Open Event Extraction from Online Text using a Generative Adversarial Network

Rui Wang[†] **Deyu Zhou**[†*] **Yulan He**[§]
[†]School of Computer Science and Engineering, Key Laboratory of Computer Network
and Information Integration, Ministry of Education, Southeast University, China
§Department of Computer Science, University of Warwick, UK
{rui_wang, d.zhou}@seu.edu.cn, yulan.he@warwick.ac.uk

Abstract

To extract the structured representations of open-domain events, Bayesian graphical models have made some progress. However, these approaches typically assume that all words in a document are generated from a single event. While this may be true for short text such as tweets, such an assumption does not generally hold for long text such as news articles. Moreover, Bayesian graphical models often rely on Gibbs sampling for parameter inference which may take long time to converge. To address these limitations, we propose an event extraction model based on Generative Adversarial Nets, called Adversarial-neural Event Model (AEM). AEM models an event with a Dirichlet prior and uses a generator network to capture the patterns underlying latent events. A discriminator is used to distinguish documents reconstructed from the latent events and the original documents. A byproduct of the discriminator is that the features generated by the learned discriminator network allow the visualization of the extracted events. Our model has been evaluated on two Twitter datasets and a news article dataset. Experimental results show that our model outperforms the baseline approaches on all the datasets, with more significant improvements observed on the news article dataset where an increase of 15% is observed in F-measure.

1 Introduction

With the increasing popularity of the Internet, online texts provided by social media platform (e.g. Twitter) and news media sites (e.g. Google news) have become important sources of real-world events. Therefore, it is crucial to automatically extract events from online texts.

Due to the high variety of events discussed online and the difficulty in obtaining annotated data for training, traditional template-based or supervised learning approaches for event extraction are no longer applicable in dealing with online texts. Nevertheless, newsworthy events are often discussed by many tweets or online news articles. Therefore, the same event could be mentioned by a high volume of redundant tweets or news articles. This property inspires the research community to devise clustering-based models (Popescu et al., 2011; Abdelhaq et al., 2013; Xia et al., 2015) to discover new or previously unidentified events without extracting structured representations.

To extract structured representations of events such as *who* did *what*, *when*, *where* and *why*, Bayesian approaches have made some progress. Assuming that each document is assigned to a single event, which is modeled as a joint distribution over the named entities, the date and the location of the event, and the event-related keywords, Zhou *et al.* (2014) proposed an unsupervised Latent Event Model (LEM) for open-domain event extraction. To address the limitation that LEM requires the number of events to be pre-set, Zhou *et al.* (2017) further proposed the Dirichlet Process Event Mixture Model (DPEMM) in which the number of events can be learned automatically from data. However, both LEM and DPEMM have two limitations: (1) they assume that all words in a document are generated from a single event which can be represented by a quadruple <entity, location, keyword, date>. However, long texts such news articles often describe multiple events which clearly violates this assumption; (2) During the inference process of both approaches, the Gibbs sampler needs to compute the conditional posterior distribution and assigns an event for each document. This is time consuming and takes long time to converge.

To deal with these limitations, in this paper, we propose the Adversarial-neural Event Model

[*]corresponding author

(AEM) based on adversarial training for open-domain event extraction. The principle idea is to use a generator network to learn the projection function between the document-event distribution and four event related word distributions (entity distribution, location distribution, keyword distribution and date distribution). Instead of providing an analytic approximation, AEM uses a discriminator network to discriminate between the reconstructed documents from latent events and the original input documents. This essentially helps the generator to construct a more realistic document from a random noise drawn from a Dirichlet distribution. Due to the flexibility of neural networks, the generator is capable of learning complicated nonlinear distributions. And the supervision signal provided by the discriminator will help generator to capture the event-related patterns. Furthermore, the discriminator also provides low-dimensional discriminative features which can be used to visualize documents and events.

The main contributions of the paper are summarized below:

- We propose a novel Adversarial-neural Event Model (AEM), which is, to the best of our knowledge, the first attempt of using adversarial training for open-domain event extraction.
- Unlike existing Bayesian graphical modeling approaches, AEM is able to extract events from different text sources (short and long). And a significant improvement on computational efficiency is also observed.
- Experimental results on three datasets show that AEM outperforms the baselines in terms of accuracy, recall and F-measure. In addition, the results show the strength of AEM in visualizing events.

2 Related Work

Our work is related to two lines of research, event extraction and Generative Adversarial Nets.

Event Extraction

Recently there has been much interest in event extraction from online texts, and approaches could be categorized as domain-specific and open-domain event extraction.

Domain-specific event extraction often focuses on the specific types of events (e.g. sports events or city events). Panem *et al.* (2014) devised a novel algorithm to extract attribute-value pairs and mapped them to manually generated schemes for extracting the natural disaster events. Similarly, to extract the city-traffic related event, Anantharam *et al.* (2015) viewed the task as a sequential tagging problem and proposed an approach based on the conditional random fields. Zhang (2018) proposed an event extraction approach based on imitation learning, especially on inverse reinforcement learning.

Open-domain event extraction aims to extract events without limiting the specific types of events. To analyze individual messages and induce a canonical value for each event, Benson *et al.* (2011) proposed an approach based on a structured graphical model. By representing an event with a binary tuple which is constituted by a named entity and a date, Ritter *et al.* (2012) employed some statistic to measure the strength of associations between a named entity and a date. The proposed system relies on a supervised labeler trained on annotated data. In (Abdelhaq et al., 2013), Abdelhaq *et al.* developed a real-time event extraction system called EvenTweet, and each event is represented as a triple constituted by time, location and keywords. To extract more information, Wang *el al.* (2015) developed a system employing the links in tweets and combing tweets with linked articles to identify events. Xia *el al.* (2015) combined texts with the location information to detect the events with low spatial and temporal deviations. Zhou *et al.* (2014; 2017) represented event as a quadruple and proposed two Bayesian models to extract events from tweets.

Generative Adversarial Nets

As a neural-based generative model, Generative Adversarial Nets (Goodfellow et al., 2014) have been extensively researched in natural language processing (NLP) community.

For text generation, the sequence generative adversarial network (SeqGAN) proposed in (Yu et al., 2017) incorporated a policy gradient strategy to optimize the generation process. Based on the policy gradient, Lin *et al.* (2017) proposed RankGAN to capture the rich structures of language by ranking and analyzing a collection of human-written and machine-written sentences. To overcome mode collapse when dealing with discrete data, Fedus *et al.* (2018) pro-

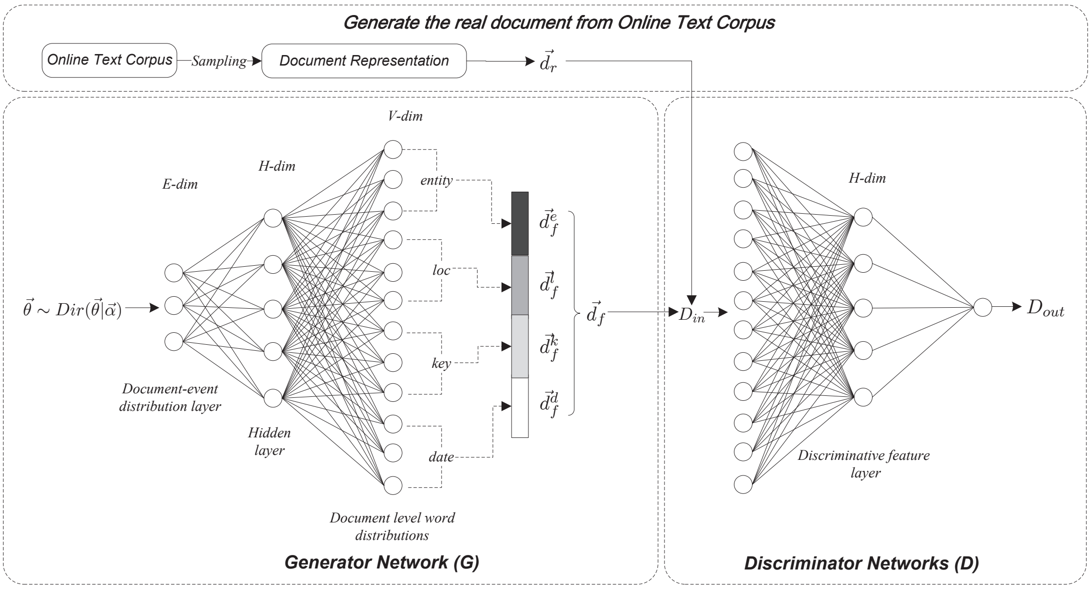
Generate the real document from Online Text Corpus
Online Text Corpus
Sampling
Document Representation
$\vec{d}_r$
V-dim
H-dim
E-dim
entity
loc
key
date
$\vec{d}^e_f$
$\vec{d}^l_f$
$\vec{d}^k_f$
$\vec{d}^d_f$
$\vec{d}_f$
$\vec{\theta} \sim Dir(\vec{\theta}|\vec{\alpha})$
Document-event distribution layer
Hidden layer
Document level word distributions
Generator Network (G)
D_{in}
H-dim
Discriminative feature layer
D_{out}
Discriminator Networks (D)

Figure 1: The framework of the Adverarial-neural Event Model (AEM), and $\vec{d}^e_f$, $\vec{d}^l_f$, $\vec{d}^k_f$ and $\vec{d}^d_f$ denote the generated entity distribution, location distribution, keyword distribution and date distribution corresponding to event distribution $\vec{\theta}$.

posed MaskGAN which used an actor-critic conditional GAN to fill in missing text conditioned on the surrounding context. Along this line, Wang *et al.* (2018) proposed SentiGAN to generate texts of different sentiment labels. Besides, Li et al. (2018) improved the performance of semi-supervised text classification using adversarial training, (Zeng et al., 2018; Qin et al., 2018) designed GAN-based models for distance supervision relation extraction.

Although various GAN based approaches have been explored for many applications, none of these approaches tackles open-domain event extraction from online texts. We propose a novel GAN-based event extraction model called AEM. Compared with the previous models, AEM has the following differences: (1) Unlike most GAN-based text generation approaches, a generator network is employed in AEM to learn the projection function between an event distribution and the event-related word distributions (entity, location, keyword, date). The learned generator captures event-related patterns rather than generating text sequence; (2) Different from LEM and DPEMM, AEM uses a generator network to capture the event-related patterns and is able to mine events from different text sources (short and long). Moreover, unlike traditional inference procedure, such as Gibbs sampling used in LEM and DPEMM, AEM could extract the events more efficiently due to the CUDA acceleration; (3) The discriminative features learned by the discriminator of AEM provide a straightforward way to visualize the extracted events.

3 Methodology

We describe Adversarial-neural Event Model (AEM) in this section. An event is represented as a quadruple $<e, l, k, d>$, where e stands for non-location named entities, l for a location, k for event-related keywords, d for a date, and each component in the tuple is represented by component-specific representative words.

AEM is constituted by three components: (1) *The document representation module*, as shown at the top of Figure 1, defines a document representation approach which converts an input document from the online text corpus into $\vec{d}_r \in \mathbb{R}^V$ which captures the key event elements; (2) *The generator G*, as shown in the lower-left part of Figure1, generates a fake document $\vec{d}_f$ which is constituted by four multinomial distributions using an event distribution $\vec{\theta}$ drawn from a Dirichlet distribution as input; (3) *The discriminator D*, as shown in the lower-right part of Figure1, distinguishes the real documents from the fake ones and its output is subsequently employed as a learning signal to update the G and D. The details of each component are presented below.

3.1 Document Representation

Each document doc in a given corpus C is represented as a concatenation of 4 multinomial distributions which are entity distribution ($\vec{d}_r^e$), location distribution ($\vec{d}_r^l$), keyword distribution ($\vec{d}_r^k$) and date distribution ($\vec{d}_r^d$) of the document. As four distributions are calculated in a similar way, we only describe the computation of the entity distribution below as an example.

The entity distribution $\vec{d}_r^e$ is represented by a normalized V_e-dimensional vector weighted by TF-IDF, and it is calculated as:

$$\begin{aligned} tf_{i,doc}^e &= \frac{n_{i,doc}^e}{\sum_{v_e} n_{v_e,doc}^e} \\ idf_i^e &= \log \frac{|C^e|}{|C_i^e|} \\ tf\text{-}idf_{i,doc}^e &= tf_{i,doc}^e \times idf_i^e \\ d_{r,i}^e &= \frac{tf\text{-}idf_{i,doc}^e}{\sum_{v_e} tf\text{-}idf_{v_e,doc}^e} \end{aligned}$$

where C^e is the pseudo corpus constructed by removing all non-entity words from C, V_e is the total number of distinct entities in a corpus, $n_{i,doc}^e$ denotes the number of i-th entity appeared in document doc, $|C^e|$ represents the number of documents in the corpus, and $|C_i^e|$ is the number of documents that contain i-th entity, and the obtained $d_{r,i}^e$ denotes the relevance between i-th entity and document doc.

Similarly, location distribution $\vec{d}_r^l$, keyword distribution $\vec{d}_r^k$ and date distribution $\vec{d}_r^d$ of doc could be calculated in the same way, and the dimensions of these distributions are denoted as V_l, V_k and V_d, respectively. Finally, each document doc in the corpus is represented by a V-dimensional (V=V_e+V_l+V_k+V_d) vector $\vec{d}_r$ by concatenating four computed distributions.

3.2 Network Architecture

3.2.1 Generator

The generator network G is designed to learn the projection function between the document-event distribution $\vec{\theta}$ and the four document-level word distributions (entity distribution, location distribution, keyword distribution and date distribution).

More concretely, G consists of a E-dimensional document-event distribution layer, H-dimensional hidden layer and V-dimensional event-related word distribution layer. Here, E denotes the event number, H is the number of units in the hidden layer, V is the vocabulary size and equals to V_e+V_l+V_k+V_d. As shown in Figure 1, G firstly employs a random document-event distribution $\vec{\theta}$ as an input. To model the multinomial property of the document-event distribution, $\vec{\theta}$ is drawn from a Dirichlet distribution parameterized with $\vec{\alpha}$ which is formulated as:

$$p(\vec{\theta}|\vec{\alpha}) = Dir(\vec{\theta}|\vec{\alpha}) \triangleq \frac{1}{\triangle(\vec{\alpha})} \prod_{t=1}^{E} \theta_t^{\alpha_t - 1} \quad (1)$$

where $\vec{\alpha}$ is the hyper-parameter of the dirichlet distribution, E is the number of events which should be set in AEM, $\triangle(\vec{\alpha}) = \frac{\prod_{t=1}^{E}\Gamma(\alpha_t)}{\Gamma(\sum_{t=1}^{E}\alpha_t)}$, $\theta_t \in [0,1]$ represents the proportion of event t in the document and $\sum_{t=1}^{E}\theta_t = 1$.

Subsequently, G transforms $\vec{\theta}$ into a H-dimensional hidden space using a linear layer followed by layer normalization, and the transformation is defined as:

$$\vec{s}_h = LN(W_h\vec{\theta} + \vec{b}_h) \quad (2)$$

$$\vec{o}_h = \max(\vec{s}_h, l_p \times \vec{s}_h) \quad (3)$$

where $W_h \in \mathbb{R}^{H\times E}$ represents the weight matrix of hidden layer, and $\vec{b}_h$ denotes the bias term, l_p is the parameter of LeakyReLU activation and is set to 0.1, $\vec{s}_h$ and $\vec{o}_h$ denote the normalized hidden states and the outputs of the hidden layer, and LN represents the layer normalization.

Then, to project $\vec{o}_h$ into four document-level event related word distributions ($\vec{d}_f^e$, $\vec{d}_f^l$, $\vec{d}_f^k$ and $\vec{d}_f^d$ shown in Figure 1), four subnets (each contains a linear layer, a batch normalization layer and a softmax layer) are employed in G. And the exact transformation is based on the formulas below:

$$\vec{h}_w^e = W_w^e\vec{o}_h + \vec{b}_w^e,\ \vec{d}_f^e = SM(BN(\vec{h}_w^e)) \quad (4)$$

$$\vec{h}_w^l = W_w^l\vec{o}_h + \vec{b}_w^l,\ \vec{d}_f^l = SM(BN(\vec{h}_w^l)) \quad (5)$$

$$\vec{h}_w^k = W_w^k\vec{o}_h + \vec{b}_w^k,\ \vec{d}_f^k = SM(BN(\vec{h}_w^k)) \quad (6)$$

$$\vec{h}_w^d = W_w^d\vec{o}_h + \vec{b}_w^d,\ \vec{d}_f^d = SM(BN(\vec{h}_w^d)) \quad (7)$$

where SM means softmax layer, $W_w^e \in \mathbb{R}^{V_e\times H}$, $W_w^l \in \mathbb{R}^{V_l\times H}$, $W_w^k \in \mathbb{R}^{V_k\times H}$ and $W_w^d \in \mathbb{R}^{V_d\times H}$ denote the weight matrices of the linear layers in subnets, $\vec{b}_w^e$, $\vec{b}_w^l$, $\vec{b}_w^k$ and $\vec{b}_w^d$ represent the corresponding bias terms, $\vec{h}_w^e$, $\vec{h}_w^l$, $\vec{h}_w^k$ and $\vec{h}_w^d$ are state vectors. $\vec{d}_f^e$, $\vec{d}_f^l$, $\vec{d}_f^k$ and $\vec{d}_f^d$ denote the generated entity distribution, location distribution, keyword distribution and date distribution, respectively, that correspond to the given event distribution $\vec{\theta}$. And each dimension represents the relevance between corresponding entity/location/keyword/date term and the input event distribution.

Finally, four generated distributions are concatenated to represent the generated document $\vec{d_f}$ corresponding to the input $\vec{\theta}$:

$$\vec{d_f} = [\vec{d^e_f}; \vec{d^l_f}; \vec{d^k_f}; \vec{d^d_f}] \tag{8}$$

3.2.2 Discriminator

The discriminator network D is designed as a fully-connected network which contains an input layer, a discriminative feature layer (discriminative features are employed for event visualization) and an output layer. In AEM, D uses fake document $\vec{d_f}$ and real document $\vec{d_r}$ as input and outputs the signal D_{out} to indicate the source of the input data (lower value denotes that D is prone to predict the input data as a fake document and vice versa).

As have previously been discussed in (Arjovsky et al., 2017; Gulrajani et al., 2017), lipschitz continuity of D network is crucial to the training of the GAN-based approaches. To ensure the lipschitz continuity of D, we employ the spectral normalization technique (Miyato et al., 2018). More concretely, for each linear layer $l_d(\vec{h}) = W\vec{h}$ (bias term is omitted for simplicity) in D, the weight matrix W is normalized by $\sigma(W)$. Here, $\sigma(W)$ is the spectral norm of the weight matrix W with the definition below:

$$\sigma(W) := \max_{\vec{h}:\vec{h}\neq\vec{0}} \frac{\|W\vec{h}\|_2}{\|\vec{h}\|_2} = \max_{\|\vec{h}\|_2\leq 1} \|W\vec{h}\|_2 \tag{9}$$

which is equivalent to the largest singular value of W. The weight matrix W is then normalized using:

$$\hat{W_{SN}} := W/\sigma(W) \tag{10}$$

Obviously, the normalized weight matrix $\hat{W_{SN}}$ satisfies that $\sigma(\hat{W_{SN}}) = 1$ and further ensures the lipschitz continuity of the D network (Miyato et al., 2018). To reduce the high cost of computing spectral norm $\sigma(W)$ using singular value decomposition at each iteration, we follow (Yoshida and Miyato, 2017) and employ the power iteration method to estimate $\sigma(W)$ instead. With this substitution, the spectral norm can be estimated with very small additional computational time.

3.3 Objective and Training Procedure

The real document $\vec{d_r}$ and fake document $\vec{d_f}$ shown in Figure 1 could be viewed as random samples from two distributions $\mathbb{P}_r$ and $\mathbb{P}_g$, and each of them is a joint distribution constituted by four Dirichlet distributions (corresponding to entity distribution, location distribution, keyword distribution and date distribution). The training objective of AEM is to let the distribution $\mathbb{P}_g$ (produced by G network) to approximate the real data distribution $\mathbb{P}_r$ as much as possible.

To compare the different GAN losses, Kurach (2018) takes a sober view of the current state of GAN and suggests that the Jansen-Shannon divergence used in (Goodfellow et al., 2014) performs more stable than variant objectives. Besides, Kurach also advocates that the gradient penalty (GP) regularization devised in (Gulrajani et al., 2017) will further improve the stability of the model. Thus, the objective function of the proposed AEM is defined as:

$$L_d = -\mathop{\mathbb{E}}_{\vec{d_r}\sim\mathbb{P}_r}[\log(D(\vec{d_r}))] - \mathop{\mathbb{E}}_{\vec{d_f}\sim\mathbb{P}_g}[\log(1-D(\vec{d_f}))] \tag{11}$$

$$L_{gp} = \mathop{\mathbb{E}}_{\vec{d^*}\sim\mathbb{P}_{d^*}}[(\| \nabla_{\vec{d^*}} D(\vec{d^*}) \|_2 - 1)^2] \tag{12}$$

$$L = L_d + \lambda L_{gp} \tag{13}$$

where L_d denotes the discriminator loss, L_{gp} represents the gradient penalty regularization loss, λ is the gradient penalty coefficient which trade-off the two components of objective, $\vec{d^*}$ could be obtained by sampling uniformly along a straight line between $\vec{d_r}$ and $\vec{d_f}$, $\mathbb{P}_{d^*}$ denotes the corresponding distribution.

The training procedure of AEM is presented in Algorithm 1, where E is the event number, n_d denotes the number of discriminator iterations per generator iteration, m is the batch size, α' represents the learning rate, β_1 and β_2 are hyper-parameters of Adam (Kingma and Ba, 2014), p_a denotes $\{\alpha', \beta_1, \beta_2\}$. In this paper, we set $\lambda = 10$, $n_d = 5$, $m = 32$. Moreover, α', β_1 and β_2 are set as 0.0002, 0.5 and 0.999.

3.4 Event Generation

After the model training, the generator G learns the mapping function between the document-event distribution and the document-level event-related word distributions (entity, location, keyword and date). In other words, with an event distribution

Algorithm 1 Training procedure for AEM

```
Input: $E$, $\lambda$, $n_d$, $m$, $\alpha'$, $\beta_1$, $\beta_2$
Output: the trained $G$ and $D$.
Initial $D$ parameters $\omega_d$ and $G$ parameter $\omega_g$
while $\omega_g$ has not converged do
    for $t = 1, ..., n_d$ do
        for $j = 1, ..., m$ do
            Sample $\vec{d_r} \sim \mathbb{P}_r$,
            Sample a random $\vec{\theta} \sim Dir(\vec{\theta}|\vec{\alpha})$
            Sample a random number $\epsilon \sim U[0, 1]$
            $\vec{d_f} \leftarrow G(\vec{\theta})$
            $\vec{d^*} \leftarrow \epsilon\vec{d_r} + (1 - \epsilon)\vec{d_f}$
            $L_d^{(j)} = -\log[D(\vec{d_r})] - \log[1 - D(\vec{d_f})]$
            $L_{gp}^{(j)} = (\| \nabla_{\vec{d^*}} D(\vec{d^*}) \| - 1)^2$
            $L^{(j)} \leftarrow L_d^{(j)} + \lambda L_{gp}^{(j)}$
        end for
        $\omega_d \leftarrow Adam(\nabla_{\omega_d} \frac{1}{m} \sum_{j=1}^{m} L^{(j)}, \omega_d, p_a)$
    end for
    Sample $m$ noise $\left\{ \vec{\theta}^{(j)} \sim Dir(\vec{\theta}|\vec{\alpha}) \right\}$
    $\omega_g \leftarrow Adam(\nabla_{\omega_g} \frac{1}{m} \sum_{j=1}^{m} \log[1 - D(G(\vec{\theta}^{(j)}))], \omega_g, p_a)$
end while
```

$\vec{\theta}'$ as input, G could generate the corresponding entity distribution, location distribution, keyword distribution and date distribution.

In AEM, we employ event seed $\vec{s}_{t\in\{1,...,E\}}$, an E-dimensional vector with one-hot encoding, to generate the event related word distributions. For example, in ten event setting, $\vec{s}_1 = [1, 0, 0, 0, 0, 0, 0, 0, 0, 0]^\intercal$ represents the event seed of the first event. With the event seed $\vec{s}_1$ as input, the corresponding distributions could be generated by G based on the equation below:

$$[\vec{\phi}_e^1; \vec{\phi}_l^1; \vec{\phi}_k^1; \vec{\phi}_d^1] = G(\vec{s}_1) \tag{14}$$

where $\vec{\phi}_e^1$, $\vec{\phi}_l^1$, $\vec{\phi}_k^1$ and $\vec{\phi}_d^1$ denote the entity distribution, location distribution, keyword distribution and date distribution of the first event respectively.

4 Experiments

In this section, we firstly describe the datasets and baseline approaches used in our experiments and then present the experimental results.

4.1 Experimental Setup

To validate the effectiveness of AEM for extracting events from social media (e.g. Twitter) and news media sites (e.g. Google news), three datasets (FSD (Petrovic et al., 2013), Twitter, and Google datasets[1]) are employed. Details are summarized below:

- *FSD dataset* (social media) is the first story detection dataset containing 2,499 tweets. We filter out events mentioned in less than 15 tweets since events mentioned in very few tweets are less likely to be significant. The final dataset contains 2,453 tweets annotated with 20 events.

- *Twitter dataset* (social media) is collected from tweets published in the month of December in 2010 using Twitter streaming API. It contains 1,000 tweets annotated with 20 events.

- *Google dataset* (news article) is a subset of GDELT Event Database[1], documents are retrieved by event related words. For example, documents which contain 'malaysia', 'airline', 'search' and 'plane' are retrieved for event *MH370*. By combining 30 events related documents, the dataset contains 11,909 news articles.

We choose the following three models as the baselines:

- **K-means** is a well known data clustering algorithm, we implement the algorithm using sklearn[2] toolbox, and represent documents using bag-of-words weighted by TF-IDF.

- **LEM** (Zhou et al., 2014) is a Bayesian modeling approach for open-domain event extraction. It treats an event as a latent variable and models the generation of an event as a joint distribution of its individual event elements. We implement the algorithm

[1] http://data.gdeltproject.org/events/index.html

[2] https://scikit-learn.org/

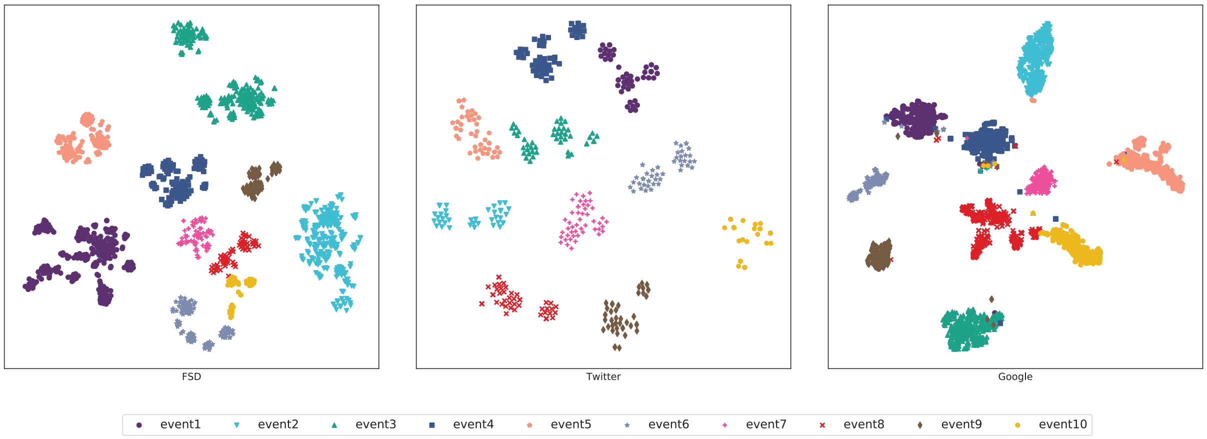

Figure 2: Visualization of the ten randomly selected events on each dataset. Each point denotes a document. Different color denotes different events.

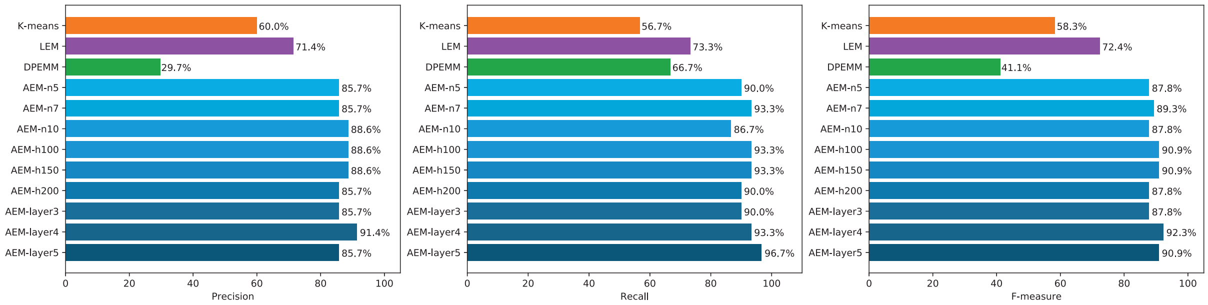

Figure 3: Comparison of methods and parameter settings,'n' and 'h' denote parameter n_d and H, other parameters follow the default setting. The vertical axis represents methods/parameter settings, the horizontal axis denotes the corresponding performance value. All blue histograms with different intensity are those obtained by AEM.

with the default configuration.

- **DPEMM** (Zhou et al., 2017) is a non-parametric mixture model for event extraction. It addresses the limitation of LEM that the number of events should be known beforehand. We implement the model with the default configuration.

For social media text corpus (FSD and Twitter), a named entity tagger[3] specifically built for Twitter is used to extract named entities including locations from tweets. A Twitter Part-of-Speech (POS) tagger (Gimpel et al., 2010) is used for POS tagging and only words tagged with nouns, verbs and adjectives are retained as keywords. For the Google dataset, we use the Stanford Named Entity Recognizer[4] to identify the named entities (organization, location and person). Due to the 'date' information not being provided in the Google dataset, we further divide the non-location named entities into two categories ('person' and 'organization') and employ a quadruple <organization, location, person, keyword> to denote an event in news articles. We also remove common stopwords and only keep the recognized named entities and the tokens which are verbs, nouns or adjectives.

4.2 Experimental Results

To evaluate the performance of the proposed approach, we use the evaluation metrics such as precision, recall and F-measure. Precision is defined as the proportion of the correctly identified events out of the model generated events. Recall is defined as the proportion of correctly identified true events. For calculating the precision of the 4-tuple, we use following criteria:

- (1) Do the entity/organization, location, date/person and keyword that we have extracted refer to the same event?

- (2) If the extracted representation contains

[3] http://fithub.com/aritter/twitter-nlp
[4] https://nlp.stanford.edu/software/CRF-NER.html

keywords, are they informative enough to tell us what happened?

Dataset	Method	Precision (%)	Recall (%)	F-measure (%)
FSD	K-means	84.0	55.0	66.5
	LEM	80.0	80.0	80.0
	DPEMM	84.6	85.0	84.8
	AEM	**88.0**	**85.0**	**86.5**
Twitter	K-means	68.0	75.0	71.3
	LEM	68.0	80.0	73.5
	DPEMM	69.2	80.0	74.2
	AEM	**72.0**	**85.0**	**77.9**
Google	K-Means	60.0	56.7	58.3
	LEM	71.4	73.3	72.4
	DPEMM	29.7	66.7	41.3
	AEM	**85.7**	**90.0**	**87.8**

Table 1: Comparison of the performance of event extraction on the three datasets.

Table 1 shows the event extraction results on the three datasets. The statistics are obtained with the default parameter setting that n_d is set to 5, number of hidden units H is set to 200, and G contains three fully-connected layers. The event number E for three datasets are set to 25, 25 and 35, respectively. The examples of extracted events are shown in Table.2.

It can be observed that K-means performs the worst over all three datasets. On the social media datasets, AEM outpoerforms both LEM and DPEMM by 6.5% and 1.7% respectively in F-measure on the FSD dataset, and 4.4% and 3.7% in F-measure on the Twitter dataset. We can also observe that apart from K-means, all the approaches perform worse on the Twitter dataset compared to FSD, possibly due to the limited size of the Twitter dataset. Moreover, on the Google dataset, the proposed AEM performs significantly better than LEM and DPEMM. It improves upon LEM by 15.5% and upon DPEMM by more than 30% in F-measure. This is because: (1) the assumption made by LEM and DPEMM that all words in a document are generated from a single event is not suitable for long text such as news articles; (2) DPEMM generates too many irrelevant events which leads to a very low precision score. Overall, we see the superior performance of AEM across all datasets, with more significant improvement on the for Google datasets (long text).

We next visualize the detected events based on the discriminative features learned by the trained D network in AEM. The t-SNE (Maaten and Hinton, 2008) visualization results on the datasets are shown in Figure 2. For clarity, each subplot is plotted on a subset of the dataset containing ten randomly selected events. It can be observed that documents describing the same event have been grouped into the same cluster.

To further evaluate if a variation of the parameters n_d (the number of discriminator iterations per generator iteration), H (the number of units in hidden layer) and the structure of generator G will impact the extraction performance, additional experiments have been conducted on the Google dataset, with n_d set to 5, 7 and 10, H set to 100, 150 and 200, and three G structures (3, 4 and 5 layers). The comparison results on precision, recall and F-measure are shown in Figure 3. From the results, it could be observed that AEM with the 5-layer generator performs the best and achieves 96.7% in F-measure, and the worst F-measure obtained by AEM is 85.7%. Overall, the AEM outperforms all compared approaches acorss various parameter settings, showing relatively stable performance.

Finally, we compare in Figure 4 the training time required for each model, excluding the constant time required by each model to load the data. We could observe that K-means runs fastest among all four approaches. Both LEM and DPEMM need to sample the event allocation for each document and update the relevant counts during Gibbs sampling which are time consuming. AEM only requires a fraction of the training time compared to LEM and DPEMM. Moreover, on a larger dataset such as the Google dataset, AEM appears to be far more efficient compared to LEM and DPEMM.

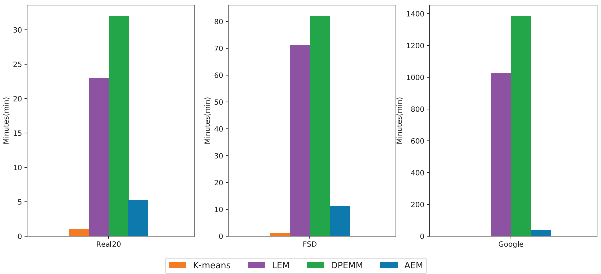

Figure 4: Comparison of training time of models.

5 Conclusions and Future Work

In this paper, we have proposed a novel approach based on adversarial training to extract the structured representation of events from online text. The experimental comparison with the state-of-the-art methods shows that AEM achieves improved extraction performance, especially on

FSD dataset			
Earthquake in Viriginia	e: nbc coast tremor east eastern l: virginia russian eal croydon washington k: earthquak feel center magnitud hit d: 2011/8/23 2011/7/23 2011/8/06 2011/9/07 2011/9/12	US debt ceiling	e: hous gifford us gabriell repres l: virginia russian eal croydon washington k: debt bill hous ceil vote d: 2011/8/01 2011/7/23 2011/8/23 2011/8/06 2011/9/13
South sudan independent	e: south sudan independ earthquak tremor l: earth senat congress york nyc k: independ celebr countri congrat challeng d: 2011/7/09 2011/8/06 2011/8/23 2011/7/23 2011/9/07	US credit downgrade	e: aaa aa yahoo standard obama l: state america tottenham congress seattl k: credit rate downgrad histori lose d: 2011/8/06 2011/7/23 2011/8/23 2011/9/07 2011/9/12
Somalia declare famine	e: somalia africa bakool southern nation l: somalia africa rome independ southern k: declar famin drought part region d: 2011/7/20 2011/7/23 2011/8/06 2011/8/23 2011/9/07	Norway youth camp attack	e: eyewit norway norweigan rock us l: norway island germani state libya k: camp attack youth bomb shoot d: 2011/7/22 2011/7/23 2011/8/23 2011/8/06 2011/8/10
Twitter dataset			
Russia hosts world cup	e: world cup william russia sport l: qatar russia china europ beij k: host cup reaction world triumph d: 2010/9/3 2010/9/10 2010/9/9 2010/9/8 2010/9/17	Larry King's last show	e: king larri cnn red vega l: uk state richardson unit south k: final show broadcast night year d: 2010/9/17 2010/9/10 2010/9/8 2010/9/9 2010/9/26
Coach Urban Meyer step down	e: meyer urban reid florida gator l: florida univers senat europ hous k: coach step univers footbal accord d: 2010/9/8 2010/9/10 2010/9/9 2010/9/18 2010/9/17	Boxer floyd Maweath is arrested	e: boxer floyd mayweath vega obama l: vega las beij europ itali k: guard boxer secur assault arrest d: 2010/9/17 2010/9/9 2010/9/18 2010/9/8 2010/9/26
Christian violence in nigeria	e: christian muslim polit concord eve l: nigeria jos congress christian of k: religion church violenc plagu peopl d: 2010/9/25 2010/9/28 2010/9/26 2010/9/6 2010/9/8	Xiaobo Liu award nobel prize	e: xiaobo liu nobel prize china l: china oslo congress continent europ k: award live nobel ceremoni dissid d: 2010/9/10 2010/9/8 2010/9/17 2010/9/9 2010/9/18
Google dataset			
Sexual assault in campus	o: university federal college department white l: obama princeton ohio columbia harvard p: mccaskill rose catherine brown duncan k: sexual assault campus title colleges	Lockett is executed death penalty in Oklahoma	o: warner state department cohen robert l: lockett oklahoma states texas ohio p: lockett clayton patton stephanie charles k: execution death penalty lethal minutes
Apple & Samsung patent jury	o: apple samsung google inc motorola l: california south santa us calif p: judge steve dunham schmidt mueller k: patent jury smartphone verdict trial	MH370	o: airlines air transport boeing najib l: malaysia australia beijing malacca houston p: najib hishammuddin hussein clark dolan k: search plane flight aircraft ocean
Afghanistan landslide	o: afghanistan united taliban kabul un l: afghanistan badakhshan kabul tajikistan pakistan p: karzai shah hill mark angela k: landslide village rescue mud province	South Africa election	o: anc national mandela congress eff l: zuma africa south africans nkandla p: zuma jacob president nelson malema k: election apartheid elections voters economic

Table 2: The event examples extracted by AEM.

long text corpora with an improvement of 15% observed in F-measure. AEM only requires a fraction of training time compared to existing Bayesian graphical modeling approaches. In future work, we will explore incorporating external knowledge (e.g. word relatedness contained in word embeddings) into the learning framework for event extraction. Besides, exploring nonparametric neural event extraction approaches and detecting the evolution of events over time from news articles are other promising future directions.

6 Acknowledgments

We would like to thank anonymous reviewers for their valuable comments and helpful suggestions. This work was funded by the National Key Research and Development Program of China (2017YFB1002801), the National Natural Science Foundation of China (61772132), the Natural Science Foundation of Jiangsu Province of China (BK20161430) and Innovate UK (grant no. 103652).

References

Hamed Abdelhaq, Christian Sengstock, and Michael Gertz. 2013. Eventweet: Online localized event detection from twitter. *Proceedings of the VLDB Endowment*, 6(12):1326–1329.

Pramod Anantharam, Payam Barnaghi, Krishnaprasad Thirunarayan, and Amit Sheth. 2015. Extracting city traffic events from social streams. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 6(4):43.

Martin Arjovsky, Soumith Chintala, and Léon Bottou. 2017. Wasserstein gan. *arXiv preprint arXiv:1701.07875*.

Edward Benson, Aria Haghighi, and Regina Barzilay. 2011. Event discovery in social media feeds. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 389–398. Association for Computational Linguistics.

William Fedus, Ian Goodfellow, and Andrew M Dai. 2018. Maskgan: Better text generation via filling in the _. *arXiv preprint arXiv:1801.07736*.

Kevin Gimpel, Nathan Schneider, Brendan O'Connor, Dipanjan Das, Daniel Mills, Jacob Eisenstein, Michael Heilman, Dani Yogatama, Jeffrey Flanigan, and Noah A Smith. 2010. Part-of-speech tagging for twitter: Annotation, features, and experiments. Technical report, Carnegie-Mellon Univ Pittsburgh Pa School of Computer Science.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680.

Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. 2017. Improved training of wasserstein gans. In *Advances in Neural Information Processing Systems*, pages 5769–5779.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Karol Kurach, Mario Lucic, Xiaohua Zhai, Marcin Michalski, and Sylvain Gelly. 2018. The gan landscape: Losses, architectures, regularization, and normalization. *arXiv preprint arXiv:1807.04720*.

Yan Li and Jieping Ye. 2018. Learning adversarial networks for semi-supervised text classification via policy gradient. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1715–1723. ACM.

Kevin Lin, Dianqi Li, Xiaodong He, Zhengyou Zhang, and Ming-Ting Sun. 2017. Adversarial ranking for language generation. In *Advances in Neural Information Processing Systems*, pages 3155–3165.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605.

Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. 2018. Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957*.

Sandeep Panem, Manish Gupta, and Vasudeva Varma. 2014. Structured information extraction from natural disaster events on twitter. In *Proceedings of the 5th International Workshop on Web-scale Knowledge Representation Retrieval & Reasoning*, pages 1–8. ACM.

Sasa Petrovic, Miles Osborne, Richard McCreadie, Craig Macdonald, Iadh Ounis, and Luke Shrimpton. 2013. Can twitter replace newswire for breaking news? In *Seventh international AAAI conference on weblogs and social media*.

Ana-Maria Popescu, Marco Pennacchiotti, and Deepa Paranjpe. 2011. Extracting events and event descriptions from twitter. In *Proceedings of the 20th international conference companion on World wide web*, pages 105–106. ACM.

Pengda Qin, Weiran Xu, and William Yang Wang. 2018. Dsgan: Generative adversarial training for distant supervision relation extraction. *arXiv preprint arXiv:1805.09929*.

Alan Ritter, Oren Etzioni, Sam Clark, et al. 2012. Open domain event extraction from twitter. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1104–1112. ACM.

Ke Wang and Xiaojun Wan. 2018. Sentigan: Generating sentimental texts via mixture adversarial networks. In *IJCAI*, pages 4446–4452.

Yu Wang, David Fink, and Eugene Agichtein. 2015. Seeft: Planned social event discovery and attribute extraction by fusing twitter and web content. In *ICWSM*, pages 483–492.

Chaolun Xia, Jun Hu, Yan Zhu, and Mor Naaman. 2015. What is new in our city? a framework for event extraction using social media posts. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 16–32. Springer.

Yuichi Yoshida and Takeru Miyato. 2017. Spectral norm regularization for improving the generalizability of deep learning. *arXiv preprint arXiv:1705.10941*.

Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. 2017. Seqgan: Sequence generative adversarial nets with policy gradient. In *AAAI*, pages 2852–2858.

Daojian Zeng, Yuan Dai, Feng Li, R Simon Sherratt, and Jin Wang. 2018. Adversarial learning for distant supervised relation extraction. *Computers, Materials & Continua*, 55(1):121–136.

Tongtao Zhang and Heng Ji. 2018. Event extraction with generative adversarial imitation learning. *arXiv preprint arXiv:1804.07881*.

Deyu Zhou, Liangyu Chen, and Yulan He. 2014. A simple bayesian modelling approach to event extraction from twitter. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 700–705.

Deyu Zhou, Xuan Zhang, and Yulan He. 2017. Event extraction from twitter using non-parametric bayesian mixture model with word embeddings. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, volume 1, pages 808–817.