Graph Based Semi-supervised Learning with Convolution Neural Networks to Classify Crisis Related Tweets

Firoj Alam[1], Shafiq Joty[2], Muhammad Imran[1]
[1]Qatar Computing Research Institute, HBKU, Doha, Qatar
[2]School of Computer Science and Engineering, NTU, Singapore
[1]{fialam, mimran}@hbku.edu.qa, [2]srjoty@ntu.edu.sg

Abstract

During time-critical situations such as natural disasters, rapid classification of data posted on social networks by affected people is useful for humanitarian organizations to gain situational awareness and to plan response efforts. However, the scarcity of labeled data in the early hours of a crisis hinders machine learning tasks thus delays crisis response. In this work, we propose to use an inductive semi-supervised technique to utilize unlabeled data, which is often abundant at the onset of a crisis event, along with fewer labeled data. Specifically, we adopt a graph-based deep learning framework to learn an inductive semi-supervised model. We use two real-world crisis datasets from Twitter to evaluate the proposed approach. Our results show significant improvements using unlabeled data as compared to only using labeled data.

Introduction

The recent emergence and wide-adaptation of microblogging platforms such as Twitter, during crises and emergency situations due to natural or man-made disasters, has been proven useful for a number of humanitarian tasks (Imran et al. 2015). Affected population post timely and useful information of various types such as reports of injured or dead people, infrastructure damage, urgent needs (food, shelter, medical assistance) on these social networks. Humanitarian organizations believe timely access to this important information in the first few hours can help significantly and can reduce both human loss and economic damage (Vieweg, Castillo, and Imran 2014; Varga et al. 2013).

In order to identify useful messages for humanitarian tasks one potential approach is to use supervised learning to automatically categorize each incoming message (e.g., tweets) into one of the two classes i.e., *relevant* and *irrelevant* (Nguyen et al. 2017). In order to design the classification model, obtaining a large amount of labeled data is a challenging task, particularly during the first few hours of a crisis situation. However, access to abundant unlabeled data is possible under such time-critical situations, as hundreds of tweets arrive each minute. Moreover, one can rely on labeled data from past similar events. In such situations, semi-supervised methods can provide effective ways to leverage unlabeled data in addition to labeled data.

Many models have been proposed for semi-supervised learning including generative models (Nigam et al. 2000), co-training, (Mitchell 1999), self-training (Mihalcea 2004), and graph-based models (Subramanya and Talukdar 2014). These methods can be categorized into two types: *transductive* and *inductive*. In the transductive setting, a learner is only applicable to the unlabeled instances observed at training time, that is, the learner does not generalize to unobserved instances. Whereas, an inductive learner generalizes to data that are not seen at the training time. Therefore, it is more desirable to have an inductive learner over a transductive one. Other reason to prefer inductive semi-supervised learning over the transductive approach is that it avoids building the graph each time it needs to infer the labels for the unlabeled instances.

Among other semi-supervised text classification approaches, Johnson et al. (Johnson and Zhang 2015) use a Convolutional Neural Networks (CNN) via region embedding in which the CNN learns a small region from embedding. Miyato et al. (Miyato, Dai, and Goodfellow 2016) used adversarial training for text classification with a small perturbations on the input word embeddings. Our motivation of using deep neural network (i.e., CNN) is that it has shown a great success in recent years in many different areas such as NLP and data mining (Collobert et al. 2011; Grover and Leskovec 2016). Apart from the improved performance, one crucial benefit of DNN is that they obviate the need for feature engineering and learn latent features automatically as distributed dense vectors. This capability of DNN has recently been extended to the semi-supervised setting (Yang, Cohen, and Salakhutdinov 2016).

In this work, we adopt a graph-based deep learning framework recently proposed by Yang et. al (Yang, Cohen, and Salakhutdinov 2016) for learning an inductive semi-supervised model to classify tweets in a crisis situation. In this framework, CNN is combined with graph-based network that learns internal representations of the input by predicting contextual nodes in a graph that encodes *similarity* between labeled and unlabeled training instances. Compared to the work of Yang et. al (Yang, Cohen, and Salakhutdinov 2016), our approach is different in several ways: 1) we construct a graph by computing the distance between tweets based on word embeddings, 2) we use a CNN to compose higher-level features from the word embeddings, and 3) for

context prediction, instead of performing a random walk, we select nodes based on their similarity in the graph.

The evaluation of the proposed approach (see Sec. Methodology) is conducted using two real-world Twitter datasets. Our results (see Sec. Results and Discussion) demonstrate the effectiveness of our approach with an improvement from 5% to 26% compared to the supervised classification. The experimental data can be accessed through `http://crisisnlp.qcri.org`.

Methodology

Let $\mathcal{D}_l = \{\mathbf{t}_i, y_i\}_{i=1}^{L}$ and $\mathcal{D}_u = \{\mathbf{t}_i\}_{i=1}^{U}$ be the set of labeled and unlabeled tweets for a particular crisis event, where $y_i \in \{1, \ldots, K\}$ is the class label for tweet $\mathbf{t}_i$, L and U are the number of labeled and unlabeled tweets, respectively, with $n = L + U$ being the total number of training instances. Our goal is to learn an inductive model $p(y_i|\mathbf{t}_i, \theta)$, where θ denotes the model parameters. We adopt the graph-based semi-supervised embedding learning framework proposed in (Yang, Cohen, and Salakhutdinov 2016). In this framework, in addition to the labeled data, a "similarity" graph G is used to learn internal representations for the input (i.e., embeddings) by exploiting relations between labeled and unlabeled training instances.

Graph Construction

Given a set of n instances (tweets in our case), a typical approach is to construct the graph based on relational knowledge source (e.g., citation links in (Lu and Getoor 2003)) or distance between instances (Zhu 2005). However, developing such a relational knowledge is not feasible for every problem. On the other hand, computing distance between $n(n-1)/2$ pairs of instances to construct the graph is also very expensive (Muja and Lowe 2014). Hence, we choose to use k-nearest neighbor-based approach for finding nearest neighbors of instances as it has been shown to be an effective approach in other studies (Dong, Moses, and Li 2011; Jebara, Wang, and Chang 2009). The nearest neighbor graph consists of n vertices and for each vertex, there is an edge set consisting of a subset of n instances. The edge is defined by the distance measure $d(i, j)$ between tweets $\mathbf{t}_i$ and $\mathbf{t}_j$, where the value of d represents how similar the two tweets are. Similar similarity-based graph has shown impressive results in learning sentence representations (Saha et al. 2017). To find the nearest instances efficiently, we used k-d tree data structure (Witten et al. 2016). The rationale of this approach is that if $\mathbf{t}_i$ is very far from $\mathbf{t}_j$ and $\mathbf{t}_k$ is close to $\mathbf{t}_j$ then without computing the distance between $\mathbf{t}_i$ and $\mathbf{t}_k$ we can infer they are far. For our graph construction, we first represent each tweet by averaging the word embedding vectors (see Sec. Crisis Word Embedding for details) of its words, and then we measure $d(i, j)$ by computing the *Euclidean* distance between the vectors. We have chosen to use *Euclidean* distance to reduce computational complexity. The number of nearest neighbor k was set to 10.

Semi-supervised Neural Network Model

Figure 1 shows the architecture of our neural network model. The input to the network is a tweet $\mathbf{t} = (w_1, \ldots, w_n)$ con-

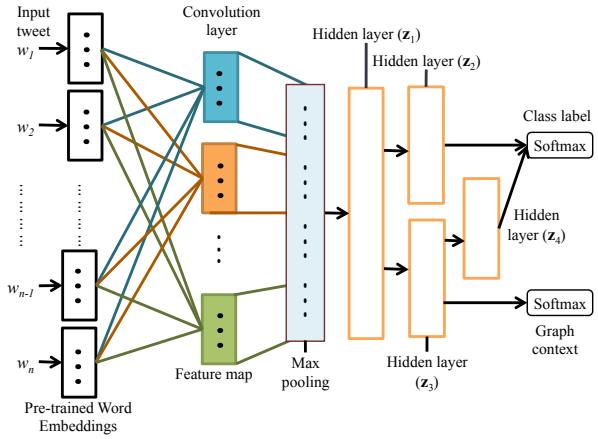

Figure 1: The architecture of the graph-based semi-supervised learning with CNN.

taining words each coming from a finite vocabulary $\mathcal{V}$. The first layer of our network maps each of these words into a distributed representation $\mathbb{R}^d$ by looking up a shared embedding matrix $E \in \mathbb{R}^{|\mathcal{V}| \times d}$. We initialized E using pretrained word vectors (more in Sec. Crisis Word Embedding). The output of the look-up layer is a matrix $X \in \mathbb{R}^{n \times d}$, which is passed through a number of convolution and pooling layers to learn higher-level feature representations.

A convolution operation applies a *filter* $\mathbf{u} \in \mathbb{R}^{k.d}$ to a window of k vectors to produce a new feature $h_t = f(\mathbf{u}.X_{t:t+k-1})$, where $X_{t:t+k-1}$ is the concatenation of k look-up vectors, and f is a nonlinear activation; we use rectified linear units (ReLU). We apply this filter to each possible k-length windows in X to generate a *feature map*, $\mathbf{h}^j = [h_1, \ldots, h_{n+k-1}]$. This process is repeated N times with N different filters to get N different feature maps. We then apply a *max-pooling* operation, $\mathbf{m} = [\mu_p(\mathbf{h}^1), \cdots, \mu_p(\mathbf{h}^N)]$, where $\mu_p(\mathbf{h}^j)$ refers to the max operation applied to each window of p features in the feature map $\mathbf{h}^i$. Intuitively, the filters compose local features into higher-level representations in the feature maps, and max-pooling extracts the most important aspects from each feature map while reducing the output dimensionality. The pooled features are passed through two fully-connected hidden layers, $\mathbf{z}_1 = f(V_1\mathbf{m})$; $\mathbf{z}_2 = f(V_2\mathbf{z}_1)$, where V_1 and V_2 are the associated weight matrices. The final activations are used for classification using a Softmax in the output layer; the formal definition of Softmax is defined below.

To leverage the unlabeled data $\mathcal{D}_u$, and to exploit the relations between training instances (labeled or unlabeled) encoded in the graph G we add a branch to the network. It takes $\mathbf{z}_1$ as input and learns internal representations by predicting a node in the graph context of the input tweet. Following (Yang, Cohen, and Salakhutdinov 2016), we use *negative sampling* to compute the loss for predicting the context node, and we sample two types of contextual nodes: one is based on the graph G to encode the structural information and the second is based on labels to incorporate label in-

formation through this branch of the network. The ratio of positive and negative samples is controlled by a random variable $\rho_1 \in (0, 1)$, and the proportion of the two context types is controlled by another random variable $\rho_2 \in (0, 1)$; see Alg. 1 of (Yang, Cohen, and Salakhutdinov 2016) for details of the sampling procedure. Let (i, j, γ) is a tuple sampled from the distribution $p(i, j, \gamma|\mathcal{D}_l, \mathcal{D}_u, G)$, where j is a context node of an input node i and $\gamma \in \{+1, -1\}$ denotes whether it is a positive or a negative sample; $\gamma = +1$ if $\mathbf{t}_i$ and $\mathbf{t}_j$ are neighbors in the graph (for graph-based context) or they both have same labels (for label-based context), otherwise $\gamma = -1$. The loss for context prediction can be written as $\mathcal{L}_c(\theta) = \mathbb{E}_{(i,j,\gamma)} \log \sigma \left(\gamma \mathbf{w}_j^T \mathbf{z}_3(i)\right)$, where $\mathbf{z}_3(i) = f(V_3\mathbf{z}_1(i))$ defines another fully-connected hidden layer (marked as *Hidden layer (*$\mathbf{z}_3$*)* in Fig. 1) having weights V_3, and $\mathbf{w}_j$ is the weight vector associated with the context node $\mathbf{t}_j$.

For the classification, we take $\mathbf{z}_3$ and pass it through another fully-connected hidden layer, $\mathbf{z}_4 = f(V_4\mathbf{z}_3)$, where V_4 is the corresponding weight matrix. Finally, the Softmax output layer does the classification $p(y = k|\mathbf{t}, \theta) = \exp\left(\mathbf{w}_k^T [\mathbf{z}_2; \mathbf{z}_4]\right) / \sum_{k'} \exp\left(\mathbf{w}_{k'}^T [\mathbf{z}_2; \mathbf{z}_4]\right)$, where $[.;.]$ denotes concatenation of two column vectors, and $\mathbf{w}_k$ are the class weights. The overall loss of the network can be written as $\mathcal{L}(\theta) = -\frac{1}{L} \sum_{i=1}^{L} \log p(y_i|\mathbf{t}_i, \theta) - \lambda \mathcal{L}_c(\theta)$, where the first part is the classification loss and the second part is the context loss, and the hyperparameter λ controls the relative strength of the two parts.

Crisis Word Embedding

We initialize the embedding matrix E in our network with pretrained word embeddings. We trained a continuous bag-of-words (CBOW) wrod2vec (Mikolov et al. 2013) model on a large crisis dataset with vector dimensions of 300, a context window size of 5 and $k = 5$ negative samples. The crisis dataset consists of different collections of tweets collected automatically using the AIDR system (Imran et al. 2014). In the preprocessing, we lowercased the tweets and removed URLs, digit, time patterns, special characters, single character, user name started with the @ symbol. The resulting dataset has about 364 million tweets and about 3 billion words. When training CBOW, we filtered out words with a frequency less than or equal to 5. The resulting trained word-embedding model contains about 2 million words.

Experiments

In this section, we first describe the datasets we used in our experiments, then the experimental setting, and finally the results.

Datasets

For the evaluation, we use two real-world Twitter datasets collected during the *2015 Nepal earthquake* and the *2013 Queensland floods*. These datasets are collected through the Twitter streaming API[1] using event-specific keywords/hashtags. To obtain the labeled examples for our task which consists of two classes *relevant* and *irrelevant*, we employed paid workers from the Crowdflower[2] – a crowdsourcing platform. We randomly sampled 11,668 and 10,033 tweets from the Nepal earthquake and the Queensland floods respectively. Given a tweet, we asked crowdsourcing workers to assign the *"relevant"* label if the tweet conveys/reports information useful for crisis response such as a report of injured or dead people, some kind of infrastructure damage, urgent needs of affected people, donations requests or offers, otherwise assign the *"irrelevant"* label.

Table 1: Distribution of the labeled datasets

Data	Relevant	Irrelevant	Train	Dev	Test
Nepal	5,527	6,141	7,000	1,166	3,502
Queensland	5,414	4,619	6,019	1,003	3,011

For evaluation, we split the datasets into 60% as training, 30% as test and 10% as development. Table 1 shows the resulting datasets.

Experimental Settings

As a part of the preprocessing of the dataset, we used the same approach that we used to train the word2vec model (see Sec. Crisis Word Embedding). We use the validation set to optimize the hyperparameters.

For our semi-supervised setting, one of the main goals was to understand how much labeled data is sufficient to obtain a reasonable result. Therefore, we experimented our system considering the smallest to all instances, such as 100, 500, 2000, 5000 and all instances. Such an understanding can help us to design the model at the onset of a crisis event with sufficient amount of labeled data. To demonstrate that the semi-supervised approach outperforms the supervised baseline, we run supervised experiments using the same number of labeled instances. In the supervised setting, only $\mathbf{z}_2$ activations in Fig. 1 are used for classification.

We trained the models using the adadelta (Zeiler 2012) algorithm. The learning rate was set to 0.1 when optimizing on the classification loss and to 0.001 when optimizing on the context loss. The maximum number of epochs was set to 200, and dropout (Srivastava et al. 2014) rate of 0.02 was used to avoid overfitting. We did *early stopping* based on the f-measure on the validation set with a patience of 25.

We used 100, 150 and 200 filters each having the window size of 2, 3 and 4, respectively, and pooling length of 2, 3 and 4 respectively. The value of λ was set to 1.0 in the semi-supervised model. We did not tune any hyperparameter (e.g., the size of hidden layers, filter size, dropout rate) in any experimental setting since the goal was to have an end-to-end comparison for the same hyperparameter setting and understand whether CNN with graph based semi-supervised approach can outperform or not. We did not filter out any vocabulary item for any of the settings. We also did not fine-tune the word embeddings on the classification task.

Results and Discussion

In Table 2, we present the classification results for different experimental settings. We computed the performances using

[1] https://dev.twitter.com/streaming/overview

[2] http://crowdflower.com

Table 2: F-measure for different experimental settings. *L* refers to labeled data, *U* refers to unlabeled data, *All L* refers to all labeled instances for that particular dataset.

Dataset	L/U	100	500	1000	2000	All L
Nepal Earthquake	L	47.11	52.63	55.95	58.26	60.89
	L+U(50k)	52.32	59.95	61.89	64.05	66.63
Queensland Flood	L	58.52	60.14	62.22	73.92	78.92
	L+U(∼21k)	75.08	85.54	89.08	91.54	93.54

weighted averaged precision, recall and F-measure. In the table, we only report the F-measure for simplicity. The rational behind choosing the weighted metric is that it takes into account the class imbalance problem. From the table, we see that as we increase the number of labeled examples (**L**), the classification performance improves – from 47.1 to 60.9 for Earthquake and from 58.5 to 78.9 for Flood, which is a common trend for supervised models.

In this study, for computational efficiency, we limit the number of unlabeled instances **U** in our semi-supervised model to $50K$ for Nepal and $\sim$21K for Queensland. We can observe that as we include unlabeled instances with labeled instances, performance significantly improves in each experimental setting giving 5% to 26% absolute improvements over the supervised models. These improvements demonstrate the effectiveness of our approach. We also notice that our semi-supervised approach can perform above 90% depending on the event. Recall that we did not tune the hyperparameters of our supervised and semi-supervised models. Therefore, these results may not be optimal. We believe that upon optimizing hyperparameters, the overall performances of our system can be further improved. From the results, we can ascertain that 500 labeled instances with unlabelled instances could be a reasonable choice at the early onset of a crisis event to design the semi-supervised model.

Conclusions

We presented a graph-based semi-supervised deep learning framework based on a CNN. The network combines a loss for predicting the class labels with a loss for predicting the context defined by a similarity graph. We constructed the similarity graph using a k-nearest neighbor approach that exploits distributed representations of tweets. Our evaluation on two crisis-related tweet datasets demonstrates significant improvements for our semi-supervised model over a supervised only baseline. There are several interesting future research directions of this work such as exploring domain adaptation and zero- or one-shot learning.

References

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

Dong, W.; Moses, C.; and Li, K. 2011. Efficient k-nearest neighbor graph construction for generic similarity measures. In *Proc. of World wide web*, 577–586. ACM.

Grover, A., and Leskovec, J. 2016. node2vec: Scalable feature learning for networks. In *Proc. of the 22nd ACM Conference on KDD*, 855–864.

Imran, M.; Castillo, C.; Lucas, J.; Meier, P.; and Vieweg, S. 2014. AIDR: Artificial intelligence for disaster response. In *ACM International Conference on World Wide Web*, 159–162.

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

Jebara, T.; Wang, J.; and Chang, S.-F. 2009. Graph construction and b-matching for semi-supervised learning. In *Proc. of Machine Learning*, 441–448. ACM.

Johnson, R., and Zhang, T. 2015. Semi-supervised convolutional neural networks for text categorization via region embedding. In *Advances in neural information processing systems*, 919–927.

Lu, Q., and Getoor, L. 2003. Link-based classification. In *ICML*.

Mihalcea, R. 2004. Co-training and self-training for word sense disambiguation. In *CoNLL*, 33–40.

Mikolov, T.; Chen, K.; Corrado, G.; and Dean, J. 2013. Efficient estimation of word representations in vector space. In *ICLR*.

Mitchell, T. 1999. The role of unlabeled data in supervised learning. In *Proceedings of the sixth international colloquium on cognitive science*, 2–11. Citeseer.

Miyato, T.; Dai, A. M.; and Goodfellow, I. 2016. Adversarial training methods for semi-supervised text classification. *arXiv preprint arXiv:1605.07725*.

Muja, M., and Lowe, D. G. 2014. Scalable nearest neighbor algorithms for high dimensional data. *IEEE Transactions on PAMI* 36(11):2227–2240.

Nguyen, D.; Mannai, K.; Joty, S.; Sajjad, H.; Imran, M.; and Mitra, P. 2017. Robust classification of crisis-related data on social networks using convolutional neural networks. In *ICWSM*, 632–635.

Nigam, K.; McCallum, A. K.; Thrun, S.; and Mitchell, T. 2000. Text classification from labeled and unlabeled documents using em. *Machine learning* 39(2):103–134.

Saha, T.; Joty, S.; Hassan, N.; and Hasan, M. 2017. Regularized and retrofitted models for learning sentence representation with context. In *CIKM*, 547–556. Singapore: ACM.

Srivastava, N.; Hinton, G. E.; Krizhevsky, A.; Sutskever, I.; and Salakhutdinov, R. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of MLR* 15(1):1929–1958.

Subramanya, A., and Talukdar, P. P. 2014. Graph-based semi-supervised learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning* 8(4):1–125.

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

Vieweg, S.; Castillo, C.; and Imran, M. 2014. Integrating social media communications into the rapid assessment of sudden onset disasters. In *International Conference on Social Informatics*, 444–461. Springer.

Witten, I. H.; Frank, E.; Hall, M. A.; and Pal, C. J. 2016. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann.

Yang, Z.; Cohen, W.; and Salakhutdinov, R. 2016. Revisiting semi-supervised learning with graph embeddings. *arXiv preprint arXiv:1603.08861*.

Zeiler, M. D. 2012. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

Zhu, X. 2005. Semi-supervised learning literature survey. Technical Report 1530, Computer Sciences, University of Wisconsin-Madison.