Applications of Online Deep Learning for Crisis Response Using Social Media Information

Dat Tien Nguyen, Shafiq Joty, Muhammad Imran, Hassan Sajjad, Prasenjit Mitra
Qatar Computing Research Institute, HBKU
Doha, Qatar
{ndat, sjoty, mimran, hsajjad, pmitra}@qf.org.qa

ABSTRACT

During natural or man-made disasters, humanitarian response organizations look for useful information to support their decision-making processes. Social media platforms such as Twitter have been considered as a vital source of useful information for disaster response and management. Despite advances in natural language processing techniques, processing short and informal Twitter messages is a challenging task. In this paper, we propose to use Deep Neural Network (DNN) to address two types of information needs of response organizations: (*i*) identifying informative tweets and (*ii*) classifying them into topical classes. DNNs use distributed representation of words and learn the representation as well as higher level features automatically for the classification task. We propose a new online algorithm based on stochastic gradient descent to train DNNs in an online fashion during disaster situations. We test our models using a crisis-related real-world Twitter dataset.

Keywords

deep learning, supervised classification, twitter, text classification, crisis response

1. INTRODUCTION

Emergency events such as natural or man-made disasters bring unique challenges for humanitarian response organizations. Particularly, sudden-onset crisis situations demand officials to make fast decisions based on minimum information available to deploy rapid crisis response. However, information scarcity during time-critical situations hinders decision-making processes and delays response efforts [4, 7].

During crises, people post updates regarding their statuses, ask for help and other useful information, report infrastructure damages, injured people, etc., on social media platforms like Twitter [26]. Humanitarian organizations can use this citizen-generated information to provide relief if critical information is easily available in a timely fashion.[1] In this paper, we consider the classification of the social media posts into different humanitarian categories to fulfill different information needs of humanitarian organizations. Specifically, we address two types of information needs described as follows:

Informativeness of social media posts: Information posted on social networks during crises vary greatly in value. Most messages contain irrelevant information not useful for disaster response and management. Humanitarian organizations do not want a deluge of noisy messages that are of a personal nature or those that do not contain any useful information. They want clean data that consists of messages containing potentially useful information. They can then use this information for various purposes such as situational awareness. In order to assist humanitarian organizations, we perform *binary classification*. That is, we aim to classify each message into one of the two classes i.e. *"informative"* vs. *"not informative"*.

Information types of social media posts Furthermore, humanitarian organizations are interested in sorting social media posts into different categories. Identifying social media posts by category assists humanitarian organizations in coordinating their response. Categories such as infrastructure damage, reports of deceased or injured, urgent need for shelter, food and water, or donations of goods or services could therefore be directed to different relief functions. In this work, we show how we can classify tweets into multiple classes.

Automatic classification of short crisis-related messages such as tweets is a challenging task due to a number of reasons. Tweets are short (only 140 characters), informal, often contain abbreviations, spelling variations and mistakes, and, therefore, they are hard to understand without enough context. Despite advances in natural language processing (NLP), interpreting the semantics of short informal texts automatically remains a hard problem. Traditional classification approaches rely on manually engineered features like cue words and TF-IDF vectors for learning [7]. Due to the high variability of the data during a crisis, adapting the model to changes in features and their importance manually is undesirable (and often infeasible).

To overcome these issues, we use Deep Neural Networks (DNNs) to classify the tweets. DNNs are usually trained using online learning and have the flexibility to adaptively learn the model parameters as new batches of labeled data arrive, without requiring to retrain the model from scratch. DNNs use distributed condensed representation of words and learn the representation as well as higher level abstract features automatically for the classification task. Distributed representation (as opposed to sparse discrete representation) generalizes well. This can be a crucial advantage at the beginning of a new disaster, when there is not enough event-specific labeled data. We can train a reasonably good DNN model using previously labeled data from other events, and then the model is

[1] http://www.napsgfoundation.org/wp-content/uploads/2013/02/NAPSG-Remote-Sensing-Webcast-022213.pdf

This article has been peer reviewed and accepted at the 4th international workshop on Social Web for Disaster Management (SWDM), co-located with the 25th Conference of Information and Knowledge Management (CIKM), October 24–28, 2016, Indianapolis, USA.

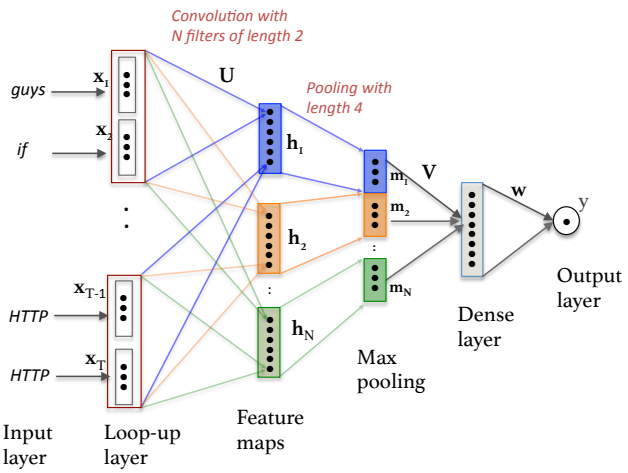

Figure 1: Convolutional neural network on a sample tweet: "guys if know any medical emergency around balaju area you can reach umesh HTTP doctor at HTTP HTTP".

fine-tuned adaptively as newly labeled data arrives in small batches.

In this paper, we use Deep Neural Network (DNN) to address two types of information needs of response organizations: (*i*) identifying informative tweets and (*ii*) classifying them into topical classes. DNNs use distributed representation of words and learn the representation as well as higher level features automatically for the classification task. We propose a new online algorithm based on stochastic gradient descent to train DNNs in an online fashion during disaster situations. Moreover, we make our source code publicly available for crisis computing community for further research at: https://github.com/CrisisNLP/deep-learning-for-big-crisis-data

In the next section, we provide details regarding DNNs we use and the online learning algorithm. Section 3 describes datasets and online learning settings. In Section 4, we describe results of our models. Section 5 presents related-work and we conclude our paper in Section 6.

2. DEEP NEURAL NETWORK

As argued before, deep neural networks (DNNs) can be quite effective in classifying tweets during a disaster situation because of their distributed representation of words and automatic feature learning capabilities. Furthermore, DNNs are usually trained using online algorithms, which nicely suits the needs of a crisis response situation.

Our main hypothesis is that in order to effectively classify tweets, which are short and informal, a classification model should learn the *key* features at different levels of abstraction. To this end, we use a Convolutional Neural Network (CNN), which has been shown to be effective for sentence-level classification tasks [13].

2.1 Convolutional Neural Network

Figure 1 demonstrates how a CNN works with an example tweet.[2] Each word in the vocabulary V is represented by a D dimensional vector in a shared look-up table $L \in \mathbb{R}^{|V| \times D}$. L is considered a model parameter to be learned. We can initialize L randomly or using pretrained word embedding vectors like word2vec [16].

Given an input tweet $\mathbf{s} = (w_1, \cdots, w_T)$, we first transform it into a feature sequence by mapping each word token $w_t \in \mathbf{s}$ to an index in L. The look-up layer then creates an input vector $\mathbf{x_t} \in \mathbb{R}^D$ for each token w_t, which are passed through a sequence of convolution and pooling operations to learn high-level abstract features.

A convolution operation involves applying a *filter* $\mathbf{u} \in \mathbb{R}^{L.D}$ to a window of L words to produce a new feature

$$h_t = f(\mathbf{u}.\mathbf{x}_{t:t+L-1} + b_t) \tag{1}$$

where $\mathbf{x}_{t:t+L-1}$ denotes the concatenation of L input vectors, b_t is a bias term, and f is a nonlinear activation function (e.g., sig, tanh). A filter is also known as a kernel or a feature detector. We apply this filter to each possible L-word window in the tweet to generate a *feature map* $\mathbf{h}_i = [h_1, \cdots, h_{T+L-1}]$. We repeat this process N times with N different filters to get N different feature maps. We use a *wide* convolution [12] (as opposed to *narrow*), which ensures that the filters reach the entire sentence, including the boundary words. This is done by performing *zero-padding*, where out-of-range (i.e., $t<1$ or $t>T$) vectors are assumed to be zero.

After the convolution, we apply a max-pooling operation to each feature map.

$$\mathbf{m} = [\mu_p(\mathbf{h}_1), \cdots, \mu_p(\mathbf{h}_N)] \tag{2}$$

where $\mu_p(\mathbf{h}_i)$ refers to the max operation applied to each window of p features in the feature map $\mathbf{h}_i$. For instance, with $p = 2$, this pooling gives the same number of features as in the feature map (because of the zero-padding). Intuitively, the filters compose local n-grams into higher-level representations in the feature maps, and max-pooling reduces the output dimensionality while keeping the most important aspects from each feature map.

Since each convolution-pooling operation is performed independently, the features extracted become invariant in locations (i.e., where they occur in the tweet), thus acting like bag-of-n-grams. However, keeping the *order* information could be important for modeling sentences. In order to model interactions between the features picked up by the filters and the pooling, we include a *dense* layer of hidden nodes on top of the pooling layer

$$\mathbf{z} = f(V\mathbf{m} + \mathbf{b_h}) \tag{3}$$

where V is the weight matrix, $\mathbf{b_h}$ is a bias vector, and f is a nonlinear activation. The dense layer naturally deals with variable sentence lengths by producing fixed size output vectors $\mathbf{z}$, which are fed to the output layer for classification.

Depending on the classification tasks, the output layer defines a probability distribution. For binary classification tasks, it defines a Bernoulli distribution:

$$p(y|\mathbf{s}, \theta) = \text{Ber}(y|\,\text{sig}(\mathbf{w^T z} + b)) \tag{4}$$

where sig refers to the sigmoid function, and $\mathbf{w}$ are the weights from the dense layer to the output layer and b is a bias term. For multi-class classification the output layer uses a `softmax` function. Formally, the probability of k-th label in the output for classification into K classes:

$$P(y = k|\mathbf{s}, \theta) = \frac{exp\ (\mathbf{w}_k^T \mathbf{z} + b_k)}{\sum_{j=1}^{K} exp\ (\mathbf{w}_j^T \mathbf{z} + b_j)} \tag{5}$$

where, $\mathbf{w}_k$ are the weights associated with class k in the output layer. We fit the models by minimizing the cross-entropy between the predicted distributions $\hat{y}_{n\theta} = p(y_n|\mathbf{s}_n, \theta)$ and the target distributions y_n (i.e., the gold labels).[3] The objective function $f(\theta)$ can

[2]The HTTP tag in the example represents URLs.

[3]Other loss functions (e.g., hinge) yielded similar results.

Algorithm 1: Online learning of CNN

1. Initialize the model parameters θ_0;
2. **for** *a minibatch* $B_t = \{\mathbf{s}_1 \dots \mathbf{s}_n\}$ *at time* t **do**
   a. Compute the loss $f(\theta_t)$ in Equation 6;
   b. Compute gradients of the loss $f'(\theta_t)$ using backpropagation;
   c. Update: $\theta_{t+1} = \theta_t - \eta_t \frac{1}{n} f'(\theta_t)$;

end

be written as:

$$f(\theta) = \sum_{n=1}^{N} \sum_{k=1}^{K} y_{nk} \; log \; P(y_n = k|\mathbf{s}_n, \theta) \tag{6}$$

where, N is the number of training examples and $y_{nk} = I(y_n = k)$ is an indicator variable to encode the gold labels, i.e., $y_{tk} = 1$ if the gold label $y_t = k$, otherwise 0.

2.2 Online Learning

DNNs are usually trained with first-order online methods like stochastic gradient descent (SGD). This method yields a crucial advantage in crisis situations, where retraining the whole model each time a small batch of labeled data arrives is impractical. Algorithm 1 demonstrates how our CNN model can be trained in a purely online setting. We first initialize the model parameters θ_0 (line 1), which can be a trained model from other disaster events or it can be initialized randomly to start from scratch.

As a new batch of labeled tweets $B_t = \{\mathbf{s}_1 \dots \mathbf{s}_n\}$ arrives, we first compute the log-loss (cross entropy) in Equation 6 for B_t with respect to the current parameters θ_t (line 2a). Then, we use backpropagation to compute the gradients $f'(\theta_t)$ of the loss with respect to the current parameters (line 2b). Finally, we update the parameters with the learning rate η_t and the mean of the gradients (line 2c). We take the mean of the gradients to deal with minibatches of different sizes. Notice that we take only the current minibatch into account to get an updated model. Choosing a proper learning rate η_t can be difficult in practice. Several adaptive methods such as ADADELTA [27], ADAM [14], etc., have been proposed to overcome this issue. In our model, we use ADADELTA.

2.3 Word Embedding and Fine-tuning

As mentioned before, we can initialize the word embeddings L randomly, and learn them as part of model parameters by backpropagating the errors to the look-up layer. Random initialization may lead the training algorithm to get stuck in a local minima. One can plug the readily available embeddings from external sources (e.g., Google embeddings [16]) in the neural network model and use them as features without further task-specific tuning. However, the latter approach does not exploit the automatic feature learning capability of DNN models, which is one of the main motivations of using them. In our work, we use pre-trained word embeddings (see below) to better initialize our models, and we fine-tune them for our task, which turns out to be beneficial.

Mikolov et al. [16] propose two log-linear models for computing word embeddings from large (unlabeled) corpuses efficiently: (*i*) a *bag-of-words* model CBOW that predicts the current word based on the context words, and (*ii*) a *skip-gram* model that predicts surrounding words given the current word.[4] They released their pre-trained 300-dimensional word embeddings trained by the skip-gram model on a Google news dataset.

[4]https://code.google.com/p/word2vec/

Since we work on disaster related tweets, which are quite different from news, we have trained *domain-specific* embeddings of 300-dimensions (vocabulary size 20 million) using the Skip-gram model of *word2vec* tool [17] from a large corpus of disaster related tweets. The corpus contains $57,908$ tweets and 9.4 million tokens.

3. DATASET AND EXPERIMENTAL SETTINGS

In this section, we describe the datasets used for the classification tasks and the settings for CNN and online learning.

3.1 Dataset and Preprocessing

We use CrisisNLP [9] labeled datasets. The CNN models were trained online using a labeled dataset related to the 2015 Nepal Earthquake[5] and the rest of the datasets are used to train an initial model (θ_0 in Algorithm 1) upon which the online learning is performed. The Nepal earthquake dataset consists of approximately 12k labeled tweets collected from Twitter during the event using different keywords like *NepalEarthquake*. Of all the labeled tweets, 9k are labeled by trained volunteers[6] during the actual event using the AIDR platform [8] and the remaining 3k tweets are labeled using the Crowdflower[7] crowdsourcing platform.

The dataset is labeled into different informative classes (e.g., affected individuals, infrastructure damage, donations etc.) and one "not-related" or "irrelevant" class. Table 1 provides a one line description of each class and also the total number of labels in each class. *Other useful information* and *Not related or irrelevant* are the most frequent classes in the dataset.

Data Preprocessing: We normalize all characters to their lowercased forms, truncate elongations to two characters, spell out every digit to `D`, all twitter usernames to `userID`, and all URLs to `HTTP`. We remove all punctuation marks except periods, semicolons, question and exclamation marks. We further tokenize the tweets using the CMU TweetNLP tool [6].

3.2 Online Training Settings

Before performing the online learning, we assume that an initial model θ_0 exists. In our case, we train the initial model using all the datasets from CrisisNLP except the Nepal earthquake. For online training, we sort the Nepal labeled data based on the time stamp of the tweets. This brings the tweets in their posting order. Next, the dataset D is divided at each time interval d_t in which case D is defined as: D = $\sum_{t=1}^{T} d_t$ where $d_t = 200$. For each time interval t, we divide the available labeled dataset into a train set (70%), dev set (10%), and a test set (20%) using ski-learn toolkit's module [19], which ensured that the class distribution remains reasonably balanced in each subset.

Based on the data splitting strategy mentioned above, we start online learning to train a binary and a multi-class classifier. For the binary classifier training, we merge all the informative classes to create one general *Informative* class. We train CNN models by optimizing the cross entropy in Equation 4 using the gradient-based online learning algorithm ADADELTA [27].[8] The learning rate and the parameters were set to the values as suggested by the authors. The maximum number of epochs was set to 25. To avoid overfitting, we use dropout [23] of hidden units and *early stop-*

[5]https://en.wikipedia.org/wiki/April_2015_Nepal_earthquake

[6]These are trained volunteers from the Stand-By-Task-Force organization (http://blog.standbytaskforce.com/).

[7]crowdflower.com

[8]Other algorithms (SGD, Adagrad) gave similar results.

Class	Labels	Description
Affected individuals	756	Reports of deaths, injuries, missing, found, or displaced people
Donations and volunteering	1021	Messages containing donations (food, shelter, services etc.) or volunteering offers
Infrastructure and utilities	351	Reports of infrastructure and utilities damage
Sympathy and support	983	Messages of sympathy-emotional support
Other useful information	1505	Messages containing useful information that does not fit in one of the above classes
Not related or irrelevant	6698	Irrelevant or not informative, or not useful for crisis response

Table 1: Description of the classes in the dataset. Column *Labels* shows the total number of labeled examples in each class

ping based on the accuracy on the validation set.[9] We experimented with $\{0.0, 0.2, 0.4, 0.5\}$ dropout rates and $\{32, 64, 128\}$ minibatch sizes. We limit the vocabulary (V) to the most frequent $P\%$ ($P \in \{80, 85, 90\}$) words in the training corpus. The word vectors in L were initialized with the pre-trained embeddings. We use rectified linear units (ReLU) for the activation functions (f), $\{100, 150, 200\}$ filters each having window size (L) of $\{2, 3, 4\}$, pooling length (p) of $\{2, 3, 4\}$, and $\{100, 150, 200\}$ dense layer units. All the hyperparameters are tuned on the development set.

4. RESULTS

In this section, we present our results for binary and multi-class classification tasks.

4.1 Binary Classification

Figure 2 shows the results for the *"informative"* vs. *"not informative"* binary classification task using online learning. The performance of the model is quite inconsistent as the size of the in-event training data varies. We observe an improvement in performance initially. However, the results dropped when the training size is between 2200 to 3900 tweets. We investigated this strange result and found that this could be due to the inconsistencies in the annotation procedure and the data sources. In our in-event (Nepal Earthquake) training data, first 3000 tweets are from CrowdFlower and the rest are from AIDR. Tweets in CrowdFlower were annotated by paid workers, where AIDR tweets are annotated by volunteers. We speculate these inconsistencies can affect the performance at the beginning, but as the model sees more AIDR data (4000+), the performance stabilizes.

4.2 Multi-Class Classification

Figure 3 summarizes the results of online training for the multi-class classification task. Since multi-class classification is a harder task than binary classification, the first training run provides very low accuracy and the results continue to drop until a good number of training examples are available, which in this case is approximately 2200 labeled tweets. As in the binary classification case, after the initial dip in performance, once over 3000 tweets are available, the performance of the classifier improves and remains stable after that.

The benefit of using online learning methods like CNN compared to offline learning methods used in classifiers like SVM, Naive Bayes, and Logistic Regression is online training. The labeled data comes in batches and retraining a model on the complete data every time with the addition of newly labeled data is an expensive task. Online training methods learn in small batches, which suits the situation in hand perfectly.

Another advantage of neural network methods is automatic feature extraction that does not require any manual feature engineering. The models take labeled tweets as input and automatically learn features based on distributed representation of words.

[9] l_1 and l_2 regularization on weights did not work well.

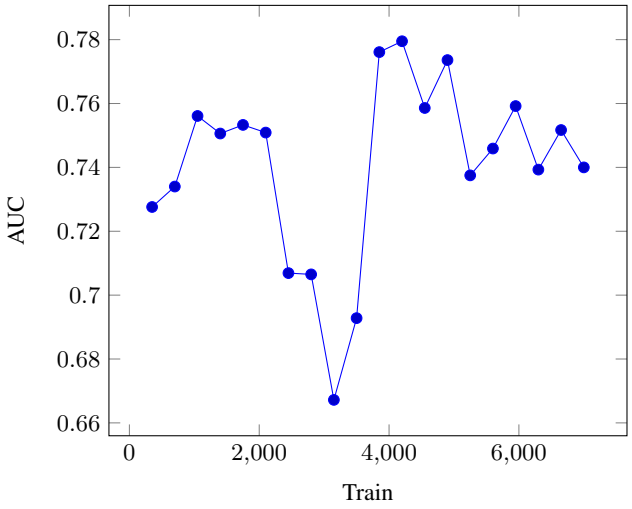

Figure 2: **Binary** classification: Performance of the CNN model with varying size of the training data

4.3 Discussion

Rapid analysis of social media posts during time-critical situations is important for humanitarian response organization to take timely decisions and to launch relief efforts. This work proposes solutions to two main challenges that humanitarian organizations face while incorporating social media data into crisis response. First, how to filter-out noisy and irrelevant messages from big crisis data and second, categorization of the informative messages into different classes of interest. By utilizing labeled data from past crises, we show the performance of DNNs trained using the proposed online learning algorithm for binary and multi-class classification tasks.

We observe that past labeled data helps when no event-specific data is available in the early hours of a crisis. However, labeled data from event always help improve the classification accuracy.

5. RELATED WORK

Recent studies have shown the usefulness of crisis-related data on social media for disaster response and management [1, 22, 24]. A number of systems have been developed to classify, extract, and summarize [21] crisis-relevant information from social media; for a detailed survey see [7]. Cameron, et al., describe a platform for emergency situation awareness [2]. They classify interesting tweets using an SVM classifier. Verma, et al., use Naive Bayes and MaxEnt classifiers to find situational awareness tweets from several crises [25]. Imran, et al., implemented AIDR to classify a Twitter data stream during crises [8]. They use a random forest classifier in

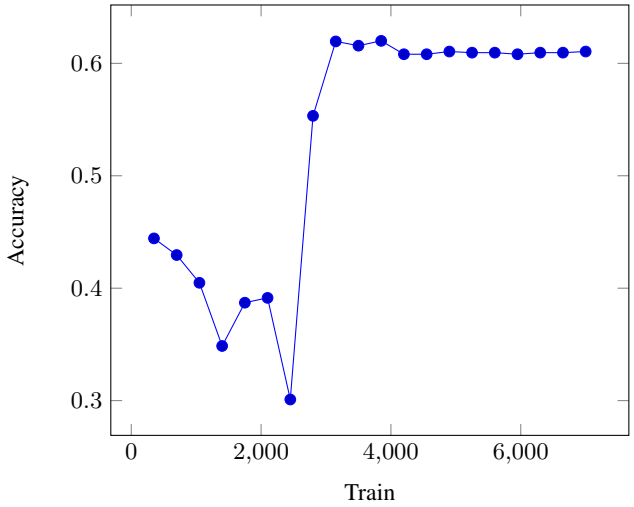

Figure 3: **Multi-class** classification: Performance of the CNN model with various sizes of training data.

an offline setting. After receiving every mini-batch of 50 training examples, they replace the older model with a new one. In [10], the authors show the performance of a number of non-neural network classifiers trained on labeled data from past crisis events. However, they do not use DNNs in their comparison.

DNNs and word embeddings have been applied successfully to address NLP problems [5, 3, 18, 15, 11]. The emergence of tools such as word2vec [17] and GloVe [20] have enabled NLP researchers to learn word embeddings efficiently and use them to train better models.

Collobert, et al. [5] presented a unified DNN architecture for solving various NLP tasks including part-of-speech tagging, chunking, named entity recognition and semantic role labeling. They showed that DNNs outperform traditional models in most of these tasks. They also proposed a multi-task learning framework for solving the tasks jointly.

Kim [13] and Kalchbrenner et al. [12] used convolutional neural networks (CNN) for sentence-level classification tasks (e.g., sentiment/polarity classification, question classification) and showed that CNNs outperform traditional methods (e.g., SVMs, MaxEnts). Caragea, Silvescu, and Tapia used CNNs to identify informative messages during disasters [3]. However, to the best of our knowledge, no previous research has shown the efficacy of CNNs to both the binary classification and the multi-class classification problems using online learning.

6. CONCLUSIONS

We presented an online learning model namely Convolutional Neural Network for the purpose of classifying tweets in a disaster response scenario. We proposed a new online learning algorithm for training CNNs in online fashion. We showed that online training of the model perfectly suits the disaster response situation. We assume that a base model trained on past crisis labeled data exists and the event-specific labeled data arrive in small batches which are used to perform online learning. The neural network models bring an additive advantage of automatic feature extraction which eases the training process when compared with offline learning methods like SVM, logistic regression. The model uses only labeled tweets for training and automatically learns features from them. We reported the results of two classification tasks (i.e. binary and multi-class). Moreover, we also provide source code for the online learning of CNN models to research community for further extensions.

7. REFERENCES

[1] A. Acar and Y. Muraki. Twitter for crisis communication: lessons learned from japan's tsunami disaster. *International Journal of Web Based Communities*, 7(3):392–402, 2011.

[2] M. A. Cameron, R. Power, B. Robinson, and J. Yin. Emergency situation awareness from twitter for crisis management. In *Proceedings of the 21st international conference companion on World Wide Web*, pages 695–698. ACM, 2012.

[3] C. Caragea, A. Silvescu, and A. H. Tapia. Identifying informative messages in disaster events using convolutional neural networks. *International Conference on Information Systems for Crisis Response and Management*, 2016.

[4] C. Castillo. *Big Crisis Data*. Cambridge University Press, 2016.

[5] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. Natural language processing (almost) from scratch. *The Journal of Machine Learning Research*, 12:2493–2537, 2011.

[6] K. Gimpel, N. Schneider, B. O'Connor, D. Das, D. Mills, J. Eisenstein, M. Heilman, D. Yogatama, J. Flanigan, and N. A. Smith. Part-of-speech tagging for twitter: Annotation, features, and experiments. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 42–47. Association for Computational Linguistics, 2011.

[7] M. Imran, C. Castillo, F. Diaz, and S. Vieweg. Processing social media messages in mass emergency: a survey. *ACM Computing Surveys (CSUR)*, 47(4):67, 2015.

[8] M. Imran, C. Castillo, J. Lucas, P. Meier, and S. Vieweg. AIDR: Artificial intelligence for disaster response. In *Proceedings of the companion publication of the 23rd international conference on World wide web companion*, pages 159–162. International World Wide Web Conferences Steering Committee, 2014.

[9] M. Imran, P. Mitra, and C. Castillo. Twitter as a lifeline: Human-annotated twitter corpora for NLP of crisis-related messages. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC)*, 2016.

[10] M. Imran, P. Mitra, and J. Srivastava. Cross-language domain adaptation for classifying crisis-related short messages. *International Conference on Information Systems for Crisis Response and Management*, 2016.

[11] S. R. Joty and E. Hoque. Speech act modeling of written asynchronous conversations with task-specific embeddings and conditional structured models. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*, 2016.

[12] N. Kalchbrenner, E. Grefenstette, and P. Blunsom. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 655–665, Baltimore, Maryland, June 2014. Association for Computational Linguistics.

[13] Y. Kim. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar, October 2014. Association for Computational Linguistics.

[14] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[15] P. Liu, S. Joty, and H. Meng. Fine-grained opinion mining with recurrent neural networks and word embeddings. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1433–1443, Lisbon, Portugal, September 2015. Association for Computational Linguistics.

[16] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[17] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119, 2013.

[18] D. T. Nguyen, K. A. A. Mannai, S. Joty, H. Sajjad, M. Imran, and P. Mitra. Rapid classification of crisis-related data on social networks using convolutional neural networks. *arXiv preprint arXiv:1608.03902*, 2016.

[19] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[20] J. Pennington, R. Socher, and C. Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, October 2014. Association for Computational Linguistics.

[21] K. Rudra, S. Banerjee, N. Ganguly, P. Goyal, M. Imran, and P. Mitra. Summarizing situational tweets in crisis scenario. In *Proceedings of the 27th ACM Conference on Hypertext and Social Media*, pages 137–147. ACM, 2016.

[22] T. Sakaki, M. Okazaki, and Y. Matsuo. Earthquake shakes twitter users: real-time event detection by social sensors. In *Proceedings of the 19th international conference on World wide web*, pages 851–860. ACM, 2010.

[23] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.

[24] I. Varga, M. Sano, K. Torisawa, C. Hashimoto, K. Ohtake, T. Kawai, J.-H. Oh, and S. De Saeger. Aid is out there: Looking for help from tweets during a large scale disaster. In *ACL (1)*, pages 1619–1629, 2013.

[25] S. Verma, S. Vieweg, W. J. Corvey, L. Palen, J. H. Martin, M. Palmer, A. Schram, and K. M. Anderson. Natural language processing to the rescue? extracting" situational awareness" tweets during mass emergency. In *ICWSM*. Citeseer, 2011.

[26] S. Vieweg, C. Castillo, and M. Imran. Integrating social media communications into the rapid assessment of sudden onset disasters. In *Social Informatics*, pages 444–461. Springer, 2014.

[27] M. D. Zeiler. ADADELTA: an adaptive learning rate method. *CoRR*, abs/1212.5701, 2012.