Catching the Long-Tail: Extracting Local News Events from Twitter

Puneet Agarwal, Rajgopal Vaithiyanathan, Saurabh Sharma and Gautam Shroff
TCS Innovation Labs - Delhi
Tata Consultancy Services, 154B, Block A, Sector 63, Noida, Uttar Pradesh 122106, India
puneet.a@tcs.com, rajgopal.v@tcs.com, saurabh38.s@tcs.com, gautam.shroff@tcs.com

Abstract

Twitter, used in 200 countries with over 250 million tweets a day, is a rich source of local news from around the world. Many events of local importance are first reported on Twitter, including many that never reach news channels. Further, there are often only a few tweets reporting each such event, in contrast with the larger volumes that follow events of wider significance. Even though such events may be primarily of local importance, they can also be of critical interest to some specific but possibly far flung entities: For example, a fire in a supplier's factory half-way around the world may be of interest even from afar. In this paper we describe how this 'long tail' of events can be detected in spite of their sparsity. We then extract and correlate information from multiple tweets describing the same event. Our generic architecture for converting a tweet-stream into event-objects uses locality sensitive hashing, classification, boosting, information extraction and clustering. Our results, based on millions of tweets monitored over many months, appear to validate our approach and architecture: We achieved success-rates in the 80% range for event detection and 76% on event-correlation; we also reduced tweet-comparisons by 80% using LSH.

Introduction

A business enterprise can potentially be affected by events that impact any entity in its eco-system, such as customers, partners, governments, competitors, etc.: For example, a fire in the factory of a remotely located supplier half-way around the world can disrupt an enterprise's supply-chain, causing delays and losses. Twitter has become a rich source of breaking news, including news that is local and possibly of limited interest to a wider global audience; in fact, such events may in fact never make it to any news channel, certainly not a global one.

Since the number of messages per event is small, we cannot hope to detect such events by observing trends on keywords, as do most techniques for event detection from Twitter. Correlation of tweets is another challenge if different words are used to describe the same event, correlation between messages through word-to-word similarity does not work. Extraction of properties (e.g. time-of-occurrence, location, etc,) is required to further correlate such potential events and merge them to create structured *event-objects*. Popular techniques of information-extraction fail to give good results because of the informal language used in tweets. Last but not the least, a common event structure is required to represent different types of events so that a single system could consume them all.

In this paper we describe our architecture for, and experience with detecting sparsely discussed news events from a voluminous stream of Twitter messages. We begin with an overview of related work and then present our approach for detecting and correlating events. The following sections describe each of the components of the proposed architecture. The section on Experiments presents our results as obtained on tweets spanning a period of approximately 9 months. We conclude with a summary of our major findings and proposed future work.

Related Work

We have been motivated by the need to detect local news events that nevertheless may be of tremendous operational value when correlated with the internal operations and transactions of even a far-flung enterprise, as described in (Shroff, Agarwal, and Dey 2011). One of the pre-requisites for such a system, as also mentioned therein, is the ability to detect structured event-objects containing precise information about each event, which is what we contribute in this paper.

Recent publications on event detection from Twitter can be segregated based on (a) nature of events or (b) the detection techniques used. Events can be very specific such as natural calamities, accidents, marketing or cultural events. For example, (Sakaki, Okazaki, and Matsuo 2010), shows how to detect earthquakes. On the other hand, events can be generic, often referred to as 'breaking news' as discussed in (Phuvipadawat and Murata 2010), and (Weng and Lee 2011). One approach is to detect events from clustered tweets as discussed in (Becker and Gravano 2011); another, as in (Weng and Lee 2011) is by extracting features from individual tweets followed by clustering.

We focus on specific events rather than generic ones (in particular, *fire-in-factory* and *labor-strike*). Since each tweet is very short it usually covers only one aspect of an event (e.g., its location, or severity etc.). Therefore merely clus-

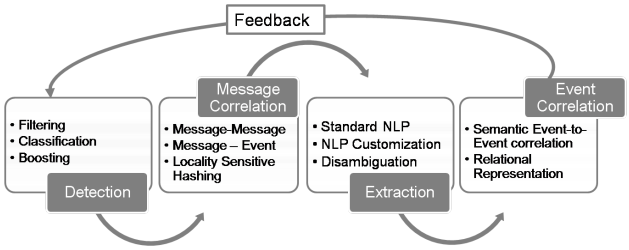

Figure 1: Logical Architecture for Creation of Event-Objects

tering tweets based on word-similarity only groups tweets describing the same aspect of the event. We therefore first classify each tweet using a supervised classifier to discard irrelevant tweets, and then group the positive tweets, first based on similarity and later based on semantic information extracted from them. We also employ locality sensitive hashing (Rajaraman and Ullman 2010) to improve processing efficiencies. Thus, compared to related work that usually rely on a single clustering step, we combine the use of LSH, a supervised classifier, and post-information-extraction clustering. As a result, we do not have to wait for more tweets to arrive before detecting an event, unlike if we had used a single clustering step.

We extract information from tweets using modifications of standard techniques: Natural language processing for information extraction from well-formed English prose have been fairly successful (Finkel, Grenager, and Manning 2005), (Kristina et al. 2003). However, information extraction from the informal language used in Twitter has still not been possible with similar level of accuracy. One approach to address this issue is to reapply some of the techniques of traditional NLP with little modifications; alternatively one could first expand the short text using normalization techniques as proposed by Xue et al (Xue, Yin, and Davison 2011) and then perform extraction information. In our scenario we found that the first approach works better.

Overview

The derivation of *event-objects* as shown in Figure 1 is a four step process beginning with the detection of a candidate-message reporting an event. A text matching algorithm then correlates these with other messages. Next aspects of the event are extracted from each cluster of tweets. Finally, before storing a potential event-object in the database, each potential event-object is merged with others based on its properties, such as time-of-occurrence and location. The input to this process is a subset of all the tweets occurring in Twitter, which can be obtained by specifying filter words, in the API made available by the website. Filter-words used for *Fire-in-Factory* tweet-stream are {*"Fire", "Blaze", "Factory", "Plant", "Mill"*}. Likewise for *Labor-Strike* {*"Strike", "Union", "Labor", "Labour", "Staff", "Management", "Workers", "Employee"*}.

Detection

To detect the messages that report occurrence of an event, we used a two step process. In the first step we reject tweets that follow a specific pattern using regular expressions, and the second step is supervised classification and boosting. Regular expressions are written based on prior domain knowledge. In case of both *fire-in-factory* and *labor-strike* we could discard about 80-90% of messages through this step.

The regular-expressions fails to discard messages such as *"If I take one more leave, my boss would fire me from the factory"*. We therefore experimented with supervised classification using both naive Bayes and SVMs. In contrast with previous approaches that first cluster similar tweets and then classify each bunch, we chose to classify every tweet as either reporting an event or not. As a result we could discover more aspects of the same event and reduce the chance of missing an event. Our classifier uses the following features: *Feature Set 1*: Parse every tweet with Stanford NER - (Finkel, Grenager, and Manning 2005), which annotates proper-names for organizations, locations and people, in a given text. *Occurrences* of location, organization or person are taken as a features. Similarly occurrences of a URL in the tweet is also taken as a feature.

Feature Set 2: Check for occurrence of numbers, associate every number with a range, and identify every such range as a feature.

Feature Set 3: Having removed pre-configured stop-words and after necessary stemming we identified all the remaining words of every message as features.

Message Correlation

Each tweet usually describes only one aspect of an event. It is observed that as time passes newer tweets with the latest information about an event begin appearing. We therefore compare every incoming tweet after classification with tweets of last 24 hours and bunch together the matching ones, and call this process Message-to-Message correlation.

If a new tweet is naively compared with all tweets in the past few hours, this could lead to a bottleneck, especially if the rate at which pairwise message-to-message correlation is possible cannot match the arrival rate of newer tweets. Instead we avoid exhaustive pairwise comparisons using *Locality Sensitive Hashing* (LSH) (Rajaraman and Ullman 2010) to discover potentially similar tweets in linear time.

LSH groups incoming messages into b buckets, which include those created for previous tweets as well as new buckets Tweets falling in the same bucket are assumed to be potentially matching. The set of unique tweets across b buckets form the set of *potentially matching tweets*; the incoming tweet is compared with each of these. Prior tweets having at least 75% similarity with the incoming tweet are considered to be matching. If the incoming tweet does not match with any of the tweets in the set of *potentially matching tweets*, a new event is created for the incoming tweet. Such sets of matching tweets define partial-events, to be further correlated based on semantic information.

Information Extraction

In this step we extract information from the tweets in each partial event bucket. As suggested in (Westermann and Jain 2007), event properties can be classified into 4 broad categories; *temporal, generative, spatial* and *descriptive* . The *temporal* aspect captures the time of occurrence, duration of the event etc. The *generative* aspect represents source information, e.g., Tweeter-ID and time of tweet. The *spatial* aspect defines event location and the *descriptive* aspect defines detailed information about the event, e.g., type of event ('fire-in-factory' or 'labor-strike').

Twitter posts do not usually follow the rules of English grammar, so accurate information extraction from these is a challenge. We have made small modifications to standard NLP-based information extraction techniques, and have used regular expressions, Named Entity Recognition (Finkel, Grenager, and Manning 2005), a combination of POS Tagger (Kristina et al. 2003) and regular-expressions, and *'Concept vocabulary based extraction'*, as described below:

Factory Information Extraction

Tweets on *fire-in-factory* or *labor-strike* often contain obfuscated factory names, e.g., *"huge fire breaks out in a cotton factory in .."*, and cause NER to give poor results. Therefore, we employ a POS Tagger on every message followed by tree traversal to identify a phrase indicating factory-information. Traversal is performed upwards, from any of leaf node having words "factory", "mill", or "plant", as long as the parent tag continues to be NP (proper-noun). In this subtree, we identify the occurrence of an article ('a' or 'the') immediately before "factory", "mill", or "plant", and take all words starting from this article to "factory", "mill", or "plant" as factory-information. Thus "a cotton factory" gets extracted as factory-information. On a set of $\tilde{3}00$ tweets containing factory information, plain NER could extract factory information from 1% of the tweets while our approach gave 78% correct results.

Location Extraction

For location extraction we employ both NER (Finkel, Grenager, and Manning 2005), which labels words in a text that are likely to be names of person, location, organization etc., as well as *'Concept vocabulary based extraction'* (CVE): In the latter we first identify phrases which could denote a location, through a concept vocabulary of all possible words indicating a location. We used World Gazeteer data from http://www.world-gazetteer.com as our vocabulary. A directed-acyclic graph of concept words is used as an index to help annotate the location phrases in tweets. This gives rise to noise, because certain locations names are based on English words, e.g., 'Friday Harbor'. We filter such noise using a naive Bayes classifier, with features being the POS tags of three words before a word, the word itself, and three words after a word, as suggested by (Califf and Mooney 1999). CVE could identify location in 74% of the tweets, whereas NER could identify location in 70% of the tweets. We combine both techniques and accept locations identified by either or both, resulting in 78% accurate location extraction.

To further disambiguate locations, we extract longitude and latitude through a combination of inverted-index search on World Gazetter data and search using Google Maps API (which is used only when no data found in the index). Consider two examples of such disambiguation: First, multiple locations could be extracted for the same event, with all also being found in the inverted-text-index. We check how many of them are very closely related or overlap spatialy. The place having maximum overlap is taken to be the event-location, e.g., if the word 'York' and 'UK' both occur, they are found to be overlapping (using their longitude and latitude) so 'York, UK' is chosen instead of choosing 'New York' which would be the top search result based on the word 'York'. Alternatively, only one location word might be identified that still returns many hits in the inverted index, e.g., if only the word 'London' is mentioned, we cannot conclusively state which of the many Londons the tweet means. For such situations we use precedence rules, e.g., choosing London, U.K. in the absence of further information.

Event-to-Event Correlation

Many partial-events refer to the same real world event, and need to be consolidated into a single *event object*. If two partial-events belong to the same real world event, their context, location, time window and certain *descriptive* properties should match. At the same time, a simple string match on, say, event-location does not work because users can use different levels of granularity to describe the event-location, e.g., 'USA' and 'New York'. To reduce the comparisons needed for such correlation, we use neighborhood search on an in-memory index of the events to calculate the neighborhood of an event: All the events with the same context, events with event-locating within a 100 kms radius, of occurring within 24-hour time window. To improve efficiency, we use Lucene's spatial index for location, and Lucene's range search for temporal properties. The event is compared with each such its neighbor for matches in the *descriptive*, *spatial* and *generative* aspects, e.g., URL, and location overlap.

Experiments

Our experiments are based on 5 million tweets collected over a period of almost 9 months starting April 2011 for fire-in-factory events and about 3 million tweets obtained for Labor-Strike events over the same period. We took manually tagged tweets in chronological order as they occurred in Twitter over 3 months (April-June, 2011). Starting with first 200 tweets as training set, we classified the next 200 tweets, then for next run we took all these 400 tweets as training set, with next incoming 200 as test set, and so on. Results of this build-phase are shown in Figure 2 marked as 'B'. As is evident, accuracies improve with continuous augmentation of training set. Though accuracies are in the of 80% range with both naive Bayes and SVM, we find that Naive Bayes does better.

Next we used our classifier trained on the April-May data on a fixed test set of 250 filtered tweets collected in January

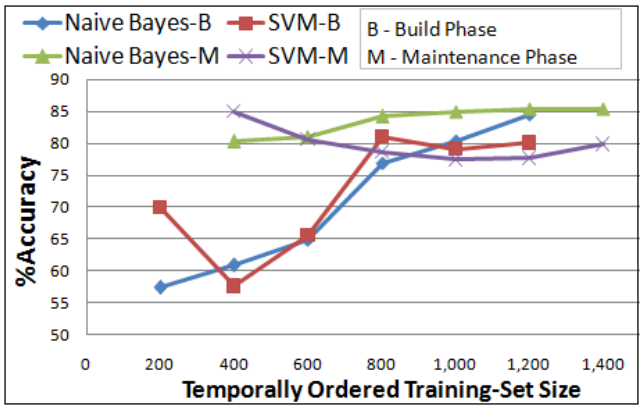

Figure 2: Comparison of classifiers for event-detection, Build and Maintenance Phases

2012. The results in this maintenance phase are shown in Figure 2 marked as 'M'. Once more, we achieve similar accuracies in the 77-85% range, with naive Bayes outperforming SVM. Thus, we conclude that our classifier based on a few month's training data works equally well even on tweets far into the future, at least for *Fire-in-Factory* events. (Similar results are observed for *Labosr-Strike* events, albeit with lower accuracies in the 60-65% range; we are still improving the training for this event and hence do not report these results in detail here.) Last but not the least, it is important to note that even with the moderate 80%-level classification accuracies, we hardly ever missed a real-world factory fire, except when no pertinent information was actually posted.

To assess the performance gain due to LSH, we measured the average number of tweets an incoming tweet was compared with, for a continuous run of the framework for about 8 days. It was found that use of LSH resulted in an 80% reduction in the number of comparisons required. Finally, to ensure uniqueness of the *event-objects*, a sample of events collected for a continuous run of 11 days was analyzed. It was found that 23.4% of the events were duplicate or alluding to same real-world event. So we conclude that our approach of message-to-message correlation coupled with event-to-event correlation is about 76.6% accurate.

Conclusion and Future Work

We have described the challenges involved in creating structured *event-objects* from Twitter posts reporting news events. We have added to the body of work event detection from Twitter by focusing on the specific case of sparsely reported events for which existing techniques do not work. In particular, we have established that event correlation is a two step process, first at the raw message level and then through semantic analysis of events. For detection of relevant tweets we have shown that our approach using supervised classification of individual tweets is able to catch the sparsely reported events in the 'long-tail'. We have also improvised standard NLP techniques so that they work on the informal language often used in Twitter. We continue to augment our running system with additional event-types, such as competitor promotions and consumer complaints, and we are finding that our overall architecture appears to work even in more general cases merely by using a few special features to augment the underlying word-based classifiers as needed.

Acknowledgements

We acknowledge Divya Garg's assistance in creation of components of this framework as part of Enterprise Information Fusion Project.

References

Becker, M. N. H., and Gravano, L. 2011. Beyond trending topics: Real-world event identification on twitter. In *Proc. of Intl. Conference on Weblogs and Social Media*, ICWSM '11.

Califf, M., and Mooney, R. 1999. Relational learning of pattern-match rules for information extraction. In *Proceedings of AAAI-99*, AAAI '99.

Finkel, J. R.; Grenager, T.; and Manning, C. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, 363–370. Stroudsburg, PA, USA: Association for Computational Linguistics.

Kristina, T.; Dan, K.; Christopher, M.; and Yoram, S. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of HLT-NAACL 2003*, pp. 252–259.

Phuvipadawat, S., and Murata, T. 2010. Breaking news detection and tracking in twitter. In *Proceedings of the 2010 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology - Volume 03*, WI-IAT '10, 120–123. Washington, DC, USA: IEEE Computer Society.

Rajaraman, A., and Ullman, J. 2010. *Mining of Massive Datasets*.

Sakaki, T.; Okazaki, M.; and Matsuo, Y. 2010. Earthquake shakes twitter users: real-time event detection by social sensors. In *Proceedings of the 19th international conference on World wide web*, WWW '10, 851–860. New York, NY, USA: ACM.

Shroff, G.; Agarwal, P.; and Dey, L. 2011. Enterprise information fusion for real-time business intelligence. In *Proceedings of the 14th International Conference*, Fusion '11.

Weng, J., and Lee, B.-S. 2011. Event detection in twitter. In *Proc. of Intl. Conference on Weblogs and Social Media*, ICWSM '11.

Westermann, U., and Jain, R. 2007. Toward a common event model for multimedia applications. In *IEEE MultiMedia, v.14 n.1*, p.19–29,.

Xue, Z.; Yin, D.; and Davison, B. D. 2011. Normalizing microtext. In *Proc. of Intl. Conference on Weblogs and Social Media*, ICWSM '11.