified user and > 50 users from trusted user data set (verified + ring 1 + ring 2)
- Link selection: > 2.2 virality score

By using the above optimizations, the final number of fake and real links that remains does not change much from before. It seems there is some correlation between user/content filters and viral filtering, in that very viral links also tend to get shared by verified users and through high-quality tweets. So, for example, we can filter by either high virality or verified users and both will remove several of the same fake links.

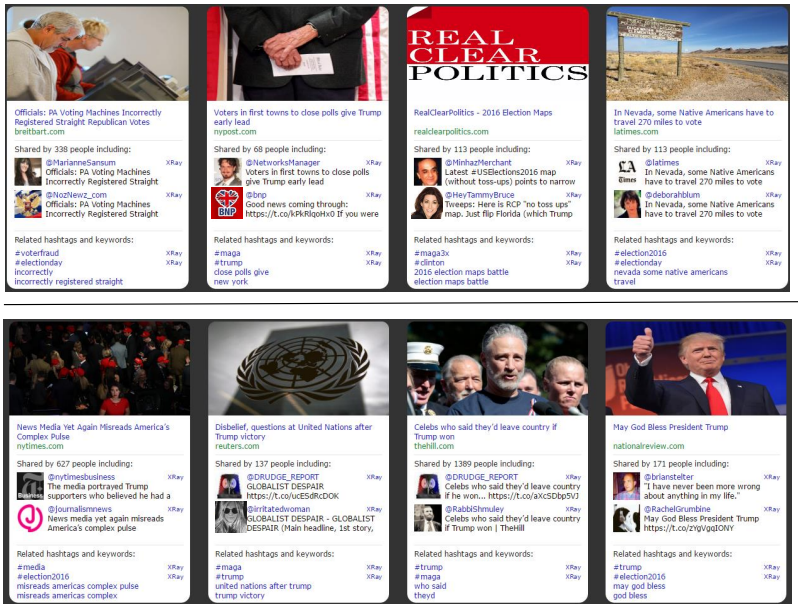

Figure 5: Wayback machine example for the two final days of the 2016 US Election. The screenshot shows search results and top-25 hashtags for both days.

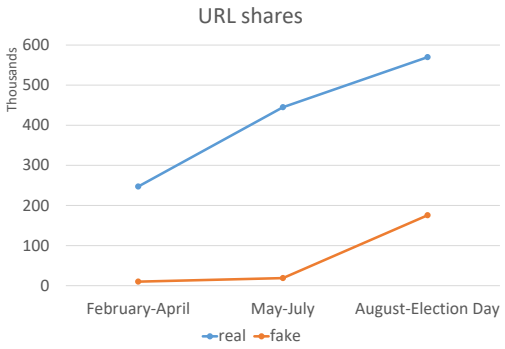

Figure 6: Total shares for the top 20 fake and top 20 real election stories.

The additional filtering conditions however have the effect of moving the filtering of some fake links to earlier steps, so we can improve the filtering and the overall computation cost of the data processing pipeline (there are less links at step 3 for which we need to compute viral scores). Additionally, we can achieve a more gradual change in precision/recall allowing us to stop at an earlier step if needed (so if we stop after step 3 we can get more real links and some fake, but not as many fake as before the optimizations). For example, before the optimizations, most of the fake filtering in the February-April bucket happens in Step 3; before and after that step there is little change on the number of links. After the optimizations (charts in the second row), each step contributes its share of filtering as shown in the figure. For completeness, we also present the charts for the other two time buckets.

Even though our techniques were not designed with the explicit goal of detecting fake links, the filtering, selection and diffusion tree computation are shown to be effective in reducing fake links.

7 CONCLUSION AND FUTURE WORK

Past research in the field of social data mining, especially on the Twitter network, has focused on extracting sentiments, trending topics, communities of similar users and summaries from tweets. Much of this work is based on formulating and evaluating sophisticated algorithms using manually selected datasets that are either small or biased towards noise-free content. Such assumptions about the underlying data makes it hard to deploy these techniques in real-world systems, where efficiency in data processing and scalable implementations are as important as the effectiveness of the system. Our approach consists on techniques that can process large scale data sets and can be implemented in a distributed infrastructure.

Here we emphasized a working system that can ingest and archive data over time, tested with a real-world data set, with the goal of prototyping a new type of search engine. Our findings suggest that mining relevant links shared by trusted users is not only a reliable mechanism for selecting high quality content, but it can also be adopted for ranking content. By applying a number of filters and viral detection techniques, we can identify high quality links discovered within a certain period of time. This temporal aspect helps not only in providing relevant and fresh content, but also with the ability to look back in time.

The content selected by users reflects a collective interest that is very rich when exploring it as an archive. The US 2016 election

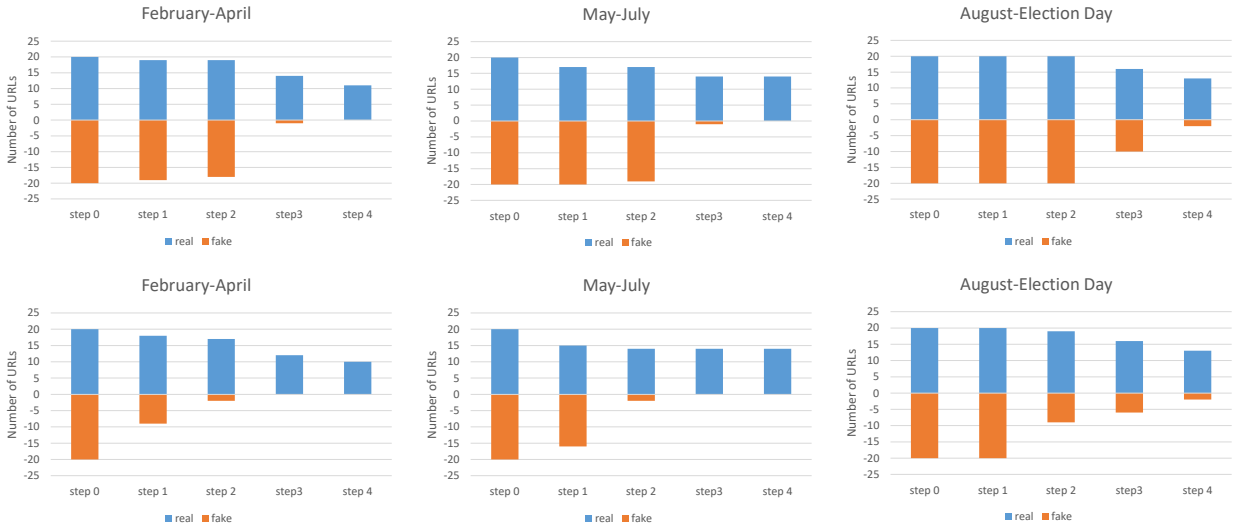

Figure 7: Fake news study on Twitter. The first row shows three charts each comparing the number of real links (colored in blue, positive y-axis) vs. fake links (colored in orange, negative y-axis) over the four steps of the data pipeline (x-axis; step 0 in the the charts means the starting data set). Second row shows the results once we apply optimizations that can help improve the filtering and the computation cost of the processing pipeline.

example shows not only that the system collects the proper data for reconstructing a story, but also that such data mirrors real events. With the wayback machine we can retrieve content and associated chatter that was relevant for a specific period in time.

We provide an extensive description of the techniques and algorithms used along with data points and other data management details so our findings can be replicated by other researchers using Twitter or similar social networks. Unfortunately, business reasons do not allow us to release the source code. As part of our evaluation results, we also demonstrate that the proposed data processing pipeline can be used to identify and filter fake links. While fake link detection was not the goal of our work, the results show that aggressive filtering should be an essential piece of any social web solution that strives for relevant content and not an afterthought like in current systems.

Future work includes personalization of the search results, organization of the links by categories, improving the quality of the pivots, better archiving, and enhancing the diversity of the links.

REFERENCES

[1] Rakesh Agrawal, Behzad Golshan, and Evangelos E. Papalexakis. 2015. Whither Social Networks for Web Search?. In *KDD*. 1661–1670.

[2] Omar Alonso, Sushma Bannur, Kartikay Khandelwal, and Shankar Kalyanaraman. 2015. The World Conversation: Web Page Metadata Generation From Social Sources. In *WWW*. 385–395.

[3] Omar Alonso, Catherine C. Marshall, and Marc Najork. 2013. Are Some Tweets More Interesting Than Others? #HardQuestion. In *HCIR '13*. 2:1–2:10.

[4] Farzindar Atefeh and Wael Khreich. 2015. A survey of techniques for event detection in twitter. *Computational Intelligence* 31, 1 (2015), 132–164.

[5] Hila Becker, Mor Naaman, and Luis Gravano. 2011. Beyond Trending Topics: Real-World Event Identification on Twitter. *ICWSM* 11 (2011), 438–441.

[6] Andrei Z Broder. 1997. On the resemblance and containment of documents. In *Compression and Complexity of Sequences 1997. Proceedings*. IEEE, 21–29.

[7] Michael Busch, Krishna Gade, Brian Larson, Patrick Lok, Samuel Luckenbill, and Jimmy J. Lin. 2012. Earlybird: Real-Time Search at Twitter. In *ICDE*. 1360–1369.

[8] Cheng Cao, James Caverlee, Kyumin Lee, Hancheng Ge, and Jin-Wook Chung. 2015. Organic or Organized?: Exploring URL Sharing Behavior. In *CIKM*. 513–522.

[9] Ronnie Chaiken, Bob Jenkins, Per-Åke Larson, Bill Ramsey, Darren Shakib, Simon Weaver, and Jingren Zhou. 2008. SCOPE: easy and efficient parallel processing of massive data sets. *PVLDB* 1, 2 (2008), 1265–1276.

[10] Jennifer Fernquist and Ed H. Chi. 2013. Perception and understanding of social annotations in web search. In *WWW*. 403–412.

[11] Carolin Gerlitz and Anne Helmond. 2013. The like economy: Social buttons and the data-intensive web. *New Media & Society* 15, 8 (2013), 1348–1365.

[12] Sharad Goel, Ashton Anderson, Jake M. Hofman, and Duncan J. Watts. 2016. The Structural Virality of Online Diffusion. *Management Science* 62, 1 (2016), 180–196.

[13] Martin Hentschel, Omar Alonso, Scott Counts, and Vasileios Kandylas. 2014. Finding Users we Trust: Scaling up Verified Twitter Users Using their Communication Patterns. In *ICWSM*.

[14] Vasileios Kandylas and Ali Dasdan. 2010. The Utility of Tweeted URLs for Web Search. In *WWW*. 1127–1128.

[15] John Langford and Tong Zhang. 2007. The Epoch-Greedy Algorithm for Contextual Multi-armed Bandits. In *NIPS*.

[16] Florian Meier and David Elsweiler. 2016. Going back in Time: An Investigation of Social Media Re-finding. In *SIGIR*. 355–364.

[17] Gilad Mishne and Jimmy J. Lin. 2012. Twanchor text: a preliminary study of the value of tweets as anchor text. In *SIGIR*. 1159–1160.

[18] Aditi S. Muralidharan, Zoltán Gyöngyi, and Ed Chi. 2012. Social annotations in web search. In *CHI*. 1085–1094.

[19] Patrick Pantel, Michael Gamon, Omar Alonso, and Kevin Haas. 2012. Social annotations: utility and prediction modeling. In *SIGIR*. 285–294.

[20] Tom Rowlands, David Hawking, and Ramesh Sankaranarayana. 2010. New-web search with microblog annotations. In *WWW*. 1293–1296.

[21] Yu Wang, David Fink, and Eugene Agichtein. 2015. SEEFT: Planned Social Event Discovery and Attribute Extraction by Fusing Twitter and Web Content. In *ICWSM*. 483–492.

[22] Jianshu Weng and Bu-Sung Lee. 2011. Event Detection in Twitter. *ICWSM* 11 (2011), 401–408.

[23] Changtao Zhong, Dmytro Karamshuk, and Nishanth Sastry. 2015. Predicting Pinterest: Automating a Distributed Human Computation. In *WWW*. 1417–1426.