Campaign Extraction from Social Media

KYUMIN LEE, JAMES CAVERLEE, and ZHIYUAN CHENG, Texas A&M University
DANIEL Z. SUI, Ohio State University

In this manuscript, we study the problem of detecting coordinated free text campaigns in large-scale social media. These campaigns—ranging from coordinated spam messages to promotional and advertising campaigns to political astro-turfing—are growing in significance and reach with the commensurate rise in massive-scale social systems. Specifically, we propose and evaluate a content-driven framework for effectively linking free text posts with common "talking points" and extracting campaigns from large-scale social media. Three of the salient features of the campaign extraction framework are: (i) first, we investigate graph mining techniques for isolating coherent campaigns from large message-based graphs; (ii) second, we conduct a comprehensive comparative study of text-based message correlation in message and user levels; and (iii) finally, we analyze temporal behaviors of various campaign types. Through an experimental study over millions of Twitter messages we identify five major types of campaigns—namely Spam, Promotion, Template, News, and Celebrity campaigns—and we show how these campaigns may be extracted with high precision and recall.

Categories and Subject Descriptors: H.3.5 [**Online Information Services**]: Web-Based Services; J.4 [**Computer Applications**]: Social and Behavioral Sciences

General Terms: Algorithms, Design, Experimentation

Additional Key Words and Phrases: Social media, campaign detection

ACM Reference Format:
Lee, K., Caverlee, J., Cheng, Z., and Sui, D. Z. 2013. Campaign extraction from social media. ACM Trans. Intell. Syst. Technol. 5, 1, Article 9 (December 2013), 28 pages.
DOI: http://dx.doi.org/10.1145/2542182.2542191

1. INTRODUCTION

Social media has become very popular in recent years, leading to new opportunities for global-scale user engagement, sharing, and interaction. Many users of social media organically engage with social media to share opinions and interact with friends; on the other hand, social media is a prime target for strategic influence. For example, there is widespread anecdotal evidence of "astro-turfing" campaigns [Films 2011], in which political operatives insert memes such as a phrase into sites like Twitter and Facebook in an effort to influence discourse about particular political candidates and topics. In addition, there are large campaigns of coordinated spam messages in social media [Gao et al. 2010], templated messages (e.g., auto-posted messages to social media sites from third-party applications announcing a user action, like joining a game or viewing a video), high-volume time-synchronized messages (e.g., many users may

An early version of this manuscript appeared in the 2011 ACM Proceedings of the CIKM Conference [Lee et al. 2011a].
Author's addresses: K. Lee (corresponding author), J. Caverlee, and Z. Cheng, Department of Computer Science and Engineering, Texas A&M University, College Station, TX; email: kyumin@cse.tamu.edu; D. Z. Sui, Department of Geography, Ohio State University, Columbus, OH.

DOI: http://dx.doi.org/10.1145/2542182.2542191

repost news headlines to social media sites in a flurry after the news has been initially reported), and so on. In the case of spam and promotion campaigns, the relative openness of many social media sites (typically requiring only a valid email address to register) suggests coordinated campaigns could be a low-cost approach for strategically influencing participants.

And in a sinister direction, there is growing evidence that tightly organized strategic campaigns are growing in significance [Motoyama et al. 2011; Wang et al. 2012]. One example is the development of sites like SubvertAndProfit (www.subvertandprofit.com), which claims to have access to "25,000 users who earn money by viewing, voting, fanning, rating, or posting assigned tasks" across social media sites. Related services can be found at fansandinvites.com, socioniks.com, and usocial.net. Even within the great firewalls of China, we have witnessed the emergence of the so-called "Wang Luo Shui Jun" or "Online Water Army" (e.g., http://shuijunwang.com). According to a recent CCTV report [CCTV 2010], online mercenaries in China help their customers by: (i) promoting a specific product, company, person or message; (ii) smearing the competitor or adversary or competitors' products or services; or (iii) deleting unfavorable posts or news articles. Most online "mercenaries" work part-time and are paid around 5 US cents per action.

User-driven campaigns—often linked by common "talking points"—appear to be growing in significance and reach with the commensurate rise of massive-scale social systems. However, there has been little research in detecting these campaigns "in the wild". While there has been some progress in detecting isolated instances of long-form fake reviews (e.g., to promote books on Amazon), of URL-based spam in social media, and in manipulating recommender systems [Gao et al. 2010; Hurley et al. 2007; Lam and Riedl 2004; Lim et al. 2010; Mehta 2007; Mehta et al. 2007; O'mahony et al. 2002; Ray and Mahanti 2009; Su et al. 2005; Wu et al. 2010], there is a significant need for new methods to support Web-scale detection of campaigns in social media.

Hence, we focus in this manuscript on detecting one particular kind of coordinated campaign, namely those that rely on "free text" posts, like those found on blogs, comments, forum postings, and short status updates (like on Twitter and Facebook). For our purposes, a campaign is a collection of users and their posts bound together by some common objective, for example, promoting a product, criticizing a politician, or inserting disinformation into an online discussion. Our goal is to link messages with common "talking points" and then extract multimessage campaigns from large-scale social media. Detecting these campaigns is especially challenging considering the size of popular social media sites like Facebook and Twitter with hundreds of millions of unique users and the inherent lack of context in short posts.

Concretely, we propose and evaluate a content-based approach for identifying campaigns from the massive scale of real-time social systems. The content-driven framework is designed to effectively link free text posts with common "talking points" and then extract campaigns from large-scale social media. Note that text posts containing common "talking points" means the contents of the posts are similar or the same. We find that over millions of Twitter messages, the proposed framework can identify hundreds of coordinated campaigns, ranging in size up to several hundred messages per campaign. The campaigns themselves range from innocuous celebrity support (e.g., fans retweeting a celebrity's messages) to aggressive spam and promotion campaigns (in which handfuls of participants post hundreds of messages with malicious URLs). Through an experimental study over millions of Twitter messages we identify five major types of campaigns—namely Spam, Promotion, Template, News, and Celebrity campaigns—and we show how these campaigns may be extracted with high precision and recall. We also find that the less organic campaigns (e.g., Spam and Promotion) tend to be driven by a higher ratio of messages to participants (corresponding to a

handful of accounts "pumping" messages into the system). Based on this observation, we propose and evaluate a user-centric campaign detection approach. By aggregating the messages posted by a single user, we find that the method can successfully discover cross-user correlations not captured at the individual message level (e.g., for two users posting a sequence of correlated messages), resulting in more robust campaign detection. In addition, we analyze each campaign type's temporal behavior to see the possibility to automatically determine a campaign's campaign type.

The rest of the manuscript is organized as follows. Section 2 highlights relevant work in spam and campaign detection, information credibility, and persuasion. Then in Section 3, we formalize the problem statement and present the datasets and evaluation metrics. Section 4 presents the proposed content-driven campaign detection approach in detail. In Section 5, we experimentally test the content-driven campaign detection in message and user levels, and analyze temporal behaviors of several campaign types that we found in the datasets. We conclude in Section 6 with some final thoughts.

2. RELATED WORK

The prior work relevant to this manuscript covers spam and campaign detection, information credibility, trust, and persuasion. We summarize several related efforts in this section.

Researchers have proposed several approaches to detect spam in emails and Web pages. Representative solutions include link analysis algorithms for link farms [Becchetti et al. 2008; Benczur et al. 2006; Gyöngyi et al. 2006; Wu and Davison 2005], data compression and machine learning algorithms for email spam [Bratko et al. 2006; Sahami et al. 1998; Yoshida et al. 2004], and machine learning algorithms for spam Web pages [Fetterly et al. 2004; Ntoulas et al. 2006].

As social networking sites become more popular, researchers have studied the categorization of spam content, analyzed spammers' behaviors, and proposed possible solutions. Grier et al. [2010] declared that blacklists are too slow in identifying incoming real-time threats on Twitter, allowing more than 90% of visitors to view a malicious Web page before it becomes blacklisted. Koutrika et al. [2008] proposed a framework to detect spam in social tagging systems, in which they built user models such as good user model and bad user model, and showed that tagging systems can be spammed by bad users. Machine learning algorithms have been used to detect video content spammers and promoters by Benevenuto et al. [2009]. Researchers also studied trending topic (hashtag) spam problems on Twitter and proposed content-based and machine-learning-based approaches to solve those problems [Irani et al. 2010; Benevenuto et al. 2010]. Social honeypots on Twitter and MySpace were deployed to collect spammers' information and to analyze their behaviors, and machine learning algorithms were used to detect spammers [Lee et al. 2010, 2011b].

In addition, researchers have begun studying group spammers and their tactics. Mukherjee et al. [2011] proposed an approach which consists of frequent pattern mining techniques, computing spam indicator value, and using SVM Rank to rank possible spam groups, to detect group review spammers. Gao et al. [2010] studied spam behavior on Facebook; their approach finds coordinated spam messages that use the same malicious URL. The Truthy system [Ratkiewicz et al. 2011] detects astro-turf political campaigns on Twitter. They first define memes consisting of hashtags, mentions, URLs, and phrases. If Twitter users post tweets or retweet a message containing one of these memes, they assumed that the users participate in a coordinated effort. The researchers detect these political campaigns as follows: (1) first identify memes; (2) compute features (network features, sentiment scores, the number of "truthy" button clicks, etc.); (3) train a classifier with binary class (either a legitimate campaign

or "truthy" campaign', i.e., astro-turf political campaign); and (4) predict unlabeled memes' class. Their approach achieved high accuracy.

Other researchers have studied information credibility, trust, and persuasion techniques in social media. Castillo et al. [2011] studied information credibility, especially in newsworthy topics in Twitter, and built a classifier to determine whether messages associated with a topic are credible or not. Given a set of confirmed trustworthy and untrustworthy nodes such as Web pages on the Web or users in social systems as inputs, researchers have studied trust propagation methods in local computation and global computation based on the taxonomy presented by Ziegler and Lausen [2005]. In a local trust computation [Levien and Aiken 1998; Mui et al. 2002; Ziegler and Lausen 2005], each node has multiple trust values measured by a single user's perspective while in a global trust computation, each node has a single trust value measured by the perspective of the whole network [Gyöngyi et al. 2004; Caverlee et al. 2008; 2010]. Young et al. [2011] present their persuasion model and the hostage negotiation corpus (a microtext corpus) [Gilbert and Henry 2010] which contains 12% persuasive utterances. Their persuasion model, based on Cialdini's persuasion model [Cialdini 2007], focuses on reciprocity, commitment and consistency, scarcity, liking, authority, and social proof. Based on the persuasion model and using the corpus, they build classifiers to detect persuasion automatically.

In the literature, researchers have proposed solutions to detect spammers or measure information credibility in both email and social systems. In contrast, our focus is on identifying campaigns from massive scale of real-time social systems, understanding what types of campaigns exist in the social systems, and analyzing temporal behaviors of various campaign types.

3. CONTENT-DRIVEN CAMPAIGN DETECTION

In this section, we describe the problem of campaign detection in social media, introduce the data, and outline the metrics for measuring effective campaign detection.

3.1. Problem Statement

We consider a collection of n participants across social media sites $U = \{u_1, u_2, \ldots, u_n\}$, where each participant u_i may post a time-ordered list of k messages $M_{u_i} = \{m_{i1}, m_{i2}, \ldots, m_{ik}\}$. Our hypothesis is that among these messages and users, there may exist *coordinated campaigns*.

Given the set of users U, a *campaign* M_c can be defined as a collection of messages and the users who posted the messages: $M_c = \{m_{ij}, u_i | u_i \in U \cap m_{ij} \in M_{u_i} \cap theme(m_{ij}) \in t_k\}$ such that the campaign messages belong to a coherent theme t_k. Themes are human-defined logical assignments to messages and application dependent. For example, in the context of spam detection, a campaign may be defined as a collection of messages with a common target product (e.g., Viagra). In the context of astro-turf, a campaign may be defined as a collection of messages promoting a particular viewpoint (e.g., the veracity of climate change). Additionally, depending on the context, a message may belong to one or multiple themes. For the purposes of this manuscript and to focus our scope of inquiry, we consider as a theme all messages sharing similar "talking points" as determined by a set of human judges.

3.2. Data

To evaluate the quality of a campaign detection approach, we would ideally have access to a large-scale "gold set" of known campaigns in social media. While researchers have published benchmarks for spam Web pages [Webb et al. 2006; TREC 2007], ad hoc text retrieval [Voorhees and Dang 2005], and other types of applications [TREC 2004; Cheng et al. 2010; Lee et al. 2011b], we are not aware of any standard social media

campaign dataset. Hence, we take in this manuscript a twofold approach for message-level campaign detection: (i) a small-scale validation over hand-labeled data; and (ii) a large-scale validation over 1.5 million Twitter messages for which ground truth is not known.

CD_{Small}. First, we sample a small collection of messages (1,912) posted to Twitter in October 2010. Over this small *campaign dataset* (CD_{Small}), two judges labeled all pairs of the 1,912 tweets as sharing similar "talking points" or not, finding 298 pairs of messages sharing similar "talking points". Based on these initial labels, the judges considered all combinations of messages that may form campaigns consisting of four messages or more, and found 11 campaigns ranging in size from four messages to eight messages. While small in size, this hand-labeled dataset allows us to evaluate the precision and recall of several campaign detection methods.

CD_{Large}. Second, we supplement the small dataset with a large collection of messages (1.5 million) posted to Twitter between October 1 and October 7, 2010. We sampled these messages using Twitter's streaming API, resulting in a representative random sample of Twitter messages. Over this large *campaign dataset* (CD_{Large}), we can test the precision of the campaign detection methods and investigate the types of campaigns that are prevalent in the wild. Since we do not have ground-truth knowledge of all campaigns in this dataset, our analysis will focus on the campaigns detected for which we can hand-label as actual campaigns or not.

Additionally, we also consider a *user-based* dataset, in which all of the messages associated with a single user are aggregated.

CD_{User}. Since the datasets CD_{Small} and CD_{Large} are collected by a random sample method from Twitter (meaning most users were represented by only one or two messages), we collected a user-focused dataset from Twitter consisting of 90,046 user profiles with at least 20 English-language messages, resulting in 1.8 million total messages.

3.3. Metrics

To measure the effectiveness of a campaign detection method, we use variations of average precision, average recall, and the average F_1 measure. The Average Precision (AP) for a campaign detection method is defined as

$$AP = \frac{1}{n} \sum_{i=1}^{n} \frac{\max CommonMessages(PC_i, TCs)}{|PC_i|},$$

where n is the total number of predicted campaigns by the campaign detection method, PC is a predicted campaign, and TC is an actual (true) campaign. $MaxCommonMessage$ function returns the maximum of the number of common messages in both the predicted campaign i (PC_i) and each of the actual (true) campaigns (TCs). For example, suppose a campaign detection method identifies a three-message campaign: $\{m_1, m_{10}, m_{30}\}$. Suppose there are two actual campaigns with at least one message in common: $\{m_{30}, m_{38}, m_{40}\}$ and $\{m_1, m_{10}, m_{35}, m_{50}, m_{61}\}$. Then the precision is $max(2, 1)/3 = 2/3$. In the aggregate, this individual precision will be averaged with all n predicted campaigns.

Similarly, we can define the Average Recall (AR) as

$$AR = \frac{1}{n} \sum_{i=1}^{n} \frac{\max Common\ Messages(PC_i, TCs)}{|TC_j|},$$

where n is the number of the predicted campaigns, and TC_j is a true campaign which has the largest common messages with the predicted campaign i (PC_i). Continuing the example from before, the recall would be $max(2, 1)/5 = 2/5$.

Finally, we can combine precision and recall as the Average F_1 measure (AF).

$$AF_1 = \frac{2 * AP * AR}{AP + AR}$$

An effective campaign detection approach should identify predicted campaigns that are composed primarily of a single actual campaign (i.e., have high precision) and that contain most of the messages that actually belong to the campaign (i.e., have high recall). A method that has high precision but low recall will result in only partial coverage of all campaigns available (which could be especially disastrous in the case of spam or promotional campaigns that should be filtered). A method that has low precision but high recall may identify nearly all messages that belong to campaigns but at the risk of mislabeling noncampaign messages (resulting in false positives, which could correspond to mislabeled legitimate messages as belonging to spam campaigns).

4. CAMPAIGN DETECTION: FRAMEWORK AND METHODS

In this section, we describe the high-level approach for extracting campaigns from social media, present the message- and user-level campaign detection in detail, and discuss a MapReduce-based implementation for efficient campaign detection.

4.1. Overall Approach

To detect coordinated campaigns, we explore in this manuscript several content-based approaches for identifying campaigns. Our goal is to find methods that can balance both precision and recall for effective campaign detection. In particular, we propose a content-driven campaign detection approach that views social media from two perspectives.

Message Level. In the first perspective, we view each message as a potential member of a campaign. Our goal is to identify a campaign as a collection of its constituent messages. In this way, we can identify related messages as shown in Figure 1. Given a set of messages (6 messages in the example), our goal is to build a message graph in which a node represents a message and if the similarity of a pair of messages is larger than a threshold (τ) then an edge exists between the pair of messages. Note that the similarity of a pair of messages means how much the pair of messages is similar in terms of number of common tokens, and a token can be defined as a n-gram word or n-gram character depending on a message similarity identification algorithm. In this way, we can identity significant subgraphs as campaigns, which should reflect multiple messages sharing the same key "talking points".

User Level. In the second perspective, rather than viewing the message as the core component of a campaign, we view *each user* as a potential member of a campaign. In this way, a campaign is composed of constituent users. This second perspective may be more reasonable in the case of campaigns that span multiple messages posted by a single user, or in the case of campaigns in which evidence of the campaign is clear at user level but perhaps not at the individual message level (say, in cases of 3 spam accounts that post similar messages in the aggregate, although no two individual messages may share the same talking points). For this perspective, we construct a graph, but where nodes represent users and their aggregated messages. Edges exist between users based on some overall measure of their similarity.

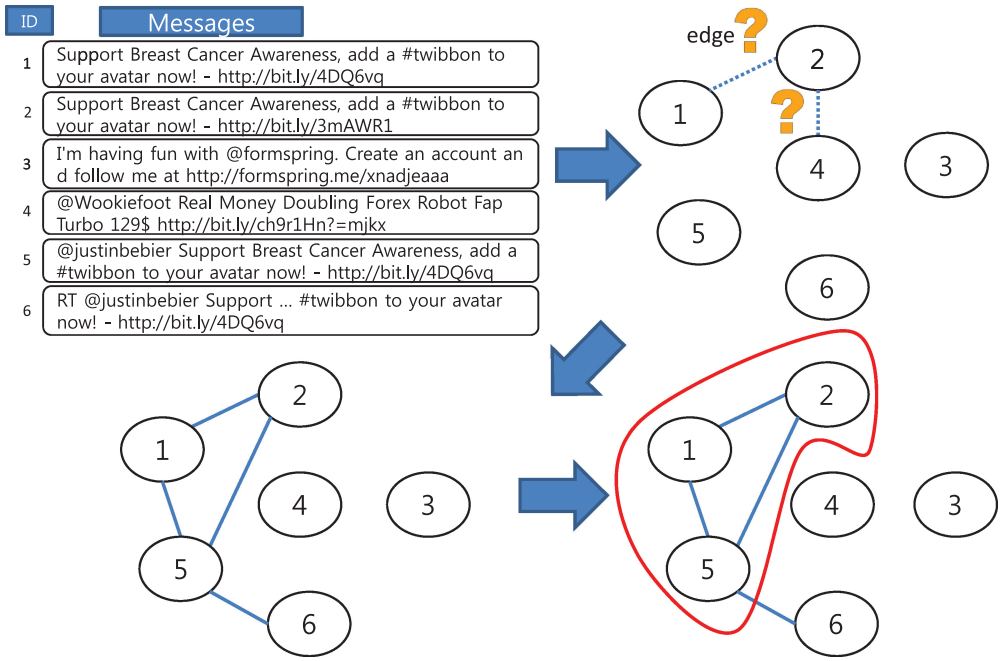


Fig. 1. Overall approach showing how to identify campaigns given a list of messages.

In the following, we detail these two approaches—at message level and at user level—in great detail.

4.2. Message-Level Campaign Detection

For the task of message-level campaign detection, we consider a graph-based framework, where we model messages in social media as a *message graph*. Each node in the message graph corresponds to a message; edges correspond to some reasonable notion of content-based correlation between messages, corresponding to pairs of messages with similar "talking points." Formally, we have the following definition.

Definition 1 (*Message Graph*). A message graph is a graph $G = (V, E)$ where every message in M corresponds to a vertex m_{ix} in the vertex set V. An edge $(m_{ix}, m_{jy}) \in E$ exists for every pair of messages (m_{ix}, m_{jy}) where $corr(m_{ix}, m_{jy}) > \tau$, for a measure of correlation and some parameter τ.

A message graph which links unrelated messages will necessarily result in poor campaign detection (by introducing spurious links). Traditional information retrieval approaches for document similarity (e.g., cosine similarity [Manning et al. 2008], KL-divergence [Manning and Schütze 1999]) as well as efficient near-duplicate detection methods (e.g., Shingling [Broder et al. 1997], I-Match [Chowdhury et al. 2002], and SpotSigs [Theobald et al. 2008]) have typically not been optimized for the kind of short posts of highly variable quality common in many social media sites (including Facebook and Twitter). Concretely we consider six approaches for measuring whether messages share similar "talking points"

—*Unigram Overlap.* The baseline unigram approach considered two messages to be correlated if they have higher Jaccard similarity than a threshold after we extract unigrams from each message and compute Jaccard similarity. The Jaccard coefficient

between the unigrams of each pair of messages A and B is used to measure the similarity of the pair of messages.

$$Jaccard(A, B) = \frac{|A \cap B|}{|A \cup B|} \leq \frac{min(|A|, |B|)}{max(|A|, |B|)}$$

—*Edit Distance.* An alternative is to consider the edit distance between two messages, that is, two messages are correlated if the number of edits to transform one message into the other is less than some threshold value. Concretely, we adopt the Levenshtein distance as a metric for measuring the amount of difference between two messages [Levenshtein 1966]. The distance is the minimum number of edits required in order to transform one message into the other.

—*Euclidean Distance.* Another similarity metric is Euclidean distance which is the length of the line segment connecting two vectors (two messages in this context). First convert messages to vectors in the vector space model and then compute their distance. The smaller the Euclidean distance between two messages, the more similar they are.

—*Shingling.* As an exemplar of near-duplicate detection, Broder's Shingling algorithm [Broder et al. 1997] views a document d as a sequence of words $w_1\ w_2\ w_3 \ldots w_n$, where n is the number of words in d. It extracts unique k-grams $\{g_1, g_2, \ldots, g_m\}$, such that m is the number of unique k-grams. For easy processing and reduction of storage usage, each g_i is encoded by 64-bit Rabin fingerprints F. The encoded value is called a shingle. Now, d's shingles $S = \{s_1, s_2, s_3, \ldots s_m\}$, such that s_i is a shingle (i.e., a signature) and $s_i = F(g_i)$. The Jaccard coefficient between the shingles of each pair of documents A and B is used to measure the similarity of the pair of documents.
If the similarity score of a pair of documents (messages) is higher than a threshold, they will be considered as near duplicates (and hence, correlated messages for our purposes).

—*I-Match.* In contrast to Shingling, the I-Match [Chowdhury et al. 2002] approach explicitly leverages the relative frequency of terms across messages. First, it defines an I-Match lexicon L based on a message frequency of each term in a collection of documents (i.e., Twitter messages). Usually, L consists of a bag of words (i.e., terms or unigrams) which have mid-idf values in the collection. I-Match extracts unigrams U from a document d and only uses some unigrams P, which have mid-idf values in the collection (i.e., $P = L \cap U$). The idea behind this approach is that infrequent and too frequent terms are not helpful to detect near-duplicate documents. Then, I-Match sorts P and concatenates it in order to make a single string, which is then encoded to a single hash value h by SHA-1; in our case, pairs of messages with identical hash values shall be considered correlated messages.

—*SpotSigs.* The final approach we consider is SpotSigs [Theobald et al. 2008], which observes that noisy content, such as navigational banners and advertisements in Web pages, may result in poor performance of traditional Shingling-based methods. By observing that stop-words rarely occur in the noisy content, SpotSigs scans a document to find stop-words as antecedents (anchors), and extracts special k-grams called "spot signatures", one of which consists of an antecedent and a k-gram after the antecedent, excluding stop-words. A hash function is applied to detect identical duplicates.

It is of course an open question how well each of these methods performs toward the ultimate goal of identifying campaigns in social media. Hence, we shall investigate experimentally in Section 5 each of these approaches for determining pairwise message correlation which guides the formation of the message graph.

Given a message graph, we propose to explore three graph-based approaches for extracting campaigns:

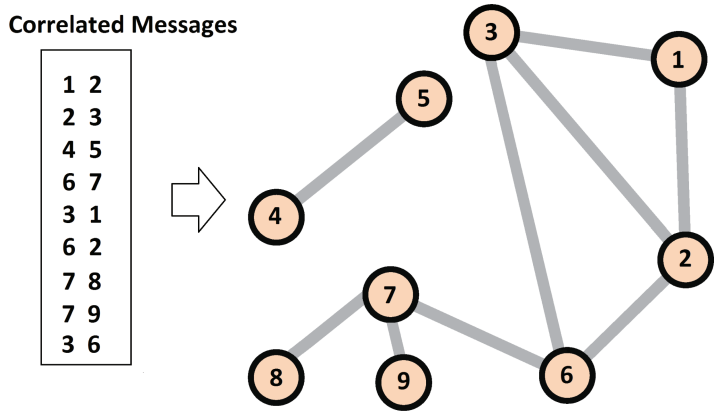


Fig. 2. In a messages graph, a node represents a message and there exists an edge between correlated messages. This figure shows an example of a message graph before extracting campaigns.

—(i) loose extraction;
—(ii) strict extraction; and
—(iii) cohesive extraction.

Experimentally, we compare these graph-based approaches versus a traditional k-means clustering approach and reach poor results for clustering as compared to the graph methods. For now, we focus our attention on extracting content-driven campaigns via graph mining.

4.2.1. Loose Campaign Extraction. The first approach for content-driven campaign detection is what we refer to as *loose campaign extraction*. The main idea is to identify as a logical campaign all chains of messages that share common "talking points". In this way, the set of all loose campaigns is the set of all maximally connected components in the message graph.

Definition 2 (*Loose Campaign*). A loose campaign is a subgraph $s = (V', E')$, such that s is a maximally connected component of G, in which s is connected, and for all vertices m_{ix} such that $m_{ix} \in V$ and $m_{ix} \notin V'$ there is no vertex $m_{jy} \in V'$ for which $(m_{ix}, m_{jy}) \in E$.

As an example, Figure 2 illustrates a collection of 10 messages, edges corresponding to messages that are highly correlated, and the two maximal components (corresponding to loose campaigns): $\{1, 2, 3, 6, 7, 8, 9\}$ and $\{4, 5\}$. Such an approach to campaign detection faces a critical challenge, however: not all maximally connected components are necessarily campaigns themselves (due to long chains of tangentially related messages). For example, a chain of similar messages A–B–C–...–Z, while displaying local similarity properties (e.g., between A and B and between Y and Z), will necessarily have low similarity across the chain (e.g., A and Z will be dissimilar since there is no edge between the pair, as in the case of messages 9 and 1 in Figure 2). In practice, such maximally connected components could contain disparate "talking points" and not strong campaign coherence.

4.2.2. Strict Campaign Extraction. A natural alternative is to constrain campaigns to be maximal cliques, what we call *strict campaigns*.

Definition 3 (*Strict Campaign*). A strict campaign $s' = (V'', E'')$ in a message graph $G = (V, E)$, in which $V'' \subseteq V$ and $E'' \subseteq E$, such that for every two vertices m_{ix} and

m_{jy} in V'', there exists an edge $(m_{ix}, m_{jy}) \in E''$ and the clique cannot be enlarged by including one more adjacent vertex (corresponding to a message in M).

To identify these strict campaigns, we can first identify all loose campaigns; by identifying all maximally connected components over the message graph, we can prune from consideration all singleton messages and are left with a set of candidate campaigns. Over these candidates, we can identify the strict campaigns through maximal clique mining. However, discovering all maximal cliques from a graph is an NP-hard problem (i.e., the time complexity is exponential). Finding all maximal cliques takes $O(3^{n/3})$ in the worst case where n is the number of vertices [Tomita et al. 2006]. Over large graphs, even with parallelized implementation over MapReduce-style compute clusters, the running time is still $O(3^{n/3}/m)$ in the worst case, where n is the number of vertices and m is the number of reducers [Wu et al. 2009].

And there is still the problem that even with a greedy approximation, strict campaign detection may overconstrain the set of campaigns, especially in the case of loosely connected campaigns. Returning to the example in Figure 2, the maximal cliques {1, 2, 3} and {2, 3, 6} would be identified as strict campaigns, but perhaps {1, 2, 3, 6, 7} form a coherent campaign even though the subgraph is not fully connected. In this case the strict approach will identify multiple overlapping campaigns and will miss the larger and (possibly) more coherent campaign. In terms of our metrics, the expectation is that strict campaign detection will favor precision at the expense of recall.

4.2.3. Cohesive Campaign Extraction. Hence, we also consider a third approach which seeks to balance loose and strict campaign detection by focusing on what we refer to as *cohesive campaigns*, which relaxes the conditions of maximal cliques.

Definition 4 (*Cohesive Campaign*). Given a message graph G = (V,E), a subgraph G' is called a *cohesive campaign* if the number of edges of G' is close to the maximal number of edges with the same number of vertices of G'.

The intuition is that a cohesive campaign will be a dense but not fully connected subgraph, allowing for some variation in the "talking points" that connect subcomponents of the overall campaign. There are a number of approaches mining dense subgraphs [Hu et al. 2005; Gibson et al. 2005; Wang et al. 2008] and the exact solution is again NP-hard in computation complexity, so we adopt a greedy approximation approach following the intuition in Wang et al. [2008]. The approach to extract cohesive campaigns requires a notion of maximum coclique $CC(m_{ix}, m_{jy})$ for all neighbors.

Definition 5 (*Maximum Coclique:* $CC(m_{ix}, m_{jy})$). Given a message graph G = (V,E), the maximum co-clique $CC(m_{ix}, m_{jy})$ is the (estimated) size of the largest clique containing both vertices m_{ix} and m_{jy}, where $m_{jy} \in V$ and m_{jy} is a neighbor vertex of m_{ix} (i.e., they are connected).

Considering all of a vertex's neighbors, we define the largest of the maximum cocliques as $C(m_{ix})$.

Definition 6 ($C(m_{ix})$). Then, $C(m_{ix})$ is the largest value between m_{ix} and any neighbor m_{jy}, formally defined as $C(m_{ix}) = max\{CC(m_{ix}, m_{jy}), \forall m_{jy} \in Neighbor(m_{ix})\}$.

With these definitions in mind, our approach to extract cohesive campaign is as follows.

(1) *Estimate each vertex's* $C(m_{ix})$. In the first step, our goal is to estimate the C values for every vertex in a candidate campaign which indicates the upper bound of the maximum clique size to which the vertex belongs. Starting at a random vertex m_{ix} in s, we compute the maximum coclique size $CC(m_{ix}, m_{jy})$, where $m_{jy} \in V'$ and m_{jy} is a neighbor vertex

of m_{ix}. Then, we compute $C(m_{ix})$. We insert m_{jy} into a priority queue and sort all m_{jy} by $CC(m_{ix}, m_{jy})$. Next, we greedily advance to the m_{jy}, which has the largest $CC(m_{ix}, m_{jy})$ among all m_{jy}, and remove it from the queue. Finally, we compute $C(m_{jy})$. We repeat this procedure for every vertex in the candidate campaign. At the conclusion of this procedure, we have an estimated $C(m_{ix})$ for every vertex.

(2) *Cohesive campaign extraction*. Given the estimated $C(m_{ix})$ for every vertex in a candidate campaign, by considering the order in which the greedy algorithm in step 1 encounters each vertex, we can consider consecutive neighbors as potential members of the same coherent campaign. Intuitively, the $C(m_{ix})$ values should be high for vertices in dense subgraphs but should drop as the algorithm encounters nodes on the border of the dense subgraph, then rise again as the algorithm encounters vertices belonging to a new dense subgraph. We identify the first vertex with an increasing $C(m_{ix})$ over its neighbor as the initial boundary of a cohesive campaign. We next include all vertices between this first boundary up to and including the vertex with a $C(m_{ix})$ value larger than or equal to some threshold (= the local peak value * λ). By tuning λ to 1, the extracted cohesive campaigns will be nearly clique like; lower values of λ will result in more relaxed campaigns (i.e., with less density). We repeat this procedure until we extract all cohesive subgraphs in the candidate campaign.

The output of the cohesive campaign extraction approach is a list of cohesive campaigns, each of which contains a list of vertices forming a cohesive subgraph.

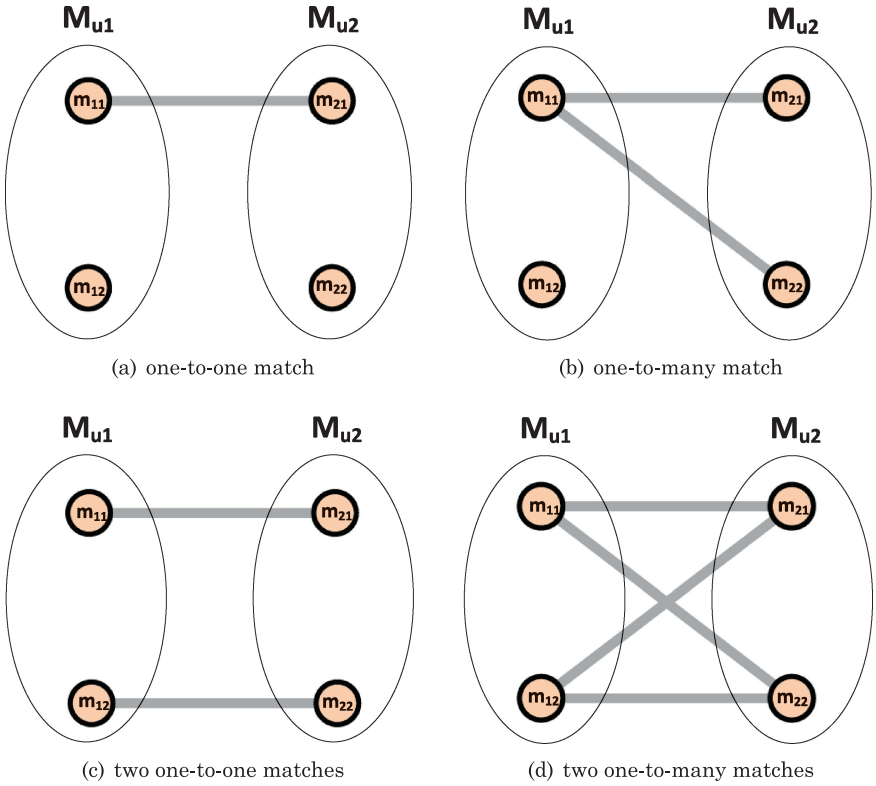
4.3. User-Level Campaign Detection

We turn our attention to a *user-aggregated* perspective. In the message-level campaign detection in the previous section, we have viewed all messages without consideration for *who* is posting the messages. By also considering user-level information, we are interested to see how this impacts campaign detection. The intuition is that by aggregating the messages posted by a single user, we may discover cross-user correlations not captured at the individual message level (e.g., for two users posting a sequence of correlated messages), leading to more robust campaign detection.

Definition 7 (*User-Aggregated Message Graph*). A user-aggregated message graph is a graph $G_u = (V, E)$ where V is a collection of n users' aggregate messages $V = \{M_{u_1}, M_{u_2}, ...M_{u_n}\}$. An edge $(M_{u_i}, M_{u_j}) \in E$ exists for every pair of vertices (M_{u_i}, M_{u_j}) in V where $confidence\ (Mu_i, Mu_j) > threshold$, for some measure of confidence and threshold. In the confidence computation, message similarity for every pair of messages (m_{ix}, m_{jy}) is computed where $corr(m_{ix}, m_{jy}) > \tau$, $m_{ix} \in M_{u_i}$, $m_{jy} \in M_{u_j}$ and $M_{u_i}, M_{u_j} \subseteq M$, for some measure of correlation and some parameter τ.

An important challenge is to define the correlation across vertices in the user-aggregated message graph, since each vertex now represents multiple messages (and so straightforward adoption of the message-level correlation approach is insufficient). For example, the two users in Figure 3 could have several different degrees of message-level correlation, based on the overlap between their messages. In the figure, we show messages $M_{u_1} = \{m_{11}, m_{12}\}$, and $M_{u_2} = \{m_{21}, m_{22}\}$ from two users u_1 and u_2 respectively. An edge represents that a pair of two messages between M_{u_1} and M_{u_2} are correlated.

To compute user-based correlation, we propose a measure called *confidence* that aggregates message-message correlation and reflects: (i) that one edge in a one-to-many match receives same weight comparing to the edge in a one-to-one edge; (ii) that extra edges in a one-to-many match receive less weight than the weight for the edge in a one-to-one match, but still credits the one-to-many match for more evidence of user-based correlation.



(a) one-to-one match (b) one-to-many match

(c) two one-to-one matches (d) two one-to-many matches

Fig. 3. Four matches of correlated messages between user u_1 and u_2.

Concretely, we calculate confidence in the following way: Given two users u_1 and u_2 and their latest k messages $M_{u_i} = \{m_{i1}, m_{i2}, \ldots, m_{ik}\}$ where i is a user id (i.e., 1 or 2 in our example). First, we compute pairwise message correlation across M_{u_1} and M_{u_2}, where pairs are $P = \{m_{1x}, m_{2y} | 1 \leq x, y \leq k\}$. If the correlation of a pair in P is larger than threshold τ, we consider the pair to be correlated. By continuing this procedure for each pair in P, we have correlated pairs P' and can calculate: (1) the number of pairs in P', $N = |\{m_{1x}, m_{2y} | corr(m_{1x}, m_{2y}) \geq \tau, 1 \leq x, y \leq k\}|$; and (2) the minimum n between number of distinct messages belonging to P' in M_{u_1} and number of distinct messages belonging to P' in M_{u_2}, where $n = MIN(|\{m_{1x}|\ m_{1x} \in M_{u_1}$ and $M_{1x} \in P'\}|,|\{m_{2y}|\ m_{2y} \in M_{u_2}$ and $m_2 j \in P'\}|)$. Now, we define that *confidence* as

$$confidence = \alpha n + (1 - \alpha)(N - n),$$

where α is the weight for the only edge in a one-to-one match or one edge in a one-to-many match, and $1 - \alpha$ is the weight for each of the extra edges in a one-to-many match. We assigned 0.95 to α to balance between αn and $(1 - \alpha)(N - n)$. Returning to Figure 3(a), (b), (c), (d), we have {N = 1, n = 1, confidence = 0.95}, {N = 2, n = 1, confidence = 1}, {N = 2, n = 2, confidence = 1.9}, and {N = 4, n = 2, confidence = 2} showing that in order of user-based correlation a < b < c < d.

4.4. MapReduce-Based Implementation

To support scalable identification of correlated messages, we implement the proposed approach over the MapReduce framework, which was introduced by Google to process

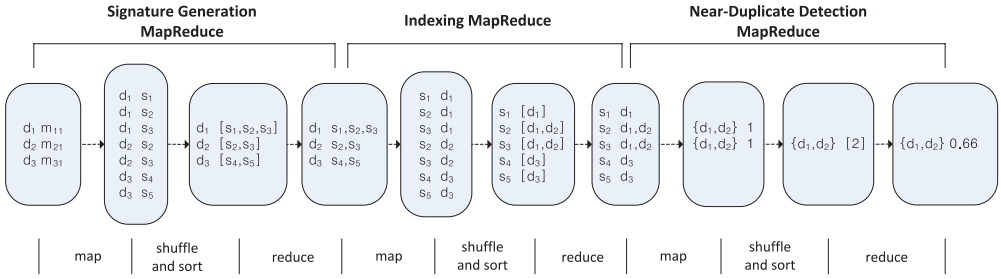


Fig. 4. Logical dataflow of the three MapReduce jobs for identifying correlated messages.

large datasets on a cluster of machines [Dean and Ghemawat 2004]. In MapReduce-style programming, each task is divided into two subfunctions: (1) a mapper: a sequence of data is inserted to a computation to generate partial results; and (2) a reducer: the results are then aggregated. We implemented our correlated message identification approach on Hadoop [Apache 2012] which can facilitate the handling of large-scale social message data.

The implementation consists of three MapReduce jobs, illustrated in Figure 4 with the following notation: (1) d_k is an auto-increasing message ID for a message; (2) m_{ij} indicates the j_{th} message from user u_i; (3) a near-duplicate detection algorithm generates three signatures (s_1, s_2, s_3) from the message m_{11}; (4) { } means a tuple and [ ] means a list. To calculate the correlation of the Jaccard coefficient (we use Jaccard coefficient in this example, but use Overlap coefficient in the experiments), we calculate each message's number of signatures in the map function of the signature generation job and pass the information associated to the message ID to later jobs. The near-duplicate detection returns pairs of near-duplicate messages (e.g., m_{11} and m_{21} have 0.66 similarity). To test the gains from a MapReduce-based implementation, we ran the message correlation component over 1.5 million Twitter messages as a MapReduce job on a small nine-node cluster and as a single-threaded (non-MapReduce) job on a single machine. The MapReduce job took only 7 minutes as compared to one day in the non-MapReduce approach, indicating the gains from parallelization.

5. EXPERIMENTAL STUDY

In this section, we explore campaign discovery over social media through an application of the framework to messages and user-aggregated messages sampled from Twitter. For message-level campaign detection, we begin by examining how to accurately and efficiently construct the campaign message graph, which is the critical first step necessary for campaign detection. We find that a short-text modified Shingling-based approach results in the most accurate message graph construction. Based on this finding, we next explore campaign detection methods over the small hand-labeled Twitter dataset, before turning our sights to analysis of campaigns discovered over the large (1.5 million messages) Twitter dataset. Based on the insights learned from the experiments in message-level campaign detection, we run user-level campaign detection to see whether we can find more evidence of spam and other coordinated campaigns. In the end of this section, we analyze the temporal patterns of campaigns, which suggests the potential of predicting a campaign's category based on its temporal pattern.

5.1. Message Level

We begin by examining message graph construction, which is the critical first step necessary for campaign detection.

5.1.1. Message Graph Construction. Recall that each node in the message graph corresponds to a message; edges correspond to some reasonable notion of "relatedness" between messages corresponding to human-labeled similar "talking points". Our first goal is to answer the question: can we effectively determine if two messages are correlated (i.e., algorithmically determine if they share similar "talking points") across hundreds of millions of short messages for constructing the message graph in the first place? This step is critical for accurate message graph formation for discovering campaigns.

Using the small *campaign dataset* (CD_{Small}), we consider the 298 pairs of messages sharing similar "talking points" (as determined by human judges) as the ground truth for whether an edge should appear in the message graph between the two messages. We can measure the effectiveness of a message correlation method by precision, recall, and F_1. Precision (P) is the fraction of predicted edges that are correct:

$$\frac{\text{\# of correctly predicted edges}}{\text{\# of predicted edges}}.$$

Recall (R) is the fraction of correct edges that are predicted:

$$\frac{\text{\# of correctly predicted edges}}{\text{\# of edges}}.$$

The F_1 measure balances precision with recall: $\frac{2PR}{P+R}$.

Identifying Correlated Messages. We investigate the identification of correlated messages through a comparative study of the six distinct techniques described in Section 4: unigram-based overlap between messages, edit distance, Euclidean distance, and three representative near-duplicate detection algorithms (Shingling [Broder et al. 1997], I-Match [Chowdhury et al. 2002], and SpotSigs [Theobald et al. 2008]). The near-duplicate detection approaches such as Shingling, I-Match, and SpotSigs have shown great promise and effectiveness by Web search engines to efficiently identify duplicate Web content, but their application to inherently short messages lacking context is unclear.

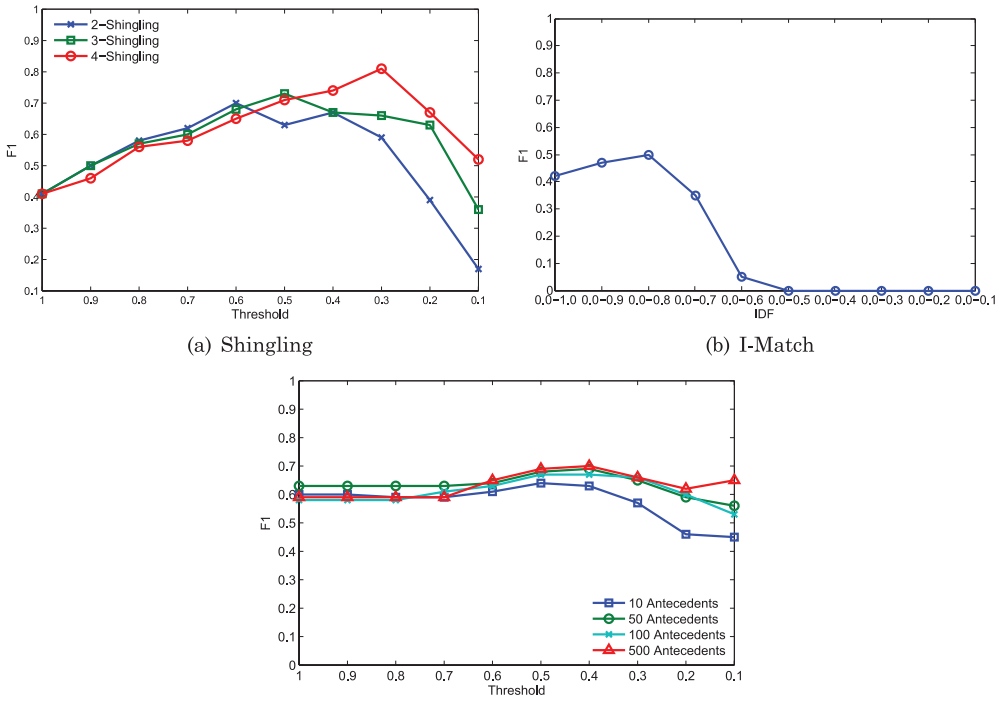
To evaluate each approach, we considered a wide range of parameter settings. For example, the quality of Shingling depends on the size of the shingle (2, 3, 4). I-Match requires minimum and maximum IDF values; we varied the min and max IDF values over the range [0.0, 1.0] in 0.1 increments and considered all possible pairs (e.g., min = 0.1, max = 0.6). SpotSigs requires a number of antecedents (which we varied across 10, 50, 100, and 500) and a specification of what antecedents will be used. As the authors of SpotSigs [Theobald et al. 2008] did in their experiments, we used stop-words as antecedents. And across all approaches, we must also set a predefined threshold value τ, above which a pair of messages are considered correlated (and hence and edge should appear in the message graph).

With this large parameter space in mind, we show in Table I the results across all approaches that optimize the F_1 score (the details of performance of Shingling, I-Match, and SpotSigs with different parameter value are shown in Figure 5).

We see that the baseline Shingling approach performs the best, with an $F_1 = 0.81$. In contrast, both I-Match and SpotSigs performed much worse (0.50, 0.70), in sharp contrast to their performance in near-duplicate detection of Web pages (with F_1 near 95%) [Theobald et al. 2008; Zhang et al. 2010]. While these approaches work well in news articles and Web pages (relatively long text), they do not work well for short text. We also observe that unigram-, edit-distance-, and Euclidean-distance-based methods perform poorly, primarily due to their low recall. This indicates that short messages

Table I. Identifying Correlated Messages

Approach	F_1	Precision	Recall
Unigram (τ = 0.8)	0.63	0.97	0.46
Edit Distance (τ = 11)	0.54	0.97	0.38
Euclidean Distance (τ = 5)	0.61	0.99	0.44
4-Shingling (τ = 0.3)	**0.81**	0.89	0.73
I-Match (IDF = [0.0, 0.8])	0.50	0.53	0.47
SpotSigs (#A = 500, τ = 0.4)	0.70	0.77	0.64



(a) Shingling (b) I-Match

(c) SpotSigs

Fig. 5. Performance of Shingling, I-Match, and SpotSigs with different parameter value.

that do share common "talking points" may be missed by these approaches which emphasize on only minor syntactic changes across messages.

Refining Shingling. Based on these results, we further explore refinements to the baseline Shingling approach. First, we vary the base tokenization unit for message comparison, which is especially critical for short messages. We consider three general approaches for extracting tokens to generate shingles: (i) word-based k-grams, in which k consecutive words are treated as base tokens; (ii) character-based k-grams, in which k consecutive characters are treated as base tokens. As compared to word-based k-grams, character-based k-grams generate more tokens but offer finer granularity of measuring message correlation; and (iii) orthogonal sparse bigrams, introduced by Cormack [2008] for lexically expanding a short message by generating sparse bigrams by the number of intervening words, each of which we denote by "?". For example, "lady gaga is unique person" generates sparse bigrams: lady + gaga, lady + ? + is, lady + ? + ? + unique, gaga + is, gaga + ? + unique, gaga + ? + ? + person, is + unique, is + ? + person, unique + person.

Table II. Refinements to Shingling

Approach	F_1	Precision	Recall
4-Shingling ($\tau = 0.3$)	0.81	0.89	0.73
Character k-grams ($k = 6$, $\tau = 0.6$)	0.74	1	0.59
OSB. ($\tau = 0.5$)	0.68	0.6	0.79
With Short Message Overlap	**0.88**	0.92	0.83

Finally, we note that straightforward application of the Jaccard coefficient over short messages may underestimate the degree of overlap between two messages, resulting in the mislabeling of correlated messages as unrelated. For example, suppose we apply 4-Shingling to the following two messages, splitting each message on whitespace and punctation:

—Here's How Apple's iPad Is Invading The Business World (AAPL, RIMM, MSFT) - San Francisco Chronicle: `http://bit.ly/dhqDGf`
—Here's How Apple's iPad Is Invading The Business World (AAPL, RIMM, MSFT) `http://bit.ly/d3ClTj`

With 18 and 15 shingles, respectively, and 11 shingles in common, the Jaccard coefficient will identify a correlation of only 0.5 (11 / (18+15−11)), even though the two messages are nearly identical. With a typical threshold τ of 0.6 or above, these two messages, though clearly correlated, would not be properly identified. Hence, we propose as a measure of correlation the overlap coefficient

$$corr_{overlap}(A, B) = \frac{|A \cap B|}{min(|A|, |B|)}$$

which in this case results in a correlation value of $11/15 = 0.73$. In general, smaller number of words in two messages will give us higher Jaccard and overlap coefficients divergence. Experimentally, we evaluate the impact of these approaches on the quality of correlated message identification.

Interestingly, as seen in Table II, neither character-based k-grams nor orthogonal sparse bigrams, which have shown promise in other short text domains, performed as well as Shingling or the short-message-optimized approach presented in this manuscript. We conjecture that word-based tokens can capture similar messages well compared to character k-grams and orthogonal sparse bigrams which may generate too many features, leading to message correlation confusion. The short message overlap optimization, however, results in the best results and so we shall use this as a core approach for generating the message graphs in all subsequent experiments.

5.1.2. Campaign Detection over Small Data. In the previous set of experiments, we evaluated several approaches to measuring message correlation. Now we turn our attention to evaluating campaign detection methods. We begin in this section with the small dataset (which recall allows us to measure precision and recall against ground truth) before considering the large dataset.

Over the hand-labeled campaigns in CD_{Small}, we apply the three graph-based campaign extraction methods: (i) loose; (ii) strict; and (iii) cohesive, over the message graph generated via the best performing message correlation method identified in the previous section. We also compare campaign extraction using a fourth approach based on text clustering. For this nongraph-based approach, we consider k-means clustering, where each message is treated as a vector with 10K bag-of-words features, weighted using TF-IDF, with Euclidean distance as a distance function. We vary the choice of k value, and report the best result.

Table III. Effectiveness Comparison of Campaign Detection Approaches

Approach	NumC	F_1	Precision	Recall
Loose	12	0.962	0.986	0.940
Strict	12	0.906	0.907	0.904
Cohesive	11	**0.963**	0.977	0.950
k-means	5	0.89	1	0.805

Table III presents the experimental results of the four campaign detection approaches. The cohesive campaign detection approach found 11 campaigns ($NumC$) like the ground truth, but missed a message in two campaigns. The strict approach found 12 campaigns, missed one message in a true campaign, and divided a true campaign to two predicted campaigns due to the strict campaign rule (all nodes in a campaign should be completely connected). The loose approach found 12 campaigns, one of which is not an actual campaign (false positive) and some predicted campaigns contain dissimilar messages due to long chains. The k-means clustering algorithm found only 5 campaigns. Overall, the cohesive and strict approaches outperformed the loose and cluster-based approaches. In practice, the ideal approach should return the same number of campaigns as the ground truth and do so quickly. In this perspective, the cohesive approach would be preferred over the strict approach because the number of its campaigns is the same with the ground truth, and it is relatively faster than the strict approach.

5.1.3. Campaign Detection over Large Data. We next examine campaign extraction from the large Twitter dataset, CD_{Large}. Can we detect coordinated campaigns in a large message graph with 1.5 million messages? What kind of campaigns can we find? Which graph technique is the most effective to find campaigns?

Message Graph Setup. Based on the best message graph construction approach identified in the previous section, we generated a message graph consisting of 1.5 million vertices (one vertex per message). Of these, 1.3 million vertices are singletons, representing messages without any correlated messages in the sample (and hence, not part of any campaign). Based on this sample, we find 199,057 vertices have at least one edge; in total, there are 1,027,015 edges in the message graph.

Identifying Loose Campaigns. Based on the message graph, we identify as *loose campaigns* all of the maximally connected components, which takes about 1 minute on a single machine (relying on a breadth-first search with time complexity $O(|E| + |V|)$. Figure 6 shows the distribution of the size of the candidate campaigns on a log-log scale. We see that the candidate campaign sizes approximately follow a power law, with most candidates consisting of 10 or fewer messages. A few candidates have more than 100 messages, and the largest candidate consists of 61,691 messages. On closer inspection, the largest candidate (as illustrated in Figure 7) is clearly composed of many locally dense subgraphs and long chains. Examining the messages in this large candidate, we find many disparate topics (e.g., spam messages, Justin Bieber retweets, quotes, Facebook photo template) and no strong candidate-wide theme, as we would expect in a coherent campaign.

Identifying Strict Campaigns. To refine these candidates, one approach suggested in Section 4 is *strict campaign detection*, in which we consider only maximal cliques as campaigns (in which all message nodes in a subgraph are connected to each other). While maximal clique detection may require exponential time and not be generalizable to all social message datasets, in this case we illustrate the maximal cliques found even though it required $\sim$7 days of computation time (which may be unacceptable

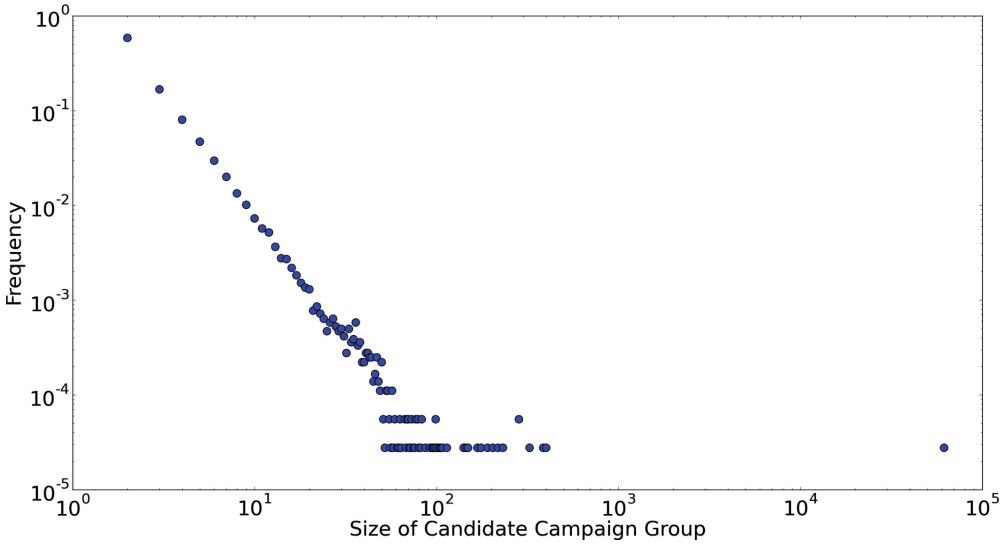


Fig. 6. This figure depicts the distribution of the size of candidate campaigns on a log-log scale. It follows a power law.

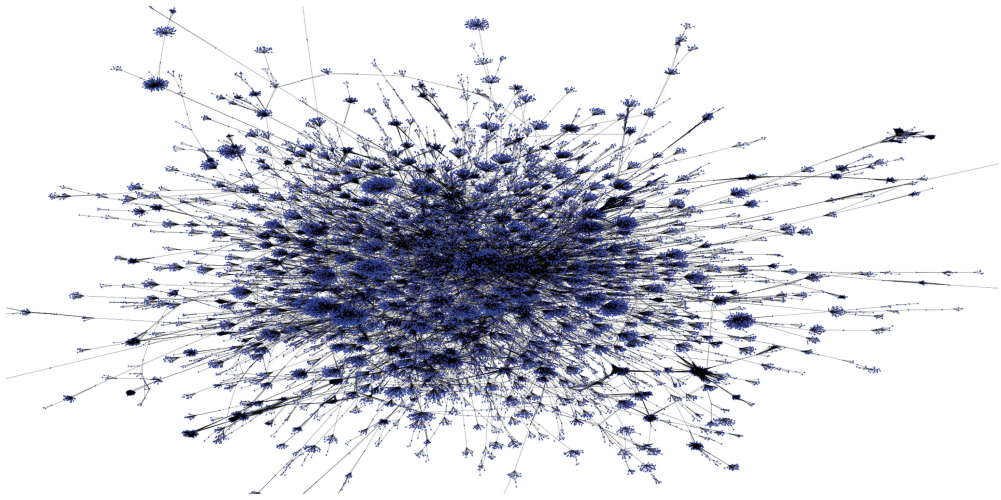


Fig. 7. This figure depicts a candidate with 61,691 vertices. A blue dot and a black line represent a vertex and an edge, respectively. The area in the center is dark because most vertices in the center are very densely connected.

for campaign detection in deployed systems). Considering the top-10 strict campaigns discovered in order of size: [559, 400, 400, 228, 228, 227, 227, 217, 217, 214], we find high overlap in the campaigns discovered. For example, the 2nd and 3rd strict campaigns (each of size 400) have 399 nodes in common. Similarly, the 4th, 5th, 6th, 7th, and 10th strict campaigns have over 200 nodes in common, suggesting that these five different strict campaigns in essence belong to a single coherent campaign (see Figure 8). This identification of multiple overlapping strict campaigns—due to noise, slight changes in message "talking points", or other artifacts of short messages—as well as the high cost

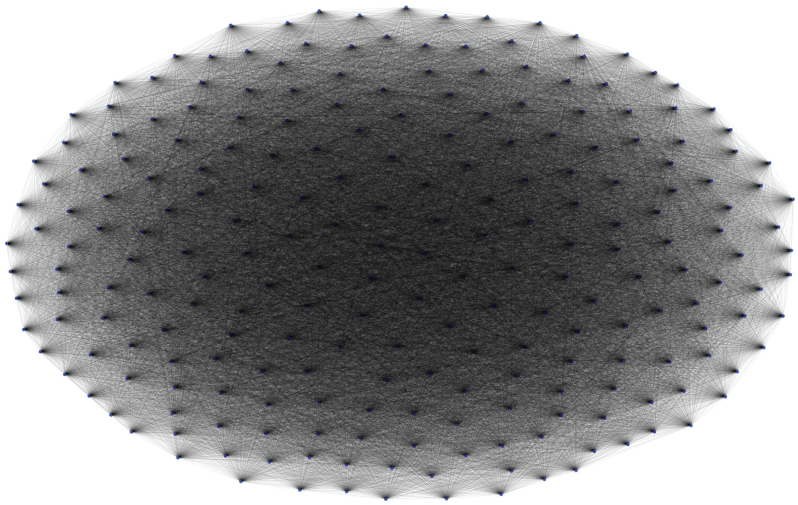


Fig. 8. An example dense subgraph campaign: the center area is dark because vertices in the area are very densely connected; this subgraph is almost fully connected except a few vertices. While strict campaign detection identifies 5 different maximal cliques, cohesive campaign detection identifies a single coherent campaign including all vertices.

of maximal clique detection suggests the cohesive campaign detection approach may be preferable.

Identifying Cohesive Campaigns. We next applied the *cohesive campaign extraction* approach to the set of candidate campaigns corresponding to maximal connected components. We assign λ to 0.95 and use the CSV tool [Wang et al. 2008] for an efficient implementation of computing each vertex m_{ix}'s $C(m_{ix})$ by mapping edges and vertices to a multidimensional space. Although computing $C(m_{ix})$ of all vertices takes $O(|V|^2 \log |V| 2^d)$ where d is a mapping dimension, the performance for real datasets is typically subquadratic. Figure 9 shows the distribution of the size of the cohesive campaigns in a log-log scale. Like the candidate campaign sizes, we see that the cohesive campaigns follow a power law. Since the cohesive campaign extraction approach can isolate dense subgraphs, we see that the large 61,691 message candidate has been broken into 609 subcomponents. Compared to strict campaign detection, the cohesive campaign extraction approach required only 1/7 the computing time on single workstation.

Examining the top-10 campaigns (shown in Table IV) we see that the cohesive campaign detection approach overcomes the limitations of strict campaign detection by combining multiple related cliques into a single campaign (recall Figure 8). The biggest campaign contains 560 vertices and is a spam campaign. The "talking point" of this campaign is an Iron Man 2 promotion of the form: "#Monthly Iron Man 2 (Three-Disc Blu-ray/DVD Combo + Digital Copy) . . . http://bit.ly/9L0aZU", though individual messages vary the exact wording and inserted link.

Based on a manual inspection of the identified campaigns, we categorize the campaigns into five categories.

—*Spam campaigns*. These campaigns typically post duplicate spam messages (changing @username with the same payload), or embed trending keywords, often with a URL linking to a malware Web site, phishing site, or a product Web site. Example: "Want FREE VIP, 100 new followers instantly and 1,000 new followers next week? GO TO `http://alturl.com/bpby`".

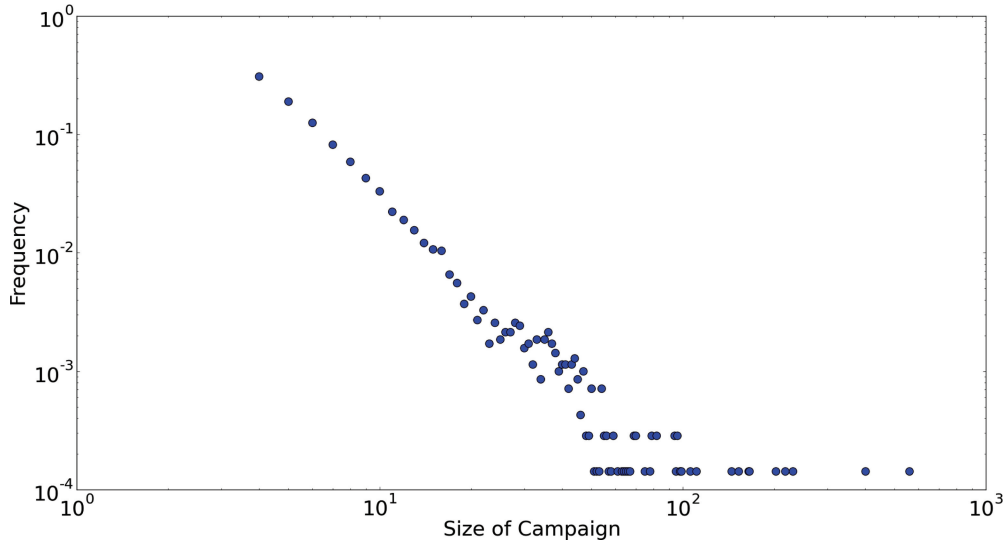


Fig. 9. This figure depicts the distribution of the size of cohesive campaigns on a log-log scale. It also follows a power law.

Table IV. Top-10 Largest Campaigns

Msgs	Users	Talking Points
560	34	Iron Man 2 spam
401	390	Facebook photo template
231	231	Support Breast Cancer Research (short link)
218	218	Formspring template
203	197	Chat template (w/ link)
166	166	Support Breast Cancer Research (full link)
165	154	Quote "send to anyone u don't regret meeting"
153	153	Justin Bieber Retweets
145	31	Twilight Movie spam
111	111	Quote "This October has 5 Fridays ..."

—*Promotion campaigns*. Users in these campaigns promote a Web site or product. Their intention is to expose it to other people. Example: "FREE SignUp!!! earn $450 Per Month Do NOTHING But Getting FREE Offers In The Mail!! `http://budurl.com/PPLSTNG`".
—*Template campaigns*. These are automatically generated messages typically posted by a third-party service. Example: "I'm having fun with @formspring. Create an account and follow me at `http://formspring.me/xnadjeaaa`".
—*News campaigns*. Participants post recent headlines along with a URL. Example: "BBC News UK: Rwanda admitted to Commonwealth: Rwanda becomes the 54th member of the Commonwealth g.. `http://ad.vu/nujv`".
—*Celebrity campaigns*. Users in these campaigns send messages to a celebrity or retweet a celebrity's tweet. Example: "@justinbieber please follow me i love youuu<3".

Some of these campaigns are organic and the natural outgrowth of social behavior, for example, a group of Justin Bieber fans retweeting a message, or a group posting news articles of interest. On closer inspection, we observe that many of the less organic

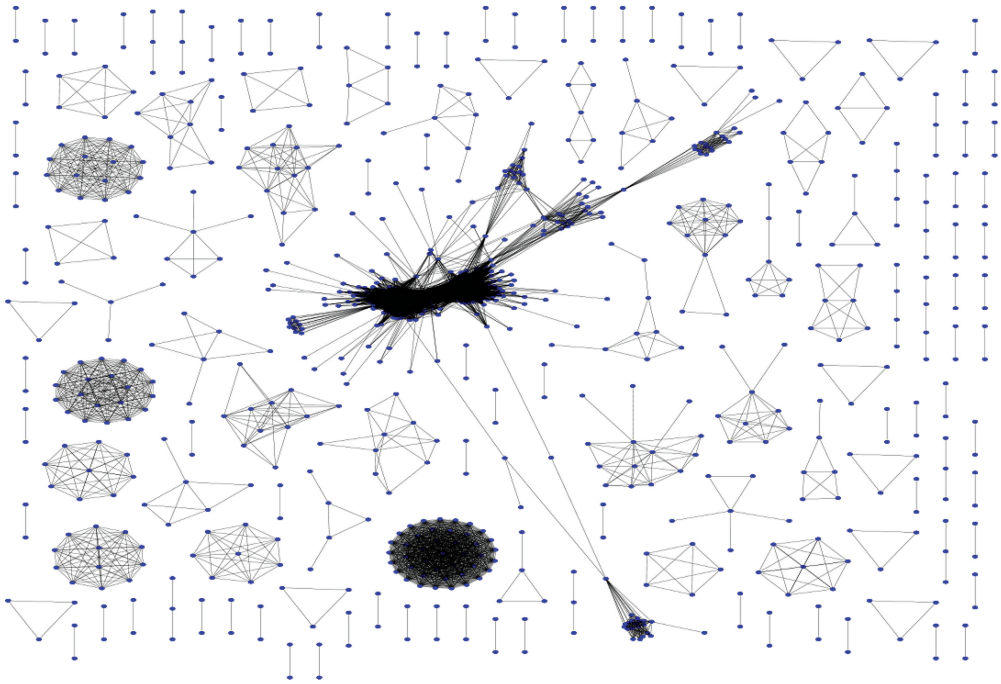


Fig. 10. 303 candidate campaigns in the user-aggregated message graph.

campaigns (e.g., spam and promotion campaigns) are driven by a higher ratio of messages to participants. For example in Table IV, the Iron Man 2 spam campaign consists of 560 messages posted by only 34 different participants. In contrast, the Justin Bieber retweet campaign consists of 153 messages posted by 153 different participants.

5.2. User Level

Based on this observation—of a handful of accounts aggressively promoting particular "talking points" in Twitter—we next turn to user-aggregated campaign detection. By collapsing multiple messages from a single user in the user-aggregated message graph, do we find more evidence of spam and other coordinated campaigns (since edges correspond to users with highly correlated messages)? What impact does the confidence threshold have on campaign detection?

Data and Setup. Since the dataset for the previous study was based on a random sample of Twitter (meaning most users were represented by only one message), we use a user-focused dataset CD_{User} from Twitter consisting of 90,046 user profiles with at least 20 English-language messages. Based on these messages, we constructed a user-aggregated message graph where each vertex corresponds to a user and an edge exists between all users passing a threshold confidence value. For a threshold of 3.8 (i.e., $n = 4$) we find 2,301 vertices with at least one edge, and a total of 89,294 edges in the user-aggregated message graph.

Campaign Detection. Following the campaign framework in Section 4.2.3, we find 303 candidate campaigns illustrated in Figure 10. Applying the cohesive campaign extraction approach we find 62 campaigns with at least four users. Through manual inspection, we labeled each of the 62 campaigns according to campaign type (see Figure 11).

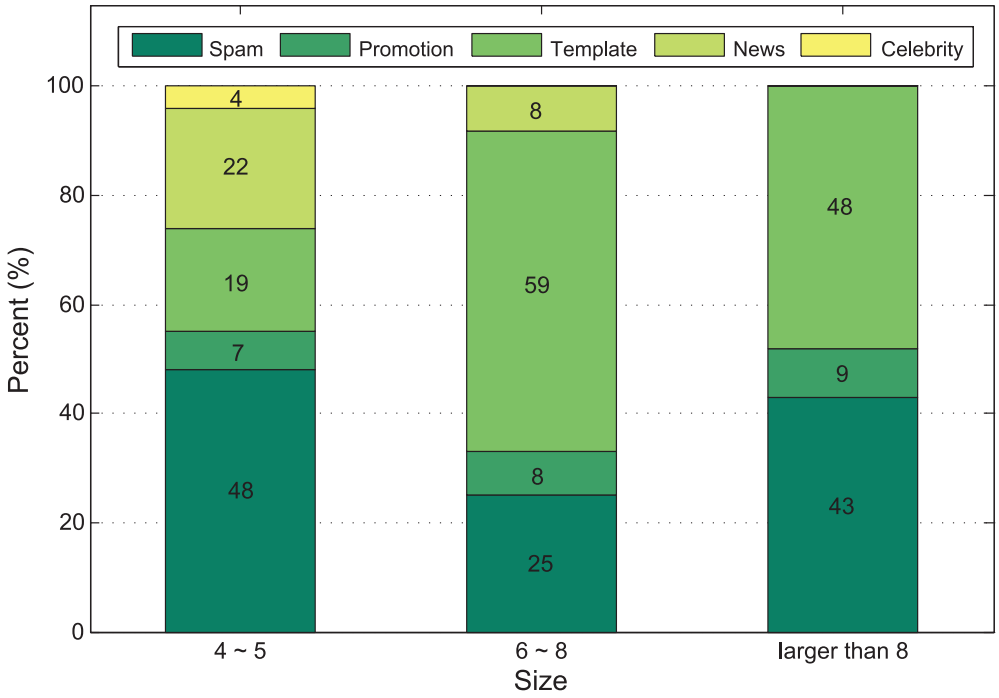


Fig. 11. Campaign type distribution (threshold = 3.8).

We observe that spam and template campaigns are major campaign types in all three partitions divided by ranges of the size.

We next analyze whether a different campaign category has significantly different content/terms in messages. To identify significant terms for the users in each category type, we identify terms with high mutual information for each campaign category. Mutual information is a standard information-theoretic measure of "informativeness" and, in our case, can be used to measure the contribution of a particular term to a category of campaign. Concretely, we build a unigram language model for each category of campaign by aggregating all messages by all users belonging to a particular campaign category (e.g, all users participating in a spam campaign). Hence, mutual information is measured as: $MI(t,c) = p(t|c)p(c)log\frac{p(t|c)}{p(t)}$ where $p(t|c)$ is the probability that a user which belongs to category c has posted a message containing term t, $p(c)$ is the probability that a user belongs to category c, and $p(t)$ is the probability of term t over all categories. That is, $p(t) = count(t)/n$. Similarly, $p(t|c)$ and $p(c)$ can be simplified as $p(t|c) = count(c,t)/count(c)$ and $p(c) = count(c)/n$ respectively, where $count(c,t)$ denotes the number of users in category c which also contain term t, and $count(c)$ denotes the number of users in category c.

Table V shows the top-10 significant terms for each campaign category. In spam campaigns, we observe that spammers have posted messages regarding increasing followers via a software service. An example message is "Hey Get 100 followers a day using http://yumurl.com/p74ZY6. Its super fast!". Note that the Twitter Safety team considers promoting such automated friend software as spam [Twitter 2012]. Promotion campaigns promote particular links or products. An example message is "if you like iq quize's then check out this free iq quiz http://tiny.cc/amazingfreeiqquiz #donttrytoholla". Messages of the users in the news campaign contain hot keywords

Table V. Top-10 Significant Terms for Each Campaign Category

Category	Top 10 Terms				
spam	followers	100	day	site	fast
	check	twitter	account	upload	twtmuzik
promotion	iq	broadcasting	stickam	stream	quiz
	michael	#140kingofpop	jackson	free	woot
news	media	social	bbc	engadget	windows
	#news	apple	android	africa	iphone
template	video	#epicpetwars	xbox	chat	#tinychat
	joined	people	playing	youtube	live
celebrity	@justinbieber	follow	justin	bieber	love
	mee	song	plss	hiii	dream

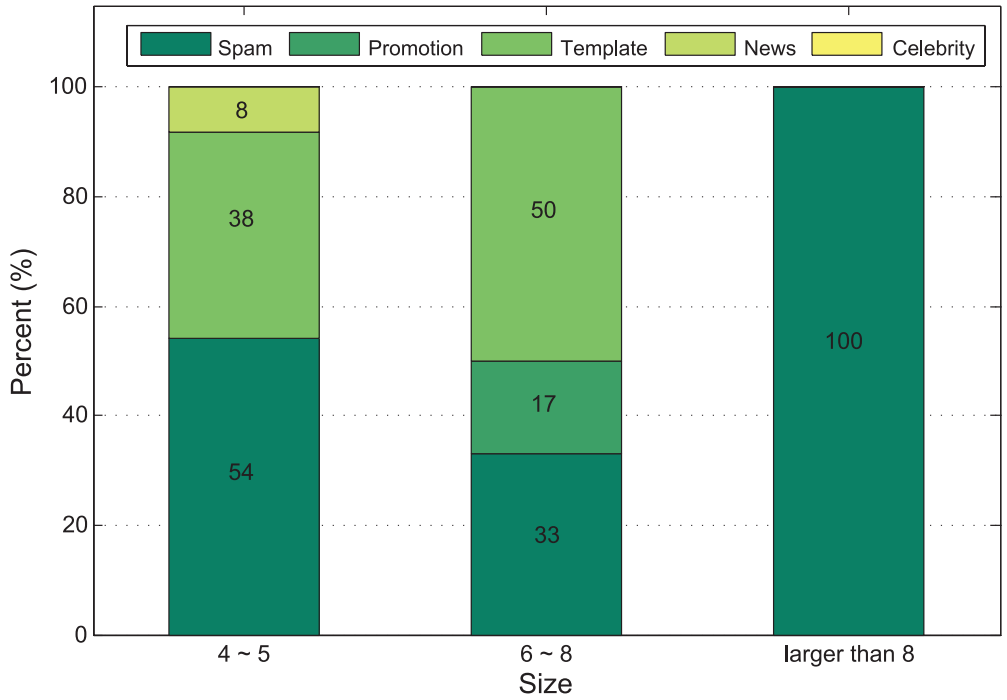


Fig. 12. Campaign type distribution (threshold = 9.5).

(e.g., social, media, android, and iphone) or media name (e.g., bbc, engadget). The significant terms in template campaigns describe a user's status (playing, xbox) or reflect a service being used (chat, #tinycat, and live). Users participating in celebrity campaigns often post messages targeting a particular celebrity (e.g., @justinbieber) expressing love or asking for the celebrity to reciprocate and follow the user.

Varying the Confidence Threshold. Now, we are interested in how the confidence threshold influences the campaigns detected. A higher confidence corresponds to more tightly correlated users (pairs who tend to post a sequence of similar messages), and would perhaps suggest a strategic rather than organic campaign. When we increase the confidence threshold to 9.5 (i.e., $n = 10$) we find 28 campaigns as shown in Figure 12. Compared to the lower confidence threshold, the proportion of spam campaigns increases to 65% compared with 42% in the previous experiment (see Table VI). Second,

Table VI. Campaign Categories for Low Confidence Threshold and High Confidence Threshold

Category	Low	High
Spam	42%	65%
Promotion	8%	3%
Template	37%	29%
News	11%	3%
Celebrity	2%	0%

Table VII. Categories of Top-50 Cohesive Campaigns

Category	Percent	Category	Percent
Spam	26%	Promotion	6%
Celebrity	34%	Template	34%

we see that for campaigns of the largest size, all are spam campaigns. This indicates that the confidence threshold can be an effective tunable knob for identifying strategic campaigns in large-scale social media. Overall, these user-aggregated message graph results show that content-based campaign detection can effectively identify campaigns of multiple types at low confidence and specifically of spam campaigns at high confidence.

5.3. Temporal Analysis

We next analyze temporal behaviors of the cohesive campaigns. Especially, we study each cohesive campaign category's temporal behaviors to see whether each campaign category has different temporal behavior. Using CD_{Large} (one-week data) may be not enough to study temporal patterns because of sparse data. In order to overcome this sparsity, we extend the one-week data to three-weeks data collected between October 1 and October 21, 2010 (again, we used Twitter streaming API which allows us to collect randomly 1% of all messages. If we can access all messages generated on Twitter, we may just need to use one-week data or even shorter data for the temporal analysis).

For temporal analysis, we selected the top-50 cohesive campaigns detected in Section 5.1.3, and added similar messages in the extended dataset into each campaign[1]. Then, we manually labeled the top-50 cohesive campaigns to one of four categories: Spam, Promotion, Celebrity, and Template. The campaign category distribution is shown in Table VII.

For temporal behavior analysis, a cohesive campaign is represented by a time-series vector $T_a = (T_{a1}, T_{a2}, \ldots, T_{an})$. Each value in the vector denotes a number of messages belonging to the campaign in a time unit (e.g., 1 day). Likewise, we create 50 time series (campaign vectors) based on 1-day unit. To make a time-series graph smooth (less fluctuated), we use two days moving average. For example, given a time series $T_a = (T_{a1}, T_{a2}, T_{a3}, \ldots, T_{an})$, two days moving average of T_a is $T'_a = (\frac{T_{a1}+T_{a2}}{2}, \frac{T_{a2}+T_{a3}}{2}, \ldots, \frac{T_{an-1}+T_{an}}{2})$.

We use dynamic time warping barycenter averaging (DBA) which is a global technique for averaging a set of sequences [Petitjean et al. 2011]. Compared to approaches like balanced hierarchical averaging or sequential hierarchical averaging, DBA avoids some of the deficiencies of these alternatives [Niennattrakul and Ratanamahatana 2007].

[1]There was no news campaign in the top-50 cohesive campaigns.

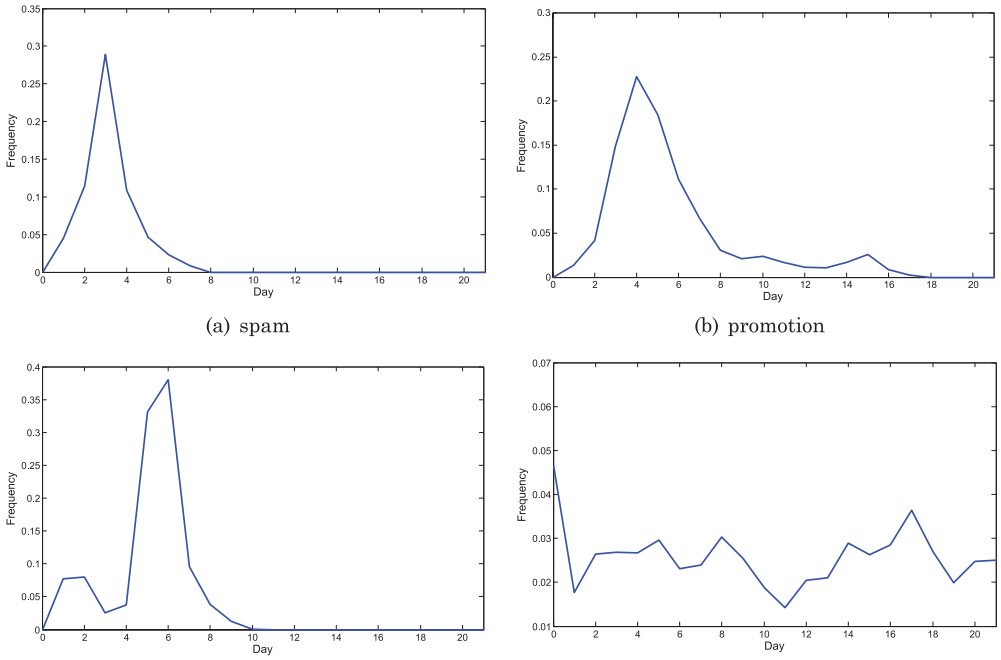


Fig. 13. Average temporal graphs of four campaign categories.

Figure 13 presents an average time series of each campaign type calculated by DBA. Spam campaigns have a sharp spike, reflecting how spammers post many similar messages at the beginning and then reduce the frequency of messages or change payload to avoid being caught by Twitter administrators. Users in promotion campaigns post messages over a longer period, suggesting that promotion and spam campaigns (though closely related) may reveal distinctions in their temporal patterns to support automatic differentiation. Celebrity campaigns have two spikes and then the frequency drops off. We conjecture that this phenomenon happens as people quickly retweet a celebrity's message (the first spike) and then the retweet passes through those users' social networks and is echoed (the second spike). Template campaigns have different temporal patterns from the others. As we can expect a temporal pattern of template campaigns, messages forming a template campaign are posted constantly and statically over time. This phenomenon makes sense because these messages are posted by third-party services or tools. Overall, each type of campaigns has different temporal pattern. This temporal analysis reveals the possibility to automatically classify a campaign type by its temporal pattern.

5.4. Summary

Through the preceding experiments, we found that it is possible to detect content-based campaigns in message and user levels in social media. Also, we found five campaign categories—namely Spam, Promotion, Template, Celebrity, and News campaigns. The proposed cohesive campaign detection approach outperformed loose and strict campaign detection approaches and a k-means clustering approach in terms of effectiveness and efficiency. The most encouraging result is that the messages posted by users who participate in negative campaigns (spam and promotion campaigns) have higher

content similarity. Temporal analysis of campaigns reveals that each campaign type has different temporal pattern, showing us the possibility to automatically determine a campaign's category.

6. CONCLUSION AND FUTURE WORK

In this manuscript, we have investigated the problem of campaign detection in social media. We have proposed and evaluated an efficient content-driven graph-based framework for identifying and extracting campaigns from the massive scale of real-time social systems. Based on the success of the system we are extending this work to incorporate adaptive statistical machine learning approaches for isolating artificial campaigns from organic campaigns. Do we find that strategically organized campaigns engage in particular behaviors that make them clearly identifiable? Our results in this manuscript suggest that campaigns are not necessarily "invisible" to automated detection methods. We are also interested in exploring whether campaigns are centralized around common types of users or are they embedded in diverse groups. How early in a campaign's lifecycle can a strategic campaign be detected with high confidence? Do we find a change in campaign membership and detection effectiveness after it reaches a critical mass? These challenges motivate our continuing research.

REFERENCES

Apache. 2012. Hadoop. http://hadoop.apache.org/.

Becchetti, L., Castillo, C., Donato, D., Baeza-Yates, R., and Leonardi, S. 2008. Link analysis for web spam detection. *ACM Trans. Web 2*, 1, 1–42.

Benczur, A. A., Csalogany, K., and Sarlos, T. 2006. Link-based similarity search to fight web spam. In *Proceedings of the SIGIR Workshop on Adversarial Information Retrieval on the Web.*

Benevenuto, F., Magno, G., Rodrigues, T., and Almeida, V. 2010. Detecting spammers on twitter. In *Proceedings of the Collaboration, Electronic Messaging, Anti-Abuse and Spam Conference (CEAS'10).*

Benevenuto, F., Rodrigues, T., Almeida, V., Almeida, J., and Gonc Alves, M. 2009. Detecting spammers and content promoters in online video social networks. In *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'09).* 620–627.

Bratko, A., Filipic, B., Cormack, G. V., Lynam, T. R., and Zupan, B. 2006. Spam filtering using statistical data compression models. *J. Mach. Learn. Res. 7,* 2673–2698.

Broder, A. Z., Glassman, S. C., Manasse, M. S., and Zweig, G. 1997. Syntactic clustering of the web. *Comput. Netw. ISDN Syst. 29*, 8–13, 1157–1166.

Castillo, C., Mendoza, M., and Poblete, B. 2011. Information credibility on twitter. In *Proceedings of the 20th International Conference on World Wide Web (WWW'11).* 675–684.

Caverlee, J., Liu, L., and Webb, S. 2008. Socialtrust: Tamper-resilient trust establishment in online communities. In *Proceedings of the 8th ACM/IEEE-CS Joint Conference on Digital Libraries (JCDL'08).* 104–114.

Caverlee, J., Liu, L., and Webb, S. 2010. The socialtrust framework for trusted social information management: Architecture and algorithms. *Inf. Sci. 180*, 1, 95–112.

Cctv. 2010. Uncovering online promotion. http://news.cntv.cn/china/20101107/102619.shtml.

Cheng, Z., Caverlee, J., and Lee, K. 2010. You are where you tweet: A content-based approach to geolocating twitter users. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management (CIKM'10).* 759–768.

Chowdhury, A., Frieder, O., Grossman, D., and McCabe, M. C. 2002. Collection statistics for fast duplicate document detection. *ACM Trans. Inf. Syst. 20*, 2, 171–191.

Cialdini, R. B. 2007. *Influence: The Psychology of Persuasion (Collins Business Essentials).* Harper Paperbacks.

Cormack, G. V. 2008. Email spam filtering: A systematic review. *Foundat. Trends Inf. Retr. 1*, 335–455.

Dean, J. and Ghemawat, S. 2004. Mapreduce: Simplified data processing on large clusters. In *Proceedings of the 6th Conference on Operating Systems Design and Implementation (OSDI'04).*

Fetterly, D., Manasse, M., and Najork, M. 2004. Spam, damn spam, and statistics: using statistical analysis to locate spam web pages. In *Proceedings of the Workshop on the Web and Databases.*

Films, L. 2011. (Astro) turf wars. www.astroturfwars.com.

GAO, H., HU, J., WILSON, C., LI, Z., CHEN, Y., AND ZHAO, B. Y. 2010. Detecting and characterizing social spam campaigns. In *Proceedings of the 10th Annual Conference on Internet Measurement (IMC'10)*.

GIBSON, D., KUMAR, R., AND TOMKINS, A. 2005. Discovering large dense subgraphs in massive graphs. In *Proceedings of the 31st International Conference on Very Large Data Bases (VLDB'05)*. 721–732.

GILBERT, I. AND HENRY, T. 2010. Persuasion detection in conversation. In Master's thesis, Naval Postgraduate School, Monterey, CA.

GRIER, C., THOMAS, K., PAXSON, V., AND ZHANG, M. 2010. @spam: The underground on 140 characters or less. In *Proceedings of the 17th ACM Conference on Computer and Communications Security (CCS'10)*. 27–37.

GYONGYI, Z., BERKHIN, P., GARCIA-MOLINA, H., AND PEDERSEN, J. 2006. Link spam detection based on mass estimation. In *Proceedings of the 32nd International Conference on Very Large Data Bases (VLDB'06)*. 439–450.

GYONGYI, Z., GARCIA-MOLINA, H., AND PEDERSEN, J. 2004. Combating web spam with trustrank. In *Proceedings of the 30th International Conference on Very Large Data Bases (VLDB'04)*. 576–587.

HU, H., YAN, X., HUANG, Y., HAN, J., AND ZHOU, X. J. 2005. Mining coherent dense subgraphs across massive biological networks for functional discovery. *Bioinf. 21*, 213–221.

HURLEY, N. J., O'MAHONY, M. P., AND SILVESTRE, G. C. M. 2007. Attacking recommender systems: A cost-benefit analysis. *IEEE Intell. Syst. 22*, 3, 64–68.

IRANI, D., WEBB, S., PU, C., AND LI, K. 2010. Study of trend-stuffing on twitter through text classification. In *Collaboration, Electronic Messaging, Anti-Abuse and Spam Conference (CEAS'10)*.

KOUTRIKA, G., EFFENDI, F. A., GYONGYI, Z., HEYMANN, P., AND GARCIA-MOLINA, H. 2008. Combating spam in tagging systems: An evaluation. *ACM Trans. Web 2,* 4, 1–34.

LAM, S. K. AND RIEDL, J. 2004. Shilling recommender systems for fun and profit. In *Proceedings of the 13th International Conference on World Wide Web (WWW'04)*.

LEE, K., CAVERLEE, J., CHENG, Z., AND SUI, D. Z. 2011a. Content-driven detection of campaigns in social media. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management (CIKM'11)*. 551–556.

LEE, K., CAVERLEE, J., AND WEBB, S. 2010. Uncovering social spammers: Social honeypots + machine learning. In *Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'10)*. 435–442.

LEE, K., EOFF, B. D., AND CAVERLEE, J. 2011b. Seven months with the devils: A long-term study of content polluters on twitter. In *Proceedings of the 5th AAAI International Conference on Weblogs and Social Media (ICWSM'11)*.

LEVENSHTEIN, V. 1966. Binary codes capable of correcting deletions, insertions and reversals. *Soviet Phys. Doklady 10*, 707.

LEVIEN, R. AND AIKEN, A. 1998. Attack-resistant trust metrics for public key certification. In *Proceedings of the 7th USENIX Security Symposium*.

LIM, E. P., NGUYEN, V. A., JINDAL, N., LIU, B., AND LAUW, H. W. 2010. Detecting product review spammers using rating behaviors. In *Proceedings of the ACM Conference on Information and Knowledge Management (CIKM'10)*.

MANNING, C. D., RAGHAVAN, P., AND SCHTZE, H. 2008. *Introduction to Information Retrieval*. Cambridge University Press.

MANNING, C. D. AND SCHUTZE, H. 1999. *Foundations of Statistical Natural Language Processing*. MIT Press.

MEHTA, B. 2007. Unsupervised shilling detection for collaborative filtering. In *Proceedings of the 22nd National Conference on Artificial Intelligence (AAAI'07)*.

MEHTA, B., HOFMANN, T., AND FANKHAUSER, P. 2007. Lies and propaganda: Detecting spam users in collaborative filtering. In *Proceedings of the 12th International Conference on Intelligent User Interfaces (IUI'07)*. 14–21.

MOTOYAMA, M., MCCOY, D., LEVCHENKO, K., SAVAGE, S., AND VOELKER, G. M. 2011. Dirty jobs: The role of freelance labor in web service abuse. In *Proceedings of the 20th USENIX Security Symposium*.

MUI, L., MOHTASHEMI, M., AND HALBERSTADT, A. 2002. A computational model of trust and reputation for e-business. In *Proceedings of the 35th Annual Hawaii International Conference on System Sciences (HICSS'02)*. 188.

MUKHERJEE, A., LIU, B., WANG, J., GLANCE, N., AND JINDAL, N. 2011. Detecting group review spam. In *Proceedings of the 20th International Conference Companion on World Wide Web (WWW'11)*. 93–94.

NIENNATTRAKUL, V. AND RATANAMAHATANA, C. A. 2007. Inaccuracies of shape averaging method using dynamic time warping for time series data. In *Proceedings of the 7th International Conference on Computational Science (ICCS'07)*.

NTOULAS, A., NAJORK, M., MANASSE, M., AND FETTERLY, D. 2006. Detecting spam web pages through content analysis. In *Proceedings of the 15th International Conference on World Wide Web (WWW'06)*. 83–92.

O'MAHONY, M., HURLEY, N., AND SILVESTRE, G. 2002. Promoting recommendations: An attack on collaborative filtering. In *Proceedings of the 13th International Conference on Database and Expert Systems Applications (DEXA'02)*. 494–503.

PETITJEAN, F., KETTERLIN, A., AND GANCARSKI, P. 2011. A global averaging method for dynamic time warping, with applications to clustering. *Pattern Recogn. 44*, 678–693.

RATKIEWICZ, J., CONOVER, M., MEISS, M., GONCALVES, B., FLAMMINI, A., AND MENCZER, F. 2011. Detecting and tracking political abuse in social media. In *Proceedings of the 5th AAAI International Conference on Weblogs and Social Media (ICWSM'11)*.

RAY, S. AND MAHANTI, A. 2009. Strategies for effective shilling attacks against recommender systems. In *Proceedings of the 2nd ACM SIGKDD International Workshop on Privacy, Security, and Trust in KDD*.

SAHAMI, M., DUMAIS, S., HECKERMAN, D., AND HORVITZ, E. 1998. A bayesian approach to filtering junk e-mail. In *Proceedings of the ICML Workshop on Learning for Text Categorization*.

SU, X.-F., ZENG, H.-J., AND CHEN, Z. 2005. Finding group shilling in recommendation system. In *Proceedings of the 14th International Conference on World Wide Web (WWW'05)*. (Special Interest Tracks and Posters).

THEOBALD, M., SIDDHARTH, J., AND PAEPCKE, A. 2008. Spotsigs: Robust and efficient near duplicate detection in large web collections. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'08)*.

TOMITA, E., TANAKA, A., AND TAKAHASHI, H. 2006. The worst-case time complexity for generating all maximal cliques and computational experiments. *Theor. Comput. Sci. 363*, 28–42.

TREC. 2004. Terabyte track. http://www-nlpir.nist.gov/projects/terabyte/.

TREC. 2007. Spam track. http://plg.uwaterloo.ca/~gvcormac/treccorpus07/.

TWITTER. 2012. The twitter rules. http://support.twitter.com/articles/18311-the-twitter-rules.

VOORHEES, E. M. AND DANG, H. T. 2005. Overview of the trec 2005 question answering track. In *Proceedings of the 14th Text Retrieval Conference (TREC'05)*.

WANG, G., WILSON, C., ZHAO, X., ZHU, Y., MOHANLAL, M., ZHENG, H., AND ZHAO, B. Y. 2012. Serf and turf: Crowdturfing for fun and profit. In *Proceedings of the 21st International Conference on World Wide Web (WWW'12)*.

WANG, N., PARTHASARATHY, S., TAN, K.-L., AND TUNG, A. K. H. 2008. Csv: Visualizing and mining cohesive subgraphs. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD'08)*. 445–448.

WEBB, S., CAVERLEE, J., AND PU, C. 2006. Introducing the webb spam corpus: Using email spam to identify web spam automatically. In *Proceedings of the Conference on Email and Anti-Spam (CEAS'06)*.

WU, B. AND DAVISON, B. D. 2005. Identifying link farm spam pages. In *Proceedings of the 14th International Conference on World Wide Web (WWW'05)*. (Special Interest Tracks and Posters).

WU, B., YANG, S., ZHAO, H., AND WANG, B. 2009. A distributed algorithm to enumerate all maximal cliques in mapreduce. In *Proceedings of the 4th International Conference on Frontier of Computer Science and Technology*.

WU, G., GREENE, D., SMYTH, B., AND CUNNINGHAM, P. 2010. Distortion as a validation criterion in the identification of suspicious reviews. In *proceedings of the SIGKDD Workshop on Social Media Analytics (SOMA'10)*.

YOSHIDA, K., ADACHI, F., WASHIO, T., MOTODA, H., HOMMA, T., NAKASHIMA, A., FUJIKAWA, H., AND YAMAZAKI, K. 2004. Density-based spam detector. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining (KDD'04)*. http://pdf.aminer.org/000/473/526/density_based_spam_detector.pdf.

YOUNG, J., MARTELL, C., ANAND, P., ORTIZ, P., AND GILBERT IV, H. 2011. A microtext corpus for persuasion detection in dialog. In *Proceedings of the 25th Workshops at the AAAI Conference on Artificial Intelligence*.

ZHANG, Q., ZHANG, Y., YU, H., AND HUANG, X. 2010. Efficient partial-duplicate detection based on sequence matching. In *Proceeding of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'10)*.

ZIEGLER, C.-N. AND LAUSEN, G. 2005. Propagation models for trust and distrust in social networks. *Inf. Syst. Frontiers 7,* 4–5, 337–358.

Received February 2012; revised September 2012; accepted September 2012