A Neural Model for Joint Event Detection and Summarization*

Zhongqing Wang[†,‡] and **Yue Zhang**[‡]
[†]Soochow University, Suzhou, China
[‡]Singapore University of Technology and Design, Singapore
wangzq.antony@gmail.com, yue_zhang@sutd.edu.sg

Abstract

Twitter new event detection aims to identify first stories in a tweet stream. Typical approaches consider two sub tasks. First, it is necessary to filter out mundane or irrelevant tweets. Second, tweets are grouped automatically into event clusters. Traditionally, these two sub tasks are processed separately, and integrated under a pipeline setting, despite that there is inter-dependence between the two tasks. In addition, one further related task is summarization, which is to extract a succinct summary for representing a large group of tweets. Summarization is related to detection, under the new event setting in that salient information is universal between event representing tweets and informative event summaries. In this paper, we build a joint model to filter, cluster, and summarize the tweets for new events. In particular, deep representation learning is used to vectorize tweets, which serves as basis that connects tasks. A neural stacking model is used for integrating a pipeline of different sub tasks, and for better sharing between the predecessor and successors. Experiments show that our proposed neural joint model is more effective compared to its pipeline baseline.

1 Introduction

It has been demonstrated that Twitter reacts to news events faster compared with traditional media [Petrovic *et al.*, 2010], and new event detection on Twitter has received extensive research attention over the past few years [Wurzer *et al.*, 2015]. Work has been carried out both in the open-domain [Ritter *et al.*, 2012] and target entities [Chen *et al.*, 2013]. Compared to the former, the latter can be more accurate by leveraging information that is specific to the target of concern. Our focus of this paper is the detection of certain types of events from Twitter, such as earthquakes and DDoS attacks. We investigate a neural network model that monitors the tweet stream for a certain event category, jointly detecting and summarizing new events under the category.

*This work has been done when the first author worked at SUTD.

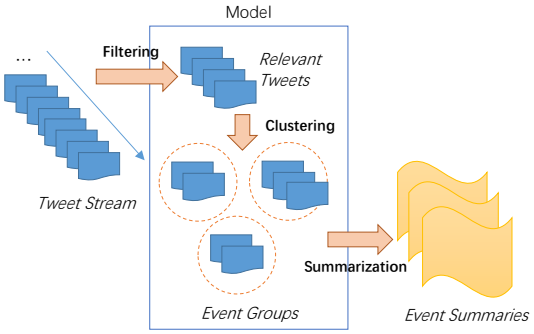

Figure 1: System Architecture.

The architecture of the system is shown in Figure 1. Given a tweets stream, our model considers three sub-tasks: tweet filtering (event mentioned tweets detection), event clustering, and event summarization. A central component of typical Twitter event detection models is *clustering* [Aggarwal and Subbian, 2012; Li *et al.*, 2012], with variations such as incremental clustering [Allan *et al.*, 1998] and locality sensitive hashing [Petrovic *et al.*, 2010]. The key idea is to group tweets of the same topic, so that a new topic can be detected if new tweets are found not belonging to existing topic clusters. Such clustering algorithms typically rely on features based on the tweet content, such as TF/IDF, for measuring the similarity between tweets [Allan, 2002].

The second sub-task is *summarization*, which is not directly involved in the detection objective, but nevertheless highly relevant, because the event clusters that are detected can be large and contain tweets of various degrees of informativeness. With regard to functionality, summarisation is often necessary for tweet event detection systems since tweet clusters can contain much noise, and feature clusters are not directly readable. We consider extractive summarization, returning event summaries as the final output.

Further, because a large part of the tweet stream contains mundane or irrelevant information [Ritter *et al.*, 2015], tweet *filtering* is a third sub-task we consider. The goal is to classify incoming tweets according to their relevance to potential new events, so that only informative tweets are kept. Filtering has been performed either before [Chen *et al.*, 2013] or after tweet clustering [Aggarwal and Subbian, 2012]. We consider filtering as a task before clustering in this paper.

The three sub-tasks form a three-stage (filtering → clustering → summarization) pipeline, in which the stages are intimately correlated. For example, a tweet that comprehensively describes an event should be scored highly in both the relevance-filtering and the extractive-summarization steps. In addition, better understanding of a tweet is helpful for both relevance-filtering and event-clustering. Inspired by this, we consider joint filtering, clustering, and summarization using a single model. Unlike traditional joint tasks in syntax, the sub tasks in our model are semantically related, which poses challenges for defining feature templates in *discrete* statistical joint models [Zhang and Clark, 2008; Finkel and Manning, 2010]. On the other hand, *neural* network models are free from manual feature engineering [Rush *et al.*, 2015; Chang *et al.*, 2016]. Hence, they are a natural choice under our setting.

Neural network models have been shown effective for multi-task learning [Collobert *et al.*, 2011] and semantic representation learning via parameter sharing [Katiyar and Cardie, 2016]. Correspondingly, we use two strategies for joint modeling. First, we take the semantic representation of tweets as a key connection factor, integrating the three tasks via *parameter sharing*. Second, we apply *neural stacking* [Chen *et al.*, 2016] to the pipeline, feeding the hidden neural layers of the predecessor sub models as additional input features to their successor sub models, and propagating the errors of successors to predecessors during training, so that information is better shared between the predecessor and successor.

Experiments show that integration of the three sub-tasks improves the performance of every stage in the pipeline. The neural network model is significantly more effective for new event detection and event summarization compared with state-of-the-art discrete and neural pipelined models. To our knowledge, we are the first to investigate neural network for joint new event detection and summarization. We release the code and data sets of this paper at `https://github.com/wangzq870305/joint_event_detection`.

2 Related Work

2.1 Event Detection and Summarization

Filtering. To filter mundane and irrelevant information from the tweet stream, Chen *et al.* [2013] proposed a classification model to extract noisy tweets before event detection, while Aggarwal and Subbian [2012] filtered noisy tweets after the event detection process. Ritter *et al.* [2015] filtered irrelevant tweets using minimum supervision by leveraging expectation regularization, and Chang *et al.* [2016] extended this method by further reducing supervision and using representation learning. Our work is related to the use of clustering for event detection, and the filtering of tweets using a neural classifier.

Detection. The task of event detection (i.e. event clustering) has been proposed in the Topic Detection and Tracking (TDT) program [Allan, 2002], the objective of which is to discover new or previously unidentified events, where each event refers to a specific happening at a specific time and place [Allan *et al.*, 1998] raised on news. Different from traditional TDT tasks [Allan, 2002], Twitter posts reflect events quickly as they unfold and hence tweets are particularly useful for first-hand news event detection.

Online clustering based approaches are popular on detecting open-domain events. For example, Aggarwal and Subbian [2012] proposed a stream-based clustering algorithm on each incoming posts. Li *et al.* [2012] used segments in tweets rather than full tweets as the basic unit of clustering. Petrovic *et al.* [2010] and Wurzer *et al.* [2015] used a Locality Sensitive Hashing (LSH) to detect and cluster events from high-volume tweet streams in constant time and space.

Summarization. On event summarization, Nichols *et al.* [2012] proposed a PageRank-based model to extract event summaries using status updates in the sports domain. Sun *et al.* [2015] proposed discrete event-driven models for headline generation. Our work touches on several strands of research within neural summarization. Rush *et al.* [2015] proposed a neural attention model for *abstractive* sentence compression, which is trained on pairs of headlines and first sentences of an article. Cheng and Lapata [2016] developed a general LSTM based framework for *extractive* single-document summarization, which uses a hierarchical document encoder and an attention-based extractor.

Different from the previous research on new event detection and summarization, we present a neural network model for integrating event filtering, clustering, and summarization jointly. To our knowledge, this is the first work to employ neural networks for collaborative event detection and summarization.

2.2 Neural Joint Modeling

There has been a line of research using discrete models to solve related NLP tasks jointly. Work has been done on joint segmentation and POS-tagging [Zhang and Clark, 2008], joint parsing and name entity recognition [Finkel and Manning, 2010], joint syntactic and semantic parsing [Li *et al.*, 2010], relation extraction [Li *et al.*, 2016], and joint syntactic and morphological synthesis [Song *et al.*, 2014]. A challenge in such joint models is the designing of manual feature templates, which capture mutual information between the integrated tasks. For our problem, the sub tasks are semantically related, which makes it rather difficult to extract non-local features that capture shared information using manual templates. In contrast, neural network models have been shown effective for inducing sentence-level sematic features automatically [Socher *et al.*, 2013], which makes them a useful tool for our joint model.

Two main approaches have been employed for neural multi-task learning. One method is to learn shared parameters between different tasks. For example, Collobert *et al.* [2011] share a hidden layer between multiple tagging tasks; Katiyar and Cardie [2016] used a shared LSTM layer to extract opinion entities and relations between entities jointly. The other method is based on stacking, feeding the hidden layer of a predecessor sub model as additional input features to its successor sub model, so that information is shared between the predecessor and successor. For example, Chen *et al.* [2016] proposed a neural stacking model for learning multiple tag sequences; Nguyen *et al.* [2016] extracted event mention, trig-

ger, and arguments from texts collectively based on LSTM stacking. Different from previous work, we investigate joint neural network for a novel task, namely new event detection and summarization, considering both shared representation and stacking learning.

3 Joint Event Detection and Summarization

Formally, the input of our system is a tweet stream, and the outputs are real-time event reports. The three main sub-tasks are defined as follows:

- *Event mention detection (Filtering)*: we classify each tweet in the stream as either being relevant or irrelevant to the events of concern. Since our goal is to detect only certain types of events (i.e., earthquake and DDOS attack events), we use a corresponding set of keywords to filter the tweet stream as a preprocessing step. Relevant classification is performed after the preprocessing step, since not all tweets that contain a keyword are relevant. For example, it has been shown that a large number of tweets that contain the word 'ddos' are in fact irrelevant to DDoS attack events, and there has been work investigating the a research question how to find out real event mentions from such tweet stream [Ritter *et al.*, 2015].
- *Event clustering*: we perform incremental clustering of tweets after event mention detection. Given an event mentioning tweet, the task is to decide whether it belongs to an existing event cluster, or describes a new event [Aggarwal and Subbian, 2012]. A key issue in this sub-task is the calculation of similarity between tweets.
- *Event summarization*: when an event cluster is sufficiently large, we create a report of the corresponding event by extracting the top-n most informative tweets [Nichols *et al.*, 2012]. The sub-task can be viewed as an extractive multi-document summarization task [Radev *et al.*, 2004].

The overall architecture of our model is shown in Figure 2, where H is the share representation of a tweet, H_d is the hidden state of event filtering, H_c is the hidden state of event clustering, and P_s is the output of event summarization. A deep neural network is used to model the three sub-tasks jointly. As mentioned in the introduction, two strategies are adopted to integrate the detection, clustering, and summarization models. First, a *representation learning* component is used to transform each incoming tweet into a dense low dimension vector H, which captures salient syntactic and semantic features of the tweet. The representation learning component is connected to all sub-task models as input and therefore acts as the shared parameters that are jointly trained across tasks.

On top of the joint vector representation, the detection, clustering, and summarization models forms a three-stage pipeline. Each sub model consists of several hidden layers that induce neural features from a vector input, with the last hidden layer being fed into a final output layer. For integration between the subtasks, we apply neural stacking [Chen *et al.*, 2016], feeding the last hidden layer of the predecessor sub models as additional input features to its successor sub models, so that information is better shared between the predecessor and successors, and back-propagation can be conducted across sub models. Results show that neural stacking yields better sub models compared with an independent pipeline. Note that, although the process of calculating similarity between two tweets is supervised with LSTM model, the clustering algorithm is an unsupervised online clustering algorithm.

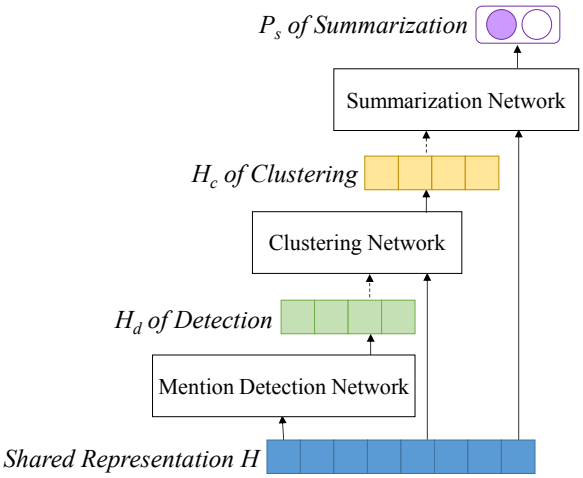

Figure 2: Overall architecture of our model.

3.1 Shared Tweet Representation

We use a standard Long Short-Term Memory (LSTM) model [Hochreiter and Schmidhuber, 1997] to learn the shared tweet representation between different tasks. Let $X = (w_1, w_2, ..., w_n)$ be a tweet, where n is the tweet length and w_i is the i-th token. We transform each token w_i into a real-valued vector x_i using the word embedding vector of w_i, obtained by looking up a pre-trained word embedding table D. We use the skip-gram algorithm to train embeddings [Mikolov *et al.*, 2013].

The LSTM is used over X to generate a hidden vector sequence $(h_1, h_2, ..., h_n)$. At each step t, the hidden vector h_t of LSTM model is computed based on the current vector x_t and the previous vector h_{t-1}, and $h_t = \text{LSTM}(x_t, h_{t-1})$. The initial state and all stand LSTM parameters are randomly initialized and tuned during training. We use $H = h_n$ as the shared representation for X.

3.2 Joint Model

Event mention detection

Event mention detection is a binary classification task, addressed using a multi-layer perceptron Formally, give an input vector H, a hidden layer is used to induce a set of high-level features:

$$H_d = \sigma(W_d^h H + b_d^h), \quad (1)$$

H_d is used as inputs to a softmax output layer:

$$P_d = \text{softmax}(W_d H_d + B_d) \quad (2)$$

Here, W_d^h, b_d^b, W_d, and b_d are model parameters. P_d is two-dimensional, where $P_d(0)$ denotes the probability of X being relevant, and $P_d(1)$ denotes the probability of X being irrelevant.

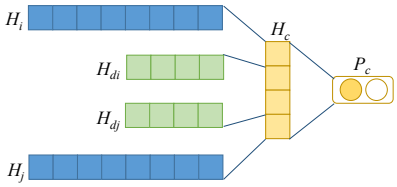

Figure 3: Siamese network for event clustering.

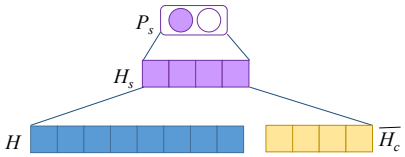

Figure 4: Multi-layer perceptron for event summarization.

Event clustering

We use the stream based clustering algorithm of Aggarwal and Subbian [2012] to cluster incoming tweets into distinct event groups, each discussing a unique event. This algorithm works incrementally with each incoming tweet in the stream calculating a similarity score between the new tweet and each tweet in the existing event clusters. The similarity between the new incoming tweet and its most similar counterpart in each event cluster is used to measure the similarity between the new tweet and the event cluster. A threshold $\mu - 3 \cdot \sigma$ is used to detect whether the new tweet belongs to an existing cluster, where μ is the mean of all previous similarity scores, and σ is the standard deviation. If the similarities between the tweet and all existing clusters are below the threshold, a new event cluster is established using the tweet. Otherwise the tweet is added to the most similar existing event cluster.

Shown in Figure 3, for calculating the similarity between two tweets X_i and X_j, we consider a Siamese network [Katiyar and Cardie, 2016; Mueller and Thyagarajan, 2016], which takes their shared representation vectors H_i and H_j, and calculates a similarity probability score P_c via parameterization

$$H_c = \sigma(W_c^h(H_i \oplus H_j) + b_c^h) \tag{3}$$

and

$$P_c = \text{softmax}(W_c H_c + B_c) \tag{4}$$

$\oplus$ denotes vector concatenation. W_c^h, b_c^b, W_c, and b_c are model parameters.

For better integration between event mention detection and event clustering, we additionally feed the hidden feature vector H_d of X_i and X_j to the Siamese network, resulting in

$$H_c = \sigma(W_c^h(H_i \oplus H_j \oplus H_{d_i} \oplus H_{d_j}) + b_c^h), \tag{5}$$

where H_{d_i} and H_{d_j} denote the hidden variables of event mention detection for X_i and X_j, respectively. This event mention detection and event clustering models are directly connected(cf. Figure 2), and the event mention detection parameters W_d^h, b_d^b, W_d, and b_d are updated also when the clustering sub model is trained.

Event summarization

In order to generate a summary for an event cluster, we rank all the tweets in the cluster using a probability score P_s. For each tweet X in the cluster, a multi-layer perceptron is used to estimate P_s, with the input being $H \oplus \overline{H_c^h}$. Here the vector $\overline{H_c^h}$ is the sum of H_c^h between the tweet X and all the other tweets in the same cluster. It serves two goals. First, $\overline{H_c^h}$ offers information about the whole cluster, which is useful for better deciding the relative of a given tweet in the cluster. Second, $\overline{H_c^h}$ also connects the clustering and summarization steps, so that information sharing is enhanced between them and back propagation is enabled (cf. Figure 2).

Formally, P_s is calculated by

$$P_s = \text{softmax}(W_s H_s + B_s) \tag{6}$$

where the intermediate hidden layer is

$$H_s = \sigma(W_s^h(H \oplus \overline{H_c^h}) + b_s^h) \tag{7}$$

W_s^h, b_s^b, W_s, and b_s are model parameters. Figure 4 shows the sub model structure.

3.3 Training

Our training objective is to minimize the cross-entropy loss between the gold labels and predicted labels on those three tasks. We apply online training, adjusting model parameters using Adagrad [Duchi *et al.*, 2011]. In order to avoid overfitting, dropout is used on word embeddings with a ratio of 0.2 [Hinton *et al.*, 2012]. The size of the hidden layers H_d, H_c, and H_s are equally set to 32. We train word embeddings using the *Skip-gram* algorithm[1], and fine-tune them during training. The size of word embeddings is 128.

4 Experiments

4.1 Data Annotation

Existing event detection and summarization data sets [Allan, 2002; Petrovic *et al.*, 2010] do not include annotated data for all the three sub-tasks that we investigate. Thus, we collect and annotate two datasets for evaluating the performance of our proposed system, one from the earthquake domain, and another from DDoS attack domain. All data were collected by using the Twitter streaming API[2], and consist of tweets from June 2013 until April 2016 with relevant domain keywords. For earthquake events, 'earthquake' is used, together with 'shake', 'refugees', 'victims', which are chosen according to PMI similarity to 'earthquake'; for DDoS events, 'ddos' and its related words 'anonymous attack', 'spoofed attack', and 'zombies host' are used as keywords using the same strategy.

For annotating events, we adopt the approach employed by NIST in labeling TDT data [Allan, 2002]. First, we define a set of events manually, thus avoiding the bias of using the output of a particular system. We choose from the set of important events during the time period of our corpus, according to Wikipedia and ddosattacks.net[3], obtaining 47 events on the earthquake domain and 170 events on the DDoS domain. The number of events in our corpus is comparable to the first TDT corpus, which contains 25 events. We then annotate the

[1] https://code.google.com/p/word2vec/
[2] https://dev.twitter.com/streaming
[3] A DDoS news website, http://www.ddosattacks.net

	Earthquake	DDoS
#Event	47	170
#Post	12090	17760
Vocabulary size	11462	15032

Table 1: Statistic of the dataset.

tweets by using the following rules: 1) A relevant tweet must explicitly mention the event and the reader should not need an external reference to infer what has happened after reading the tweet. 2) The main purpose of the tweet should be to inform of the event, and stories that only briefly refer to the main event are labeled off-topic. The statistics of the data sets is shown on Table 1.

For event mention detection training, we obtain positive samples from the annotated event mention posts, and negative samples by selecting those posts that contain the event keyword but do not mention any real event.

For event summarization, we use the content of the corresponding earthquake Wikipedia page or DDoS news pages as reference summaries, since manually written articles can serve as rational representations of the events.

We randomly choose 10 events as training data for the earthquake domain, 80 events as training data for the DDoS domain, and the remaining events as testing data.

4.2 Evaluation Metrics

Clustering. We use the standard TDT evaluation procedure [Allan, 2002], where normalized Topic Weighted Minimum Cost (C_{min}) is taken for evaluating clustering accuracy. C_{min} is a linear combination of miss and false alarm probabilities, which allows comparing different methods based on a single value metric. A lower value of C_{min} indicates better performance.

Summarization. We use ROUGE-1.5.5 [Lin, 2004] for summary evaluation, which has been adopted by several shared tasks in DUC and TAC. We report unigram overlap (ROUGE-1) for assessing informativeness, extracting top-10 most relevant posts to build the summary of each event.

4.3 Effectiveness of Event Mention Detection

We conduct an experiment on the earthquake domain to verify the effectiveness of event mention detection (i.e. filtering) for event detection. Table 2 indicates the event clustering performance with/without the event mention detection as a filtering step. We use two similarity calculation strategies for clustering: *Cosine* is a traditional strategy which is used on [Aggarwal and Subbian, 2012] and use bag-of-words as document representation, and *LSTM* means calculating the similarity between two tweets with tweet representation using the LSTM based Siamese network [Mueller and Thyagarajan, 2016]. Note that the proposed joint model use the same Siamese network to measure the similarity between two tweets. We take *Random* clustering as a low baseline.

Both the cosine and LSTM similarity strategies outperform the random approach significantly (p-value < 0.01 using t-test). In addition, similarity strategies with event mention filtering always outperform those without event mention filtering (p-value < 0.01 using t-test). It indicates that event mention detection is very important and effective for event detection. Table 3 shows the examples of irrelevant tweets filtered by the proposed model. These tweets contain the 'earthquake' keyword, but do not mention any earthquake event.

Method	C_{min}
Random	86.2
Cosine − filtering	65.8
Cosine + filtering	60.9
LSTM − filtering	64.4
LSTM + filtering	58.8

Table 2: Results of event mention detection.

Irrelevant Tweets
balochistan push is causing political earthquakes
i have bad feeling. hope no earthquake happen
when you know that an earthquake has occurred, stand by for a tsunami emergency message
independant scientifc organization and provider of real-time earthquake warning

Table 3: Example of irrelevant tweets filtered by proposed model.

Method	Clustering	Summarization
LSTM-Pipeline	58.8	18.2
LSTM-Joint	52.2	19.4
+Detect	50.2	20.6
+Cluster	47.2	20.1
JEDS	**45.8**	**21.3**

Table 4: Effectiveness of joint modeling. The performance of clustering is measured by C_{min} (%), and ROUGE-1 (%) is used to measure the performance of summarization.

4.4 Effectiveness of Joint Modeling

The main research question that we address in this paper is whether joint model improves the accuracy of each stage in our pipeline setting. Table 4 illustrates the results of different ablation baselines, where *LSTM-Pipeline* uses a LSTM to learn a separate representation of each tweet, which is used for event clustering and summarization in a pipeline setting. In this setting, there is no parameter sharing, and a separate LSTM representation is used in each subtask. *LSTM-Joint* employs the LSTM-based shared representation H (without stacking and back-propagation between tasks) to learn the different tasks jointly. *+Detect* uses the shared representation and stacked event mention detection with clustering, and *+Cluster* uses the shared representation and stacked event clustering and summarization. JEDS is the proposed Joint Event Detection and Summarization system, which uses the shared LSTM tweets representation between all subtasks, and stack all subtasks with back-propagation training.

From the table we can find that, 1) The simple joint model, which only shares the representation of each task, can outperform the basic pipeline system (p-value < 0.01 using t-

Method	C_{min}
LSH	66.7
AS12	60.9
JEDS	**45.8**

Table 5: Comparison of clustering algorithms.

Method	ROUGE-1
AS12+LexRank	18.8
AS12+CL16	19.6
LSH+LexRank	17.2
LSH+CL16	19.1
JEDS	**21.3**

Table 6: Comparison of summarization algorithms.

test). It indicates the effect of the representation learning for the joint model. 2) Stacking of event mention detection and clustering is highly effective for the joint model, considering back propagation from an event clusterings to mention detection (+*Detect*) or from event summarization to clustering (+*Cluster*). 3) Based on sharing representation and stacking for all of the sub-tasks, the proposed joint model outperforms all baseline models in both clustering and summarization stages (p-value < 0.01 using t-test). It also shows that summarization information can indeed enhance event detection.

4.5 Comparison to State-of-the-Art

We show the final results for event detection and summarization, comparing our proposed joint model with following state-of-the-art baseline pipeline systems.

- *AS12* [Aggarwal and Subbian, 2012] is a popular stream based clustering model for event detection. Given an event mentioning tweet, the model is to decide whether it belongs to an existing event cluster, or describes a new event. It use bag-of-words as the document representation of each tweet.
- *LexRank* [Erkan and Radev, 2004] is a traditional extractive text summarization model; it uses the PageRank algorithm to rank and extract the relevant sentences for generating the summary for each event.
- *LSH* [Wurzer *et al.*, 2015] is a state-of-the-art event detection algorithm. It uses Locality Sensitive Hashing (LSH) to detect and track events on unbounded high volume tweet streams in constant time and space. It use bag-of-words as the document representation of each tweet.
- *CL16* [Cheng and Lapata, 2016] is a state-of-the-art text summarization model, which uses LSTM model and attention mechanism to select the relevant sentences from each event.

We integrate the above models in the pipeline setting, and compare them with the proposed model. The event detection and summarization results are given in Table 5 and Table 6, respectively, where the input of summarization includes pipelines with different event detection methods. As can be seen from Table 5, the stream-based clustering model (AS12) outperforms the LSH model in the clustering phase. Our joint model significantly outperforms both methods (p-value $<$ 0.01 using t-test). From Table 6, we can find that the LSTM summarization model (CL16) outperforms the LexRank model in the summarization phase, which can be explained by the fact that neural network models can capture a richer features representation compared to discrete models. In addition, our joint model outperforms the pipeline models significantly (p-value < 0.01 using t-test), indicating the effectiveness of joint learning for mitigating the error propagation, and benefits from the inter-dependencies among different tasks.

Method	Clustering	Summarization
AS12+LexRank	64.4	15.5
LSH+CL16	57.8	16.5
JEDS	**38.3**	**18.7**

Table 7: Results on the DDoS domain.

4.6 Results on the DDoS Domain

The results above are all obtained on the earthquake domain. Here, we show the performance of the proposed system on the DDoS domain. We compare our results with the pipeline models of AS12+LexRank and LSH+CL16. From Table 7, we can find that the proposed joint model brings significant improvements over the state-of-the-art baseline models (p-value < 0.01 using t-test), which again indicates the effectiveness of the joint model with both shared representation and stack-propagation, and is consistent with the above experiments. The performance of clustering on the DDoS domain is higher than that on the earthquake domain, which may be because DDoS attack events are easier to separate out from mundane posts compared to earthquake events. However, summarization results on DDoS domain are lower than earthquake domain, it may be due to the fact that DDoS events contain more technical terms.

5 Conclusion

We presented a joint model to detect, cluster, and summarize events collectively by using global shared representation and stacking between different sub-tasks. Experiments demonstrated that our proposed joint model was more effective compared to pipeline baseline models. The neural joint system outperformed state-of-the-art baselines that employed discrete or neural models for new event detection and summarization.

Acknowledgments

The corresponding authors is Yue Zhang. We thank our anonymous reviewers for their constructive comments, which helped to improve the paper. This work is supported by the Temasek Lab grant IGDST1403012 at Singapore University of Technology and Design.

References

[Aggarwal and Subbian, 2012] Charu C Aggarwal and Karthik Subbian. Event detection in social streams. In *SDM*, volume 12, pages 624–635. SIAM, 2012.

[Allan *et al.*, 1998] James Allan, Jaime G Carbonell, George Doddington, Jonathan Yamron, and Yiming Yang. Topic detection and tracking pilot study final report. 1998.

[Allan, 2002] James Allan, editor. *Topic Detection and Tracking: Event-based Information Organization*. Kluwer Academic Publishers, Norwell, MA, USA, 2002.

[Chang *et al.*, 2016] Ching-Yun Chang, Zhiyang Teng, and Yue Zhang. Expectation-regulated neural model for event mention extraction. In *Proceedings of NAACL*, 2016.

[Chen *et al.*, 2013] Yan Chen, Hadi Amiri, Zhoujun Li, and Tat-Seng Chua. Emerging topic detection for organizations from microblogs. In *Proceedings of SIGIR*, 2013.

[Chen *et al.*, 2016] Hongshen Chen, Yue Zhang, and Qun Liu. Neural network for heterogeneous annotations. In *Proceedings of EMNLP*, 2016.

[Cheng and Lapata, 2016] Jianpeng Cheng and Mirella Lapata. Neural summarization by extracting sentences and words. In *Proceedings of ACL*, 2016.

[Collobert *et al.*, 2011] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel P. Kuksa. Natural language processing (almost) from scratch. *JMLR*, 12:2493–2537, 2011.

[Duchi *et al.*, 2011] John C. Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *JMLR*, 12:2121–2159, 2011.

[Erkan and Radev, 2004] Günes Erkan and Dragomir R Radev. Lexrank: Graph-based lexical centrality as salience in text summarization. *JAIR*, 22:457–479, 2004.

[Finkel and Manning, 2010] Jenny Rose Finkel and Christopher D. Manning. Hierarchical joint learning: Improving joint parsing and named entity recognition with non-jointly labeled data. In *Proceedings of ACL*, 2010.

[Hinton *et al.*, 2012] Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*, 2012.

[Hochreiter and Schmidhuber, 1997] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[Katiyar and Cardie, 2016] Arzoo Katiyar and Claire Cardie. Investigating lstms for joint extraction of opinion entities and relations. In *Proceedings of the ACL*, 2016.

[Li *et al.*, 2010] Junhui Li, Guodong Zhou, and Hwee Tou Ng. Joint syntactic and semantic parsing of chinese. In *Proceedings of ACL*, 2010.

[Li *et al.*, 2012] Rui Li, Kin Hou Lei, Ravi Khadiwala, and Kevin Chen-Chuan Chang. TEDAS: A twitter-based event detection and analysis system. In *Proceedings of ICDE*, 2012.

[Li *et al.*, 2016] Fei Li, Yue Zhang, Meishan Zhang, and Donghong Ji. Joint models for extracting adverse drug events from biomedical text. In *Proceedings of IJCAI*, 2016.

[Lin, 2004] Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out: Proceedings of the ACL-04 workshop*, 2004.

[Mikolov *et al.*, 2013] Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. In *Proceedings of NIPS*, 2013.

[Mueller and Thyagarajan, 2016] Jonas Mueller and Aditya Thyagarajan. Siamese recurrent architectures for learning sentence similarity. In *Proceedings of AAAI*, 2016.

[Nguyen *et al.*, 2016] Thien Huu Nguyen, Kyunghyun Cho, and Ralph Grishman. Joint event extraction via recurrent neural networks. In *Proceedings of NAACL*, 2016.

[Nichols *et al.*, 2012] Jeffrey Nichols, Jalal Mahmud, and Clemens Drews. Summarizing sporting events using twitter. In *Proceedings of IUI*, 2012.

[Petrovic *et al.*, 2010] Sasa Petrovic, Miles Osborne, and Victor Lavrenko. Streaming first story detection with application to twitter. In *Proceedings of NAACL*, 2010.

[Radev *et al.*, 2004] Dragomir R. Radev, Hongyan Jing, Magorzata Sty, and Daniel Tam. Centroid-based summarization of multiple documents. *IPM*, 40(6):919–938, 2004.

[Ritter *et al.*, 2012] Alan Ritter, Oren Etzioni, Sam Clark, et al. Open domain event extraction from twitter. In *Proceedings of the KDD*. ACM, 2012.

[Ritter *et al.*, 2015] Alan Ritter, Evan Wright, William Casey, and Tom M. Mitchell. Weakly supervised extraction of computer security events from twitter. In *Proceedings of WWW*, 2015.

[Rush *et al.*, 2015] Alexander M. Rush, Sumit Chopra, and Jason Weston. A neural attention model for abstractive sentence summarization. In *Proceedings of EMNLP*, 2015.

[Socher *et al.*, 2013] Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of EMNLP*, 2013.

[Song *et al.*, 2014] Linfeng Song, Yue Zhang, Kai Song, and Qun Liu. Joint morphological generation and syntactic linearization. In *Proceedings of AAAI*, 2014.

[Sun *et al.*, 2015] Rui Sun, Yue Zhang, Meishan Zhang, and Dong-Hong Ji. Event-driven headline generation. In *Proceedings of ACL*, 2015.

[Wurzer *et al.*, 2015] Dominik Wurzer, Victor Lavrenko, and Miles Osborne. Twitter-scale new event detection via k-term hashing. In *Proceedings of EMNLP*, 2015.

[Zhang and Clark, 2008] Yue Zhang and Stephen Clark. Joint word segmentation and POS tagging using a single perceptron. In *Proceedings of ACL*, 2008.