Joint Learning with Keyword Extraction for Event Detection in Social Media

Guandan Chen[1,2], Wenji Mao[1,2], Qingchao Kong[1,2], Han Han[3,*]
[1]The State Key Laboratory of Management and Control for Complex Systems, Institute of Automation, Chinese Academy of Sciences, China
[2]University of Chinese Academy of Sciences, China
[3]CNCERT/CC, China
{chenguandan2014, wenji.mao, qingchao.kong}@ia.ac.cn, hanh@cert.org.cn

***Abstract*— Event detection for social media is an important social media analytics task in security domain, which can provide valuable information for decision making, intelligence analysis and public management. Traditional event detection methods either rely on simple term frequency based features, or take the bag of words assumption. Recently, some deep neural network based event detection methods are proposed. However, these methods still have some drawbacks. Firstly, they do not provide an effective way to learn the connection between message representation and event representation, but simply use average of message representations as the event representation. Secondly, representations in the hidden space lack interpretability compared to the traditional event keyword representation. To deal with these weaknesses, we propose an event detection approach joint learning with keyword extraction. We provide an episode learning strategy to enable the training of event representation update. In addition, by joint learning with keyword extraction, the model is more explainable, and can achieve a better performance. As the selection of a set of keywords is a combinatorial problem and non-differential, we also employ reinforcement learning in our approach. Experiments on a public available dataset show the superiority of our approach compared with baseline methods.**

Keywords—event detection, keyword extraction, deep learning

I. INTRODUCTION

Social media is now one of the mainstream platforms for people to discuss recent events. Although it has been an important information source for recent events, information overload in social media makes it hard to acquire the needed information for event analysis. Event detection addresses this issue by filtering out non-event tweets and clustering them according to the discussed events. Event detection thus can provide valuable event information for public management, decision making, and plays an important role in security domain.

Event detection methods in previous work mainly fall into three categories, namely, term weighting based approaches clustering based approaches and topic model based approaches. As social media delivers a continuous stream of messages, the discussions of recent events show a burst pattern of event word frequencies. Term weighting based approaches [1-5] detect event words according to this phenomenon. However, these methods mainly focus on mining event words, and do not directly provide event related messages, which is important for further deep analysis of events in many applications. On the other hand, clustering-based approaches [6, 7] cluster messages according to some text features, e.g. tf-idf. The main drawbacks of these methods is that they usually require feature-engineering to design useful features. Topic model based approaches usually assume a document generation process, and detect events by optimizing document generation probability. These methods take the bag of words assumption, and event representation relies on high dimension word probability distribution. To solve the problems of these methods, Chen et al. [8] propose a neural similarity metric learning based event detection method, which achieves the best performance on a Twitter dataset. This method takes modern deep neural network to model text content. As an online method, it uses a memory module to store and update event representations. However, this method suffers from two drawbacks: (1) the event representations are the average of the tweet representations, which may be distorted by noise. (2) the tweets and event representations are all vectors in a hidden space, which lack interpretability.

To address these problems, in this paper, we propose an event detection approach joint learning with keyword extraction. Our approach reconciles explainable event keywords and the good performance of modern deep neural network. Firstly, we use recurrent neural network and attention mechanism to learn tweet representation, without the bag of words assumption. Secondly, we propose an episode learning method, which learns an event representation update function using tweet sequence, instead of simple average approach. Thirdly, our method jointly learns with keywords extraction, which is more explainable. As selecting a set of keywords is a combinatorial problem, which is non-differential, we also employ reinforcement learning to learn to extract keywords, which can optimize keyword extraction from event detection without extra annotation about keywords.

The contributions of this paper are three-fold:

- We propose a novel approach for event detection joint learning with keyword extraction, which combines explainable keyword representation and the good performance of deep neural network.
- As selecting a set of keywords is a combinatorial problem and non-differential, we employ reinforcement learning for keyword extraction.
- Experimental results on a public available dataset shows the superiority of our proposed method.

* Corresponding author: Han Han

978-1-5386-7848-0/18/$31.00 ©2018 IEEE

II. Related Work

Previous methods on event detection mainly fall into three categories, namely, term weighting based approaches clustering based approaches and topic model based approaches.

A. Term Weighting based Approaches

Social media delivers a continuous stream of messages, and discussions of new events will cause burst patterns of event keywords. Based on this phenomenon, term weighting based approaches design some term frequency scores, and take words with salient scores as event keywords. Early studies uses some scores similar to tf-idf, e.g. peakiness score [1] and trending score [2]. However, these methods did not provide an effective way to merge keywords related to the same event. To deal with this issue, EDCoW [3], ET [4] and MABED [5] merge words according to similarity based on some frequency signals. To compute similarity, EDCoW leverages wavelets of word frequency, while ET uses both frequency signal and text content, and MABED computes similarity according to the frequency of mention mark "@".

B. Clustering based Approaches

Term weighting based methods mainly focus on mining event words, and do not directly provide event related messages. In contrast, clustering based approaches aggregate messages belonging to the event, and thus are more convenient for further deep analysis about the event, e.g. stance detection, opinion mining, etc. Most of these approaches rely on text feature designing. UMASS [6, 9, 10] uses tf-idf as feature and takes incremental clustering algorithm. Pohl et al. [7] proposes an event detection method based on the tf-idf feature and self-organizing map clustering algorithm. Recently, Chen et al. [8] propose a neural similarity metric based event detection method (NSMED). NSMED uses modern deep neural network to model text sequence, and reaches better performance compared to the traditional approaches.

C. Topic Model based Approaches

Topic model based approaches assume a document generation process, and then infer topic distributions of documents by optimizing generation probability. For example, online-LDA [11] is one of these methods. In addition, MGeLDA [12] adds a generation layer about hashtag to the original LDA, which generates hashtag and document mutually. The main drawback of these methods is that they all rely on the bag of words assumption.

III. Proposed Method

Our proposed method processes social media stream in an online manner, and assigns each message to the event that it discusses. Our method relies on several components. Firstly, the encoder maps each message into a vector representation in the hidden space. The memory module stores the event representations and keywords. The actor takes an action from the following three options, according to tweet representations, event representations and keywords: (1) add an event; (2) take the message as noise and drop it; (3) select the event that the message discusses, and update the event representation and keywords. Figure 1 shows the model architecture.

A. Encoder

Each message is a sequence of words, i.e. $[w_1, w_2, \dots, w_n]$. We map each word into an embedding vector using Glove [13], a widely used word embedding model, i.e.

$$x_j = Ew_j, j \in [1, n]$$

where E is the embedding matrix, w_j is one-hot representation of the *j*-th word, and x_j is the embedding vector of the *j*-th word.

Then we employ a bidirectional gated recurrent units (GRU) [14] to model the sequence. The GRU processes elements in the sequence in a recurrent manner, and at each step it updates a state vector according to the previous state vector and the current input. Specifically, GRU is computed from

$$z_j = \sigma\left(W_z x_j + U_z h_{j-1} + b_z\right)$$

$$r_j = \sigma\left(W_r x_j + U_r h_{j-1} + b_r\right)$$

$$C_j = \sigma\left(W_h x_j + U_h\left(r_j \odot h_{j-1}\right) + \mathrm{b_h}\right)$$

$$h_j = \left(1 - z_j\right) \odot h_{j-1} + z_j \odot C_j$$

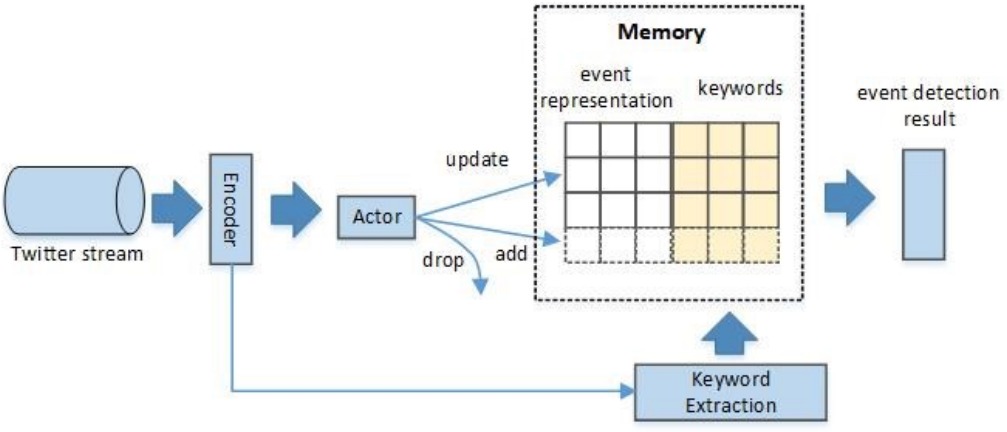

Fig. 1. Architecture of the Proposed Model

where z_j and r_j are update gates and reset gates respectively, controlling information from previous states or inputs. Here W, U and b are trainable parameters, and $\odot$ is element-wise multiplication.

Bidirectional GRU processes the sequence from both the head to the tail and the tail to the head, and then concatenate two states as output. It is computed from

$$\vec{h}_j = \overrightarrow{GRU}(x_j, \vec{h}_{j-1})$$

$$\overleftarrow{h}_j = \overleftarrow{GRU}(x_j, \overleftarrow{h}_{j-1})$$

$$\mathrm{h_j} = [\vec{h}_j, \overleftarrow{h}_j], \mathrm{j} \in [1, \mathrm{n}]$$

where $\vec{h}_j$ and $\overleftarrow{h}_j$ represents the j-th state produced by GRU from two directions respectively, x_j is the j-th input vector, h_j is the j-th state, and $[\cdot,\cdot]$ represents for concatenation operation.

Finally, we use attention mechanism [15] to map the sequence $[h_1, h_2, \dots, h_n]$ into a vector. The attention mechanism output the weighted summing of the sequence. By allocating different attention weights, it can emphasize informative states and suppress less useful ones, which is computed from:

$$\mathrm{z} = \mathrm{Att}([h_1, h_2, \dots, h_n])$$

where z is the tweet representation. Specifically, attention mechanism is computed as

$$u_j = \tanh\left(W_h h_j + b_h\right), j \in [1, n]$$

$$\alpha_j = \frac{\exp\left(u_s^{\mathrm{T}} u_j\right)}{\sum_r \exp(u_s^{\mathrm{T}} u_r)}$$

$$z = \sum_j \alpha_j h_j$$

We first map each state into a vector representation u_j in a hidden space. Then, an attention weight α_j is computed from similarity between u_j and a trainable vector u_s, which indicates informative words over the whole text sequence, where W_h and b_h are trainable weights.

B. Actor

We store event representations and keywords in a memory module [16]. Specifically, the memory module consists a matrix and a keyword collection. The j-th row M_j in the matrix is the vector representation for the j-th event stored in the memory module, and its keywords is $\{k_{j,1}, k_{j,2}, \dots, k_{j,m}\}$ for $\mathrm{j} \in [1, H]$,where H is the number of events in the memory module.

According to the events in the memory module and tweet representation, the actor selects an action from the following options:

- If the tweet does not discuss an event, it will be considered as noise and discarded.
- If the tweet discusses an event that is not stored in the memory module, we add the event into the memory module.
- If the tweet discusses an event in the memory module, we will update the event representation and keywords.

The action is selected according to:

$$x_{j,a}^{(kw)} = Ew_{j,a}, a \in [1, m]$$

$$s_j^{(kw)} = Att([\mathrm{x}_{j,1}^{(kw)}, \mathrm{x}_{j,2}^{(kw)}, \dots, \mathrm{x}_{j,m}^{(kw)}])$$

$$z_j^{(M)} = ReLU\left(Dense\left([M_j + z + s_j^{(kw)}, M_j \odot \mathrm{z} \odot s_j^{(kw)}]\right)\right)$$

$$z^{(M)} = MaxPool([z_1^{(M)}, z_2^{(M)}, \dots, z_H^{(M)}])$$

$$p^{(act)} = \mathrm{softmax}\left(\mathrm{Dense}\left(\mathrm{z}^{(M)}\right)\right)$$

where $ReLU$ represents rectified linear unit, a widely used activation function in neural network models, defined as $f(x) = max(0, x)$. $Dense$ is a dense connected layer, which is defined as $f(x) = Wx + b$ with trainable parameters W and b. Here $\odot$ is element-wise multiplication, and $p^{(act)}$ is the probabilities to select three actions (i.e. add, update or drop).

Event representation operation. If the action is "update", we select the event that the message discusses about, which is computed from

$$z_j^{(u)} = ReLU\left(Dense\left(z_j^{(M)}\right)\right)$$

$$p^{(u)} = softmax([\mathrm{z}_1^{(u)}, \mathrm{z}_2^{(u)}, \dots, \mathrm{z}_{\mathrm{H}}^{(u)}])$$

where $\mathrm{z}_{\mathrm{j}}^{(u)}$ is the feature for the j-th event in the memory module, and $p^{(u)}$ is the probability distribution of the events that the tweet discusses.

Suppose we need to update the j-th event, the "update" operation is computed from

$$g_u = \sigma\left(Dense\left([M_j, z]\right)\right)$$

$$C = \tanh\left(Dense\left([M_j, z]\right)\right)$$

$$M_j' = g_u \odot M_j + (1 - g_u) \odot C$$

where g_u plays a role like gates, controlling the information coming from old representation, C is a vector representation, and M_j' is the new representation for the j-th event. The "add" operation is similar to "update" operation, which replaces M_j with an initial state M_0.

Keyword operation. The keyword selection probability is computed from

$$p^{(kw)} = softmax\left(Dense\left(z_j^{(M)}\right)\right)$$

where $p^{(kw)}$ is the probability of selecting a word as keyword. During training we randomly sample m keywords according to $p^{(kw)}$, while during testing we select m words with the largest probability. The "add" operation is similar to "update" operation, only replacing old keywords with a set of special symbols.

C. Leanring

Algorithm: Episode generation

Input:

Sequence length: N

Number of events in the episode: K

All events in corpora: E

Non-event tweets in corpora: T

Output:

A message sequence and optimal action sequence

Algorithm:

Init: Episode event set $E_e = \{\}$, Sequence $S = []$

For *i* in [1, N]

  Generate two random numbers $r_{noise}, r \in [0,1]$

  If $r_{noise} < p_{noise}$

    Randomly select a message m from T

  Else

    If $|E_e| < K$ and $r < p_{new}$

      Randomly select an event e from $E - E_e$

    Else

      Randomly select an event e from E_e

    Randomly select a message m discussing e

  Add m to S

Generate optimal action sequence $y^{(act)}$and $y^{(u)}$ according to the events that messages discuss

Return S, $y^{(act)}$, $y^{(u)}$

Fig. 2. Episode generation algorithm

Episode learning strategy. We learn the parameter weights in the model from the training dataset. However, it is hard to be optimized if we use the entire message stream because of the difficulty of learning from long sequences. Thus, we adopt episode learning strategy. In this strategy, we randomly sample a short sequence of tweets, which belong to K different events and its length is N. Figure 2 shows the detailed process of generating an episode. During training, we already know the optimal actions (i.e. add, drop and update) and events to be updated, so the loss is the cross-entropy computed from the action distribution, i.e.

$$L_{action} = \sum_i y_i^{(act)} \log p_i^{(act)}$$

$$L_e = \sum_j y_j^{(u)} \log p_j^{(u)}$$

where L_{action} and L_e are the losses for taking actions and selecting events for updating respectively. $y^{(act)}$ and $y^{(u)}$ are the optimal actions and events for updating respectively. $p_i^{(act)}$ and $p_j^{(u)}$ are the predicted probabilities of actions and events respectively.

Reinforcement learning. We adopt reinforcement learning for two main reasons. Firstly, the selection of keywords is a combinatorial problem and non-differential, which can not be learned using gradient backpropagation. Secondly, cross-entropy is not consistent with metrics used during testing, e.g. NMI [17] or B-cubed [18]. Thus, we also train the model using reinforcement learning. Specifically, PPO [19] algorithm is adopted, which is an optimization algorithm for reinforcement learning. The reward signal is the NMI metric.

During training, we alternatively use gradient backpropagation and reinforcement learning to train the model.

IV. Experiment

A. Dataset

We use a public available dataset [20] to evaluate our approach. The dataset was collected via Twitter platform from December 10, 2012 to January 7, 2013. As one of the largest social media platforms, Twitter owns millions of active users. Users all over the world discuss and share recent events on Twitter, which makes Twitter a valuable source of timely information. The dataset contains 120 million tweets in total, and 15M tweets among them discuss events. It includes 506 different events, all manually annotated.

We use NMI [17] and B-cubed [18] as the evaluation metrics, which are widely used when we know the ground-truth. NMI is a metric from information theory perspective, which measures how much information is shared between the predicted events and ground-truth events. B-cubed computes precision and recall associated with each message individually. Then it is computed from average precision and average recall as harmonic average of them.

B. Baseline Methods

We select the following methods as the baseline methods.

- UMASS [6]: Adopts tf-idf as features, and detects events using incremental clustering algorithm.

- LSH [21]: Considering the large volume of social media data, Petrovic et al. proposes an event detection method based on locality-sensitivity hashing (LSH).
- tfidf+SOM [7]: Adopts tf-idf as features, and detect events by self-organizing map (SOM) clustering algorithm.
- LDA [22]: Latent Dirichlet Allocation, assumes the document generation processes, and optimizes text generation probability, then it clusters texts using topic distribution as features.
- MGeLDA [12]: An extension of LDA by adding a layer about hashtag, which generates documents and hashtag mutually.
- NSMED [8]: A neural similarity metric based event detection method.

C. Results

Table I shows the performances of our proposed method and baseline methods. It can be seen that our proposed method outperforms the baseline methods, with a significant improvement in both NMI and B-cubed metrics. Compared to the variation of our model, without joint learning with keyword extraction, our model improves NMI by more than 2% and improves B-cubed by 1%. This shows the superiority of joint learning with keyword extraction. NSMED is the best baseline method based on deep model, which shows the superiority of modern deep neural network. NSMED takes the average of tweet representation as the event representation, while our model learns event representation updating function from data and thus can reach better performance. It shows the necessity of learning to update event representations. UMASS and tfidf+SOM both rely on tf-idf features, which have lower performances. LDA and MGeLDA are topic model based methods, but inferior to the best method in event detection task. Although LSH is very fast, it shows relatively worse performance compared to other methods.

TABLE I. COMPARISION OF DIFFERENT METHODS

Method	***NMI***	***B-cubed***
UMASS	0.709	0.270
LSH	0.681	0.268
tfidf+SOM	0.702	0.307
LDA	0.635	0.299
MGeLDA	0.641	0.351
NSMED	0.710	0.358
EDJKE (without keyword extraction)	0.721	0.369
EDJKE	**0.744**	**0.380**

D. Impact of Parameters

We also investigate the impact of some parameters in our method, e.g. K and N in the episode learning strategy. Table II and Table III shows the comparison results with respect to NMI and B-cubed respectively.

TABLE II. IMPACT OF K AND N (NMI)

N \ *K*	*2*	*4*	*8*	*16*	*32*	*64*
2	0.745	0.726	0.748	0.742	0.731	0.742
4	0.742	0.74	0.742	0.745	0.739	0.733
8	0.714	0.708	0.741	0.744	0.0504	0.745
16	0.131	0.042	0.0712	0.107	0.157	0.0427
32	0.134	0.168	0.179	0.177	0.12	0.0425
64	0.095	0.164	0.161	0.101	0.149	0.175

TABLE III. IMPACT OF K AND N (B-CUBED)

N \ *K*	*2*	*4*	*8*	*16*	*32*	*64*
2	0.379	0.344	0.37	0.37	0.364	0.37
4	0.371	0.371	0.379	0.378	0.365	0.363
8	0.355	0.357	0.364	0.38	0.189	0.38
16	0.241	0.183	0.212	0.254	0.302	0.183
32	0.24	0.313	0.304	0.309	0.268	0.183
64	0.24	0.31	0.286	0.247	0.257	0.316

Impact of K. K is the number of events in a generated episode. We found that K does not influence performance of the model too much. This is reasonable, because the model mainly learns to distinguish tweets discussing different events, and when K increases, the model can still learn this information.

Impact of N. N is the length of the sequence in an episode. N has a great impact on the performance of the model, because it is difficult for a model to learn from a very long sequence for gradient vanish or gradient explosion.

E. Case Study

To investigate the quality of the extracted keywords, we provide a case study in Table IV. We found that the model can extract high quality keywords for some events, e.g. “Nigeria”, “flood” in the first event, which is an event about floods in Nigeria. This shows joint learning with keyword extraction can help better event detection. However, there are still some issues that we can not deal with. Firstly, it is hard to know the number of keywords, and in our experiment we set it to 3, but event 3 has another important keyword “Ukraine” that our model did not extract. Secondly, our model can not deal with acronyms like “ASEM” in event 4.

TABLE IV. CASE STUDY OF KEYWORD EXTRACTION

Event 1	Floods in Nigeria, number of people affected
keywords	Nigeria, flood, kill
sample tweets	Nigeria: Millions displaced by floods struggle with escalating food prices #Occupy #OpESR #OWS Agency: Nigeria floods kill 363 people, displace 2.1 m. Nigeria floods kill 363 people, displace 2.1 million: agency: ABUJA (Reuters) - Nigeria's worst flooding in at least... Officials: Months of flooding across Nigeria kill 363 people, displace 2.1 million others

Event 2	Judge in Australia rules that Standard & Poor's misled investors in its rating of two pre-crisis issues of structured debt
keywords	Australia, crisis, investor
samle tweets	"Misleading and deceptive" Standard & Poor's loses landmark Australia case over ratings given to financial products S&P Misled Investors On CPDO Notes Pre-Financial Crisis: Australian Court | Economy Watch Standard & Poor's Found Guilty of Misleading Investors: Australia's Federal court issued a ... S&P has been found guilty of misleading investors by awarding its triple A rating to a complex structured
Event 3	housands of people protest against alleged fraud in the recent parliamentary elections in Ukraine.
keywords	protest, rally, election
sample tweets	Ukraine opposition calls protest rally over election count Ukrainian opposition parties promise to refuse from the seats they won in protest against unfair election Opposition parties call for protest rally near the Central Election Commission building in central Kiev on 5 November. Hundreds rally in protest over "stolen" Ukraine election
Event 4	Asian and European leaders meet at the Ninth Asia–Europe Meeting in Vientiane.
keywords	attend, Laos, leave
sample tweets	Cambodian PM heads for Laos to attend ASEM Summit PM leaves for Laos to attend ASEM Chinese premier leaves for Asia-Europe Meeting, Laos visit PM Raja Pervez Ashraf leaves Islamabad for Laos on Sunday to represent Pakistan at the 2 day Asia-Europe Meeting Summit in Vientiane, Laos

V. CONCLUSION

In this paper, we propose an event detection approach joint learning with keywords extraction. Our approach reconciles explainable event keyword representation and good performance of modern deep neural network. Firstly, we use recurrent neural network and attention mechanism to learn tweet representation, without the bag of words assumption. Secondly, we propose a scenario learning method, which learns the event representation update function using tweet sequences, instead of simple average approach. Thirdly, our method jointly learns with keywords extraction, which is more explainable. As selecting a set of keywords is a combinatorial problem, which is non-differential, we employ reinforcement learning to automatically extract keywords. Experiments on a public available dataset show that our method outperforms the baseline methods that we compare with.

ACKNOWLEDGMENT

This work is supported in part by Ministry of Science and Technology of China (Grant No. 2016QY02D0305), National Natural Science Foundation of China (Grant No. 71702181, 71621002 and 71472175), Key Program of the Chinese Academy of Sciences (Grant No. ZDRW-XH-2017-3).

REFERENCES

[1] 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.

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

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

[4] R. Parikh and K. Karlapalem, "Et: events from tweets," in *Proceedings of the 22nd international conference on world wide web*, 2013, pp. 613-620: ACM.

[5] A. Guille and C. Favre, "Mention-anomaly-based event detection and tracking in twitter," in *Advances in Social Networks Analysis and Mining (ASONAM), 2014 IEEE/ACM International Conference on*, 2014, pp. 375-382: IEEE.

[6] C. C. Aggarwal and K. Subbian, "Event Detection in Social Streams," in *Proceedings of the 2012 SIAM international conference on data mining*, 2012, vol. 12, pp. 624-635: SIAM.

[7] D. Pohl, A. Bouchachia, and H. Hellwagner, "Automatic sub-event detection in emergency management using social media," in *International Conference on World Wide Web*, 2012, pp. 683-686.

[8] G. Chen, Q. Kong, and W. Mao, "Online event detection and tracking in social media based on neural similarity metric learning," in *Intelligence and Security Informatics (ISI), 2017 IEEE International Conference on*, 2017, pp. 182-184: IEEE.

[9] T. Umass *et al.*, "Detections, Bounds, and Timelines: UMass and TDT-3," *IN PROCEEDINGS OF TOPIC DETECTION AND TRACKING WORKSHOP (TDT-3,* 2000.

[10] G. Kumaran and J. Allan, "Text classification and named entities for new event detection," in *International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2004, pp. 297-304.

[11] 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.

[12] C. Xing, Y. Wang, J. Liu, Y. Huang, and W.-Y. Ma, "Hashtag-Based Sub-Event Discovery Using Mutually Generative LDA in Twitter," in *AAAI*, 2016, pp. 2666-2672.

[13] J. Pennington, R. Socher, and C. Manning, "Glove: Global Vectors for Word Representation," in *Conference on Empirical Methods in Natural Language Processing*, 2014, pp. 1532-1543.

[14] J. Chung, C. Gulcehre, K. H. Cho, and Y. Bengio, "Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling," *arXiv preprint arXiv:1412.3555,* 2014.

[15] Z. Yang, D. Yang, C. Dyer, X. He, A. Smola, and E. Hovy, "Hierarchical attention networks for document classification," in *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2016, pp. 1480-1489.

[16] Ł. Kaiser, O. Nachum, A. Roy, and S. Bengio, "Learning to Remember Rare Events," *arXiv preprint arXiv:1703.03129,* 2017.

[17] A. Strehl and J. Ghosh, "Cluster ensembles---a knowledge reuse framework for combining multiple partitions," *Journal of machine learning research,* vol. 3, no. Dec, pp. 583-617, 2002.

[18] E. Amigó, J. Gonzalo, J. Artiles, and F. Verdejo, "A comparison of extrinsic clustering evaluation metrics based on formal constraints," *Information retrieval,* vol. 12, no. 4, pp. 461-486, 2009.

[19] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347,* 2017.

[20] A. J. Mcminn, Y. Moshfeghi, and J. M. Jose, "Building a large-scale corpus for evaluating event detection on twitter," in *ACM International Conference on Information & Knowledge Management*, 2013, pp. 409-418.

[21] M. Osborne and V. Lavrenko, "Streaming first story detection with application to Twitter," in *Human Language Technologies: the 2010 Conference of the North American Chapter of the Association for Computational Linguistics*, 2010, pp. 181-189.

[22] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *Journal of Machine Learning Research,* vol. 3, pp. 993-1022, 2003.