Feature-rich event detection in social media streams

Denis Antiukhov
aphex@mit.edu

ABSTRACT

Social networking platforms such as Twitter have emerged in recent years, creating a radically new mode of communication between people. Monitoring and analyzing rich and continuous flow of user-generated content can yield unprecedentedly valuable information, which would not have been available from traditional media outlets. However, learning from Twitter streams poses new challenges, as compared to traditional media.

Traditional approaches to topic/event detection involve clustering based on semantic features of documents. In this project, we implement and study different clustering models based on certain Twitter-specific features that are less susceptible to noise, which include geo-positional data, time stamps, hashtags and check-ins. We also implement a clustering algorithm based on bursty behavior of n-grams in Twitter documents and use it as a baseline.

The clusters produced by our model contain valuable information about live events, but they also do contain a lot of noise. We treat this as a binary classification problem and use a multilayer perceptron to classify and rank the clusters based on their textual, social, temporal and other features. Finally, we summarize top-ranking clusters and output the results to the end user.

Keywords

Social Media Analytics, Event Detection, Twitter, Social Media, Digital Journalism, News Aggregation

1. INTRODUCTION

The rise of Social Media platforms in recent years brought about a radically new mode of communication between people. Every day, users send more than 500 million tweets (as of 2015) on every possible topic. Interactions and communication in Twitter often reflect real-world events and dynamics, and important events like elections, disasters, concerts, and football games can have immediate and direct impact on the volume of tweets posted. Because of its real-time and global nature, many people use Twitter as a primary source of news content, in addition to sharing daily life, emotion and thoughts. Journalists also increasingly adopt social media as professional tools and are gradually altering their processes of news selection and presentation . They use Twitter to monitor the newsworthy stories that emerge from the crowd, and to find user-generated content to enrich their stories. However, it is very hard for a person to spot the useful information in Twitter without being overwhelmed by an endless stream of redundant tweets.

In contrast to conventional media, event detection from Twitter streams poses new challenges. Twitter streams contain large amounts of meaningless messages and polluted content, which negatively affect the detection performance. In addition, traditional text mining techniques are not suitable, because of short length of tweets, large number of spelling and grammatical errors, and the frequent use of informal and mixed language [1]. Event detection techniques presented in literature address these issues by adapting techniques from various fields to the uniqueness of Twitter.

As a response to this problem, we propose a system to detect unspecified novel, newsworthy topics/events as they are published on Twitter. In order to overcome the challenges posed by the nature of user-generated content, we focus on using document features less susceptible to human error: hashtags, check-ins and geo tags. Provided with an up-to-date database of latest tweets, the proposed system automatically mines the social stream using a sliding window approach, providing a set of headlines and complementary information that summarize the detected topics. Our proposed event detection approach is based on a combination of aggressive data pre-processing, document clustering based on certain twitter-specific features, cluster classification and post-processing.

2. RELATED WORK

Event detection has long been addressed in the TDT (Topic Detection and Tracking) [2] program, an initiative sponsored by the Defense Advanced Research Projects Agency, concerned with event-based organization of textual news document streams. The motivation for the TDT research initiative was to provide core technology for news monitoring tools from multiple sources of traditional media (including newswire and broadcast news) to keep users updated about news and developments. Originally, the TDT consisted of three main tasks: segmentation, detection, and tracking. These tasks attempt to segment news text into cohesive stories, detect new (unforeseen) events, and track the development of a previously reported event.

Methodologically, general-purpose topic detection can produce two types of complementary outputs: either the documents in the collection are clustered or the most important terms or keywords are selected and then clustered. In the first method, referred to as document-pivot, a topic is represented by a cluster of documents, whereas in the latter, commonly referred to as feature-pivot, a cluster of keywords is produced instead.

Both methods have advantages and disadvantages. Document-pivot methods suffer from cluster fragmentation problems and, in a streaming context, they often depend on arbitrary thresholds for the inclusion of a new document to an existing topic. On the other hand, feature-pivot methods are commonly based on the analysis of associations between terms, and often capture misleading term correlations. In general, the two approaches can be considered complementary and, depending on the application, one may be more suitable than the other.

Simple document-pivot approaches cluster documents by leveraging some similarity metric between them. The work by Murata [8] follows this direction to provide a method for breaking news detection in Twitter. Tweets retrieved using targeted queries or hashtags are converted into a bag-of-words representation weighted with boosted tf-idf, emphasizing important entities such as names of countries or public figures. Tweets are then incrementally merged by considering the textual similarity between incoming tweets and existing clusters.

Dimensions other than text can also be used to improve the quality of clustering. TwitterStand [5] uses a leader-follower clustering algorithm that takes into account both textual similarity and temporal proximity. Each cluster center is represented using a centroid tf-idf vector and the average post-time. Sensitivity to noise (which is a known problem for document-pivot methods) and fragmentation of clusters are drawbacks of this approach. Manual selection of trusted information providers and periodic defragmentation runs are needed to mitigate such effects.

The task of First Story Detection (FSD) discussed by Petrovic et al. [7] is closely related to document-pivot TDT. The goal is to detect the first document discussing a topic in a large corpus. A new story is created by a document having low similarity with all previously detected clusters. For fast retrieval of nearest neighbors for the incoming document locality sensitive hashing is used; however, such a solution is problematic when the nearest neighbors are not very similar to the query document.

3. SYSTEM ARCHITECTURE

Our approach to event detection in this project is based on:

1. Aggressive tweet and term filtering, designed to remove noisy tweets and vocabulary
2. A combination of clustering models based on Twitter-specific features, aimed at mitigating noise
3. A machine-learning approach to cluster classification which uses a feature-rich representation of clusters in order to remove clusters consisting of meaningless messages, unrelated content and rumors;

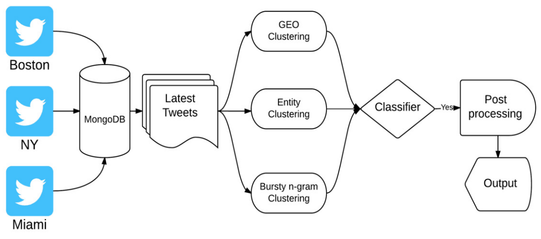

Figure 1: Overview of proposed system architecture

4. Cluster post-processing, during which we merge and summarize qualifying clusters, extract additional information (like Instagram photos and videos, popular hashtags, geodata, etc) to provide a more informative output to the user.

We describe our method in detail in the following subsections.

3.1 Data collection and storage

For collecting the Twitter stream we use a self-developed tool built against Twitter public API. The distinguishing feature of this software lies in it's ability to connect to multiple data streams at the same time. Streams are typically defined by a set of coordinate constraints: for this project we defined 5 streams corresponding to Boston, New York, Chicago, Miami and Vancouver cities. During the course of the project, a total of 8 million tweets were collected and written to database, resulting in a dataset of 30 GB size.

To organize storage, we chose to use MongoDB. This choice was motivated mainly by three reasons: 1. MongoDB stores data in serialized json format, which is also used by Twitter API to encode tweets. 2. Support of spatial and temporal indexing, which comes very useful in the framework of event detection; 3. Deep query-ability. MongoDB supports dynamic queries on documents using a document-based query language, which facilitates easy and fast information retrieval.

3.2 Document structure

As we mentioned above, Twitter API allows to collect streams of tweets documents that are represented in json format. Apart from the textual content, these json objects contain a high amount of relevant metadata. A key distinctive feature of our project lies in the fact that we try to leverage this data for clustering, in order to overcome the noisy nature of textual content, and to build a feature-rich representation of clusters for more accurate classification. For that reason, it is important to describe the object structure in detail. Each document in our collections holds information about:

- unique tweet id
- textual content, hashtags, user mentions, language

- timestamp of creation
- geo-data: timezone, country, city, approximate or precise creation coordinates
- urls, and attached multimedia
- user data: name, id, number of friends, followers, total tweets

These are just the fields that are used in our system. Overall, each stored json object holds more than 100 fields, description of them all is outside the scope of this report.

3.3 Pre-processing

Our event-detection algorithm begins with retrieving a collection of latest tweets, that lie within a certain time-window. During our experiments, we found that a window of 3 hours works best for our system. Most events that we will be looking for are well contained within such a time period. Considering a larger timeframe results in more documents being retrieved, which considerably increases computational cost. For Boston and Miami, a 3-hour window normally contains an average of 15000 documents, for New York this amount is doubled.

The next step is concerned with filtering out very noisy documents. We begin with normalizing the text. We remove hyperlinks, digits and other punctuation, extract user mentions and hashtags. Next, we tokenize the remaining clean text, and remove stop words. We check the structure of the resulting tweet, and filter out tweets that have more than 2 user mentions, more than 5 hashtags, or less than 3 clean text tokens. We also drop all documents for which the language field (as seen in the json) is not equal to EN. The idea behind this structure-based filtering is that tweets that have too many user mentions or hashtags, but lack enough clean text features, do not carry enough news-like content, or are generally too noisy to be meaningfully clustered. As a result of this simple preprocessing step, approximately 50% tweets are discarded. The remaining documents proceed to the next processing stage: clustering.

4. CLUSTERING

Clustering is the primary approach to data organisation in TDT. Various clustering-based algorithms have been successfully employed for both retrospective and new event detection tasks [4] [9] [10]. In our work, we wanted to study, how using tweet features not susceptible to noise can help overcome the challenges posed by the noisy nature of user-generated content. To this end, we consider 3 different clustering models, which we describe in detail in the following sections.

4.1 Geo model

As we mentioned in 3.2, documents stored in our collection include geo positional data. All documents hold information about some bounding box, within which they have been created. This bounding box is usually rather large, even compared to the area defined by the stream constraints (e.g. Manhattan vs New York), and does not allow to infer much about the document itself. However, as GPS-capable smartphones become more and more widespread, more tweets begin to include precise geo position. In our dataset, approximately 13% of all documents hold precise coordinates. Users reporting from a scene of certain event are expected to be close to each other: hence, we can leverage available geo-data for event detection. So, our first clustering model relies solely on available tweet coordinates.

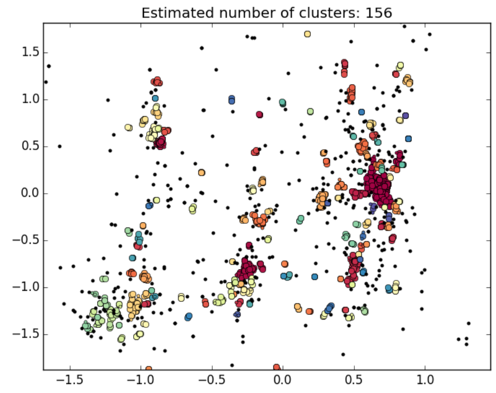

Figure 2: DBSCAN Clusters for greater Boston area

We proceed as follows: from the documents that passed the preprocessing step, we extract all that contain precise coordinates. This results in a collection of 2000-3000 documents for an average timeframe of 3 hours. We scale the resulting matrix of (lat, long) coordinates using standardization. Then, we apply DBSCAN algorithm to cluster the dataset.

We chose DBSCAN for this model for the following reasons:

- Does not require the number of clusters to be specified in advance
- Can find arbitrarily shaped clusters
- Has a notion of noise, and is robust to outliers

We had to rely on empirical observations to choose the epsilon parameter for DBSCAN. From our experiments, we found that a value of $\epsilon = 0.03$ is optimal, roughly corresponding to a size of a large concert hall. For reference, below we provide a visualization of clustering results for greater Boston area.

4.2 Entity model

Another interesting (and useful) feature particular to Twitter and other microblogging services is the widespread usage of hashtags. By definition, hashtag is a type of label or metadata tag used on social network which makes it easier for users to find messages with a specific theme or content. Users create and use hashtags by placing the #hash character. Users, relating to a certain event are likely to use similar hashtags in their tweets. This observation has been exploited by [8] and [4] in their works on event detection.

However, we go a little further and extract one additional feature that is becoming more and more widespread in microblog services. We are talking about Foursquare check-ins. Foursquare is a social networking service centered around venue recommendation and location sharing. By taking into account the places a user goes and the other users whose advice they trust, Foursquare provides recommendations of the places to go around a user's current location. At the core of Foursquare functionality lies the check-in: term used to identify when a member has physically visited (or checked in to) a venue. Since 2010, Foursquare accounts can be linked to Twitter: in that case, when performing a check-in, a user may opt to share it on his Twitter network. This will result in a message like this:

”Having fun at Santacon! #santacon #santaconnewport @ Newport, Rhode Island”

In the above example, the check-in begins with @ symbol and the first letters are capitalized. Due to such behavior, it can be easily extracted from raw text. In our dataset, 10% of tweets contain check-ins. We refer to hashtags and checkins as entities. Entities are useful for event detection for two main reasons:

- Users tend to use same entities when referring to a certain concept, venue, or event
- Entities are generated automatically (hashtags have an autocomplete feature in Twitter app), and thus are not susceptible to misspelling.

With that in mind, we propose out second, entity-based clustering model. It works as follows:

1. Extract all documents that contain two or more different entities.
2. Build a corpus of all entities detected within a time window.
3. Count occurrences, drop entities that occur less then 5 times (again, this threshold was chosen empirically).
4. Use the remaining entities to build a vocabulary for the entity space.
5. Represent documents that contain the frequent entities with a binary count vectorizer in entity space.
6. Apply hierarchical clustering to the vector representations, using cosine as a metric of distance.

Table 1: Availability of frequent entities

	Boston	New York	Miami	Chicago
hashtag	11.7%	14.5%	14.7%	9.1%
check-in	5.2%	8.5%	12.9%	4.3%
both	2.2%	4.7%	6.8%	1.7%

The 5th step, of course, results in even more documents being discarded, since we only consider documents that contain frequent entities. In the above table we show, how many of the original documents contain at least one popular entity. This data is averaged over one week.

This approach significantly reduces the size and dimensionality of the dataset. Hence, performing hierarchical clustering becomes not computationally intensive and is completed in a matter of seconds on a reasonably-specced computer. We cut the resulting dendrogram at 0.5 distance threshold: this results in clusters being tight enough.

4.3 Bursty n-gram model

Our final model is based on a paper [6] ”Sensing trending topics in Twitter”. This model is concerned with extracting bursty n-grams to detect events . Bursty n-gram is defined as an n-gram which starts appearing unusually often, as compared to historical usage statistics. We also use this language based model as a baseline to evaluate our results. We implement this model without any modifications. So as to avoid repetition, we just enumerate the clustering steps below.

1. From the window tweet corpus, create a binary tweet-term matrix, where the vocabulary terms are only bigrams and tri-grams, that occur in at least 10 tweets
2. Reduce the matrix to only the subset of rows containing at least 5 terms (tweets with at least 5 tokens from the vocabulary). This step results in 90% of tweets being discarded.
3. Compute hierarchical clusters, again, based on cosine distance between samples.
4. Cut the dendrogram at 0.5 distance threshold
5. Rank the resulting hierarchical clusters, based on the df-idf weighting statistic proposed in the paper mentioned above:

$$df - idf_t = \frac{df_i + 1}{log\Big(\frac{\sum_{j=1}^{t} df_{i-j}}{t} + 1\Big) + 1}$$

Here, df_i corresponds to frequency of an n-gram within a time window i. This statistic discounts the term-frequency in the current time window using the average frequency in the previous t time windows. Setting the parameter t controls, how much of the history affects the current weight of a term.

Using this approach has an obvious downside: in order to compute the historical document frequencies of n-grams in step five, we need to retrieve and process t times more data from the database, corresponding to t time windows. In our experiments, we primarily used a value of t = 3, to account for the last 9 hours of historical observations. We used the bursty model primarily as a baseline for comparison with our two other models.

4.4 Model comparison

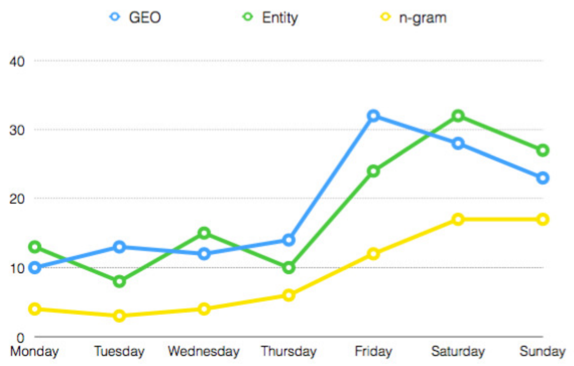

Figure 3: Number of true events detected in NY, per day of the week

Based on the gold annotations from the training set that we developed for the classification stage, above we present some statistics on how many true event clusters does each model actually capture. We observe, that despite aggressive filtering employed by our clustering algorithms, they still surpass the state-of-the-art n-gram based approach in terms of newsworthy event detection.

5. CLASSIFICATION

While the methods described above do retrieve clusters corresponding to new and newsworthy topics and events, they also produce many clusters containing spam, rumors, and meaningless collections. We treat this as a binary classification problem, and solve it using a Machine Learning approach, where class T means true news event cluster and class F represents false cluster containing spam, hot topics and heterogeneous collections. So, in the next processing step, all clusters detected by the 3 models are represented with a set of cluster-level features, and classified by a feed-forward multilayer perceptron NN model. All clusters in class T are passed to the post-processing step and eventually form a news event result. Features used to represent clusters are described in the next sections.

5.1 Textual similarity

While we do not rely on textual similarity of documents in our clustering algorithms, we use it as a metric for cluster cohesiveness. So, in order to evaluate, how similar are the documents in each cluster in terms of textual content, we employ two models.

5.1.1 TF-IDF

Traditional data representation for event detection involves the term vectors, weighted using the classical term frequency inverse document frequency (tf-idf) approach, which evaluates how important a word is to a document in a corpus. In order to learn the vocabulary and weights for the model, we used our whole 8-million document dataset. This resulted in a vocabulary of 75000 terms. Documents were represented as term vectors, similarity was evaluated using cosine as a metric of distance.

5.1.2 word2vec

While traditional term-vector approach works well with larger documents, using it in twitter framework has it's downsides due to the fact that tweets often contain only 3-5 clean text tokens. Consider the following example:

”The Paris attacks are still fresh in the minds of Parisians and tourists here for a visit #prayforparis”
”Commemorating the victims of #ISIS bombings”

Both tweets are related to the same event and relate to the same, however the TF-IDF vector similarity will be close to zero for these two documents, because the terms for the two do not intersect. In order to overcome this challenge we make use of the recent developments in DNN-based natural language processing. word2vec is a model used to produce so-called word embeddings. By analyzing the co-occurance of terms in the training set, this model maps each word to a vector of typically several hundred elements, which represent that word's relation to other words. We used an implementation of CBOW word2vec written in tensorflow, trained the model on our tweet dataset, and obtained the 256-dimensional embeddings to help us evaluate the similarity of documents. In order to represent documents, we summed the vectors corresponding to top-5 word in each document (determined from the TF-IDF weight). This worked quite well: for instance, our model assigns 0.817 similarity to the example documents presented above.

5.2 Cluster-level features

We compute four types of cluster-level features for the classifier, representing statistical, social and textual information. Some of the features are designed to filter out heterogeneous clusters, while others help distinguish news events from hot topics.

5.2.1 Textual features

1. size: number of tweets in cluster
2. unique unigrams: normalised number of unique unigrams found within cluster
3. mean tfidf: mean cosine similarity of tweets, represented using a weighted TF-IDF vectorizer
4. mean word2vec: mean cosine similarity of tweets, represented using a word2vec model

5.2.2 Entity features

1. unique hashtags: number of unique hashtags found
2. unique check-ins: number of unique check-ins found
3. frequent entities: number of popular entities found. Entity is deemed frequent if it was included in the vocabulary by the entity model, at step number 3.
4. mentions: number of tweets that include @user mentions

5.2.3 Meta features

1. timeframe: distance in time between the newest and oldest tweet in cluster

2. unique links: number of unique hyperlinks contained within cluster

3. instagram links: number of hyperlinks pointing to instagram photo or video content

4. frequent entities: number of popular entities found. Entity is deemed frequent if it was included in the vocabulary by the entity model, at step number 3.

5. bounding box: represents the size of the bounding box within which all tweets are contained.

5.2.4 *Social features*

1. unique users: number of unique tweet authors

2. mean friends: average number of friends of cluster authors

3. mean followers: average number of followers of cluster authors

4. retweets: represent how many times tweets in cluster were retweeted

5.3 Learning the model

In order to learn the model using the features described above, we had to obtain a training set. So, we ran our clustering models on historical data from our dataset, and labeled the resulting clusters by hand. We considered a time frame between November 27 and December 8, and repeated the clustering process, window by window, city by city. As a result, we obtained a dataset containing 250 true events and 2800 false event clusters. Due to the fact that the classes were very imbalanced, we had to employ oversampling for the true event dataset.

We employed a regularized multilayer perceptron model as a classifier. We used back-propagation and gradient descent with adaptive step size for learning the weights. Since the true event dataset was rather small, we used 10-fold cross validation. We experimented a lot with the number of hidden layers and the number of nodes within. First of all, we figured out that a network with two hidden layers and a sigmoid activation function achieved the best result. Then we started experimented with the number of nodes in hidden layers.

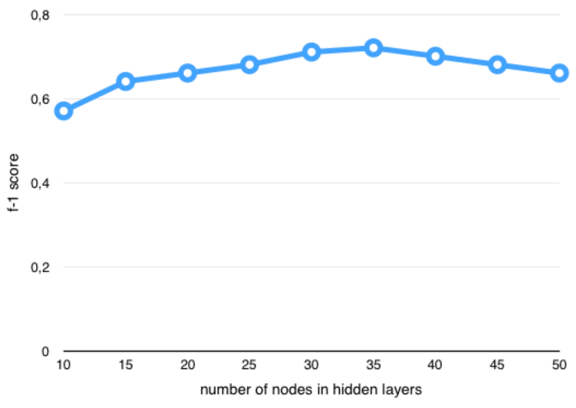

Figure 4: Adjusting number of hidden nodes

The model with 35 nodes achieved the best f-1 score in our experiments. We also utilized early stopping for regularization: from cross-validation we have learnt that stopping at iteration 3000 yields the lowest generalization error. Despite the fact that our training set was very limited, we were able to achieve reasonably good classification error. The classification report for our best-performing model is provided below:

Table 2: Classification report

	Precision	Recall	f1 score	support
True	0.72	0.79	0.74	433
False	0.76	0.67	0.70	437
avg/total	0.74	0.73	0.72	870

5.4 Sample output

The events classified as True are passed to a post-processing step. There, clusters are summarized, instagram hyperlinks are extracted and corresponding pictures are presented to the user. Currently, summarization is performed simply by extracting top-5 most frequently occurring entities. Below we present an example of an output of our system. Since one does not simply download images from Instagram (developer access is required), in the actual output only the links are present: we followed the links and extracted the photos by hand for the purpose of this demonstration. Implementing an automatic Instagram picture scrapper is possible, but will require some more work.

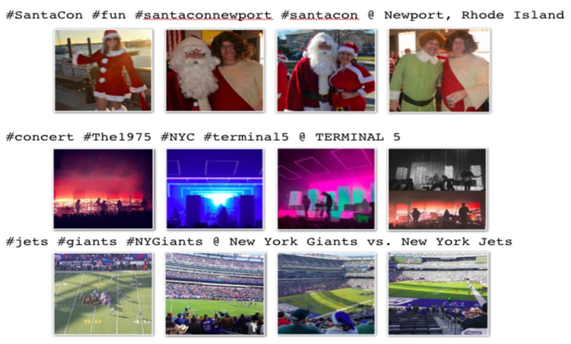
#SantaCon #fun #santaconnewport #santacon @ Newport, Rhode Island
#concert #The1975 #NYC #terminal5 @ TERMINAL 5
#jets #giants #NYGiants @ New York Giants vs. New York Jets

Figure 5: Sample system output

6. CONCLUSIONS

We proposed 3 clustering models and a feature-rich classifier to recognize news events for event detection. We defined novel statistical, social and textual features for the classifier and trained and 2 hidden layer neural network model using a self-developed event dataset. Experiments showed the effectiveness of the proposed method. The feature-rich event filter led to significantly higher precision and recall when compared to the state-of-the-art baseline system Twevent by Chenliang Li et al [3]. In our experiments we observed that a news event can be detected more than once in one time window, which each appearance representing one aspects of the event. Merging these sub-events into a hierarchy will be explored in the future work.

7. FUTURE WORK

We have shown that using non-textual features of documents available in Twitter for clustering and classification provides a considerable boost to the quality and quantity of events detected. The next step would be to combine the described clustering models into one model. This will require computing some weighted combination of document similarities based on entity, geodata, text and thus can be seen as a multi-kernel learning problem. We plan to tackle this problem in the future work.

8. REFERENCES

[1] C. C. Aggarwal and C. Zhai. A survey of text clustering algorithms. in mining text data. *Springer: New York*, 2012.

[2] J. Allan. Topic detection and tracking: event-based information organization. *Springer, volume 12*, 2002.

[3] Aixin Sun Chenliang Li. Twevent: segment-based event detection from tweets. *CIKM '12 Proceedings of the 21st ACM international conference on Information and knowledge management*, 2012.

[4] Luis Gravano Hila Becker, Mor Naaman. Beyond trending topics: Real-world event identification on twitter. *Fifth International AAAI Conference on Weblogs and Social Media*, 2011.

[5] Hanan Samet er al. Jagan Sankaranarayanan. Twitterstand: News in tweets. *University of Maryland*, 2009.

[6] Carlos Martin Dancausa et al. Luca Maria Aiello, Georgios Petkos. Sensing trending topics in twitter. *IEEE Transactions on Multimedia*, 2013.

[7] Sasa Petrovic. Real-time event detection in massive streams. *University of Edinburgh*, 2012.

[8] Tsuyoshi Murata Swit Phuvipadawat. Breaking news detection and tracking in twitter. *2010 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*, 2010.

[9] T. Pierce Yaang, Y. and J. Carbonell. A study of retrospective and on-line event detection. *SIGIR 98, ACM, New York, NY*, 1998.

[10] J. Zhang J. Carbonell Yang, Y. Topic-conditioned novelty detection. *KDD 02, ACM, New York, NY*, 2002.