Towards Practical Usage of a Domain Adaptation Algorithm in the Early Hours of a Disaster

Hongmin Li
Kansas State University
hongminli@ksu.edu

Doina Caragea
Kansas State University
dcaragea@ksu.edu

Cornelia Caragea
University of North Texas
ccaragea@unt.edu

ABSTRACT

Many machine learning techniques have been proposed to reduce the information overload in social media data during an emergency situation. Among such techniques, domain adaptation approaches present greater potential as compared to supervised algorithms because they don't require labeled data from the current disaster for training. However, the use of domain adaptation approaches in practice is sporadic at best. One reason is that domain adaptation algorithms have parameters that need to be tuned using labeled data from the target disaster, which is presumably not available. To address this limitation, we perform a study on one domain adaptation approach with the goal of understanding how much source data is needed to obtain good performance in a practical situation, and what parameter values of the approach give overall good performance. The results of our study provide useful insights into the practical application of domain adaptation algorithms in real crisis situations.

Keywords

Twitter, Domain adaptation, Disaster, Classification

INTRODUCTION

With the prevalent usage of social media, we often see newspapers or TV stations citing texts and images posted on Twitter by eye-witnesses of emergency events. Such first-hand information during emergencies has the potential to also help emergency teams to improve situation awareness and response (Castillo 2016; Hughes et al. 2014; Reuter et al. 2015; Starbird et al. 2010; Landwehr and Carley 2014; Palen, Vieweg, et al. 2011). In practice, however, large-scale disaster response organizations haven't extensively adopted the use of such social media data (Tapia and Moore 2014). A survey of emergency managers in U.S.A. at county level (Plotnick et al. 2015) and surveys of emergency staff in some European countries (Reuter et al. 2015) show that although the responding officers see social media as being useful, many challenges still impede the extensive adoption of social media data in operations. These challenges include both management challenges such as lack of guidance, lack of staff, and technical challenges such as information overload, and reliability or trustworthiness of the information source.

Many recent research works have used machine learning approaches on social media data gathered during emergency events. Such works have suggested that automated approaches based on machine learning can enable management and response organizations sift through large amounts of information, and prioritize the information to be carefully analyzed based on relevance, reliability, trustworthiness, etc. (Imran, Elbassuoni, et al. 2013; Imran, Mitra, et al. 2016; C. Caragea, Squicciarini, et al. 2014; C. Caragea, Silvescu, et al. 2016; Li, Guevara, et al. 2015; Imran, Castillo, et al. 2015). However, machine learning algorithms cannot be easily deployed and used in emergency situations due to different challenges, depending on the algorithms.

One big challenge for supervised learning algorithms is that such algorithms require labeled data from a current disaster of interest, i.e. tweets labeled as relevant to the disaster or not-relevant, tweets labeled as relevant and

informative versus tweets labeled as relevant but not-informative, etc. Unfortunately, it is not realistic to assume that labeled data for a current disaster is readily available, as trustworthy labels require significant time and effort. However, it is reasonable to assume that labeled data is available for a prior disaster, called "source", and several works have focused on learning supervised classifiers for the "target" disaster based on a "source" disaster (Imran, Elbassuoni, et al. 2013; Imran, Mitra, et al. 2016). One drawback of this approach is that the classifiers learned from a prior source disaster may not generalize well to the target disaster (Imran, Elbassuoni, et al. 2013; Verma et al. 2011), as the target disaster may have unique characteristics in terms of its nature, location, etc. and may also cause different social media response (Palen and Anderson 2016).

Domain adaptation algorithms (Li, Guevara, et al. 2015) that make use of target unlabeled data in addition to source labeled data represent a good alternative to supervised classifiers learned from labeled source data only, given that unlabeled data from the target disaster accumulates quickly. The algorithm proposed in (Li, Guevara, et al. 2015) is an iterative algorithm based on Expectation Maximization (EM). A variant of this algorithm, based on the idea of self-training, has been introduced in our prior work (Li, D. Caragea, et al. 2017). Our prior experimental results have shown that the self-training variant gives results comparable, and sometimes better, than the EM variant. More importantly, both variants result in classifiers that are significantly better than the supervised classifiers learned from labeled source data only. Furthermore, the self-training variant is more appropriate for scenarios where more unlabeled target data may become available during the algorithms' later iterations. Such scenarios better reflect the reality, making the self-training approach more desirable when attempting to use domain adaptation algorithms in practice during an emergent disaster.

However, there are other challenges that prevent the self-training domain adaptation approach from being deployed in a real situation. First, to be able to successfully deploy this approach, it is important to understand how much source labeled data the algorithm needs, as too little labeled data may result in inaccurate classifiers, while too much labeled data may not be worth the effort of labeling it. Thus, our first objective is to study the variation of the performance of this domain adaptation algorithm with the amount of labeled source data. Second, the algorithm has two parameters that need to be tuned, a parameter that controls the weights assigned to the source and target data during training, and a parameter that controls the number of target instanced added to the classifier at each iteration. Parameter tuning is not a challenge specific to domain adaptation. In fact, many supervised learning algorithms require tuning for best performance, a process that can be time consuming. But in addition to being time consuming, in a domain adaptation scenario, parameter tuning requires labeled data from the target disaster. Under the assumption that labeled data is not available for the target disaster, parameter tuning becomes impractical.

To address these issues, we use a relatively large number of (source, target) pairs and study the performance of the self-training domain adaptation algorithm under different amounts of labeled source data, with the goal of identifying the size of source data that the algorithm needs. We then study how the performance changes when varying the parameters of the algorithm. The objective is to identify parameter values that work well for a large number of (source, target) pairs, and use those values as default values when deploying a domain adaptation in practice, thus avoiding the need for parameter tuning in a real scenario.

In addition, we study the performance of the algorithm with the number of iterations. As mentioned earlier, the algorithm terminates when there is no change in the labels of the current instances in the target data, in between two consecutive iterations. We can think of this condition as a pseudo-convergence condition, in the sense that if the labels don't change in between two consecutive iterations, the classifiers will presumably not change much if the most confidently labeled instances at the last iteration are still added to the training data. However, in practice, the addition of the most confidently labeled instances can result in a classifier with performance different from the performance of the classifier at pseudo-convergence. The objective is to understand if the algorithm should terminate when the pseudo-convergence condition is met, or if it should run for a certain number of iterations (dependent on the number of target unlabeled instances to be added to the training data).

We summarize our main contributions as follows:

- We evaluate the performance of the domain adaptation algorithm with self-training on a relatively large set of (source, target) pairs, when using different amounts of labeled source data. The goal is to identify a good trade-off between the amount of labeled source data necessary for good performance and the effort to label that data.
- We evaluate the performance of this algorithm when varying its parameters. The goal is to identify parameter values that give good results, in general, and use those values as default values in a realistic scenario, where tuning is impractical.

- We also evaluate the performance of the algorithm when varying the number of iterations. The goal is to identify the number of iterations that leads to the best performance, in general, and use that number as a default value in a realistic scenario.

METHOD

We focus on the task of classifying tweets posted during a disaster as related to the disaster, or *on-topic*, and not related to the disaster, or *off-topic*. This is one of the most important classification tasks as many other tasks during a disaster are performed on tweets relevant to a disaster. Furthermore, this is a task where supervised classifiers learned from source only do not generalize well (Li, Guevara, et al. 2015). While any algorithm, including Naïve Bayes, Random Forest (RF), Support Vector Machines (SVM), Logistic Regression (RF), etc. can be used with domain adaptation, we choose to use the Naïve Bayes algorithm as it doesn't have any parameters that need to be tuned. Furthermore, our prior work (Li, D. Caragea, et al. 2017) has shown that for our classification task, Naïve Bayes gives better results than the RF, SVM, LR algorithms with default parameters (without tuning).

We implemented a version of self-training domain adaptation based on (Li, Guevara, et al. 2015; Herndon and D. Caragea 2015). The self-training domain adaptation builds a weighted Naïve Bayes Bernoulli (Manning et al. 2008) classifier, which linearly combines source and target data in an iterative fashion, to simultaneously estimate the prior $P(c_i)$ and the likelihood $P(w_j|c_i)$, as follows:

$$P(c_i) = (1-\gamma)P_{SL}(c_i) + \gamma P_{TU}(c_i) \tag{1}$$

$$P(w_j|c_i) = (1-\gamma)P_{SL}(w_j|c_i) + \gamma P_{TU}(w_j|c_i) \tag{2}$$

In the equations above, c_i represents a class label, w_j is a feature in the feature set or vocabulary V, the probability subscript (SL or TU) denotes the type of data used to estimate that probability, i.e. SL denotes source labeled data, TU denotes target unlabeled data, and δ is a parameter that controls how fast we shift the weight from source to target data. This parameter is defined as $\gamma = \min(t * \delta, 1)$, where $t = \{0, 1, 2, \cdots\}$ is the iteration number. Initially, $t = 0$, $\gamma = 0$, which means that only source labeled data is used. Then, according to the Bayes Theorem, we estimate the posterior class label c_i for a new instance d as:

$$P(c_i|d) \propto P(c_i) \prod_{j \in |V|} P(w_j|c_i) \tag{3}$$

At each iteration, the current classifier (originally trained only from source labeled data) is used to classify the remaining target unlabeled data (originally all the target unlabeled data). The most confidently classified unlabeled instances (e.g., top k instances in each class) are moved to the training set, with hard (e.g., 0/1) labels, to be used in subsequent iterations. By default, the algorithm runs until convergence, where "convergence" means that the labels of the remaining target unlabeled instances don't change in between two consecutive iterations (Yarowsky 1995). The domain adaptation approach with self-training is summarized in Algorithm 1 below.

So the algorithm has two parameters that need to be tuned, a parameter δ which defines how fast we shift the weight from source to target data, and another parameter k which defines how many instances of each class to add at each iteration while training.

DATASET

CrisisLexT6 dataset is a collection of English tweets from six disasters, published by (Olteanu et al. 2014). This dataset is collected through Twitter API based on keywords and geo-locations of affected areas. The six disasters all occurred between October 2012 and July 2013 in USA, Canada and Australia. There are approximately 10,000 tweets for each disaster, all manually labeled as on-topic or off-topic through the crowdsourcing platform Crowdflower. We used the same pre-processing as in (Li, Guevara, et al. 2015) to clean the tweets, including removing non-printable ASCII characters; replacing URLs, email addresses, and usernames with an URL/EMAIL/USERNAME placeholder, removing RT (i.e., re-tweet) and duplicate tweets etc. Furthermore, we represented both source disaster tweets and target disaster tweets as feature vectors (using the bag-of-words 0/1 representation), where the features are words that appear in the target disaster with frequency as least 10. Thus, the vocabulary size is different for different target disasters, but generally each vocabulary consists of around 1000 features for the source/disaster pairs that we chose to experiment with. We do not perform additional feature selection as we assume that target labeled data is not available. The statistics for the final dataset are shown in Table 1, sorted based on the time when each disaster happened.

Algorithm 1: Naïve Bayes Domain Adaptation algorithm with self-training

1. Given: Target unlabeled data TU, source labeled data SL and target test data TT
2. Initialize $TU_{hard} = \phi$ and $TU_{left} = TU$, where TU_{hard} is the set of instances with hard labels assigned by self-training, and TU_{left} is the set of unlabeled instances left in TU
3. Use SL and TU_{hard} to simultaneously compute the prior and likelihood with Equations 1 and 2, respectively. At the first iteration, only SL is used
4. Compute the posterior class probability of target instances still unlabeled, TU_{left}, with Equation 3
5. Select the k most confidently labeled instances from each class c_i based on probability ranking, move them to TU_{hard} (to use for training in the next iteration) and remove them from TU_{left}
6. **while** *(labels assigned to instances in TU_{left} change) or (maximum number of iteration not reached)* **do**

   M-step: Simultaneously compute the prior and likelihood using Equations 1 and 2, respectively, using a combination of SL and TU_{hard} weighted based on $\gamma = \min(t * \delta, 1)$, where t is the iteration number

   E-step: Compute the posterior class probability of target instances still unlabeled, TU_{left}, with Equation 3 and select the k most confidently labeled instances as in Step 4

   end
7. Use the final classifier to predict the labels of the target test instances TT

Table 1. Number of instances for each disaster in the dataset, before and after cleaning

	Before Cleaning			After Cleaning		
Crisis	On-topic	Off-topic	Total	On-topic	Off-topic	Total
2012_Sandy_Hurricane	6138	3870	10008	5261	3752	9013
2013_Queensland_Floods	5414	4619	10033	3236	4550	7786
2013_Boston_Bombings	5648	4364	10012	4441	4309	8750
2013_West_Texas_Explosion	5246	4760	10006	4123	4733	8856
2013_Oklahoma_Tornado	4827	5165	9992	3209	5049	8258
2013_Alberta_Floods	5189	4842	10031	3497	4714	8211

EXPERIMENTAL SETUP

Our experimental setup is designed to help us understand the use of the domain adaptation algorithm from a practical point of view. First, we want to understand how many source instances are needed to learn an accurate classifier for a target disaster in a domain adaptation setting. Second, we aim to understand how the performance of the domain adaptation classifier varies with parameters δ and k values, and to select good overall values to use in practice. Third, we aim to study how the performance varies with different numbers of iterations, and to identify an appropriate number of iterations for good performance. More specifically, we ask the following questions:

- *How many source labeled instances are needed to build an accurate classifier for the target?*
- *What values should we use in practice for the parameters δ and k of the domain adaptation algorithms?*
- *How many iterations are needed to build an accurate classifier for the target?*

To answer these questions, we follow the timeline of the six disasters in the dataset and select a variety of disaster pairs to perform experiments. Except for Hurricane Sandy, which does not have a prior disaster in this dataset, all the other disasters are used as target disasters in one or more pairs. We end up with 11 pairs, which cover natural disaster pairs, man-made disaster pairs, and also natural and man-made disaster pairs. When reporting the results, we arrange pairs having the same target disaster but different source disasters together, and represent each pair with its initials of the source and target disasters, as shown in Table 2.

Table 2. Source-Target pairs of disasters used in the experiments

Pair	Source Disaster	Target Disaster
SH -> QF	2012_Sandy_Hurricane	2013_Queensland_Floods
SH -> BB	2012_Sandy_Hurricane	2013_Boston_Bombings
QF -> BB	2013_Queensland_Floods	2013_Boston_Bombings
SH -> WTE	2012_Sandy_Hurricane	2013_West_Texas_Explosion
BB -> WTE	2013_Boston_Bombings	2013_West_Texas_Explosion
SH -> OT	2012_Sandy_Hurricane	2013_Oklahoma_Tornado
QF -> OT	2013_Queensland_Floods	2013_Oklahoma_Tornado
BB -> OT	2013_Boston_Bombings	2013_Oklahoma_Tornado
SH -> AF	2012_Sandy_Hurricane	2013_Alberta_Floods
QF -> AF	2013_Queensland_Floods	2013_Alberta_Floods
BB -> AF	2013_Boston_Bombings	2013_Alberta_Floods

Training and test data: For each pair, we use 5-fold cross-validation to select target unlabeled data and target test data. Specially, we split the target data into 5 folds; at each rotation, one fold is used as target test data (TT), and three folds are used as target unlabeled data (TU) to be used in domain adaptation together with source labeled data (SL). The last fold is reserved for potential usage as target labeled data in future work. To choose different amounts of source labeled data for each pair, we randomly select 250, 500, 1000, 2000 instances from each class (on-topic/off-topic), and then finally include all instances from each c lass. Thus, we end up with SL-500, SL-1000, SL-2000, SL-4000 and SL-ALL number of source instances, respectively. The number of all instances for each disaster is around 8000 after cleaning, except for Hurricane Sandy whose number is around 9000.

Parameter tuning: To report best performance for a pair, we tune parameters δ and k during a validation step. We randomly select one of the three target unlabeled (TU) folds as validation data (TTV), and use the other two folds of TU as target unlabeled data for validation (TUV). We use SL+TUV for training and TTV and select the best values for the parameters based on TTV. After tuning, the whole TU is used to learn the final classifier for the target, and performance is estimated using the target test set TT. The values used for δ are: {0.001, 0.01, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9}, and the values used for k are {1,5,10}.

Experiments: We perform several experiments on each pair of disasters. The first experiment is domain adaptation with self-training running until convergence, with parameter values tuned based on the validation data. We refer to this experiment as NB-STT-Conv. The second experiment, NB-STF-Conv, is similar to the first one, except that the algorithm runs with fixed values for parameters δ and k . We compare the results of these experiments to understand how much is lost, if anything, by fixing parameters.

To see whether the performance can still improve after convergence, we also vary the number of iterations for fixed δ and k parameters, beyond convergence. The following value are considered for the number of iterations: {10, 50,100,150,200,250,300}, and the best number of iterations, among those considered, is identified. We refer to the experiment where the algorithm runs with fixed parameters δ and k, and fixed number of iterations as NB-STF-Iter. The results of all experiments are reported in terms of the area under the ROC curve (auROC), but other measures (e.g., accuracy) show similar trends.

EXPERIMENTAL RESULTS AND DISCUSSION

To answer our research questions, we first run the algorithm using source datasets of different sizes and tune parameters. We then analyze how the performance varies with different values for parameters δ and k, when running until convergence. We identify the general best values for those parameters. Then, using the selected δ and k values, we study the performance of the algorithm when increasing the number of iterations, and identify a good number of iterations to use instead of convergence. We answer the research questions in the following discussion.

How many source labeled instances are needed to build an accurate classifier for the target?

Table 3 shows the variation of the performance with the size of the source dataset. We performed column-wise paired t-tests $p \leq 0.05$) to compare the results in a row and identify values that are significantly better than their counterparts. The best values for each row/pair are shown in bold. As can be seen, the performance generally increases with the amount of labeled source data. However, between using 4000 instances and using all instances

Table 3. Variation of the performance of the domain adaptation algorithm with the size of the source dataset. The supervised Naïve Bayes classifier learned from all source data is used as a baseline. Performance is reported as weighted auROC values obtained using parameter tuning (averaged over 5 folds). The algorithm terminates upon convergence (NB-STT-Conv). The best value in each row is shown in bold (based on a t-test with $p \leq 0.05$).

Pair	Baseline	SL-500	SL-1000	SL-2000	SL-4000	**SL-All**
SH -> QF	0.911	0.945	0.955	0.964	0.971	**0.974**
SH -> BB	0.753	0.877	0.914	0.926	0.935	**0.941**
QF -> BB	0.820	0.840	0.863	0.866	**0.890**	**0.890**
SH -> WT	0.853	0.973	0.972	0.976	**0.986**	**0.987**
BB -> WT	0.983	0.969	0.972	0.972	**0.980**	**0.984**
SH -> OT	0.865	0.921	0.932	0.944	**0.953**	**0.951**
QF -> OT	0.880	**0.916**	**0.921**	**0.919**	**0.926**	**0.924**
BB -> OT	0.905	0.919	0.924	0.927	**0.942**	**0.944**
SH -> AF	0.830	0.925	0.942	0.956	**0.970**	**0.972**
QF -> AF	0.860	0.880	0.890	0.892	0.918	**0.922**
BB -> AF	0.806	0.898	0.912	0.935	**0.950**	**0.950**

(approximately 8000), the performance does not increase much. In fact, for 7 out of 11 pairs, the results obtained with 4000 instances are as good as the results obtained with all source instances, suggesting that the effort that goes into data labeling is not worth beyond 4000 instances. Furthermore, we can also see that as few as 500 source instances can produce classifiers with performance close to 90%, and most of the time better than the performance of the supervised classifiers learned from all the source data. Therefore, if labeling 4000 instances is not possible, the domain adaptation algorithm can still help as compared to the supervised learning algorithm.

What values should we use in practice for the parameters δ and k of the domain adaptation algorithms?

Figures 1 and 2 show the variation of performance with parameters δ and k, respectively. In each figure, the variation of performance when 500 source instances are used is shown on the left, whereas variation when 4000 source instances are used is shown on the right. We focus on 500 and 4000 instances, respectively, to understand if the best values for parameters are different for smaller versus larger source datasets. In all cases, the algorithm is run to convergence.

Subplots (a), (b), (c) in Figure 1, show the variation of the performance with δ when 500 source instances are used, and k is fixed to 1, 5, 10, respectively. Similarly, subplots (d), (e), (f) in Figure 1, show the variation of the performance with δ when 4000 source instances are used, and k is fixed to 1, 5, 10, respectively. We can see that when we have 4000 source instances, the best results are generally obtained for a very small value of δ, specifically 0.001, regardless of the value used for k . This result suggests that a source with 4000 instances produces a reasonably good classifier in the first place, and therefore the shift from the source to the target should be done slowly to allow for the accumulation of accurately labeled target instances in the training set. Another interesting observation is that for values of δ greater than 0.1, the performance does not change much. This is because when δ is large, the algorithm shifts all the weight to the target data in a small number of iterations. For example, when $\delta = 0.2$, the weight assigned to the target will be 1.0 at the sixth iteration ($\min(5 * 0.2, 1) = 1$), and thus the classifier solely depends on the self-training of the target unlabeled instances added in the first 5 iterations, which will lead the algorithm to converge very fast.

When 500 source instances are used, the best results are also obtained with small values of δ, but the best value is not consistent as 0.001. In some cases, the best δ value is 0.01 or 0.1, which shows that the shift from source to target happens faster when the original classifier learned from source is not very good. In effect, a higher weight will be assigned to the target data, which is still small in size and possibly not very accurate. Given that, from a practical point of view, it is desirable to have a larger amount of source data (approximately 4000), as that makes it easier to find good overall values for the parameter δ.

Subplots (a), (b), (c) in Figure 2 show the variation of the performance with k when 500 source instances are used, and δ is fixed to 0.001, 0.01, 0.1, respectively. Similarly, subplots (d), (e), (f) in Figure 2 show the variation of the performance with δ when 4000 source instances are used, and δ is fixed to 0.001, 0.01, 0.1, respectively. By analyzing these plots, we can see that when δ is very small, for example 0.001, the performance is more steady, generally increasing very slowly with k, with some exceptions (e.g., SH -> AF in Figure 2 (a)). In particular, for

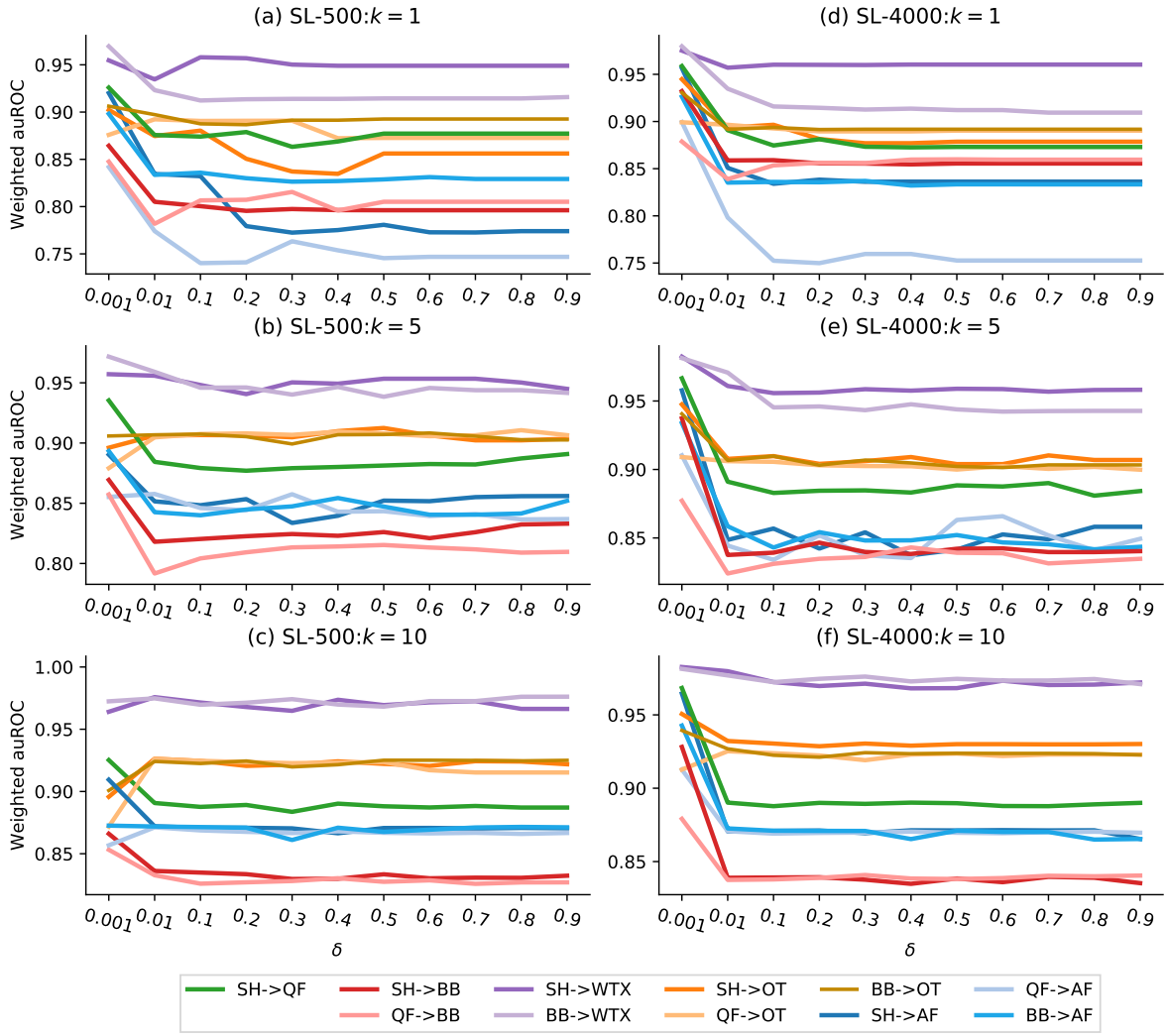

Figure 1. Variation of the performance (auROC) with δ for two sizes of the source dataset: SL-500 (left) and SL-4000 (right). The parameter k is fixed to 1, 5, and 10, respectively. The algorithm terminates upon convergence.

4000 source instances and $\delta = 0.001$, the best performance is observed for either $k = 5$ or $k = 10$. The performance decrement from $k = 5$ to $k = 10$ that is observed for some pairs can be explained by the addition of mislabeled target instances to the training data, which can easily happen when too many target instances are added at once. When δ is 0.01 or 0.1, the increase/decrease pattern is less consistent overall, although for many pairs larger k is better.

Figures 1 and 2 together suggest that the best performance overall is obtained when the source data consists of approximately 4000 instances, parameter δ is set to 0.001 and parameter k is set to 5 or 10. Furthermore, even when only 500 source instance are available, the same parameters can be used.

To understand if performance is sacrificed when fixing parameters as opposed to tuning them, we compare the two settings when the algorithm runs to convergence. The results are shown in Table 4. As can be seen, the results obtained using fixed parameter values are as good as the results obtained using tuned values for most pairs, with only two exceptions for ST-4000 (QF -> BB, QF->OT) and two exceptions for ST-500 (QF->OT and SH -> WTE), where the performance with fixed values is just slightly worse than the performance with tuned values.

How many iterations are needed to build an accurate classifier for the target?

Using the findings about best overall parameters, our next objective is to compare the performance when the algorithm terminates upon convergence versus performance when the algorithm terminates after a fixed number of iterations, identified as a good overall number of iterations during validation. Specifically, we run experiments with fixed parameters δ and k, and vary the number of iterations. The results on the target validation data are shown in Figure 3, for 500 source instances (left) and 4000 source instances (right). The dot on each curve represents

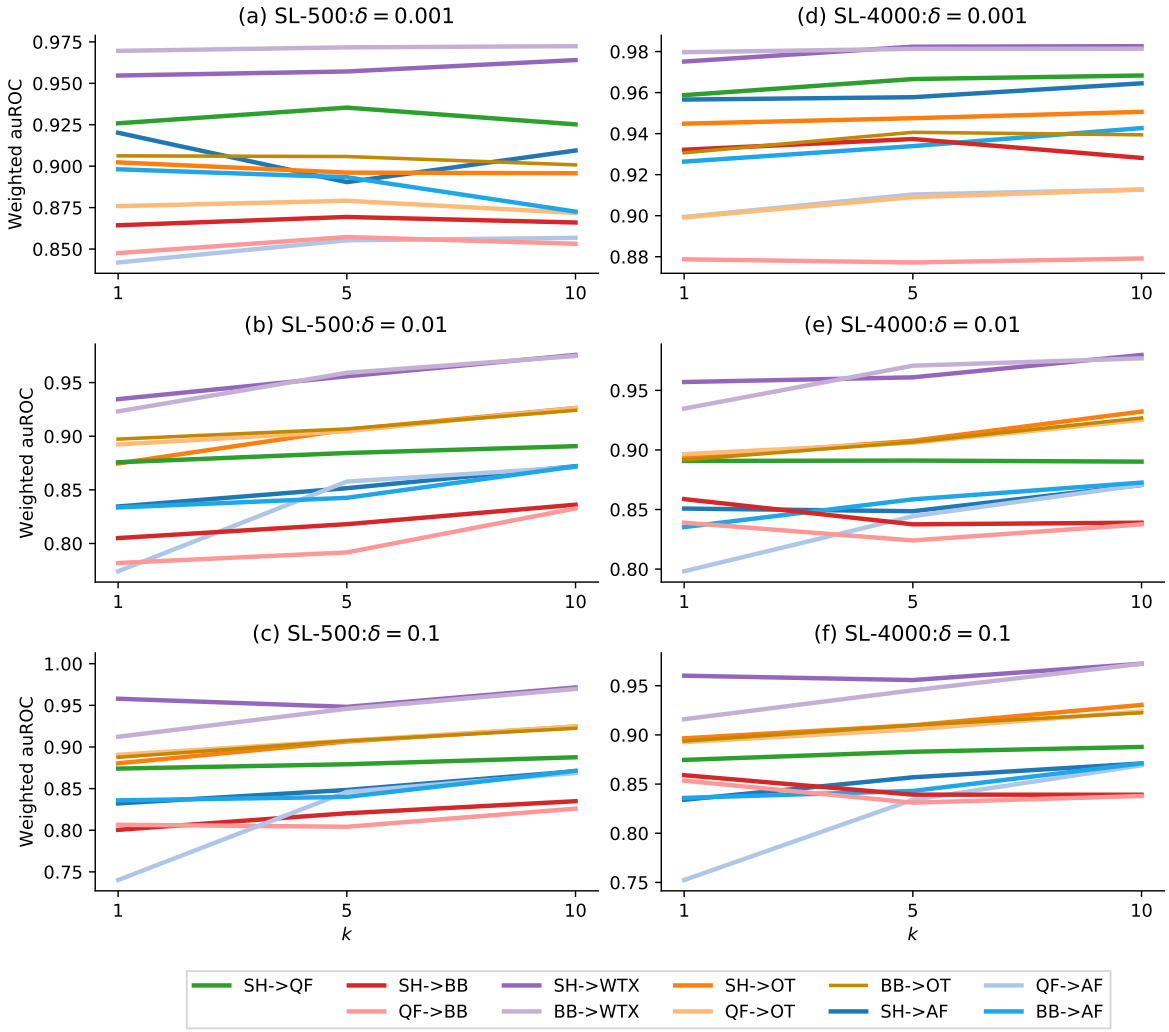

Figure 2. Variation of performance (auROC) with k for two sizes of the labeled source data: SL-500 (left) and SL-4000 (right). Parameter δ is fixed to 0.001, 0.01 and 0.1, respectively. The algorithm terminates upon convergence.

the number of iterations at "convergence" for the corresponding pair on a curve. As can be seen, the performance generally increases with the number of iterations, beyond the number of iterations at which "convergence" is reached. However, after a certain number of iterations (equivalently, after a certain number of target instances have been added to the training set), the performance does not change much. As expected, the number of iterations at which performance stabilizes is larger for $k = 5$ as compared to $k = 10$, as less target instances are added to the training data, at each iteration, for $k = 5$. In particular, the performance becomes stable around 250/300 iterations for $k = 5$ and around 150 iterations for $k = 10$. For $k = 5$, 300 iterations correspond to 5x2x300=3000 target instances being added to the training data, while for $k = 10$, 150 iterations correspond to 2x10x150=3000 target instances as well. This result suggests that the number of target instances to be included in the training dataset needs to be greater than 3000 for best performance. However, it is important to note that the algorithm can start with the unlabeled target data available at the onset of a disaster. As more unlabeled target data becomes available at a later time, that data can be used in subsequent iterations of the domain adaptation algorithm, in an online fashion.

Table 5 shows the results of the algorithm when run with fixed parameters $\delta = 0.001$ and $k = 5$ and two termination conditions, respectively: convergence (NB-STF-Conv) and fixed number of iterations, specifically 300 iterations (NB-STF-Iter). As can be seen, the results are almost aways better when using a fixed number of iterations, therefore running the algorithm beyond pseudo-convergence is generally advantageous.

In summary, our empirical results suggest that fixing the algorithm's parameters δ and k, and fixing the number of iterations n can be done without scarifying performance. In turn, this finding makes it possible to use our domain

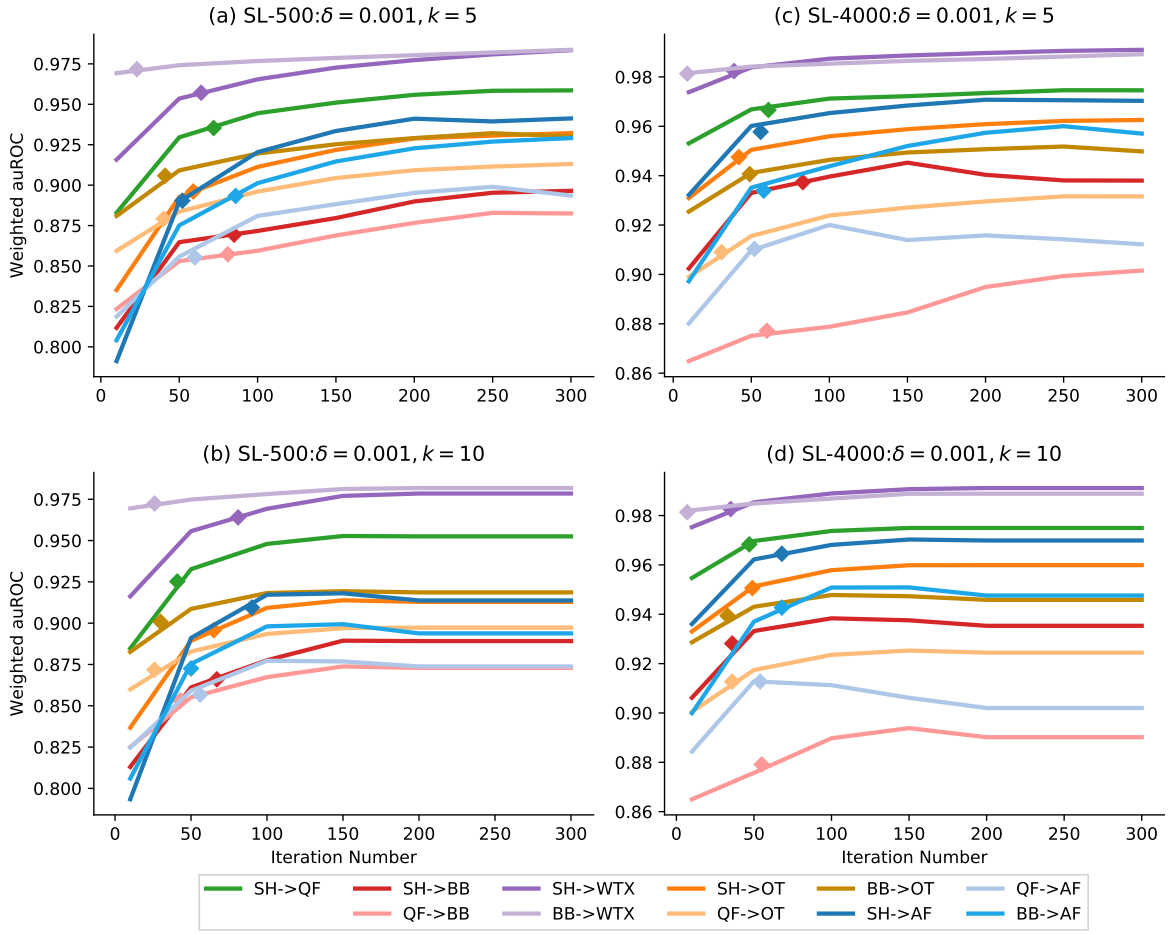

Figure 3. Variation of performance (auROC) with the number of iterations for two sizes of the labeled source data: SL-500 (left) and SL-4000 (right). The dots on each curve represent the performance at "convergence."

adaptation approach in a practical situation, where target labeled data for tuning parameters is not available, but batches of unlabeled target data accumulate quickly in an online fashion.

RELATED WORK

In the context of automatically classifying disaster data, several research papers have studied supervised learning algorithms in regard to transferring information from a prior source disaster to a current target disaster (Imran, Elbassuoni, et al. 2013; Imran, Mitra, et al. 2016; Verma et al. 2011; Rudra et al. 2015). Verma et al. (2011) used natural language processing techniques together with machine learning algorithms, both Naïve Bayes and Maximum Entropy, to identify situational awareness tweets belonging to four crisis events. They studied how well the Maximum Entropy classifiers performed across the four events and found that the classifiers didn't generalized well across different types of disasters. In a similar study, Imran, Mitra, et al. (2016) used Random Forest classifiers and achieved good results for disasters of the same type, but not for disasters of different types. Imran, Elbassuoni, et al. (2013) performed experiments on two disasters, Joplin Tornado (as source) and Hurricane Sandy (as target), with the goal of identifying information nuggets. After classifying different types of informative (casualties, donations, etc.) tweets with Naïve Bayes classifiers, they used a machine-learning sequence labeling algorithm, conditional random fields (CRF), to extract useful information, such as the number of casualties or the name of the infrastructure. Their experiments showed that using source data only results in a significant drop in the detection rate, while not affecting significantly the recall.

Most of these supervised algorithms, e.g., Maximum Entropy, Random Forest, are well developed and have been used extensively in text classification, but have hyper-parameters that need to be t uned. When applying these supervised algorithms to classify disaster social media data, default parameter values may not produce models as good as those obtained when tuning parameters. Furthermore, supervised classifier learned from a prior source disaster only may not perform well on an emergent target disaster as suggested by (Verma et al. 2011; Imran,

Table 4. Tuning results (NB-STT-Conv) versus results with fixed parameters $\delta = 0.001$ and $k = 5$ (NB-STF-Conv). In both cases, the algorithm terminates upon convergence. Best value in each column of a pair is shown in bold (row-wise t-test with $p \leq 0.05$).

Pair	Experiment	SL-500	SL-4000
SH -> QF	NB-STT-Conv	**0.945**	**0.971**
	NB-STF-Conv	**0.949**	**0.967**
SH -> BB	NB-STT-Conv	**0.877**	**0.935**
	NB-STF-Conv	**0.873**	**0.938**
QF -> BB	NB-STT-Conv	**0.840**	**0.890**
	NB-STF-Conv	**0.855**	0.878
SH -> WTE	NB-STT-Conv	**0.973**	**0.986**
	NB-STF-Conv	0.959	**0.985**
BB -> WTE	NB-STT-Conv	**0.969**	**0.980**
	NB-STF-Conv	**0.973**	**0.983**
SH-> OT	NB-STT-Conv	**0.921**	**0.953**
	NB-STF-Conv	**0.910**	**0.931**
QF -> OT	NB-STT-Conv	**0.916**	**0.926**
	NB-STF-Conv	0.893	0.914
BB -> OT	NB-STT-Conv	**0.919**	**0.942**
	NB-STF-Conv	**0.911**	**0.942**
SH -> AF	NB-STT-Conv	**0.925**	**0.970**
	NB-STF-Conv	**0.928**	**0.965**
QF -> AF	NB-STT-Conv	**0.880**	**0.918**
	NB-STF-Conv	**0.864**	**0.919**
BB -> AF	NB-STT-Conv	**0.898**	**0.950**
	NB-STF-Conv	**0.894**	**0.946**

Table 5. Results with fixed parameters when the algorithm runs to convergence (NB-STF-Conv) or for a fixed number of iterations (NB-STF-Iter), specifically 300 iterations. Best value in each column of a each pair is shown in bold (row-wise t-test with $p \leq 0.05$).

Pair	Experiment	SL-500	SL-4000
SH -> QF	NB-STF-Conv	0.949	**0.967**
	NB-STF-Iter	**0.957**	**0.972**
SH -> BB	NB-STF-Conv	0.873	**0.938**
	NB-STF-Iter	**0.893**	0.935
QF -> BB	NB-STF-Conv	0.855	0.878
	NB-STF-Iter	**0.880**	**0.898**
SH -> WTE	NB-STF-Conv	0.959	0.985
	NB-STF-Iter	**0.980**	**0.990**
BB -> WTE	NB-STF-Conv	0.973	0.983
	NB-STF-Iter	**0.981**	**0.987**
SH-> OT	NB-STF-Conv	0.910	0.950
	NB-STF-Iter	**0.938**	**0.962**
QF -> OT	NB-STF-Conv	0.893	0.914
	NB-STF-Iter	**0.919**	**0.933**
BB -> OT	NB-STF-Conv	0.911	0.942
	NB-STF-Iter	**0.934**	**0.952**
SH -> AF	NB-STF-Conv	0.928	**0.965**
	NB-STF-Iter	**0.940**	**0.968**
QF -> AF	NB-STF-Conv	0.864	**0.919**
	NB-STF-Iter	**0.902**	**0.916**
BB -> AF	NB-STF-Conv	0.894	0.946
	NB-STF-Iter	**0.932**	**0.959**

Elbassuoni, et al. 2013; Li, Guevara, et al. 2015 and our own work (Li, D. Caragea, et al. 2017)). At last, the works mentioned before use source data sets of varying sizes, and it is not clear how that affects the performance.

In terms of unsupervised domain adaptation, Li, Guevara, et al. (2015) proposed a domain adaptation algorithm based on EM. Their algorithm makes use of target disaster unlabeled data, together with source disaster labeled data, and produces classifiers that are better than the supervised classifiers learned from labeled source data only. Li, Guevara, et al. (2015) experimented with three classification tasks from two disasters, Hurricane Sandy (used as source) and Boston Bombing (used as target), with promising results on the task of identifying tweets relevant to a certain disaster. As opposed to the algorithm proposed by Li, Guevara, et al. (2015), our domain adaptation variant uses self-training, enables the learning process to start quickly (when only a small number of target unlabeled data is available) and allows for incremental updates of the target unlabeled data, as more target unlabeled data becomes available. This makes our variant more appropriate for a real scenario.

C. Caragea, Silvescu, et al. (2016) explored the use of Convolutional Neural Networks (CNN) to classify informative tweets from six flood e vents. Their idea to identify overall good parameters for the models is similar to the idea in our paper. To implement this idea, they used three flood disaster datasets with labeled instances, tuned parameters on one dataset and tested them on the other two test datasets to see how well the tuned parameters generalize. Nguyen et al. (2016) used convolutional neural networks (CNN) to classify crisis related tweets. They assumed that some target labeled data is available and used two simple supervised domain adaptation techniques to combine prior source disasters data with current disaster labeled data during training. One technique was weighting the prior source disasters data, while regularizing the modified m odel. The other technique was simply selecting a subset of the prior source disaster tweets, specifically those that were correctly labeled by a target-based classifier. Experimental results showed that CNNs with the simple instance selection domain adaptation technique gave better results. One drawback of these approaches is the requirement that some target labeled data is available.

At last, Zhang and Vucetic (2016) proposed another supervised domain adaptation approach that requires target labeled data, and used the dataset that we also used. Their proposed approach first clustered words from unlabeled tweets, and then trained a logistic regression classifier on labeled disaster tweets represented with the word clusters as features. They varied the number of current disaster labeled instances and found that the performance was generally better with more labeled data. This result is similar to our result that suggests that the more source labeled data, the better the performance. However, we found out that for our domain adaptation approach, more source labeled data can help up to a point when the performance stabilizes (in our experiments, the performance stabilized at 4000 source labeled instances). Zhang and Vucetic (2016) also varied the number of unlabeled instances from the current disaster and/or other source disasters, and even pre-trained a vocabulary based on word clustering. They found that more unlabeled data gives better results, in general, with some exceptions. In our experiments, the performance stabilized around 3000 target unlabeled instances.

CONCLUSION

The value of social media data, such as Twitter data, in emergency situations has been widely recognized by researchers. However, emergency organizations haven't extensively adopted such data in practice due to many challenges, including, information overload and reliability of data, among others. At the same time, many machine learning and data analysis techniques and systems have been proposed to address the information overload problem. Unfortunately, many machine learning algorithms require parameter tuning to build a good model, and this represents a big challenge when labeled data for tuning is not available (as it's the case in a current crisis situation). To address this challenge, we studied an unsupervised domain adaptation algorithm inspired by Li, Guevara, et al. (2015) on the the very first task aimed at reducing information overload on Twitter disaster data, specifically the task of classifying tweets as related to the disaster of interest or not related.

Our study provides recommendations for good overall parameter values to be used in practice for the domain adaptation algorithm with self-training. Furthermore, our study provides recommendations with respect to the number of labeled source instances to be used for good performance, and also with respect to the number of iterations needed for good performance.

We used the CrisisLexT6 dataset and constructed eleven source and target disaster pairs to experiment with. We showed that in general more source labeled data can produce better results, but performance does not change significantly after a certain amount, for example after 4000 instances (2000 on-topic and 2000 off-topic). If that amount of source labeled data is not available, our experiments show that even 500 source labeled instances can result in a relatively good classifier. This provides some insights into how much effort we should put into labeling data. Based on the analysis of performance variation with parameters, we recommend small values for δ (e.g., 0.001) to very slowly shift the weight from source to target and get good overall results. Furthermore, our study suggests that adding more than 1 instances from each class at each iteration of the algorithm (e.g., adding 5 or 10 instances from each class) benefits the final classifier. Finally, we showed that best overall results are obtained when fixing the shifting speed δ as 0.001, adding $k = 5$ instances from each class at each iteration, and running 300 iterations. This suggests that we can potentially use these parameter values as default in practice without sacrificing performance, and thus help disaster management and response teams prioritize the information that they need to more carefully analyze.

There are of course limitations to our work. While our results could contribute to a decrease in the amount of information that a human analyst needs to consider, by identifying tweets related to a disaster of interest, additional analysis is needed on other classification tasks, for example multi-class classification tasks, to understand how the algorithm behaviors generalize to different classification tasks. Thus, as part of future work, we plan to analyze the domain adaptation algorithm with self-training on more classification tasks. For example, we plan to consider the task of classifying disaster related tweets further into subcategories based on the user publishing the tweets or based on the tweet content - a tweet can be contributed by eyewitnesses, by victims or by news agents, etc., or a tweet can be about casualty, infrastructure, etc. As deep learning techniques have shown their potential in the context of social media data posted during emergencies, we plan to extend existing deep learning algorithms into unsupervised domain adaptation approaches, or supervised domain adaptation approaches that require a very small amount of target labeled data. Finally, we also plan to study how to choose a good source disaster for a particular target disaster, when we have different source disasters available. Alternatively, we plan to explore approaches for combining multiple source disasters.

ACKNOWLEDGMENTS

The computing for this project was performed on the Beocat Research Cluster at Kansas State University. We thank the National Science Foundation for support from grant CNS-1429316, which funded the cluster. We also thank the

National Science Foundation for support from the grants IIS-1526542 and CMMI-1541155 to Cornelia Caragea. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either express or implied, of the National Science Foundation. We also wish to thank our anonymous reviewers for their constructive comments.

REFERENCES

Caragea, C., Silvescu, A., and Tapia, A. H. (2016). "Identifying Informative Messages in Disasters using Convolutional Neural Networks". In: *13th Proceedings of the International Conference on Information Systems for Crisis Response and Management, Rio de Janeiro, Brasil, May 22-25, 2016.*

Caragea, C., Squicciarini, A. C., Stehle, S., Neppalli, K., and Tapia, A. H. (2014). "Mapping moods: Geo-mapped sentiment analysis during hurricane sandy". In: *11th Proceedings of the International Conference on Information Systems for Crisis Response and Management, University Park, Pennsylvania, USA, May 18-21, 2014.*

Castillo, C. (2016). *Big Crisis Data: Social Media in Disasters and Time-Critical Situations*. Cambridge University Press.

Herndon, N. and Caragea, D. (2015). "Empirical Study of Domain Adaptation Algorithms on the Task of Splice Site Prediction". In: *Biomedical Engineering Systems and Technologies: 7th International Joint Conference, BIOSTEC 2014, Angers, France, March 3-6, 2014, Revised Selected Papers*. Ed. by G. Plantier, T. Schultz, A. Fred, and H. Gamboa. Cham: Springer International Publishing, pp. 195–211.

Hughes, A. L., Denis, L. A. S., Palen, L., and Anderson, K. M. (2014). "Online public communications by police & fire services during the 2012 Hurricane Sandy". In: *CHI Conference on Human Factors in Computing Systems, CHI'14, Toronto, ON, Canada - April 26 - May 01, 2014*, pp. 1505–1514.

Imran, M., Castillo, C., Diaz, F., and Vieweg, S. (2015). "Processing Social Media Messages in Mass Emergency: A Survey". In: *ACM Comput. Surv.* 47.4, 67:1–67:38.

Imran, M., Elbassuoni, S., Castillo, C., Diaz, F., and Meier, P. (2013). "Practical extraction of disaster-relevant information from social media". In: *22nd International World Wide Web Conference, WWW '13, Rio de Janeiro, Brazil, May 13-17, 2013, Companion Volume*, pp. 1021–1024.

Imran, M., Mitra, P., and Srivastava, J. (2016). "Cross-Language Domain Adaptation for Classifying Crisis-Related Short Messages". In: *13th Proceedings of the International Conference on Information Systems for Crisis Response and Management, Rio de Janeiro, Brasil, May 22-25, 2016.*

Landwehr, P. M. and Carley, K. M. (2014). "Social Media in Disaster Relief". In: *Data Mining and Knowledge Discovery for Big Data: Methodologies, Challenge and Opportunities*. Ed. by W. W. Chu. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 225–257.

Li, H., Caragea, D., Caragea, C., and Herndon, N. (2017). "Disaster Response Aided by Tweet Classification with a Domain Adaptation Approach". In: *Journal of Contingencies and Crisis Management* Special Issue on HCI in Critical Systems.In press.

Li, H., Guevara, N., Herndon, N., Caragea, D., Neppalli, K., Caragea, C., Squicciarini, A. C., and Tapia, A. H. (2015). "Twitter Mining for Disaster Response: A Domain Adaptation Approach". In: *12th Proceedings of the International Conference on Information Systems for Crisis Response and Management, Krystiansand, Norway, May 24-27, 2015.*

Manning, C. D., Raghavan, P., Schütze, H., et al. (2008). *Introduction to information retrieval*. Vol. 1. 1. Cambridge university press Cambridge.

Nguyen, D. T., Al-Mannai, K., Joty, S. R., Sajjad, H., Imran, M., and Mitra, P. (2016). "Rapid Classification of Crisis-Related Data on Social Networks using Convolutional Neural Networks". In: *CoRR* abs/1608.03902.

Olteanu, A., Castillo, C., Diaz, F., and Vieweg, S. (2014). "CrisisLex: A Lexicon for Collecting and Filtering Microblogged Communications in Crises". In: *Proceedings of the Eighth International Conference on Weblogs and Social Media, ICWSM 2014, Ann Arbor, Michigan, USA, June 1-4, 2014.*

Palen, L. and Anderson, K. M. (2016). "Crisis informatics-New data for extraordinary times". In: *Science* 353.6296, pp. 224–225.

Palen, L., Vieweg, S., and Anderson, K. M. (2011). "Supporting "Everyday Analysts" in Safety- and Time-Critical Situations". In: *Inf. Soc.* 27.1, pp. 52–62.

Plotnick, L., Hiltz, S. R., Kushma, J. A., and Tapia, A. H. (2015). "Red Tape: Attitudes and Issues Related to Use of Social Media by U.S. County-Level Emergency Managers". In: *12th Proceedings of the International Conference on Information Systems for Crisis Response and Management, Krystiansand, Norway, May 24-27, 2015.*

Reuter, C., Ludwig, T., Friberg, T., Pratzler-Wanczura, S., and Gizikis, A. (2015). "Social Media and Emergency Services?: Interview Study on Current and Potential Use in 7 European Countries". In: *IJISCRAM* 7.2, pp. 36–58.

Rudra, K., Ghosh, S., Ganguly, N., Goyal, P., and Ghosh, S. (2015). "Extracting Situational Information from Microblogs During Disaster Events: A Classification-Summarization Approach". In: *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*. CIKM '15. Melbourne, Australia: ACM, pp. 583–592.

Starbird, K., Palen, L., Hughes, A. L., and Vieweg, S. (2010). "Chatter on the red: what hazards threat reveals about the social life of microblogged information". In: *Proceedings of the 2010 ACM Conference on Computer Supported Cooperative Work, CSCW 2010, Savannah, Georgia, USA, February 6-10, 2010*, pp. 241–250.

Tapia, A. H. and Moore, K. (2014). "Good Enough is Good Enough: Overcoming Disaster Response Organizations' Slow Social Media Data Adoption". In: *Computer Supported Cooperative Work (CSCW)* 23.4, pp. 483–512.

Verma, S., Vieweg, S., Corvey, W. J., Palen, L., Martin, J. H., Palmer, M., Schram, A., and Anderson, K. M. (2011). "Natural Language Processing to the Rescue? Extracting "Situational Awareness" Tweets During Mass Emergency". In: *Proceedings of the Fifth International Conference on Weblogs and Social Media, Barcelona, Catalonia, Spain, July 17-21, 2011*.

Yarowsky, D. (1995). "Unsupervised Word Sense Disambiguation Rivaling Supervised Methods". In: *Proceedings of the 33rd Annual Meeting on Association for Computational Linguistics*. ACL '95. Cambridge, Massachusetts: Association for Computational Linguistics, pp. 189–196.

Zhang, S. and Vucetic, S. (2016). "Semi-supervised Discovery of Informative Tweets During the Emerging Disasters". In: *CoRR* abs/1610.03750.