What's up on Twitter? Catch up with TWIST!

Marina Litvak and **Natalia Vanetik** and **Efi Levi** and **Michael Roistacher**
Department of Software Engineering
Sami Shamoon College of Engineering
Beer Sheva, Israel
{marinal,natalyav}@sce.ac.il
{efilvefi,mikiroistacher}@gmail.com

Abstract

Event detection and analysis with respect to public opinion and sentiment in social media is a broad and well-addressed research topic. However, the characteristics and sheer volume of noisy Twitter messages make this a difficult task. This demonstration paper describes a TWItter event Summarizer and Trend detector (TWIST) system for event detection, visualization, textual description, and geo-sentiment analysis of real-life events reported in Twitter.

1 Introduction

Twitter grows rapidly. Efficient, accurate, and scalable real-time analysis of Twitter content is in demand and requires integration of sophisticated approaches for natural language processing, signal processing, and more. Instead of considering single tweets, Twitter events[1] are detected and analysed in many existing approaches (Becker et al., 2011; Long et al., 2011; Weng and Lee, 2011; Cordeiro, 2012; Osborne et al., 2014; Preotiuc-Pietro et al., 2012). Our method is directed toward *unspecified event detection*, where an event has not been previously identified. Because no prior information is available, the classic approach to detection of such events exploits temporal bursts of Twitter stream features such as *hashtags* and specific words. TWIST defines an event as a collection of hashtags and extends the Event Detection with Clustering of Wavelet-based Signals (EDCoW) algorithm of Weng and Lee (2011) by performing an additional text analysis of tweets. This extension enables TWIST to distinguish between events that occur at the same time and share similar wavelet signals, but have a different content. After events are detected, each event can be summarized using both internal content from Twitter, and external sources. Finally, all detected events are analysed by their sentiment distribution over a world map, and visualized in the resolution of countries. This feature enables the TWIST user to see whether and how the geolocation of Twitter users affects their opinions, and how the sentiments and opinions regarding the same political or other event can be different over different countries. TWIST utilizes unsupervised learning for most of its stages, except sentiment analysis. TWIST does not rely on external ontologies. It incoporates external sources, automatically retrieved, for summarizing events. Also, TWIST detects geolocation of event-related tweets (and not geolocation of events) and visualizes their sentiments on a map. TWIST architecture is flexible, its implementaion uses the JavaScript Object Notation (JSON) format for the processed data that enables other domains to be easily integrated. All this makes TWIST convenient for a potentially wide range of users, from private individuals and businesses, to data scientists and other professionals dealing with different kinds of data analysis.

2 System description

Our system is written in C#, as a standalone application, and it uses a MySQL database. The system (see Figure 1) is composed of several modules that are described below.

2.1 The Twitter stream

The dataset object of analysis is retrieved using the Twitter streaming API that, using the default access level, returns a random sample of all public tweets. This access level provides a small proportion of

[1] A Twitter event is a collection of tweets and re-tweets that discuss the same subject in a relatively short (minutes, hours or days) time period.

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: System Demonstrations, pages 213–217, Osaka, Japan, December 11-17 2016.

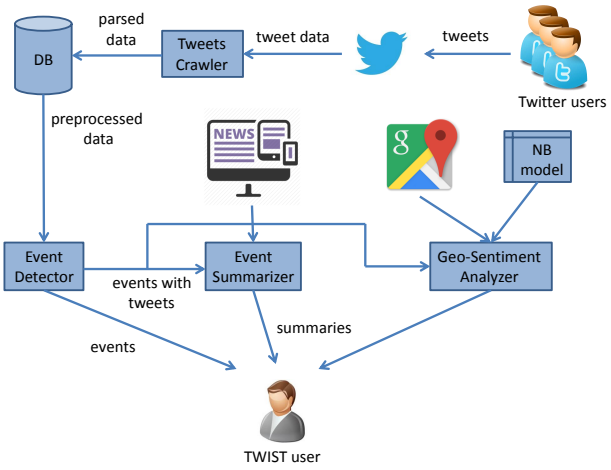


Figure 1: Data flow of TWIST.

all public tweets (1%). The data are returned in JSON, as a set of documents, one per tweet. Given the average number of 140 million tweets sent per day in Twitter, the size of the data retrieved by the streaming API in a 24 hour time span is roughly $1,400,000$ tweets. The 140 character limit of tweets gives an expected data stream of 196 MBytes per day or 2269 bytes per second. The Tweets Crawler uses Tweetinvi[2] to retrieve tweets data and store it in a database. We are mainly interested in hashtags, as an explicit annotation of a tweet's main theme.

2.2 Event detector

This component detects events by operating hashtag wavelets, following three stages of the EDCoW algorithm enumerated below, and integrated with text analysis. The system enables a user to follow after the evolution of the event detection process by visualizing the results of each simple stage. The configuration file enables a user to set multiple parameters of the event detector, such as time period to work on, sampling time interval, and the hashtag minimal count threshold.

Wavelet analysis

A wavelet is a wave-like oscillation with an amplitude that begins at zero, increases, and then decreases back to zero. The EDCoW algorithm detects events by grouping a set of hashtags with similar burst patterns. This algorithm has three components: (1) signal construction, (2) filtering and cross-correlation computation, and (3) graph partitioning.

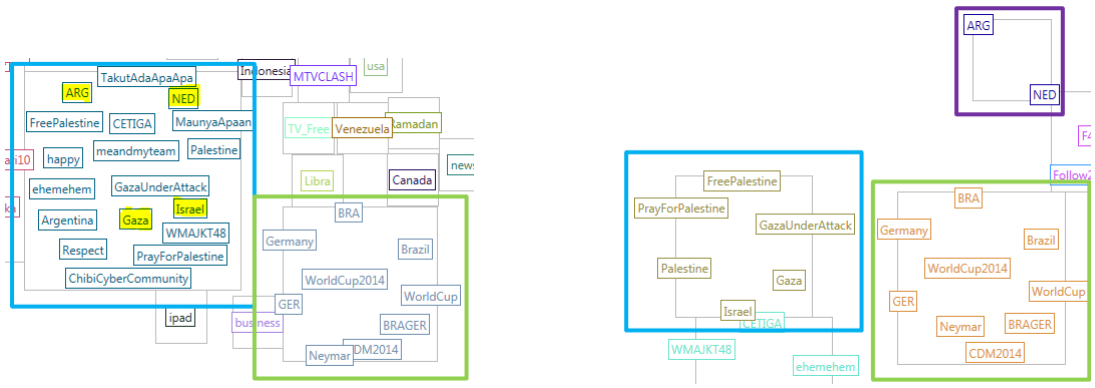
The **first** stage of the EDCoW algorithm constructs a signal for each individual hashtag that appears in tweets, using its *tf-idf* values. The **second** part of signal processing builds the smooth signal with the help of a sliding window that covers a number of initial sample points. We use the Savitzky-Golay filter (Press and Teukolsky, 1990). After autocorrelation is computed, irrelevant signals (with low values) are discarded, thereby efficiently eliminating noise from the meaningful data. The values of cross-correlation (similarity) for the remaining signals are stored in a matrix. Correlation values that are too low are set to zero[3]. The **third** stage of the algorithm views event detection as a graph-partitioning problem for a weighted graph whose adjacency matrix is the wavelet cross-correlation matrix. TWIST uses the Girvin-Newman algorithm (2006). After modularity-based graph partitioning (third stage), the clusters of hashtag wavelets representing events are displayed to the user. TWIST extends the EDCoW algorithm by analysing the textual content of tweets. The latter requires text pre-processing that is performed on all collected tweets before they are stored in a database and then analysed.

Text preprocessing

We perform the following preprocessing steps for each collected tweet: (1) tokenization, (2) part-of-speech tagging and filtering, (3) stop-words removal, and (4) stemming for remaining words (Porter, 1980). The result of preprocesing is a collection of normalized terms and hashtags, linked to their tweets. The frequency-based statistics (term frequency–inverse document frequency) are then calculated

[2] A Twitter C# library `https://tweetinvi.codeplex.com/`

[3] TWIST uses *median value* as a boundary in both cases



(a) Without text analysis

(b) With text analysis

Figure 2: Events detected by TWIST

and also stored in our database.

Text similarity analysis for better event detection

Our system extends the third stage of the EDCoW algorithm by integrating a text similarity knowledge between tweets into a graph representation. The motivation behind this idea was dictated by a possible situation where two or more unrelated events evolve at the same time, following the same burst pattern. In such a case, a wavelet analysis will not distinguish between these events, and only analysing the *content* of tweets may point to the differences between them.

In TWIST, the weights on graph edges are calculated as a weighted linear combination of cross-correlation values computed during the second stage, together with *textual similarity* scores for every pair of signals. Every signal is represented by the texts of all tweets belonging to it. TWIST enables the end user to choose between two classic metrics for textual similarity–Jaccard and cosine similarity. Given a cross-correlation score cc_{ij} and a similarity score sim_{ij} between signals i and j, the weight on edge between signal nodes is computed as $w_{ij} = \alpha \times cc_{ij} + (1 - \alpha) \times sim_{ij}$, where $0 \leq \alpha \leq 1$ is a system parameter. Figure 2 shows different clusters, representing events, that were detected without taking tweet text into account (by using hashtag wavelets only), and with analysis of tweets.

2.3 Event summarizer

This module enables a user to obtain a summary of an event of interest. A user can choose to see either an *internal profile* for the selected event, which includes the most salient hashtags, keywords, and tweets, or an *external profile* that contains sentences from the most salient external links mentioned in tweets. The module enables a user to configure multiple internal parameters that affect the summary quality. The summarization approach in both cases follows a strictly extractive principle, as being most appropriate for the Twitter domain, both in terms of accuracy and efficiency. We used two state-of-the-art algorithms in the summarization process – TextRank (Mihalcea and Tarau, 2004) and Lingo (Osiski et al., 2004) – for text ranking and clustering, respectively. The following subsections describe adaptation of those algorithms in our system.

2.3.1 Internal profile

The internal profile is built from the most salient hashtags, keywords, and tweets. The tweets with the highest PageRank score are retrieved from a weighted tweets graph, with nodes standing for tweets, according to the TextRank method. A tweets graph is built on the tweets that are filtered by length and keywords coverage in order to reduce graphs size and TextRank processing time. Hashtags are considered as extremely important keywords and give a higher impact to a coverage score. A similarity between tweets for weighting the graphs edges can be calculated by either Jaccard similarity between sets of tweets terms or cosine similarity between their tf-idf vectors. The keywords and hashtags are ranked by their *tf-idf* score and the top-ranked ones are extracted.

2.3.2 External profile

The external profile of the detected event is compiled by extracted parts from the relevant external sources. This profile is created in TWIST by (1) retrieving the relevant sources; (2) preprocessing and

ranking relevant sources; and (3) summarizing the top-ranked sources.

The relevant sources for each detected event are retrieved by collecting, analyzing, and filtering links appearing in tweets. The sources that do not contain enough meaningful text are filtered. Then, the document graph, with nodes standing for sources and edges for lexical similarity between them, is built, and the eigenvalue centrality is computed. For final ranking of external sources, their eigenvalue centrality, the keyword coverage, and link frequency counts are used.

The summarization is performed by (1) selecting theme sentences, and then (2) ranking and selecting theme sentences into a summary. We consider theme to be a group of lexically similar sentences, and retrieve all event-related themes by a clustering of sentences collected from relevant sources. We use the Lingo clustering algorithm that, in addition to clusters, also provides a label and a score for each cluster. Then, given clusters, we select theme sentences closest to cluster centroids as representatives of their clusters. A summary that describes the detected event must cover as many its important themes as possible. Given theme sentences, we rank them using the TextRank approach–an undirected graph of sentences with the lexical similarity relationships is built from event theme sentences, and the PageRank algorithm is applied to find sentence scores–and compile a summary from the top-ranked sentences.

2.4 Geo-sentiment analyser

The geo-sentiment analyser is the only module of TWIST that needs annotated data for supervised learning. It uses the NaiveBayes algorithm, as one of the most simple and reliable classification methods for textual data. The textual data is preprocessed and represented in a bag-of-words format, per tweet. Each bag is labeled by a particular detected event (event detector output) and country (we filter out tweets that lack geolocation data). Then a trained model[4] of NaiveBayes is used for a sentiment classification of tweets, and the resulting statistics are displayed on the world map. A user can choose to see majority sentiment, with corresponding color, and full distribution statistics for the event of interest. We use the Google GeoChart API for a map visualization.

3 Pilot study and experiments

We performed a pilot study over a two day period, during which $7,549,339$ tweets, published between 08/07/14 24:00 and 10/07/14 24:00, and covering 95889 hashtags, were collected. The data collection fell during the period of the Football World Cup 2014 and the Israeli Protective Edge Operation in Gaza (Wikipedia, 2014). There is a need to stress that we collected tweets only until midnight, so it is likely that we collected only partial events (for instance, the football game between Argentina and Holland took place in the late hours of the evening, and, therefore, only some of the tweets about the game were collected).

Using pure wavelet similarity according to the EDCoW algorithm resulted in inaccurate event detection, when different unrelated events mistakenly fell into the same category. As an example, signals related to the football game between Holland and Argentina fell in the same cluster with signals hashtagged by Gaza. The system detected one event perfectly–the World Cup 2014. The clique contains BRA, GER, WorldCup2014, Brazil, and similar hashtags. However, the Protective Edge Operation event was not detected. After the text analysis component was activated, Gaza and World Cup 2014 were detected as separate unrelated events. The Protective Edge Operation event is detected properly. The event contains hashtags such as PrayForPalestine, GazaUnderAttack, and FreePalestine. The demo video of TWIST demonstrates its usage in the demo mode, on the data described above, with results for all stages, including: event detection, summarization of main events, and their geo-sentiment analysis. The video can be found here: `https://youtu.be/JH4-YU8mL9A`. One can also run TWIST in a regular mode, where a new data will be collected and analysed in a real time.

The accuracy of NaiveBayes classifier for the sentiment classification was evaluated on the Sentiment140 dataset and resulted in 76%, using 10-fold cross validation. The experimental results for the TextRank summarizer can be found in its original paper (Mihalcea and Tarau, 2004). The improvement in event detection, using text analysis, was determined by human evaluation during our pilot study. It takes approximately one minute for TWIST to analyse over 7 million tweets.

[4] trained on the Sentiment140 dataset (`http://www.sentiment140.com`)

4 Conclusions and future work

In this work we present a system we call TWIST, which aims at detecting and describing events in Twitter during a pre-defined period of time. TWIST extends the EDCoW algorithm by Weng and Lee (2011) by text analysis of tweets. As a pilot study showed, the proposed extensions improve the quality of event detection. Also, TWIST applies summarization techniques for describing the detected events and performs their geo-sentiment analysis.

In future, we intend to experiment with more sophisticated summarization techniques and methods for linking news to Twitter events (see (Guo et al., 2013)). Also, we plan to evaluate TWIST performance using human evaluations.

Acknowledgments

We want to thank Daniel Miles, Dror Binder, and Shahar Sabag for their contribution to TWIST implementation and support.

References

Hila Becker, Mor Naaman, and Luis Gravano. 2011. Beyond trending topics: Real-world event identification on twitter. *ICWSM*, 11:438–441.

Mário Cordeiro. 2012. Twitter event detection: Combining wavelet analysis and topic inference summarization. In *Doctoral Symposium on Informatics Engineering, DSIE*.

Weiwei Guo, Hao Li, Heng Ji, and Mona Diab. 2013. Linking tweets to news: A framework to enrich short text data in social media. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 239–249.

Rui Long, Haofen Wang, Yuqiang Chen, Ou Jin, and Yong Yu. 2011. Towards effective event detection, tracking and summarization on microblog data. In *Web-Age Information Management*, pages 652–663. Springer.

Rada Mihalcea and Paul Tarau. 2004. Textrank: Bringing order into texts. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Mark EJ Newman. 2006. Modularity and community structure in networks. *Proceedings of the National Academy of Sciences*, 103(23):8577–8582.

M. Osborne, S. Moran, R. Mccreadie, A. Von Lunen, M. Sykora, E. Cano, N. Ireson, C. Macdonald, I. Ounis, Y. He, T. Jackson, F. Ciravegna, , and A. OBrien. 2014. Real-time detection, tracking, and monitoring of automatically discovered events in social media. In *52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 37–42.

Stanisaw Osiski, Jerzy Stefanowski, and Dawid Weiss. 2004. Lingo: Search results clustering algorithm based on singular value decomposition. In *Advances in Soft Computing, Intelligent Information Processing and Web Mining, Proceedings of the International IIS: IIPWM04 Conference*, pages 359–368.

M.F. Porter. 1980. An algorithm for suffix stripping. *Program*, 14(3):130137.

Daniel Preotiuc-Pietro, Sina Samangooei, Trevor Cohn, Nicholas Gibbins, and Mahesan Niranjan. 2012. Trendminer: An architecture for real time analysis of social media text. In *Workshop on Real-Time Analysis and Mining of Social Streams (RAMSS), International AAAI Conference on Weblogs and Social Media (ICWSM)*.

William H Press and Saul A Teukolsky. 1990. Savitzky-golay smoothing filters. *Computers in Physics*, 4(6):669–672.

Jianshu Weng and Bu-Sung Lee. 2011. Event detection in twitter. *ICWSM*, 11:401–408.

Wikipedia. 2014. Protective edge operation. `http://en.wikipedia.org/wiki/2014_IsraelGaza_conflict`.