# Online Event Detection and Tracking in Social Media Based on Neural Similarity Metric Learning

Guandan Chen[1,2], Qingchao Kong[1,2], Wenji Mao[1,2]
[1]The State Key Laboratory of Management and Control for Complex Systems, Institute of Automation, Chinese Academy of Sciences, China
[2]School of Computer and Control Engineering, University of Chinese Academy of Sciences, China
{chenguandan2014, qingchao.kong, wenji.mao}@ia.ac.cn

***Abstract*—The ever-growing number of users makes social media a valuable information source about recent events. Event detection and tracking plays an important role in decision-making and public management. Despite recent progress, the performance of event detection and tracking is still limited. The majority of existing work lacks an effective way to judge whether a text related to a certain event, due to the limitations of semantic representation and heuristic similarity metric. In this paper, we present an online event detection and tracking method based on similarity metric learning using neural network. Our method first trains a classification model to identify event related texts. To detect and track events, we adopt a clustering-based approach. Specifically, we use neural network to jointly learn a similarity metric and low dimension representation of events, and then use a memory module to store and update event representation. Experiments on Twitter dataset show the effectiveness of our proposed method.**

***Keywords—event detection and tracking; deep neural network; social media***

## I. INTRODUCTION

Social media has become the mainstream platform for people to discuss recent events, which makes it a valuable information source of recent events and opinions about them. However, information overload in social media makes it hard to acquire information related to a certain event. Event detection and tracking try to solve this problem by collecting and aggregating information related to interesting events. Event detection and tracking plays an important role in the security domain. As a social sensor, it provides data for impact analysis, decision making and public management.

Event detection and tracking is not a trivial task because of the following reasons. Firstly, the number of tweets in social media is very large. Secondly, there are various expressions of the same meaning, and many informal expressions in social media. Moreover, the online contents in social media keep updating and new events keep appearing.

To address these problems, we propose an event detection and tracking method based on neural network. We train a classification model to detect event related tweets, and jointly learns a neural similarity metric and low dimension representations for events. We use a memory module store and update low dimension representation of recent events dynamically, which enables us efficiently track events by computing similarity between event representation and tweet representation.

Our method has the following advantages: (1) our similarity metric is based on deep neural network, and learned from data, which makes it more powerful and robust than previous related work; (2) we introduce attention mechanism in our neural network to make it focus on most important features; (3) the online paradigm and low dimension representation of events ensure time and space effectiveness of our method. Experiments on a large dataset from Twitter demonstrate the effectiveness of our proposed method.

## II. RELATED WORKS

The event detection and tracking methods fall into the following categories: *topic modeling based methods*, *term weighting based methods* and *clustering based methods*. For topic modeling based methods, Online-LDA [1] is the represent method. However, The work in [2] shows that Online-LDA can not handle events that are reported in parallel.

As the discussion of recent events will cause the burst of word frequency signal in social media, term weighting based methods use such burst patterns to detect event words. These approaches identify words with salient scores as event words, compared to scores in other time windows. For example, [3] uses peakiness score, which is similar to TF-IDF metric, and [4] uses trending score, which normalized term frequency of an n-gram with regard to other n-grams in the same time window. However, these methods lack an effective way to merge words related to the same events. To solve this problem, clustering based methods cluster words using frequency and content similarity. These methods use different similarity metrics. EDCoW [5] compute cross correlations by wavelets of word frequency. ET [6] uses both frequency and text content to compute similarity. [7] leverages the frequency of mention mark @ as a signal of words, and use similarity of this signal to cluster event words.

Compared to other event detection and tracking methods, clustering based methods have better performance in general. In this paper, we propose a clustering based method to event detection and tracking. We take tweets as an example of social media form. To detect event related tweets, most clustering based methods use anomaly of word frequency signal (except for [6], which uses statistical n-gram language model). In contrast, our method uses a deep neural network to extract text related representation. Moreover, our method jointly learns representation and similarity metric of tweets and events from data.

## III. PROPOSED METHOD

In this section, we formulate the event detection and tracking problem, and introduce our neural network based event detection and tracking method.

### A. Problem Formulation

We are dealing with sample tweets from Twitter stream. Our goal is to identify and aggregate the tweets related to real word events. Specifically, for the recent tweets $\{T_1, T_2, \dots T_n\}$, we detect event-related tweets $\{T_1, T_2, \dots, T_m\}$ that are discussing about some events. Suppose we already detected $k$ events $E = \{e_1, e_2, \dots, e_k\}$, for each event-related tweet $T$, if the event $e_i$ that $T$ is discussing about belong to $E$, we update $e_i$ in $E$, otherwise we add $e_i$ to $E$. Each $e_i$ contains the related tweets and an event feature, so the update operation add new tweet $T$ into the tweet set, and update the feature according to $T$. We define the magnitude of an event as the number of related tweets. The model would output the top $k$ events with largest magnitude.

### B. Overview of the Proposed method

Our method consists of two phases. Firstly, it is event related tweets detection, which would filter tweets not related to a real world event. Secondly, in event tracking phase, we aggregate tweets related to the same event together. As an online manner, if a tweet related to an event stored in the memory module, we fetch event id from memory module and update the memory content of this event. Otherwise, a new event id is assigned to the event and the representation of the event is added to the memory module. A similarity metric is learned to judge whether a tweet related to the events in the memory module.

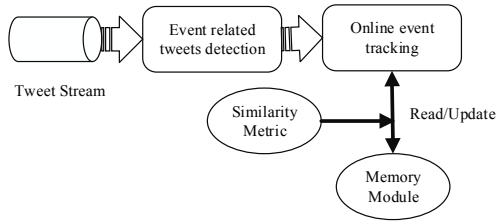


Fig. 1. Framework of our event detection and tracking method

### C. Event Related Tweets Detection

There are many tweets not related to a real world event in the Twitter stream, e.g. tweets about daily chores. It will provide benefits for our event tracking task to filter these tweets.

Given a tweet, we represent it as a sequence of words $[w_1, w_2, \dots, w_n]$. We map each word $w_j$ into a vector $x_j$ using glove, which is a widely used word embedding model. Then we use bidirectional gated recurrent units (GRU) with attention mechanism to map each tweet into a vector in the hidden space. The bidirectional GRU map a sequence into another sequence, i.e. $[h_1, h_2, \dots, h_n] = BiGRU([w_1, w_2, \dots, w_n])$, where each state $h_j$ encodes the context information of $w_j$. Then we introduce attention mechanism[8] to aggregate the states, and encode the tweet into a fix-length vector, i.e. $s = Att([h_1, h_2, \dots, h_n])$. This attention operation maps a sequence of states into a vector by weight summing.

We use a sigmoid function to predict whether a tweet is event related, i.e. $y = \sigma(\mathrm{w_s} s + b)$. If $\mathrm{y} > 0.5$, the tweet is considered as event related. We use cross-entropy as our loss function, and use Adagrad to optimize the parameter weights.

### D. Similarity Metric Learning

In event tracking phase, we aggregate tweets related to the same event together. We learn a similarity metric to judge whether two tweets related to the same event. Given tweet pairs like $(T_1, T_2)$, we try to learn a function $f(T_1, T_2)$, which outputs a value close to 1 if $T_1$ and $T_2$ discuss about the same event, outputs a value close to 0 or else. We limit the function further, that it can be represented as $f(T_1, T_2) = g(T_1) \cdot g(T_2)$. In this way, we only need to learn a function $g(T)$ to map a tweet to a vector in a hidden space, which can be also low dimension representation of tweets.

Our $g(T)$ function is a neural network with similar structure in event related tweets detection. Each tweet is encoded into a fixed length vector $s$ using bidirectional GRU with attention mechanism. Then we normalize the vector s to $\tilde{\mathrm{s}} = \frac{s}{\|s\|}$, so that $\tilde{\mathrm{s}}$ has a fixed modulus 1. The similarity between two vectors can be conveniently computed using $\tilde{\mathrm{s}}_{\mathrm{i}} \cdot \tilde{\mathrm{s}}_j$.

To learn parameter weights in $g(T)$. We sample positive tweet pairs which related to the same event and negative pairs which related to different events. For each tweet pair $(T_1, T_2)$, $y = g(T_1) \cdot g(T_2)$ gives the prediction result of weather these tweets belong to the same event. We use cross-entropy as our loss function, and use Adagrad to optimize the parameter weights.

### E. Event Tracking Using Memory Module

The event representation is weighting sum of the vector representations of related tweets. We use a memory module to store and update representations of events. The structure of our memory module is the same as [9], the main difference is that we take a parametrized update strategy. The memory module is defined as a triple: $M = (K, V, A)$, with $K \in R^{m \times d}, V \in R^m, A \in R^m$. $K$ stores low dimension representations of events. $V$ holds the ids of events. $A$ holds the ages of events since last update. $m$ is the memory size, and $d$ is the dimension of event representation.

Given a tweet $T$, its low dimension representation is computed by $g(T)$ from learned similarity metric, i.e. $\tilde{s} = g(T)$. Then we select an entry $k$ in memory by nearest neighbor. The similarity is computed by $sim_i = \tilde{s} \cdot K_i$. If the similarity exceeds threshold $\sigma$, we will add this tweet to the event with id $V_i$, set the age of the event as $A_i = 0$, and update the event representation as:

$$K_i = \frac{K_i + \alpha(\tilde{s} - K_i)}{\|K_i + \alpha(\tilde{s} - K_i)\|}$$

This operation slightly shifts the event representation $K_i$ to the tweet representation $\tilde{s}$. Here, $\alpha$ is a parameter to control the update rate. If the similarity does not exceed threshold $\sigma$, we will add a new event to the memory. The oldest memory entry $n' = argmax_i A_i$ is selected to store information of the new event, i.e.$\mathrm{K}_{n'} = \tilde{\mathrm{s}}, \mathrm{V}_{n'} = v, A_{n'} = 0$, where $v$ is the unique id for the new event. After every update, the age of events in memory module will increment automatically.

## IV. EXPERIMENT

### A. Dataset and Experiment Setting

We collect tweets using Twitter public API (https://dev.twitter.com/streaming/overview). We annotate 619 events from Aug 10, 2016 to Sept 10, 2016. To train event related tweets detection model, we randomly sample 33,808 event related and 33,808 non-event related tweets. Similarly, for learning similarity metric, we sample 1000,000 tweet pairs that belong to the same event as positive samples, and 1000,000 pairs that belong to different events as negative samples. We use 9,563,979 tweets during Nov 10, 2016 to Dec 10, 2016 to evaluate event detection and tracking method.

In our experiment, word embedding dimension is 100, GRU size as 100, memory size is 1000. Memory updating rate $\alpha$ is set to 0.4, and the similarity threshold $\sigma$ is set to 0.5.

### B. Baseline Methods

We compare our methods to the following baseline methods: (1) *PS* [3]: it uses burst patterns of Peakiness Score to detect and track events, which is a normalized word frequency metric. (2) *TS* [2]: it uses Trending Score to detect and track events, which is a normalized n-gram frequency according to other n-grams in the same time window. (3) *MABED* **[7]**: it uses the number of co-occurrence times with mention mark @ as a signal of a word. [7] detects event words using the anomaly of this signal, and clusters event words using the dynamic of this signal. *MABED* has better performance than *EDCoW* [5] and *ET* [6], so we do not compare our method to these two methods here.

### C. Evaluation metrics

We take the same evaluation method as in [7], that two humans are asked to judge whether the detected events are meaningful or duplicate, and only top 40 events with largest magnitude are judged considering that this is a time consuming task. Precision, recall, F1, and DERate (the fraction of duplicate events) are used as evaluation metrics.

### D. Model Comparisons and Results

TABLE I shows the performance of our method and baseline methods. Our method gets the best performance over all evaluation metrics. The PS method and TS method have lower performance. Further analysis of the events detected by PS and TS shows that their performance mainly reduced by spam words. The MABED method clusters some irrelevant words together and sometimes fails to merge same event together because the time overlap of events does not reach given threshold. This shows that MABED is unstable. Instead, our method learns a similarity metric from data, and the supervised manner ensures its robustness. The DERate of our method is 0.057, which shows the learned similarity metric can effectively reduce duplicate events, while keeping most meaningful events.

TABLE I. PERFORMANCE OF DIFFERENT METHODS

| Methods | *Precision* | *Recall* | *F1* | *DERate* |
|---|---|---|---|---|
| PS | 0.425 | 0.398 | 0.375 | 0.118 |
| TS | 0.450 | 0.409 | 0.375 | 0.167 |
| MABED | 0.732 | 0.650 | 0.585 | 0.200 |
| Our method | **0.800** | **0.774** | **0.750** | **0.063** |

To further show the effectiveness of learned similarity metric, we provide some examples from our detected events in TABLE II (in which bold words have highest attention weights). We observe that these tweets contain many different words, which will have low similarity using simple bag of words model. Our model successfully aggregates these tweets to the same event by assigning high attention weights on the keywords.

TABLE II. SAMPLE EVENTS DETECTED BY OUR METHOD

| Events | Sample tweets |
|---|---|
| Lackawanna fire | Another photo sent by a viewer of the #Lackawanna **fire** at former **Bethlehem Steel** site. |
| | The **fire** at the former **Bethlehem Steel** facility likely started when a hot lightbulb landed on some combustibles. |
| London tram derailed | British Transport Police has said five people have been **killed** after a **tram derailed** in Croydon - the driver has been arrested. |
| | The number of people killed after a **London tram derailed** rises to seven. |

## V. CONCLUSION

In this paper, we propose an event detection and tracking method based on neural network. We use a neural network to detect event related tweets. Then we jointly train a similarity metric and low dimension representation of events. It enable us to use a memory module to store and update representations of events dynamically. Our method efficiently tracks events by computing similarity between tweet representation and event representation. Experiments on Twitter dataset show that our method outperforms baseline methods.


## ACKNOWLEDGEMENT

This work is supported in part by NSFC Grant #71621002, #61671450 and #71472175, and the Ministry of Science and Technology of China Major Grant.


## REFERENCES


[1] J. H. Lau, N. Collier, and T. Baldwin, "On-line Trend Analysis with Topic Models:\# twitter Trends Detection Topic Model Online," in *COLING*, 2012, pp. 1519-1534.

[2] L. M. Aiello *et al.*, "Sensing trending topics in Twitter," *IEEE Transactions on Multimedia,* vol. 15, no. 6, pp. 1268-1282, 2013.

[3] D. A. Shamma, L. Kennedy, and E. F. Churchill, "Peaks and persistence: modeling the shape of microblog conversations," in *Proceedings of the ACM 2011 conference on Computer supported cooperative work*, 2011, pp. 355-358: ACM.

[4] J. Benhardus and J. Kalita, "Streaming trend detection in twitter," *International Journal of Web Based Communities,* vol. 9, no. 1, pp. 122-139, 2013.

[5] J. Weng and B.-S. Lee, "Event detection in twitter," *ICWSM,* vol. 11, pp. 401-408, 2011.

[6] R. Parikh and K. Karlapalem, "Et: events from tweets," in *WWW*, 2013, pp. 613-620: ACM.

[7] A. Guille and C. Favre, "Mention-anomaly-based event detection and tracking in twitter," in *ASONAM*, 2014, pp. 375-382: IEEE.

[8] Z. Yang, D. Yang, C. Dyer, X. He, A. Smola, and E. Hovy, "Hierarchical attention networks for document classification," in *NAACL-HLT*, 2016, pp. 1480-1489.

[9] Ł. Kaiser, O. Nachum, A. Roy, and S. Bengio, "Learning to remember rare events," *arXiv preprint arXiv:1703.03129,* 2017.