Predicting Future Popularity Trend of Events in Microblogging Platforms

Manish Gupta
UIUC
Champaign, IL
gupta58@illinois.edu

Jing Gao
State Univ. of New York
Buffalo, NY
jing@buffalo.edu

ChengXiang Zhai
UIUC
Champaign, IL
czhai@illinois.edu

Jiawei Han
UIUC
Champaign, IL
hanj@cs.uiuc.edu

ABSTRACT

The fast information sharing on Twitter from millions of users all over the world leads to almost real-time reporting of events. It is extremely important for business and administrative decision makers to learn events' popularity as quickly as possible, as it can buy extra precious time for them to make informed decisions. Therefore, we introduce the problem of predicting future popularity trend of events on microblogging platforms. Traditionally, trend prediction has been performed by using time series analysis of past popularity to forecast the future popularity changes. As we can encode the rich Twitter dynamics using a rich variety of features from microblogging data, we explore regression, classification and hybrid approaches, using a large set of popularity, social and event features, to predict event popularity. Experimental results on two real datasets of 18382 events extracted from ~133 million tweets show the effectiveness of the extracted features and learning approaches. The predicted popularity trend of events can be directly used for a variety of applications including recommendation systems, ad keywords bidding price decisions, stock trading decisions, dynamic ticket pricing for sports events, etc.

Keywords

Twitter, Popularity trend prediction, Retweet versus popularity ratios, Events, Aspects, Microblogs

INTRODUCTION

On an average, about 1620 tweets[1] are posted on Twitter every second. Twitter users act as a massive sensor force by reporting events in real time. Trendy events get discussed exhaustively on Twitter and event analysis can be used to extract mass opinions from such discussions. Knowledge about current events is helpful but knowledge about the future trend of an event is even more precious. Such knowledge is critical for a variety of business and administrative applications. Future popularity trend of events indicates the intensity with which people would react, and hence can directly impact policy decisions.

ASIST 2012, October 28-31, 2012, Baltimore, MD, USA.

Applications

Trend prediction is important for many business and administrative decision makers. To promote business, sales prediction and sensing of future consumer behavior can help business decision makers in budget planning, marketing campaigns and resource allocation. Predictive models can be used to predict the outcome of political, sports, and entertainment (movie release, music album launch, TV shows) events. News media may promote news stories which are expected to make greater impact to attract more readers. Political leaders can tune their speeches based on the expected near-future responses to their previous speeches, during election campaigns. If the masses do not respond to certain points in the manifesto, they may be de-amplified over others for future speeches. An estimate of the turnout at a stadium based on Twitter discussions about a game can help in deciding ticket prices dynamically. Also, knowing beforehand the number of people that could turn up, organizers can plan for additional staff to inform and direct passengers as well as ensure a greater availability of public transportation mechanisms[2].

The following examples further illustrate why predicting near-future trends is important. A good or a bad news about a company may not have much impact on the stock price of a company unless it gets viral thanks to the network effect of social networks. Knowledge about future popularity trend of such news could be especially beneficial to day traders. Advertisers (keyword-targeted campaigns) might use popularity trend analysis to buy ad space proactively (at lower prices) next to search result pages. Event words related to events with "high increase" future popularity trends can be chosen if the advertisers know future popularity events in advance. Online traffic, a business might receive, due to network effect of social sites is very helpful for small businesses. However, too much traffic can cause huge losses to small businesses[3] which have limited

[1] http://blog.twitter.com/2011/03/numbers.html

[2] http://tinyurl.com/7j9773o

[3] http://tinyurl.com/7h7347v

small manpower. Beforehand knowledge of expected traffic can help them have better control, e.g., by closing the offer early. Similarly, websites can proactively adjust their server bandwidth or CPU processing power, if they are notified about the potential of some of their webpages (e.g., blogs, products) getting viral on Twitter.

Existing Solutions

Many tools have been developed to conduct trend analysis. For example, Google Trends[4] can be exploited effectively by online marketers, bloggers, and opinion tracking companies. However, Google Trends does not plot any social volume for events nor does it predict future popularity trends of events. Besides Google trends, a large number of applications have been built around analyzing and churning Twitter data[5]. However, none of them tries to perform futuristic predictions either. Even Twitter Trends shows "what's hot now" instead of "what will continue to be hot in the near future". Having a system that can predict future popularity trend of an event will help decision makers plan ahead, prioritize tasks, and make better decisions. Knowledge about future popularity trend of events is even useful for Twitter itself. It can recommend events that are more likely to have popularity increase, rather than recommend currently popular events. To the best of our knowledge, this is the first work on future popularity trend prediction of events for microblogging platforms.

Challenges

Predicting future popularity of events is challenging due to the following reasons: (1) Popularity of events on Twitter can be characterized using various factors, e.g., ramp up rate, scale back rate, number of local peaks, etc. (2) Twitter is an online social network website where content is user-generated. Thus, there exist events that are not so popular in news media, but get lots of attention and discussions among Twitter users. (3) Events on Twitter get popular often because of the propagation effect of the network. (4) As new aspects of a news story unfold, discussions related to that story on Twitter gain new peaks of popularity. Thus, in a way, future popularity of an event is correlated with the number of an event's new aspects, which are revealed gradually. Therefore, due to the heterogeneous factors that affect events' popularity, we must develop a method that takes all these aspects into consideration.

Brief Overview of the Proposed Approach

Our goal is to estimate future popularity of events in the next time interval. As Twitter is a social network website, there exist a variety of features that contribute to the future popularity trend. In this paper, we carefully select and encode in-Twitter and out-of-Twitter dynamics as features. Our rich feature set includes: (1) *Popularity* (2) *Ratios* (3) *Aspects* and *Subordinate Words* (4) *URL* (5) *Friends* and *Followers* and (6) *Event Category*.

We tried multiple algorithms to predict popularity trend based on these features, including linear regression, classification, time series and hybrid approaches. We observe that classification models perform better than the other approaches. We also investigate the discriminativeness of features. Interestingly, *Ratios* turn out to be the most discriminative feature set, performing significantly better than past tweets (*Pop*) or retweets (*RT*). *Aspects* and *Subordinate Words* related to the event are more informative than *URLs* or *Social* features for this task. On the other hand, in the initial phases of an event "life", *URLs* act as a good predictor while *Ratios* features perform better in the later phases.

Summary

Our key contributions are as follows. (1) We introduce a novel problem of predicting future popularity trend of events for microblogging platforms. (2) We propose an event detection scheme designed specifically for the purpose of popularity prediction. Further, we systematically explore a huge set of features, which capture Twitter dynamics effectively, using various learning approaches for our problem. (3) Experimental results on two real datasets show that the proposed features are discriminative and effective in detecting popularity trend of events. We also perform an in-depth analysis of a large variety of experimental settings like varying training data size, analysis for different categories and ranks of events, cross dataset analysis, etc.

This paper is organized as follows. In the next section, we formalize the problem. Then, we briefly describe the proposed popular event detection methodology. We discuss features and learning approaches for predicting the future popularity trend of events on Twitter. Next, we discuss datasets and experimental results with detailed insights. Finally, we discuss related work and conclude the paper.

FUTURE POPULARITY TREND PREDICTION PROBLEM

Let us begin by defining a few basic concepts.

Event: An event e corresponds to a real world event expressed as a set of words chosen from the vocabulary V. The set of words forming the event can be divided into two groups: core words C_e and subordinate words S_e. Thus, $e = C_e \cup S_e$.

Core words are the main representative words for the event, e.g., Twitter Trends words. These are bursty keywords and are identified using a temporal TF-IDF definition as detailed later. Subordinate words are words which appear very frequently together with the core words in many tweets. Note that a subordinate word for an event e_1 might be a core word for some other similar event e_2. Representing events in terms of bursty keywords has been proposed earlier (Kleinberg, 2003; Mathioudakis and

[4] http://www.google.com/trends

[5] http://www.squidoo.com/twitterapps

Koudas, 2010). Such a definition of an event can be considered similar to the traditional definition of a topic in information retrieval. A topic extracted from tweets belonging to a short time interval can be regarded as a coherent event. Like a topic, an event is a probability distribution over words with high probability mass concentrated on core words, relatively lower mass on subordinate words and zero mass on other words.

For example, for the Oil Spill Disaster event, $C_e = \{$bp, oil, spill$\}$ while $S_e = \{$deepwater, explosion, marine, life, scientists, static, kill, mud, cement, plug, 9m, gushed, barrels, gulf, mexico$\}$.

To compute popularity of an event in a time interval, we need to map tweets to events. A tweet is a short document and so it can be represented as a bag of words. Intuitively, a tweet t can be related to an event e if most of its words belong to $C_e \cup S_e$. Formally, we define popularity of an event as follows.

Popularity of an Event: Contribution of the tweet m (containing n unique words) to the event e, $c(m, e)$, is defined as $\frac{\sum_{w \in m} I[w \in C_e \cup S_e]}{n}$, where I[.] is an indicator function which is 1 when the condition is true and 0 otherwise. If $c(m, e) \geq \psi$, we consider that the tweet m supports the event e, where ψ is a constant threshold. Popularity of an event e in a time interval t, $P_t(e)$, is defined as the number of tweets in t, supporting event e.

Future Popularity Trend Prediction Problem

Based on the above definitions, we can formulate the future popularity trend prediction problem as a regression problem. However, predicting the exact popularity value is not easy. Also, for a large number of applications, one does not need fine granularity predictions. Hence, rather than casting the problem as a regression problem, we choose to frame it as a K-class classification problem. E.g., fixing K=3, the classes could be "increase", "decrease" or "almost no change" in popularity.

Input: A set of candidate events C which have high popularity in a time interval t.

Output: A popularity trend class label ($k \in K$) corresponding to the transition from time interval t to the time interval $t + 1$ for each event $e \in C$.

Problem: Predict the future popularity trend class label by effectively exploiting all the information available in terms of feature values.

Exact ranges of popularity changes corresponding to each of these class labels can be fixed by performing clustering on the training instances with respect to the percent popularity change values. Note that percentage change in popularity can be as low as -100% but is unbounded on the positive side. Hence, care should be taken to ignore the positive outliers before clustering. Number of clusters (K) can be set as per the needs of the application.

Note that this problem is different from existing trend analysis work, because we aim at predicting the popularity trend of events in the next time interval instead of the current interval. Also, the proposed problem is different from predicting whether an event will become popular or not in the future because we focus on the changes in popularity, rather than absolute popularity.

IDENTIFYING CURRENTLY POPULAR EVENTS

In the previous section, we formalized the problem of predicting future popularity trend of events. In this section, we evaluate various alternatives for event detection modeling and then briefly describe an aspect-based agglomerative clustering approach for event detection.

Event Detection Models

Let us discuss potential candidate techniques for event detection and why none of them are directly applicable for the proposed task. **Latent Dirichlet Allocation (LDA) (**Blei et al., 2003) is a technique for identifying topics from text, but (1) it needs number of topics as input; and (2) it often mixes multiple events together when used over a single document of all tweets in a time interval. This happens mainly because of numerous spam tweets containing words from multiple events. Hence, we prefer to work on groups of words rather than individual words. The **phrase graph generation method** (Sharifi et al., 2010) summarizes tweets related to a trending phrase. But, it gives much importance to the position of the words and requires that all the words including stop words be arranged in a particular sequence only, across a large number of tweets. However, tweets are usually written in an informal way and hence may not contain exact phrases. **Twitter Trends Phrases** focus only on bursty events. So, Trends phrases may miss certain events if these events have been popular for quite some time, even if the event is still quite popular. This makes it difficult to track the events in a continuous manner. **GroupBurst** (Mathioudakis and Koudas, 2010) detects events by combining two core words together if they co-occur in many tweets. While this works quite well, it may fail if a large number of tweets contain more than one event. Often promotional tweets include many popular words from multiple events as hashtags. While these approaches work for a variety of applications, they may not be suitable for the proposed task. Therefore, we design a more robust approach to produce candidate events. However, the proposed popularity prediction method in the next section is general and can be applied on currently popular events detected using any mechanism.

Event Detection Methodology

Figure 1 summarizes the proposed event detection methodology. We pre-process tweets, get Twitter-specific list of stop words and then find core words related to events. Next, we find subordinate words which form the context for the core words. We group these subordinate words into sub-topics (or aspects) of an event. Core words (with their aspects) can be merged using an agglomerative clustering algorithm to obtain a set of candidate events.

Thus, an event is expressed as a collection of core words and aspects. Finally, spam and non-English events are removed to obtain a list of currently popular events. In the following, we briefly describe some of the important modules of this event detection method.

Stopwords and Average Word Frequency

For a document stream, stopwords should be words which have high document frequency and are also popular across multiple time intervals. We consider a word as a stopword if it lies within top few frequent words for a large number of time intervals of the tweet stream. We also add standard IR stopword list supplied by the Java IR package Iglu[6] to the stopword list. We analyze the word frequencies over the entire dataset to obtain the average daily frequency of each word (f_{avg}).

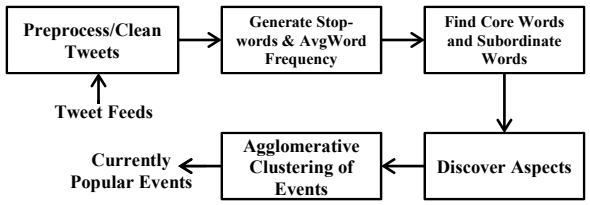

Figure 1: Event Detection Pipeline

Core Words and Subordinate Words

Intuitively, words which become suddenly popular in the tweet stream are the best representatives of an emerging event. We rank words by a temporal TF-IDF score defined as follows:

$$\text{Core Word Score: } CWS(w) = \frac{f(w)}{f_{avg}(w)} \quad (1)$$

where $f(w)$ is the frequency of the word w in the current time interval. Then, we select top few words with highest CWS scores as core words.

Subordinate words for the event are words that occur quite frequently with a core word in the current interval of the tweet stream. Let us denote the co-occurrence frequency of word w and w' by $f(w, w')$. For each core word w, the words w' with frequency $(f(w') \geq f_{sub})$ and high co-occurrence frequency $(f(w, w') \geq f_{cooccur})$ over the time interval form the set of subordinate words.

Aspects

Now we have an event represented as $e = (c, S_e)$ where c is the core word and S_e is the set of subordinate words. Next, we can find sub-topics (or aspects) for event e as follows. Let T_e be the set of tweets related to the event (i.e. tweets that contain the core word c). For every such tweet $t \in T_e$, we find all the words present in the tweet such that they belong to S_e. Let us call this maximal subset of words that belong to both S_e and t as S_{et}. Here, we assume that one tweet discusses only one aspect. We compute S_{et} for each tweet $t \in T_e$. Most frequent S_{et}'s are called aspects of event e. The core word is then also appended to each aspect. At this stage, an event (a topic) is represented by a core word and a set of aspects (sub-topics).

Events

Agglomerative clustering is then applied on the space of aspects to cluster events. Clustering is initialized by considering each core word as a cluster. We merge two clusters if they have a large number of matching sub-topics. Such merging of clusters is performed iteratively until no more merges can be done. Different from *GroupBurst* (Mathioudakis and Koudas, 2010), we do not cluster events based on the co-occurrence of words, instead, we perform clustering in a much richer space. Events are represented by a set of core words, set of subordinate words, and a set of sub-topics (or aspects). We give an example illustrating core words and aspects of a particular event as follows. The event "Facebook CEO Mark Zuckerberg files for IPO" could be described by core words "ipo, facebook, zuckerberg, files". Some of the aspects discovered by our methodology were "facebook, ipo, privacy, filing, documents", "facebook, ipo, zuckerberg, billions", "200, choe, david, ipo, facebook", "filing, hacker, ipo, zuckerberg", "artist, graffiti, ipo, facebook".

PREDICTING FUTURE POPULARITY OF EVENTS

In the previous section, we presented an event detection pipeline which used agglomerative clustering in the space of aspects to form coherent events with detailed context. The event detection module outputs a list of top few popular events for every hour. In this section, we will discuss how we extract features related to event popularity, sub-topics, event words, out-of-Twitter discussions, presence in news, and social network structure. These features could then be exploited by various learning approaches to solve the proposed problem.

Popularity Features (Pop and RT)

Traditionally, temporal data mining tasks have used historical data value series to predict future values. Such tasks generally consider recent history as more important than past history. Hence, we include the popularity for the past 24 hours (recent history at a fine level) and popularity across past 10 days (past history at a coarser level) as features.

Popularity (Pop)

To remove daily trends, we normalize the popularity for each hour (relative to the average traffic during that hour of the day) to avoid such pseudo-"highly negative" popularity change instances. These popularity-based features can be represented as $F_{Pop} = \{P_{hour_0}(e),\ P_{hour_{-1}}(e), \ldots, P_{hour_{-23}}(e), P_{day_0}(e), P_{day_{-1}}(e), \ldots, P_{day_{-9}}(e)\}$, where $P_{hour_{-r}}(e)$ and $P_{day_{-r}}(e)$ are the number of tweets related to the event e, r hours and r days back respectively. A basic analysis provides us the following insights. (1) Breaking news events show clear distinguishable peaks.

[6] http://sourceforge.net/projects/iglu-java/

They have a very short left duration but a long right duration. Users discuss a lot about such events after the event has already taken place in the actual world, for example, the event "7.2 magnitude Turkey earthquake". (2) Predictable events often are not characterized by clear peaks. They have a long left duration and optionally also have a long right duration. For example, events such as "elections", "christmas", "weather changes" ramp up very slowly. (3) About half of the events have more than one mode (**local maxima**) over their life cycle. (4) Events differ a lot with respect to their highest popularity and also with respect to their duration. In summary, events on Twitter show different kinds of temporal profiles with respect to popularity. If the training set can capture such variety of profiles, Pop features could be quite useful. However, they may not work well for certain categories or for new events.

Retweets (RT)

An event can get popular if it receives either many original tweets or retweets. Retweeting takes lower effort compared to posting new tweets, as new tweets need new content. The retweet-based features can be represented as $F_{RT} = \{R_{hour_0}(e),\ R_{hour_{-1}}(e), \dots,\ R_{hour_{-23}}(e),\ R_{day_0}(e),\ R_{day_{-1}}(e),\ \dots, R_{day_{-9}}(e)\}$, where $R_{hour_{-r}}(e)$and $R_{day_{-r}}(e)$ are the number of retweets related to the event e, r hours and r days back respectively. If an event has a large number of initial original tweets but relatively few retweets, the event may not last long. In contrast, an event with many original tweets and many follow-up retweets is the one that has caught momentum and is likely to keep attracting attention. Different features may behave very differently during different phases of an event's life. Hence, to capture the dynamics related to the life cycle of an event, we include the number of hours for which the event has been popular (age_{Pop}) and the number of hours for which the event is being retweeted (age_{RT}) in our feature set. We also include a feature $hour_0$ which is the hour of the day when the event is detected as popular.

Ratios Features

In the *Popularity* feature set, we considered *Pop* and *RT* features. However, relative change in popularity of event for the past few consecutive pairs of time intervals may also be interesting. Hence, we consider $Pop{:}Pop$ features $F_{Pop:Pop} = \{PP_{hour_{0,-1}}(e),\ PP_{hour_{-1,-2}}(e), \dots,\ PP_{hour_{-22,-23}}(e),\ PP_{day_{0,-1}}(e),\ PP_{day_{-1,-2}}(e),\ \dots,\ PP_{day_{-8,-9}}(e)\}$, where $PP_{hour_{-r,-(r+1)}}(e)$ and $PP_{day_{-r,-(r+1)}}(e)$ are the ratios $P_{hour_{-r}}(e){:}\,P_{hour_{-(r+1)}}(e)$ and $P_{day_{-r}}(e){:}\,P_{day_{-(r+1)}}(e)$ respectively. Similarly, we also include the $RT{:}RT$ features in our feature set. These features capture "sudden shocks" in the Pop and RT time series more clearly. The ratio of retweets to all tweets for an event seems to be a good indicator about how bursty it might get in the next time interval. Hence, we include the $RT{:}Pop$ features. $F_{RT:Pop} = \{RP_{hour_0}(e),\ RP_{hour_{-1}}(e), \dots,\ RP_{hour_{-23}}(e),\ RP_{day_0}(e),\ RP_{day_{-1}}(e),\ \dots,\ RP_{day_{-9}}(e)\}$, where $RP_{hour_{-r}}(e)$ and $RP_{day_{-r}}(e)$ are the ratios $R_{hour_{-r}}(e){:}\,P_{hour_{-r}}(e)$ and $R_{day_{-r}}(e){:}\,P_{day_{-r}}(e)$ respectively. This ratio intuitively captures how much percent of current popularity is due to retweets versus organic original tweets. It provides a clear identification of the proportion of users interested in sharing versus creating fresh new content about the event.

Social Features (Followers and Friends)

Followers on Twitter are one-way friends. Friends represent two-way friendships. Popularity of an event can increase if there are new fresh tweets or there are a lot of retweets. Also, on an average, a tweet will be retweeted more if the original user has a large number of followers and friends (sharing capability). Thus, number of followers and friends of users who are currently tweeting or retweeting about the event is an important feature. We represent our social features as $F_{FF} = \{Fo_{hour_0}(e),\ Fo_{hour_{-1}}(e), \dots,\ Fo_{hour_{-9}}(e),\ Fr_{hour_0}(e),\ Fr_{hour_{-1}}(e),\ \dots, Fr_{hour_{-9}}(e)\}$, where $Fo_{hour_{-r}}(e)$ and $Fr_{hour_{-r}}(e)$ are the number of followers and friends respectively, of people who tweeted about the event e, r hours back.

Out-of-Twitter Popularity (URLs)

Events are posted on Twitter in real-time. Events that are sensationalized by media (like Casey Anthony's murder trial or Amanda Knox case) are expected to last for multiple days. Users who post the original tweets related to the event often post a URL which acts as an evidence of the genuineness of the news. URLs often refer to extensively written articles on news websites or blog pages, related to the topic of the tweet. Therefore, the number of unique URLs posted by Twitter users can approximate the popularity of the event in out-of-Twitter world. As more and more aspects of the event appear in the news, users keep on posting new URLs. Thus, as the real world event story develops, the number of unique URLs continues to grow. Hence, we include time series of the number of links posted by Twitter users related to the event in the past 10 hours as *URLs* features. The *URLs* features can be represented as $F_{URL} = \{U_{hour_0}(e),\ U_{hour_{-1}}(e), \dots,\ U_{hour_{-9}}(e)\}$, where $U_{hour_{-r}}(e)$ is the number of tweets containing URLs about the event e, r hours back. As our experiments show, these features play a major role when the event has just started, or for events of certain categories like business or sports for which news is a better media for discussions

Event Features (Aspects and Subordinate Words)

All the features mentioned above are essentially features of tweets or that of users. Feature values for an event can be computed by aggregating the values across all tweets or users related to the event. In the Event Features feature set, we include features that are inherently related to the event.

Aspects

A news story (event) in real world is generally composed of a smooth flow of different aspects. If the news story keeps

evolving (new facts get revealed or new sub-events take place), people keep discussing about it. When lots of aspects are discussed about an event, the event will last longer with a high probability. When relatively low frequency aspects are discussed a lot in a very short duration, the event may not be a coherent event and may just represent social gossip. On the other hand, a genuine news event is characterized by a smooth increase in the number of aspects being discussed over time. If we notice that the number of unique aspects being discussed are decreasing, we know that the discussion related to the event is losing steam. Based on these intuitions, we include aspect-based time series as features. Our aspect-based features can be represented as $F_{Aspect} = \{A_{hour_0}(e), A_{hour_{-1}}(e), \dots, A_{hour_{-9}}(e)\}$, where $A_{hour_{-r}}(e)$ is the number of aspects for the event e, r hours back.

Subordinate Words

Subordinate words of an event form the rich context of an event. If an event consists of a sequence of relatively small number of subordinate words each with high frequency, one would expect the event to be somewhat peaked in nature. Such an event would be expected to have a short duration with a large number of highly negative or highly positive popularity changes during this short interval. On the other hand, if the event consists of a large number of subordinate words, the event would have many sub-topics being discussed, each of which may ramp up or slack down at different points in the event life time. Hence, such an event may last for a longer time and would be characterized by many relatively flat or small negative or positive popularity changes. The *Subordinate Word* features can be represented as $F_{SW} = \{S_{hour_0}(e), S_{hour_{-1}}(e), \dots, S_{hour_{-9}}(e)\}$, where $S_{hour_{-r}}(e)$ is the number of subordinate words for event e, r hours back.

Event Category

Events belonging to categories such as entertainment, sports and politics are generally short-lived. These are the categories where fresh news keeps coming in very frequently. As a result, discussions on Twitter switch to new events quickly for such categories. On the other hand, events in some categories like technology last for longer time period. We detect the category of an event by looking up the words related to the event in the headlines of popular news websites and extracting their ways of categorization. For example, the event represented by the set of words "chronicles, nonton, voyage, 3d, treader, nntn, narnia, rapunzel" will be categorized into the category *entertainment* because Yahoo! News[7] and WorldNews[8] associate it with that category. We categorize the events not found in news to *Others* category.

[7] http://news.yahoo.com/entertainment

[8] http://wn.com/Entertainment

In this section, we discussed different features that could be useful to predict the future popularity trend of events. Most of the extracted features are Twitter-centric, except for the *URLs* feature set which captures out-of-Twitter popularity. Features such as total blog volume, degree of event coverage in news, and other social media may be interesting to explore. However, these features may be redundant because Twitter core and subordinate words almost cover all the words used in news headlines. Also, Twitter network dynamics are quite different from that of news and blogs, so these features may not complement the dynamics in Twitter. So, we focus on Twitter-centric features in this paper.

Learning Approaches

We experimented with a variety of regression models, classification algorithms and hybrid methods.

Linear Regression

Linear regression model captures the periodicity and trend of data and thus can forecast the future of the events. A linear regression model is represented by Eq. 2.

$$x_t = \sum_{i=1}^{h} \beta_i x_i + \epsilon_t \quad (2)$$

where x_t is the predicted future value and other x_i's are the past popularity values, h is the total history size and ϵ_t is the noise term. β's can be learned using multiple training instances.

Auto-Regression AR(p)

A global β across all instances may not be a good fit, so we explore auto-regression model, which is similar to linear regression, except that learning is done using the past series for the same instance. An auto regression model with lag p can be expressed as shown in Eq. 3.

$$x_t = c + \sum_{i=1}^{p} \psi_i x_{t-i} + \epsilon_t \quad (3)$$

where ψ's are the parameters, c is a constant and ϵ_t is white noise.

Auto-Regressive Moving Average ARMA(p,q)

An auto-regression model can be combined with a moving averages model to obtain an ARMA (autoregressive moving averages) model. An $ARMA(p,q)$ model with lag p and q for the AR and the MA parts respectively can be expressed as shown in Eq. 4.

$$x_t = c + \sum_{i=1}^{p} \psi_i x_{t-i} + \sum_{i=1}^{q} \theta_i \epsilon_{t-i} + \epsilon_t \quad (4)$$

Thus, a $ARMA(p,q)$ model is a linear regression of the current value of the series against previous (unobserved) q white noise error terms or random shocks, and the previous p values of the series.

Vector Auto-Regression (VAR)

Each event has multiple other associated time series like retweets, ratios, followers, etc. This motivates us to explore a *Vector Auto-Regression (VAR) model* which is a statistical model used to capture the linear interdependencies among multiple time series. A VAR consists of a set of K endogeneous variables $x_t = (x_{t_1}, x_{t_2}, \dots, x_{t_K})$. The $VAR(p)$ process is then defined as follows.

$$x_t = A_0 + \sum_{i=1}^{p} A_i x_{t-i} + \epsilon_t \quad (5)$$

where A_0 is a vector of intercept parameters and other A_i's are $K \times K$ coefficient matrices and ϵ_t represents white noise with mean 0. In our setting, x_{t_1} represents popularity; other x_{t_i}'s could be retweet, ratios, followers, etc. p is the lag parameter. Thus, the first row of A_p denotes the dependence of the future popularity values on popularity, retweets, ratios, followers, etc. p hours back.

Classification Models

We could treat each of the time series values as individual features and learn a model using classifiers such as support vector machines, k-nearest neighbors, and Naïve Bayes or decision trees.

Hybrid Approaches

Regression models such as AR and VAR capture time dependencies (periodicity and trend) but cannot learn across instances. Classification models learn across instances but have no natural way of encoding time dependencies. Therefore, we can also try a hybrid approach as follows. We learn an AR/VAR model for each event. Next, we use the coefficients of the time series model as normalization weights. Using VAR, we replace the feature value x_{t-j_k} with $x_{t-j_k} \times A_j(0,k)$. Such modified feature values capture the time dependency between the future popularity value and the value of the k^{th} attribute, j hours back. Now, one can learn a classification model such as SVM based on modified features.

EXPERIMENTS AND RESULTS

In this section, we discuss our experiments which aim at answering the following questions: (1) How do different modeling approaches (regression, classification and hybrid) compare with each other? (2) How does the category or the current phase of an event play a role in determining its future popularity trend? (3) How does the accuracy of our models vary with the amount of training data, number of features used, and cross-dataset evaluation?

Dataset Details

We used the Twitter Streaming API[9] to obtain about ~133 million tweets for Dec 2010 (D2010) and for Mar 2011 (D2011). For event detection, we tested different values for f_{sub} and $f_{cooccur}$ and observed the quality of events obtained.

Using the proposed event detection methodology with f_{sub}=5 and $f_{cooccur}$=5, we obtain a maximum of 15 events per hour for both the datasets. We use ψ= 0.5 to identify if a tweet belongs to an event. The prediction task is to accurately predict the future popularity trend for each instance (event) in the next hour. Note that since our task is to predict future popularity trend on Twitter, it is natural to compute the actual label for each instance using event popularity in the next hour on Twitter.

A large number of popular events die out very quickly and hence show negative percent popularity changes. About 22% events have positive popularity changes for the next hour. This motivates us to have one class for the positive range and have more classes for the negative range. Based on the analysis of our datasets, we define the correspondence between popularity changes and the class labels as shown in Table 1.

Popularity Change	Label
Less than -75%	DEC3
-75% to -50%	DEC2
-50% to -25%	DEC1
-25% to 0%	FLAT
More than 0%	INC

Table 1: Popularity Changes and Class Labels

The distribution of the class labels is fairly uniform (each class has ~17-25% instances) for both the datasets. A large percentage of the INC instances belong to TopNews and Others categories, while many DEC3 instances belong to Sports and Others categories. Events in all of these categories are inherently trendy and hence are characterized by a peaked behavior. The Others category mainly consists of events which are non-news events like "myhomelesssignwouldsay". Due to the network effect of Twitter, such "trendy" events often gain public attention very quickly and hence tend to suddenly gain popularity. For Sports events, users' attention is maintained while the event is going on and then suddenly drops soon after the real world event has finished.

Basic Accuracy Results

For accuracy computations, we use the accuracy matrix specified in Table 2. For example, if an instance with an actual label of INC is labeled as FLAT by our method, we consider the accuracy of our approach for that instance as 0.75. This is intuitive as the classes are not completely disjoint, but have a natural ordering and hence the penalty for classifying an INC instance as FLAT should be less than the penalty for classifying it as DEC3.

Table 3 shows the accuracy results using different feature sets and different learning approaches. For classification models, 10-fold cross-validation accuracy is shown. Among classifiers (decision trees, Naïve Bayes and Support Vector Machines (SVMs)), SVMs turn out to be the most accurate for this popularity trend prediction task. Among the AR

[9] https://stream.twitter.com/1/statuses/sample.json

models, we found that AR with lag 1 provides the best results. Among the VAR models, we tried lag values from 2 to 6 and found the best results at lag=5. AR models perform better than the VAR models. Linear Regression model performs well for *Pop* features but does not show much gain in accuracy when all the features are considered together. AR(1) provides good accuracy, but for many instances (~10%), no AR(1) model could be learned, so overall accuracy turns out to be low. When an AR, ARMA or a VAR model could not be learned, we randomly selected a class label for that instance. Comparing all the methods, SVMs using all the features provide the best accuracy. Interestingly, "ratios of retweets and all tweets (*Ratios*)" are the best features, performing significantly better than "past tweets (*Pop*) or retweets (*RT*)". Out-of-Twitter popularity measured by the number of URLs has been found to be a good predictor for other tasks (Petrovic et al., 2011; Suh et al., 2010). However, *Popularity* features (*Pop* and *RT*) perform better than the *Social* and *URLs* features for our task. Number of sub-topics (*Aspects*) or frequent words (*Subordinate Words*) related to the event are better than *URLs* or *Social* features. *Ratios* features perform quite well and the accuracy further improves when we add the *Popularity* (*Pop* and *RT*) features. Overall, when we use all the features with SVM, we obtain the best accuracy of 73.54% and 74.23% for the two datasets respectively. SVMs work well than time series models mainly because time series models cannot capture long distance temporal dependencies effectively. Also time series models do not perform any learning across instances. Thus, the learned model may suffer from sparsity of data, especially for new events. We also tried the hybrid approaches by normalizing feature values using time series model coefficients, but did not find much improvement over the SVM models. Since SVM turns out to be the best model and the feature set *Pop+RT+Ratios* provides reasonable results, we perform further analysis based on SVM classifier results for this feature subset.

Top Discriminative Features

Most of the important features correspond to the values 1-4 hours back or 18-22 hours back. This also explains why time series models with a lag of 5-6 cannot capture all the dependencies. Note that using a lag value of 20 with AR models is not feasible because then there would not be enough data for the auto-regression to be learned.

Classified as → Actual↓	DEC3	DEC2	DEC1	FLAT	INC
DEC3	1	0.75	0.5	0.25	0
DEC2	0.75	1	0.75	0.5	0.25
DEC1	0.5	0.75	1	0.75	0.5
FLAT	0.25	0.5	0.75	1	0.75
INC	0	0.25	0.5	0.75	1

Table 2: Accuracy Matrix for Evaluation

Computation Time

Note that all of these feature values (*Pop, RT, Ratios*) can be efficiently computed by processing a fixed size history of the tweet feed stream and hence involve low computational costs. Also, our event detection mechanism takes less than five minutes to process a feed of one hour on an average. On the other hand, one could also use Twitter Trend words to represent events, rather than using our event detection mechanism. Thus, it is quite realistic to use our future popularity prediction module as part of online applications like an online recommendation system.

Features	Method	D2010	D2011
URLs	SVM	60.58	63.14
Social (Followers + Friends)	SVM	61.48	61.64
Event (Aspects + Subordinate Words)	SVM	67.17	66.66
Pop	SVM	69.96	67.48
Only RT	SVM	63.61	61.46
Pop+RT	SVM	70.47	68.28
Ratios	SVM	72.75	73.21
Pop+RT+Ratios	SVM	72.81	73.43
Pop+RT+Ratios	Decision Trees	70.95	69.79
Pop+RT+Ratios	Naïve Bayes	69.42	66.5
All	SVM	**73.54**	**74.23**
Pop	Linear Regression	70.62	71.5
All	Linear Regression	70.66	69.64
Pop	AR(1)	71.21	70.81
Pop	AR(2)	69.3	69.21
Pop	ARMA(1,1)	69.68	68.96
Pop+RT+Ratios	VAR(2,5)	65.95	65.71

Table 3: Accuracy Results

Feature Selection

We performed feature selection using Information gain, PCA and CfsSubsetEval. Cross-validation using top 50 features (ranked with respect to Information gain) or using all the features selected by PCA or CfsSubsetEval does not provide better accuracy than our *Pop +RT +Ratios* feature set, further pointing out that each of the features plays an important role.

Varying Model Parameters

Next, we study the change in accuracy of our models with change in number of features, size of the training data and the rank of the events in the dataset.

Limiting the Number of Features

Using *Pop+RT+Ratios* set of features, we tested the impact of varying the size of history time series (from 10 days to 2 hours) on the accuracy. We reduced the time series for each of the *Pop*, *RT*, and *Ratios* features. Cross validation accuracy results are shown in Table 4. We observe that reducing the history size to past five hours still provides fairly good accuracy. However, drastically reducing the size of the time series to two, causes significant loss in accuracy.

Varying the Size of Training Data

We varied the amount of training data for the best model for both the datasets. The cross-validation results for different percentage of data used, are shown in Table 5. Even with just 10% of the data, the models provide decent accuracy, and thus the models can be trained using small amounts of data.

High Ranked Events versus Low Ranked Events

Next, we repeated the above experiments using the *Pop* features on datasets which consider events at rank 1-5, 6-

10, 11-15 and 16-20 respectively. We want to check the performance changes for lower ranked events versus the higher ranked events. Table 6 shows the accuracy (cross-validation accuracy within that particular sub-dataset) of the four sub-datasets derived from the two sets of Tweet feeds. Table 6 shows that *Pop+RT+Ratios* features work well for lower-ranked events. Accuracy is a little lower for the top-ranked events. For the top-ranked events, we observed that the *URLs* features have high information gain. We believe this is because popular events are often retweeted and have many URLs associated with them.

Limit	D2010	D2011
10 days	72.81	73.43
24 hours	72.47	70.94
15 hours	71.30	70.22
10 hours	70.81	69.94
5 hours	70.55	69.67
2 hours	68.94	67.98

Table 4: Accuracy (%) when the History Size (i.e. #Features) is Limited

% of data	D2010	D2011
10	72.65	70.93
20	73.00	72.38
40	73.91	72.95
60	73.22	72.74
80	73.28	72.66
100	72.81	73.43

Table 5: Accuracy (%) when the Amount of Training Data is Varied

Varying Train and Test Sets

Next, we perform analysis of our classifier by learning cross dataset models, models for different event categories, and models for different stages of an event life cycle.

Cross Dataset Analysis

Table 7 shows cross-dataset accuracy results. For this experiment, we used the *Pop+RT+Ratios* feature set. Our models provide reasonably good cross dataset accuracy, and thus they generalize well.

Learning Different Models for Different Categories

We trained separate models for each category to observe if the important features vary across categories. For each of the categories we ranked the features using information gain. For most of the categories *Pop* and *RT*:*Pop* Ratio features appear at the top. URLs features have high information gain for Business and Sports categories while Social features are important for the Entertainment and Politics categories. This clearly supports the intuition that entertainment and politics stories get discussed so often and are propagated efficiently through the network, while business and sports stories are more rigorous and are often supported by statistics presented by news agencies. This also explains why a model only based on *Pop+RT+Ratios* does not perform as well as the one using *All* features for these categories as shown in Table 8.

Analyzing Different Phases of an Event Life Cycle

We observed the popularity series of each event for a period of 10 days before and after the hour when it was detected as popular. We computed its left and right duration. Then we categorized each instance into one of the four phases: Phase1 (new), Phase2, Phase3, Phase4 (old). Thus, we divided each of the D2010 and D2011 datasets into four sub-datasets. Next, we used each of these sub-datasets to learn a model using *Pop+RT+Ratios* feature sets and tested it on the other sub-datasets. Table 9 shows the results. The diagonal entries correspond to 10-fold cross validation values. Notice that none of the train sets (except the Phase1 itself) perform well for events in Phase1. Information gain based ranking of features in Phase1 dataset revealed that some of the *URLs* features appear at the top, along with the *Pop*, *RT* and *Ratios* features. On the other hand, for the other phase datasets, almost all features in the top 100 are from *Pop*, *RT* and *Ratios* sets. This is quite interesting. We can explain this based on the following intuition. For a new event, *URLs* are important as there are not many tweets and retweets in history; for other phases, the event has gained sufficient history.

Sub-dataset	D2010	D2011
0to5	70.77	68.32
6to10	73.48	72.36
11to15	74.22	75.21
16to20	73.97	75.97

Table 6: Accuracy (%) for Different Sub-Datasets based on Event Ranks

Train Set	Test Set	Accuracy
D2010	D2011	70.64
D2011	D2010	71.92

Table 7: Cross Dataset Results

Category	D2010	D2011
Technology	74.72	71.12
Top	73.47	72.55
Business	73.44	73.64
Others	73.42	72.00
Entertainment	72.38	70.37
Travel	71.76	67.53
Politics	71.34	71.71
Health	71.27	70.88
Sports	70.12	68.77

Table 8: Accuracy (%) for Different Categories

Train → Test ↓	Phase1	Phase2	Phase3	Phase4
Phase1	70.24,69.86	70.34,70.43	70.45,70.75	70,65.21
Phase2	66.12,67.97	72.35,71.44	76.92,74.73	75.22,73.5
Phase3	62.9,62.43	71.06,68.68	76.64,76.14	75.05,75
Phase4	60.01,59.8	67.82,66.12	74.36,73.85	77.18,75.49

Table 9: Accuracy (%) for Events at Different Phases of Event Life (D2010, D2011)

RELATED WORK

Our work is related to research in the areas of event detection, current event trend analysis and predictive analysis on Twitter and other platforms.

Event Detection

Our future event popularity problem assumes an input of currently popular events as candidates. There has been extensive research for detecting Twitter events like epidemics (Lampos et al., 2010), wildfires, hurricanes, floods (Starbird et al., 2010), earthquakes (Sakaki et al., 2010) and tornados. One can use Twitter Trends words, Latent Dirichlet Allocation (LDA) (Blei et al., 2003) or Phrase Graph Generation Method (Sharifi et al., 2010) for detecting events. However, we want to analyze events in detail so as to be able to derive interesting features which could in turn be useful for future predictions.

Current Event Trend Analysis

Events have been modeled and analyzed over time using keyword graphs (Sayyadi et al., 2009), link-based topic models (Lin et al., 2010), and infinite state automatons (Kleinberg, 2003). (Leskovec et al., 2009; Yang and Leskovec, 2011; Swan et al., 2000) also study event trends.

Predictive Analysis on Twitter and Other Platforms

Previous work discusses a variety of prediction tasks for Twitter like box office forecasting of movies (Asur and Huberman, 2010), predicting retweetability of tweets (Hong et al., 2011; Petrovic et al., 2011; Suh et al., 2010), predicting for a pair of users, whether a tweet written by one will be retweeted by the other user (Zaman et al., 2010), and predicting information diffusion (Yang and Counts, 2010). Such prediction models deal with single messages and hence, cannot be used directly to predict the future popularity of an event, which is a collection of messages and their retweets. As opposed to a message, an event has notions of "lifetime" and the "community discussing the event". Also, our work is focused on generic events unlike movie forecasts (Asur and Huberman, 2010) where specific features like Hollywood Stock Exchange time series could be exploited as features. Besides Twitter, there has been work on future popularity of social media content on Digg and Youtube (Lerman and Hogg, 2010; Szabo and Huberman, 2010). But, the design of the user interface of Twitter and Digg are quite different resulting in a huge difference in social dynamics of the two networks.

CONCLUSIONS

We explored the possibility of detecting news events from Twitter feeds. We proposed an event detection method tuned for the task. We systematically explored the performance of a large set of *URLs*, *Social*, *Event*, *Popularity*, and *Ratios* features using various learning approaches for popularity detection. *Ratios* turn out to be the best features, performing significantly better than "past tweets (*Pop*) or retweets (*RT*)". *Aspects* and *Subordinate Words* related to the event are better than *URLs* or *Social* features for this task. In the initial phases of an event "life", *URLs* act as a good predictor while *Ratios* features perform better in the later phases. SVMs work better than simple time series models that can account only for *Pop* features. The learning models and features were observed to work reasonably well across events in different phases of their life and across categories. We believe that prediction analysis of events from microblogging platforms is a rich area of research. The proposed popularity trend prediction framework can be further extended to deal with some interesting tasks on microblogging data, such as combining it with sentiment analysis for stock price prediction and investigating the discriminative ability of out-of-Twitter phenomena on popularity trend of events on Twitter.

Acknowledgements

Research was sponsored in part by the Cyber Security project (W911NF-11-2-0086) and NSCTA project (W911NF-09-2-0053) of U.S. Army Research Laboratory and NSF IIS-0905215. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

REFERENCES

Asur, S. and Huberman, B. A. (2010). Predicting the Future with Social Media. WIC, 492–499.

Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003). Latent Dirichlet Allocation. JMLR, 3:993–1022.

Hong, L., Dan, O., and Davison, B. D. (2011). Predicting Popular Messages in Twitter. WWW, 57–58.

Kleinberg, J. (2003). Bursty and Hierarchical Structure in Streams. DMKD, V7:373–397.

Lampos, V., Bie, T. D., and Cristianini, N. (2010). Flu Detector-Tracking Epidemics on Twitter. PKDD, 599–602.

Lerman, K. and Hogg, T. (2010). Using a Model of Social Dynamics to Predict Popularity of News. WWW, 621–630.

Leskovec, J., Backstrom, L., and Kleinberg, J. (2009). Meme-Tracking and the Dynamics of the News Cycle. SIGKDD, 497–506.

Lin, C. X., Zhao, B., Mei, Q., and Han, J. (2010). PET: A Statistical Model for Popular Events Tracking in Social Communities. SIGKDD, 929–938.

Mathioudakis, M. and Koudas, N. (2010). TwitterMonitor: Trend Detection over the Twitter Stream. SIGMOD, 1155–1158.

Petrovic, S., Osborne, M., and Lavrenko, V. (2011). RT to Win! Predicting Message Propagation in Twitter. ICWSM.

Sakaki, T., Okazaki, M., and Matsuo, Y. (2010). Earthquake Shakes Twitter Users: Real-Time Event Detection by Social Sensors. WWW, 851–860.

Sayyadi, H., Hurst, M., and Maykov, A. (2009). Event Detection and Tracking in Social Streams. ICWSM.

Sharifi, B., Hutton, M.-A., and Kalita, J. (2010). Summarizing Microblogs Automatically. HLT, 685–688.

Starbird, K., Palen, L., Hughes, A. L., and Vieweg, S. (2010). Chatter on the Red: What Hazards Threat Reveals about the Social Life of Microblogged Information. CSCW, 241–250.

Suh, B., Hong, L., Pirolli, P., and Chi, E. H. (2010). Want to be Retweeted? Large Scale Analytics on Factors Impacting Retweet in Twitter Network. SOCIALCOM, 177–184.

Swan, R. and Allan, J. (2000). Automatic Generation of Overview Timelines. SIGIR, 49–56.

Szabo, G. and Huberman, B. A. (2010). Predicting the Popularity of Online Content. Communications of the ACM, 53:80–88.

Yang, J. and Counts, S. (2010). Predicting the Speed, Scale, and Range of Information Diffusion in Twitter. ICWSM.

Yang, J. and Leskovec, J. (2011). Patterns of Temporal Variation in Online Media. WSDM, 177–186.

Zaman, T. R., Herbrich, R., Van Gael, J., and Stern, D. (2010). Predicting Information Spreading in Twitter. CSSWC NIPS Workshop.