Semi-Supervised Granular Classification Framework for Resource Constrained Short-texts

Towards Retrieving Situational Information During Disaster Events

Samujjwal Ghosh*
cs16resch01001@iith.ac.in
Indian Institute of Technology Hyderabad
Hyderabad, Telangana

Maunendra Sankar Desarkar
maunendra@iith.ac.in
Indian Institute of Technology Hyderabad
Hyderabad, Telangana

ABSTRACT

During the time of disasters, lots of short-texts are generated containing crucial situational information. Proper extraction and identification of situational information might be useful for various rescue and relief operations. Few specific types of infrequent situational information might be critical. However, obtaining labels for those resource-constrained classes is challenging as well as expensive. Supervised methods pose limited usability in such scenarios. To overcome this challenge, we propose a semi-supervised learning framework which utilizes abundantly available unlabelled data by self-learning. The proposed framework improves the performance of the classifier for resource-constrained classes by selectively incorporating highly confident samples from unlabelled data for self-learning. Incremental incorporation of unlabelled data, as and when they become available, is suitable for ongoing disaster mitigation. Experiments on three disaster-related datasets show that such improvement results in overall performance increase over standard supervised approach.

CCS CONCEPTS

• **Information systems** → **Information extraction**; *Clustering and classification*; Incomplete data; *Data mining*; Social networks; • **Human-centered computing** → *Social network analysis*; • **Computing methodologies** → *Supervised learning by classification.*

KEYWORDS

Semi-supervised Learning, Social Media, Short-text Classification, Crisis Management

ACM Reference Format:
Samujjwal Ghosh and Maunendra Sankar Desarkar. 2020. Semi-Supervised Granular Classification Framework for Resource Constrained Short-texts : Towards Retrieving Situational Information During Disaster Events. In *12th ACM Conference on Web Science (WebSci '20), July 6–10, 2020, Southampton, United Kingdom.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3394231.3397892

WebSci '20, July 6–10, 2020, Southampton, United Kingdom

ACM ISBN 978-1-4503-7989-2/20/07…$15.00
https://doi.org/10.1145/3394231.3397892

1 INTRODUCTION

At the time of disasters, micro-blogging platforms like Twitter work as a real-time source of situational information [7, 10, 12] because of it's easy accessibility and vast outreach [17]. Short-texts generated during disasters contains important actionable information such as infrastructure damage, availability and requirements of various resources (medicines, medical tool-kits, food, water, tent), etc. A portion of these short-texts may be called situation-aware information as they represent factual and actionable information related to the disaster. Proper identification and categorization of different types of situation-aware information can help NGOs and various agencies to plan and deploy disaster response effectively.

There have been several efforts in the past to effectively extract and classify short-texts with situational information posted during disasters. We categorize them into two separate classes based on the type of data they use. Most commonly used and effective approach has been supervised learning, using only labelled data, by feature boosting [4], incorporation of disaster specific lexicons [13], curated features [1], convolutional neural networks [2]. Supervised approaches, although useful, are very data-hungry. Obtaining sufficient labelled data during an ongoing disaster is expensive. Even when labelled data is available, distribution of types of information in the data might be heavily skewed towards a few classes, which results in class-imbalanced data. Few particular classes might have very less representation due to their inherent nature of less occurrence in the world. We identify such classes as resource-constrained classes. Generally, the performance of supervised models for resource-constrained classes is poor. However, with proper use of unlabelled data, classifier performance for resource-constraint classes as well as overall performance could be improved.

The second class of approach, Semi-Supervised Learning (SSL) comes to the rescue as it can exploit the abundantly available unlabelled data along with the help of a small amount of labelled data. By incorporating latent information in the form of newly selected samples from a large amount of unlabelled data, self-learning semi-supervised learning can efficiently handle resource-constrained classes. Self-learning refers to feeding in predictions of the trained model on the unlabelled data as new target values for re-training. It has been observed that self-learning often outperform it's supervised counterpart. The study [11] by authors Mobahi et al. argue that self-distillation, a form of self-learning technique, may reduce over-fitting by amplifying regularization. However, the authors also warn that performance may deteriorate after a few iterations due to incorporation of misclassified data.

In this work, we utilize abundantly available unlabelled data to improve the performance of a supervised multi-label classifier by self-learning. We propose a novel adaptive self-learning based framework which improves the performance of the supervised model. The framework is designed to easily incorporate the unlabelled data in an incremental setting, which can also handle imbalanced resource-constrained classes. Effective utilization of unlabelled data outperforms classifiers trained with only labelled data. This performance gap increases as the number of labelled samples decrease for resource-constrained classes.

2 RELATED WORK

There have been several studies of binary short-text classification using semi-supervised learning. In [21], the authors Zhang et al. used brown clustering on the unlabelled data to cluster words. They used these clusters as dense features to decide if a tweet is relevant to the disaster in consideration or not. Clustering similar words help in utilizing words that rarely occur in the labelled data. However, word clusters limit the incremental capability of the approach as the inclusion of newer sample requires a large amount of rework. Moreover adding many new words may change the clusters significantly, thereby resulting in entirely new feature representations for the existing samples. Also, this study mostly concentrates on different clustering choices for effective retrieval of relevant samples and does not focus on any granular classification. Authors in [9] proposed a domain adaptation approach between a source and a target disaster with the help of unlabeled data. The approach utilizes the labelled data from a source disaster along with unlabelled data generated during an emerging target disaster to classify short-texts generated during target disaster to either relevant or irrelevant categories. The domain adaptation was achieved between the labelled source disaster and an unlabelled target disaster by Expectation-Maximization with the help of Naive Bayes Classifier. However, the proposed approach assumes the same class labels between source and target datasets. Granular class labels might not be aligned in different disasters. To the best of our knowledge, our work is *first to employ semi-supervised learning on granular multi-label short-text classification for disaster management.*

Several semi-supervised algorithms have been proposed in the literature for various scenarios. In [6], authors Go et al. used smileys present in the unlabeled tweets as distant supervision for sentiment classification of short-texts. However, to filter situational information in disasters, we need to avoid short-texts with sentiments as they usually do not contain any actionable information. Also, distant supervision is useful when some form of latent relationship is present between indicator and class labels. Distant supervision might be possible for a few granular classes but not be feasible for all disaster scenarios. Co-training is a form of semi-supervised learning where most confident samples of one classifier are fed into the other classifier(s). In [15], Pekar et al. applied co-training to filter disaster-related short-texts by using two different classifiers. They used the text of the tweet as a feature set for one classifier and the image associated with the tweet on the second classifier to improve their overall model accuracy. Though this approach is beneficial, for co-training to work efficiently, disjoint feature sets are required so that one classifier can learn from each other. Without sufficiently disjoint feature set, a single classifier trained with all features might work better. For an ongoing disaster, the proportion of generated short-texts associated with an image are scarce.

Semi-supervised learning based on self-learning was first proposed by Yarowsky et al. [20] for word sense disambiguation. Authors in [14], used self-learning semi-supervised learning to identify relevant tweets related to 2015 Chennai, India floods and used the relevant tweets to annotate the affected areas in Google map for people to use. They selected all tweets for re-training, which was marked relevant by the initial supervised classifier. Authors Edo et al. [3] proposed a semi-supervised approach for relevance filtering in syndromic surveillance. They selected all samples with a confidence score above a specific threshold t to retrain the initial classifier. Lee et al. [8] proposed a semi-supervised approach for sentiment classification based on self-learning. They re-trained a neural network by randomly selecting samples from unlabelled data along with their classes predicted by the neural network. Different sampling rates $(0.4, 0.6, 0.8, 1.0)$ were experimented to show the effect in the performance. The sampling of unlabelled data in [8] was performed after deciding the class labels, which means that sampling was done on the samples for which confidence was higher than the threshold of that class. As a result, when the sampling rate is 1.0 in [8] it becomes equivalent to [3] and [14].

3 FRAMEWORK

We propose a self-learning semi-supervised learning framework which utilizes the readily available unlabelled data to improve classifier performance over a classifier trained only with labelled data. We explicitly tune the framework in a way such that our framework can handle classes with deficient representation in the data. The proposed framework works in two phases. In the first phase, we train an initial supervised classifier $f_i,\ i = 0$ on the available labelled data L. The second phase is an iterative approach to use the unlabelled data U to improve the performance of the initial classifier f_0. In iteration i $(i > 0)$ of the second phase, we randomly sample a batch of unlabelled data $U_i \subseteq U$ and predict possible class confidence scores using the classifier trained on the previous iteration f_{i-1}. A *small* set of *confident* samples Γ_i^c $\left(\bigcup_c \Gamma_i^c = \Gamma_i; \Gamma_i \subseteq U_i\right)$ for each class from this unlabelled batch U_i is then selected to be assimilated to the training set for iteration $(i + 1)$. Sections 3.1.1 and 3.1.2 elaborate on how the size and the confidence scores are decided.

Any remaining samples $(U_i - \Gamma_i)$, from that batch, are discarded and not reintroduced in future. We discard such samples based on the assumption that these tweets do not contain any situational information. Then, we label the selected samples with a class c if the confidence score for that class is higher than threshold T_i^c (Section 3.1.2 describe how the threshold T_i^c is calculated) of that class. This technique of labelling samples to re-train a model is called self-label learning [19]. This new set of samples with their labelled classes are then combined with the existing training set and used to re-train the classifier f_i to get an improved classifier f_{i+1}. This iterative approach of prediction and assimilation is applied batch-wise until any of the stopping criteria (Described in Section 3.2) are met.

To handle resource-constrained classes, we crafted our framework to take special care for imbalanced classes. The framework achieves this by explicitly improving the representation of those classes in each iteration by exploiting the behaviour of classifiers for resource-constrained classes. Next, we'll look into the criteria used to increase the representation of resource-constrained classes.

3.1 Sample Selection Criteria

Here, we describe the criteria used to select the samples Γ_i from unlabelled batch U_i. We restrict the selection of samples using two different criteria. We decide on the maximum number of samples added per class based on the distribution of classes in the training set in that iteration. Along with the restriction on the number of samples per class, we also restrict the assignment of a sample to a class based on the class-specific threshold. Below we describe the exact procedure we used for class assignment and selection of a sample.

3.1.1 Capacity Criterion. This criterion sets an upper limit on the number of samples to be selected per class. This criterion has two effects; it minimizes the class imbalance in the new train set and also works as a deterrent for the selection of low confidence samples. It specifically helps in enforcing a more significant presence for the resource-constrained classes during re-training. The capacity of a class depends on the number of short-texts present in the training set during the current iteration. Let n^c be the number of short-texts from class c present in the train set and n_i is the total number of samples in training set for iteration i. We decide the maximum capacity for class c as,

$$k_i^c = \left(1 - \frac{\log n_i^c}{\log n_i}\right) \times n_i^c \tag{1}$$

Note that the value of n_i and n_i^c changes in each iteration. log was used to handle heavily skewed (positive or negative) data. It should be noted that the lesser number of samples a class has in the training set, higher will be the capacity k^c for that class. However, our other criterion (explained later in Section 3.1.2) mitigates the pitfall of incorporating mislabelled samples for resource-constrained classes.

Analysis of Capacity criterion: Equation 1 ensures a higher capacity for resource-constrained classes. As n_i^c decreases, $\log n_i^c$ decreases, and hence the quantity $\left(1 - \frac{\log n_i^c}{\log n_i}\right)$ reaches close to 1. However, maximum capacity is capped at the number of samples in the training set of that class in the current iteration.

3.1.2 Confidence Criterion. The confidence criterion has dual purposes; initially, it limits the incorporation of low confidence, possibly erroneous, short-texts to the new train set. Later, we use the same threshold to convert confidence scores to class labels. A sample is assigned to a class only if the confidence score of that sample for that class is higher than the threshold of the class. We observed during our experiments that *Recall is usually considerably lower than Precision for resource-constrained classes* as demonstrated for class 4 and 5 in Figure 2. A possible reason for this behaviour is due to the lack of presence of sufficient features for those classes. In our classification scheme, the features to represent the samples are extracted from the data. Identification of characteristic features for a class depends upon the training samples available for that class. Lower presence in the train set results in the lack of appropriate representation in the feature space leading to low Recall. We exploit this behaviour to identify ways to boost the performance of these classes. The lower threshold is preferred for resource-constrained classes which helps to expand the feature space. However, a small threshold presents the challenge of selecting low confidence misclassified samples and in turn, lowering the Precision. To handle both scenarios, we carefully adapt the threshold for each class based on the performance of that class on the validation set.

$$T_i^c = min\left([T_{i-1}^c - \Delta_i^c], 0.9999\right), \text{for } i > 1 \tag{2}$$

Δ_i^c in Equation 2 is decided based on the Precision and Recall values of class c. Δ_i^c should be positive for classes with low Recall score, which decreases the threshold and negative for low Precision classes to increase the threshold. To conform to this behaviour, we define Δ_i^c as,

$$\Delta_i^c = \frac{(P_i^c - R_i^c)}{2} \tag{3}$$

Where P_i^c and R_i^c are the Precision and Recall scores for class c in iteration i respectively. Equation 3 achieves this by subtracting Recall from Precision and then dividing this difference by a constant to dampen its effect. The value of the constant was set to 2 in our experiments based on empirical evidence. However, this simplified equation does not capture the desired behaviour of increasing the threshold in each iteration. As a general rule, we want to increase the threshold with each iteration to compensate for any misclassified sample. Towards this desired effect, we include another term to the RHS of Equation 3 to get a modified formulation for Δ_i^c as given below in Equation 4.

$$\Delta_i^c = \frac{(P_i^c - R_i^c)}{2} - i\epsilon F_i^c \tag{4}$$

Where F_i^c is the F_1 score for class c in iteration i. The additional term also has the added advantage of boosting the threshold even when Precision and Recall are the same, i.e. $P_i^c == R_i^c$. We make use of Precision, Recall and F_1 scores of each class to boost the threshold accordingly for that class. This way, boosting is done according to the performance of that particular class.

To increase the boosting term in Δ_i^c in each iteration, we multiply the F_i^c with the iteration number i. However, to adjust the influence of boosting, we multiply a small constant ϵ. ϵ is chosen using the validation set, and fixed across all experiments for a dataset. The value of ϵ was considered to be within the range $[e^{-4}, e^{-2}]$. We calculate the initial threshold T_1^c using the classifier on the validation set.

$$T_1^c = max\left(\left[\frac{(P_0^c - R_0^c)}{2} - \epsilon F_0^c\right], 0.5\right), \text{for } i = 1 \tag{5}$$

Where P_0^c, R_0^c and F_0^c are the Precision, Recall and F_1 scores for class c on the validation set. To calculate the Precision, Recall and F_1 scores, we use 0.5 as the initial threshold T_0.

The confidence criterion can be summarized as,

$$T_i^c = \begin{cases} 0.5, & \text{if } i = 0 \\ max\left(\left[\frac{P_0^c - R_0^c}{2} - \epsilon F_0^c\right], 0.5\right), & \text{if } i = 1 \\ min\left(\left[T_{i-1}^c - \left(\frac{P_i^c - R_i^c}{2} - i\epsilon F_i^c\right)\right], 0.9999\right), & \text{if } i > 1 \end{cases} \tag{6}$$

Table 1: Change in threshold: example scenarios.

Precision	Recall	Δ (Change)
0.9 (High)	0.4 (Low)	0.1750 (Decrease)
0.4 (Low)	0.9 (High)	-0.3249 (Increase)
0.9 (High)	0.9 (High)	-0.1218 (Increase)
0.4 (Low)	0.4 (Low)	-0.0541 (Slight Increase)

$i = 0$ signifies supervised approach, $T_0^c = 0.5$ is used to convert class confidence scores to calculate the Precision, Recall and F_1 scores. There is no sample selection step during supervised learning. 0.5 was used to calculate scores for all results in this paper.

Analysis of Threshold Criterion: Equation 2 works as a filter for misclassified samples. Usually, when P_i^c is high and R_i^c is low, T_i^c is decreased from previous value. Similarly, when P_i^c is low and R_i^c is high, T_i^c is increased. However, this behaviour changes when P_i^c and R_i^c are both high or low. When P_i^c and R_i^c both are high or low, the term $i \epsilon F_i^c$ will increase the threshold T_i^c. However, the amount of increment depends on the values of P_i^c and R_i^c. We summarize these different possibilities with their effects on the Δ for few examples in Table 1. For these examples, we considered $i = 1$ and $\epsilon = e^{-2}$.

3.1.3 Confidence Score to Class Label. After the thresholds T_i and capacities k_i are decided for each class, we select the top samples for each class such that both Equation 1 and 2 are satisfied. Presence of both the criteria makes sure that the incorporation of selected samples improves the Recall score without harming the Precision. After the samples are selected, we convert the confidence scores to labels using Equation 7.

Let s_i^c be the predicted confidence score for a short-text s for class c in the i^{th} iteration. We assign the sample s to class c based on the following rule:

$$s_i^c = \begin{cases} 1, & \text{if } s_i^c \geq T_i^c \text{ and } s \in \Gamma_i^c \\ 0, & \text{otherwise} \end{cases} \tag{7}$$

Note that sample s might still be added to the selected set Γ_i for other classes. To ensure sufficient availability of highly confident resource-constrained class samples, the batch size for the unlabelled data should be substantial.

3.2 Stopping Criteria:

The iteration stops if any of the below-mentioned criteria are met.

- **No performance improvement:** If the performance (Macro-F_1 score) on the validation set does not increase in three consecutive iterations, we stop the process. In such a case, the former highest-performing model is chosen as the candidate model.
- **Unlabelled pool exhausted:** If all the unlabelled data were exhausted, the iterative process stops with the last highest-performing model as the candidate model.

3.3 Incremental Learning and Vocabulary Expansion

One of the most significant advantages of the proposed framework is that it works in an incremental setup. As a result, a newly collected set of unlabelled samples may be incorporated into the framework on-the-fly as soon as those are available. The batch size of the unlabelled data may vary among iterations. The flexibility to vary batch size is crucial as the incoming number of samples is not known apriori during an ongoing disaster.

Table 2: Vocabulary expansion of each dataset (Iteration 0 represents supervised iteration).

Iteration	Kaggle[1]	FIRE16	SMERP17
0	10633	2986	3046
1	12535	3307	3411
2	-	3604	3755
3	-	3860	4105
4	-	4210	4432
5	-	4569	4820

Our framework gradually increases the vocabulary of the feature space by incorporating unknown words from unlabelled data. Table 2 lists the vocabulary expansion for all the datasets. Addition of essential words helps expand the feature set, which improves the Recall score of the model. This technique is especially helpful for resource-constrained classes, and in disaster scenarios, where accumulation of sufficient number of short-texts might not be possible.

4 DATASET

We experiment with our proposed framework on two different types of disaster datasets. Initially, we use the "Kaggle NLP with Disaster Tweets (Kaggle)"[2] competition data as a binary classification task to predict disaster relevant tweets. Kaggle contains tweets posted during various disasters with "relevant" or "not relevant" as class labels. We use the file "train.csv" from the competition as the labelled data and break into train, validation and test sets for our training and evaluation purposes. The "test.csv" file was used as the unlabelled data. After cleaning and processing, the dataset contains a total of 9897 samples. 76.9% is of the dataset is labelled and the remaining 23.1% is unlabelled as seen in Figure 1.

We also use two disaster-related tweet datasets called "Forum for Information Retrieval Evaluation" 2016 (FIRE16) [5] and "Social Media for Emergency Relief and Preparedness (SMERP17)" [18] containing tweets with granular class information like infrastructure damage, resource required and other actionable information. FIRE16 and SMERP17[3] contain labelled and unlabelled tweets posted during 2015 earthquake in Nepal and 2016 earthquake in central Italy respectively. Each dataset comes with a set of labelled tweets tagged with predefined classes based on the situational information it contains.

Usually, each short-text contains situational information related to a single class as described in the TREC format. However, sometimes a short-text might contain information relevant to multiple classes. In such a scenario, the short-text is tagged with all the relevant classes. We could collect a total of 50, 774 and 71, 659 labelled or unlabelled tweets of FIRE16 and SMERP17. Out of which,

[1]Kaggle has very limited unlabelled data, all of which was incorporated in 1 iteration.
[2] https://www.kaggle.com/c/nlp-getting-started/overview
[3]This work is not a part of the SMERP task, we re-purposed the dataset for short-text classification task.

Table 3: Class details and distribution of samples for each dataset.

(a) Kaggle

Class	Title	Train	Val	Test
0	Irrelevant	2279	760	1303
1	Relevant	1717	573	981
	Sample count	3996	1333	2284

(b) FIRE16

Class	Title	Train	Val	Test
1	Resources Available	271	122	183
2	Resources Required	149	49	93
3	Medical Resources Available	160	67	104
4	Medical Resources Required	52	26	33
5	Requirements / Availability of Resources at Specific Locations	96	84	62
6	Activities of Various NGOs / Government Organizations	181	84	106
7	Infrastructure Damage and Restoration Reported	108	64	80
	Sample count	759	326	466

(c) SMERP17

Class	Title	Train	Val	Test
1	Resources Available	174	43	93
2	Resources Required	119	30	64
3	Infrastructure Damage, Restoration and Casualties Reported	1129	282	605
4	Rescue Activities of Various NGOs / Government Organizations	202	50	108
	Sample count	1624	405	807

labelled data constitutes approx 3.05% and 3.71% from each dataset as seen in Figure 1. Note that the short-text count mentioned here may vary from the original dataset. The Twitter policy does not allow direct short-text sharing, tweets were downloaded before the experiments, and some tweets may not be retrieved if they are deleted or made private. Remaining 49, 223 and 68, 964 tweets were unlabelled and henceforth called unlabelled set. Below we discuss the labelled data in more detail.

4.1 Labelled Data

The Kaggle has 7613 tweets labelled with either relevant (1) or irrelevant (0) as class labels. A class-wise detail of the data is provided in Table 3a. Remaining 2284 tweets are unlabelled. The FIRE16 and SMERP17 labelled data contains 1551 and 2695 tweets assigned to 7 and 4 predefined classes, respectively. In the original dataset, the classes have numbers as $FMT1, FMT2, \cdots, FMT7$ for FIRE16 and $SMERP\text{-}T1, SMERP\text{-}T2, \cdots, SMERP\text{-}T4$ for SMERP17. We denote each class by their numeric value in this paper. Figure 1 summarise the dataset details along with class distribution. A detailed class-wise count of samples are provided in Tables 3b and 3c.

5 EXPERIMENTAL SETUP

We divided the labelled data into Train, Validation and Test sets with 50%, 20% and 30% of the total short-texts respectively. Tables 3a, 3b and 3c show exact counts of short-texts used for Train, Validation and Test sets for Kaggle, FIRE16 and SMERP17 respectively. Sample count is the number of samples present in each set[4]. The same set of train, validation and test were used across all experiments. Train and Test set were used for model training and testing, respectively. Validation set was used for various hyper-parameter selection and to calculate capacity and threshold values. Details of the hyper-parameter selection are described in Section 5.1.

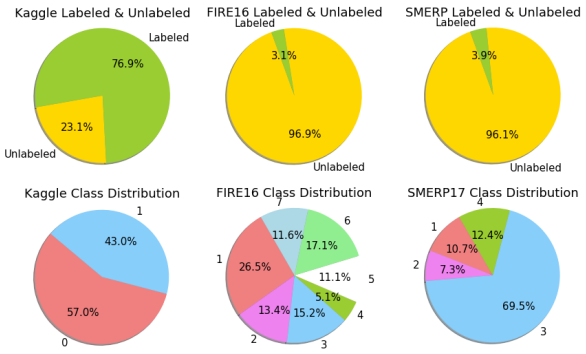

Figure 1: Dataset Details: The first row shows the proportion of Labelled and Unlabelled data for each dataset. The second row shows the distribution of different classes in the labelled data.

We initially experimented with the following machine learning algorithms: Gaussian Naive Bayes (GNB), Decision Tree (DT), Random Forest (RF), AdaBoost (AB), k-Nearest Neighbor (kNN), Logistic Regression (LR) and Support Vector Machine (SVM) to decide the best performing classifier for each dataset. After our extensive experiments gave evidence that for Kaggle, LR and for FIRE16 and SMERP17, SVM is best-performing classifier (see Table 4), we use those classifiers for all subsequent experiments for that dataset. Platt scaling [16] was used to get individual class confidence scores. As the short-texts may belong to more than one class, One-vs-Rest approach was taken for multi-label classification. Standard text pre-processing technique was used before feeding the text to the classifiers. We used Term Frequency - Inverse Document Frequency (TFIDF) to generate features for our algorithms. Note that the proposed framework is agnostic to the feature generation and algorithm used in this paper. It can be used with any classifier and any vocabulary-based feature generation technique such as embedding based techniques.

5.1 Hyper-Parameter Selection

The proposed framework has two different types of hyper-parameters. The first set of parameters could be thought of as modelling choices for the framework such as "batch size" for unlabelled data in each iteration and the value of ϵ in Equation 4. Out of these, "batch size" for unlabelled data had the least effect in performance except when it was too small. In such cases, the learning capability of the framework reduces drastically due to the lack of distinct features. These hyperparameters were selected using the validation set.

[4]Note that the sample count is not the same as the summation of the values in that column due to multi-label nature of the datasets.

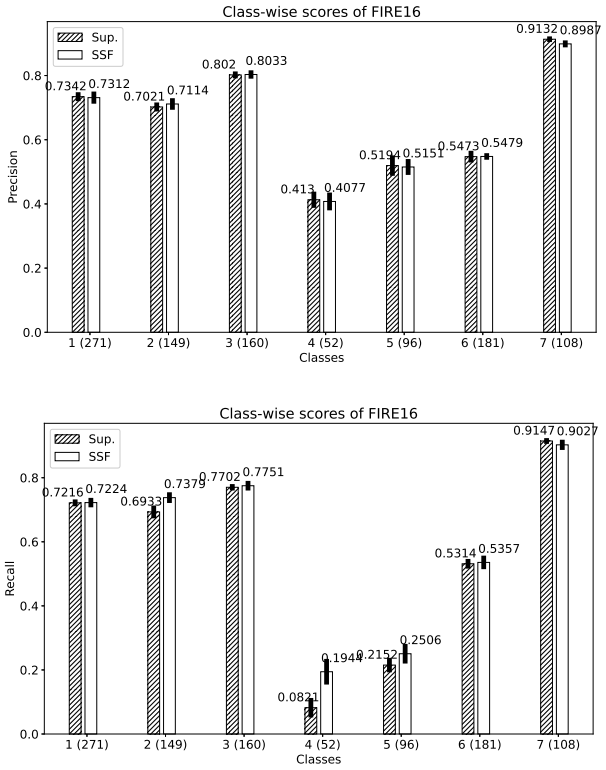

Figure 2: Class-wise Precision (SD) and Recall (SD) scores for FIRE16. Number of training examples for the class is mentioned in brackets after the class label in x-axis.

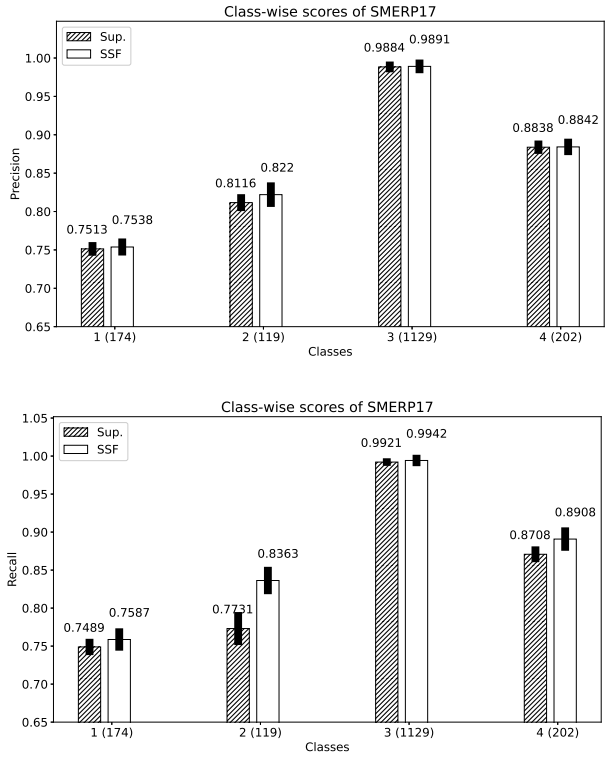

Figure 3: Class-wise Precision and Recall scores for SMERP17. Number of training examples for the class is mentioned in brackets after the class label in x-axis.

The other set of parameters that needed to be tuned are algorithm specific parameters. We used the same validation set for both types of hyper-parameter selection. Once the best model parameters were decided using the validation set, we tuned the algorithm specific parameters such as regularization parameter in SVM or penalty for Logistic Regression. Regularization parameter indicates the cost of a wrong classification. Higher value signifies costly misclassification so that algorithm learns to make fewer mistakes. However, this comes with the trade-off in the shrunk margin between classes in SVM algorithm. All the hyper-parameters were tuned during algorithm selection. Once the algorithm was decided for each dataset, the hyper-parameters were fixed for all subsequent experiments.

In the next Section, we present our results and analyze the performance of the different algorithms in detail.

6 RESULTS AND ANALYSIS

We answered several interesting questions related to multi-label short-text classification to granular classes. Table 5 contains the most comprehensive result related to our study. Test set was used to generate all results. We ran each experiment 10 times to calculate the average score along with their Standard Deviation (SD). In Figures 2, 3, 7 and 8 black rectangle on top of bars signify the SD calculated over 10 runs.

Which is the best classifier for each dataset? We applied various off-the-shelf classifiers (see Section 5 for the list of classifiers) to decide the best classifier on each dataset. Results from this experiment are mentioned in Table 4. LR works best for binary classification task on Kaggle for both supervised (Sup.) and Semi-Supervised Framework (SSF) approaches. However, for multi-label classification task on FIRE16 and SMERP17, SVM with one-vs-rest approach outperforms all other classifiers.

Is our framework effective over supervised approach? To verify the effectiveness of our framework over the standard supervised approach, we tested the performance of the best performing classifier, with and without our framework, on the test set. Findings from Table 4 shows that our framework SSF using unlabelled data improves overall performance over supervised learning Sup. for all datasets. In 19 out of 21 classifier-dataset combinations in Table 4, our approach SSF improves classifier performance except for GNB and kNN algorithm on the SMERP17 data. This improvement is due to the incorporation of critical out-of-vocabulary words from unlabelled data which helps expand the feature space. Incorporation of new words to the vocabulary helps to improve the Recall score. Figures 2 and 3 compare the improvement in Precision and Recall scores between Sup. and SSF approach. Improvement in Kaggle is minimal as the unlabelled data is limited, which restricts the expansion of the feature space. Our framework works efficiently when sufficient unlabelled data is supplied as seen with the other two datasets.

Second and last column in Table 5 summaries the class-wise performance between Sup. and SSF For FIRE16, F_1 score for 5 out of 7 classes improves with a marginal drop for class 1. We can see that performance for class 4 and 5 in FIRE16 improved by a large margin

Table 4: Best performing [Macro-F_1 (SD)] classifier calculated over 10 runs for each dataset. "Sup." and "SSF" denotes supervised and semi-supervised performance respectively.

Algo	Kaggle		FIRE16		SMERP17	
	Sup.	***SSF***	***Sup.***	***SSF***	***Sup.***	***SSF***
GNB	0.6111 (.0053)	0.7024 (.0099)	0.4678 (.0126)	0.4874 (.0123)	0.6370 (.0045)	0.6312 (.0056)
AB	0.6485 (.0054)	0.6672 (.0036)	0.5567 (.0057)	0.5653 (.0072)	0.7972 (.0095)	0.8250 (.0052)
DT	0.6499 (.0021)	0.7043 (.0049)	0.5365 (.0043)	0.5397 (.0089)	0.7523 (.0063)	0.7624 (.0083)
RF	0.6862 (.0067)	0.7138 (.0074)	0.5485 (.0096)	0.5661 (.0081)	0.7877 (.0092)	0.8020 (.0094)
kNN	0.7332 (.0126)	0.7465 (.0165)	0.5061 (.0235)	0.5548 (.0156)	0.8275 (.0269)	0.7789 (.0307)
LR	**0.7609** (.0027)	**0.7699** (.0041)	0.5595 (.0058)	0.5879 (.0063)	0.8331 (.0065)	0.8500 (.0083)
SVM	0.7351 (.0068)	0.7376 (.0097)	**0.5868** (.0085)	**0.6118** (.0102)	**0.8523** (.0026)	**0.8660** (.0064)

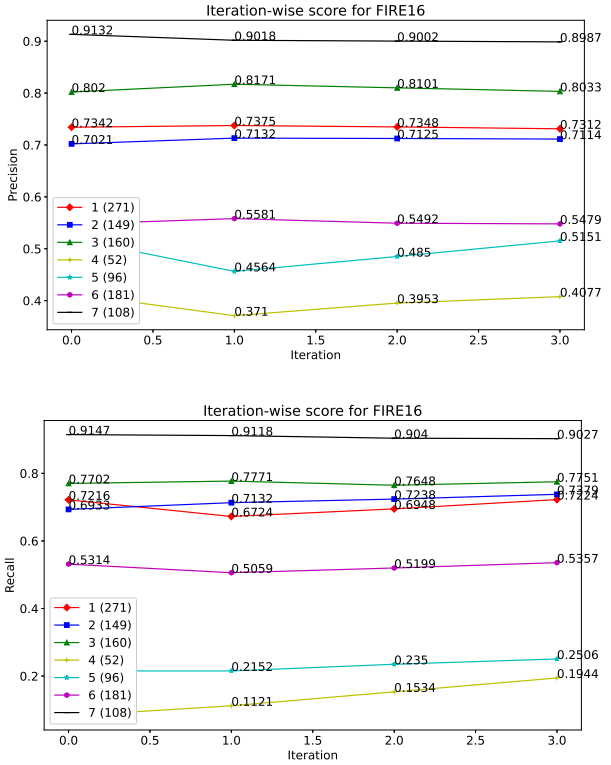

Figure 4: Iteration-wise Precision and Recall scores for individual classes of FIRE16.

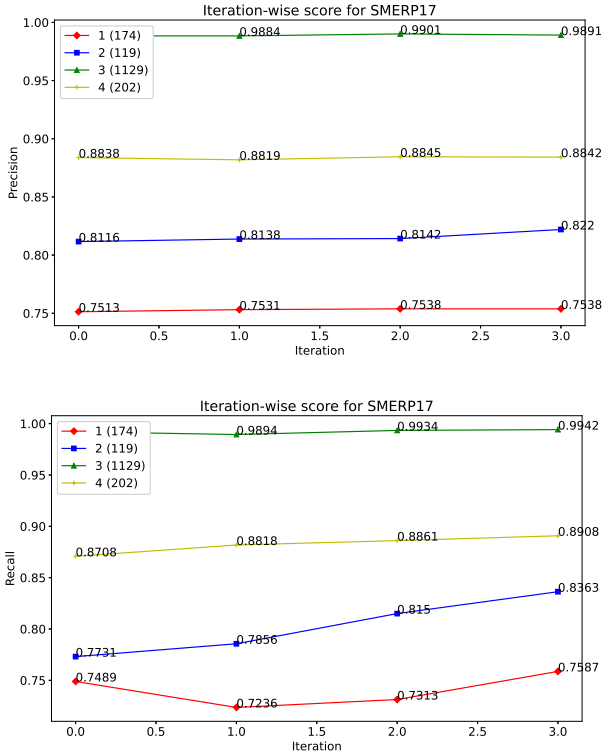

Figure 5: Iteration-wise Precision and Recall scores for individual classes of SMERP17.

of 88.67% and 10.78% respectively due to significant improvement in Recall as seen in Figure 2. Supervised performance for class 7 is high with a comparatively lower number of samples in the training set because of distinct feature set. Incorporation of new samples results in low performance for class 7. For SMERP17, the framework outperforms Sup. for all classes.

Does Capacity criterion make a difference? In Table 5, column "Sampling (1.0)" refers to a scenario when there was only threshold criterion in place [3, 8, 14]. All samples for which the predicted confidence scores were above the threshold were selected and assigned to the corresponding class(es). In such a scenario, we found that most of the predicted samples belong to the class with majority representation in training set with a high confidence score. This phenomenon happens because the availability of a large number of training samples for the majority class helps in better learning for the class, and thereby resulting in higher and more reliable confidence scores for the class.

Moreover, a similar distribution of the samples can be expected to be present in the unlabelled dataset. Hence, a large number of examples from the unlabelled data have high confidence scores for the majority class, whereas, for the minority or resource-constrained classes, much lesser number of samples have such high confidence scores. Hence, the method ends up adding more number of majority class examples to the training set during the iterations of the semi-supervised framework. Adding samples without capacity limit works best when the size of the unlabelled data is limited, or

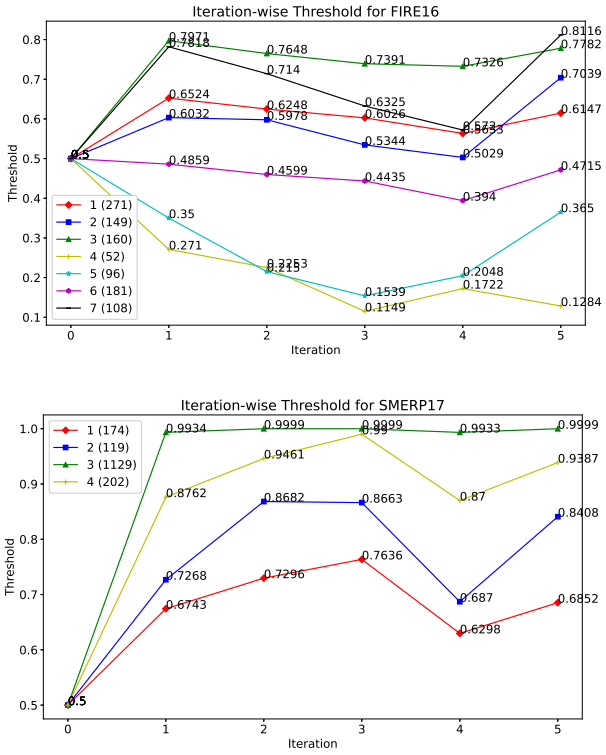

Figure 6: Evolution of thresholds with iteration for FIRE16 (top) and SMERP17 (bottom).

the proportion of relevant samples are very high in the unlabelled data. This behaviour can be seen for Kaggle in Table 5a.

Does Threshold criterion make a difference? The purpose of the Threshold criterion is to carefully filter out low confidence samples based on the performance of that class. Performance of a class is related to the number of training samples. Our Threshold criterion lowers the threshold to incorporate a more significant amount of data for resource-constrained classes, as seen in Figure 6. On the other hand, increases the threshold for well-performing classes. This careful selection helps to improve the Recall for resource constrained classes without harming the Precision of other classes. Without any restriction on the threshold, very low confidence samples might get selected, resulting in lower Precision. We again turn to Table 5 to observe this behaviour. Columns "Add count (25, 50 and 75)" refers to scenarios where a fixed number of top samples for each class are selected without any restriction on the confidence. The Precision score for class 1 in FIRE16 and class 4 in SMERP17 is high[5] (0.7357 and 0.7576) when only (25) samples are selected. However, it falls sharply (0.6975 and 0.7361) as more samples (75) are selected, resulting in a lower F_1 score.

Are both criteria necessary? With the previous two experiments, we found that there is a need to have capacity restriction to avoid adding majority class samples. On the other hand, a threshold restriction makes sure that very low confidence samples are not incorporated for self-learning.

[5]Due to lack of space, Precision and Recall tables were not presented in the paper.

Is the proposed framework effective over other existing semi-supervised baselines? The proposed framework is compared with four baselines; supervised classifier trained only on labelled data, random sampling-based semi-supervised classifier as mentioned in [8, 14] and finally with [3] which has a threshold restriction but without any restriction on the number of samples to be added per class. To make a fair comparison, we use the same threshold value in all the baselines to convert the confidence score to class labels. The result are summarised in Table 5. Our framework outperforms all the other baseline except for Kaggle. Selecting all samples above a threshold, as proposed in [3], is suitable when the distribution of relevant samples are high in the unlabelled data. However, typically that's not the scenario for short-texts generated during disasters. A significant portion of the generated short-texts are either sentiment of people or irrelevant to the disaster. The proposed framework performs better as such scenarios in FIRE16 and SMERP17, as seen in Table 5. Random sampling with different rates (0.4, 0.6, 0.8) [8] performs well in a few specific cases when a small set of samples are assigned to that class, such as for class 6 for FIRE16.

How effective is our framework for resource-constrained classes? Next, we experimented with our framework in the resource constrained scenario. It can be seen from Figures 4 and 5 for FIRE16 and SMERP17 that the Precision for all the classes remain almost constant across iteration. However, the Recall scores for the resource-constrained classes keep on improving. This improvement happens due to the careful selection of the new examples to be added to the training set at each iteration. Iteration 0 refers to supervised iteration, initial dip at iteration 1 is due to the improper initialization of thresholds. The framework updates the threshold of each class according to the performance.

To verify the effectiveness of our framework in resource constrained scenario, we randomly selected samples of a class from the training set to make that class resource-constrained. FIRE16 already has two resource-constrained classes (Class 4 and 5) with less than 100 samples. For remaining classes in FIRE16 and SMERP17, we randomly select 80%, 50% and 20% of samples for a class from the training set. Note that selecting samples of a class may result in a small number of samples being removed from other classes due to multi-label nature.

Figures 7 and 8 summarize our findings for resource-constrained classes for FIRE16 and SMERP17. Results show as we reduce the number of training samples for a class, performance gap between our framework over the supervised approach increases. For example, the F_1 score difference between Sup. and SSF for class 1 of FIRE16 increases from 0.58% to 3.32% and finally 22.30% for 80%, 50% and 20% of the training set. Similar patterns of more significant performance gap can be found for all classes.

On improvements in performance and its impact: From the above analysis, we see that the proposed framework results in overall performance improvement for all datasets. Almost all the cases, the Recall scores of the classes improved, without any significant reduction in the Precision. For disaster mitigation, even a small improvement in the Recall would indicate being able to identify and retrieve more number of relevant short-texts for the classes, where classes are connected with situations at the affected regions. Identifying more number of such situational information would

Table 5: Performance [Macro-F_1 (SD)] comparison between our approach and baselines calculated over 10 runs.

(a) Kaggle

Class	Sup.	Add Count			Sampling				SSF
		25	*50*	*75*	*0.4*	*0.6*	*0.8*	*1.0*	
0	0.8054 (.0021)	0.8056 (.0027)	0.8073 (.0033)	0.8075 (.0039)	0.8087 (.0064)	0.8104 (.0050)	0.8106 (.0042)	**0.8121** (.0034)	0.8101 (.0043)
1	0.7165 (.0043)	0.7175 (.0047)	0.7203 (.0052)	0.7200 (.0063)	0.7243 (.0068)	0.7283 (.0057)	0.7297 (.0051)	**0.7311** (.0058)	0.7298 (.0041)

(b) FIRE16

Class	Sup.	Add Count			Sampling				SSF
		25	*50*	*75*	*0.4*	*0.6*	*0.8*	*1.0*	
1	**0.7278** (.0116)	0.7247 (.0141)	0.7047 (.0148)	0.6915 (.0163)	0.6903 (.0216)	0.6814 (.0204)	0.6794 (.0188)	0.6770 (.0173)	0.7267 (.0168)
2	0.6976 (.0170)	0.7097 (.0185)	0.6973 (.0193)	0.6934 (.0201)	0.7121 (.0234)	0.7145 (.0221)	0.7192 (.0245)	0.6894 (.0267)	**0.7244** (.0173)
3	0.7857 (.0104)	0.7831 (.0158)	0.7731 (.0171)	0.7702 (.0165)	0.7874 (.0198)	0.7878 (.0202)	0.7831 (.0191)	0.7657 (.0179)	**0.7889** (.0140)
4	0.1395 (.0278)	0.1790 (.0326)	0.2045 (.0341)	0.2180 (.0333)	0.2395 (.0352)	0.2425 (.0346)	0.2484 (.0361)	0.2512 (.0348)	**0.2632** (.0338)
5	0.3043 (.0271)	0.3074 (.0311)	0.3127 (.0327)	0.3226 (.0357)	0.3183 (.0348)	0.3262 (.0353)	0.3322 (.0324)	0.3341 (.0249)	**0.3371** (.0276)
6	0.5392 (.0166)	0.5342 (.0171)	0.5107 (.0168)	0.5262 (.0192)	0.5402 (.0214)	0.5414 (.0207)	**0.5423** (.0176)	0.5373 (.0221)	0.5417 (.0156)
7	**0.9139** (.0089)	0.9063 (.0154)	0.8963 (.0147)	0.8938 (.0151)	0.8904 (.0142)	0.9018 (.0187)	0.9024 (.0183)	0.8967 (.0158)	0.9006 (.0133)

(c) SMERP17

Class	Sup.	Add Count			Sampling				SSF
		25	*50*	*75*	*0.4*	*0.6*	*0.8*	*1.0*	
1	0.7500 (.0098)	0.7510 (.0120)	0.7523 (.0114)	0.7471 (.0138)	0.7413 (.0141)	0.7542 (.0104)	0.7451 (.0176)	0.7504 (.0121)	**0.7562** (.0129)
2	0.7918 (.0163)	0.7922 (.0144)	0.7854 (.0155)	0.7819 (.0146)	0.7924 (.0154)	0.8039 (.0187)	0.8113 (.0137)	0.7890 (.0172)	**0.8290** (.0170)
3	0.9902 (.0041)	0.9814 (.0047)	0.9773 (.0071)	0.9724 (.0065)	0.9781 (.0059)	0.9795 (.0066)	0.9817 (.0054)	0.9744 (.0097)	**0.9916** (.0032)
4	0.8772 (.0095)	**0.8879** (.0108)	0.8815 (.0115)	0.8768 (.0172)	0.8721 (.0119)	0.8704 (.0126)	0.8698 (.0154)	0.8457 (.0213)	0.8874 (.0130)

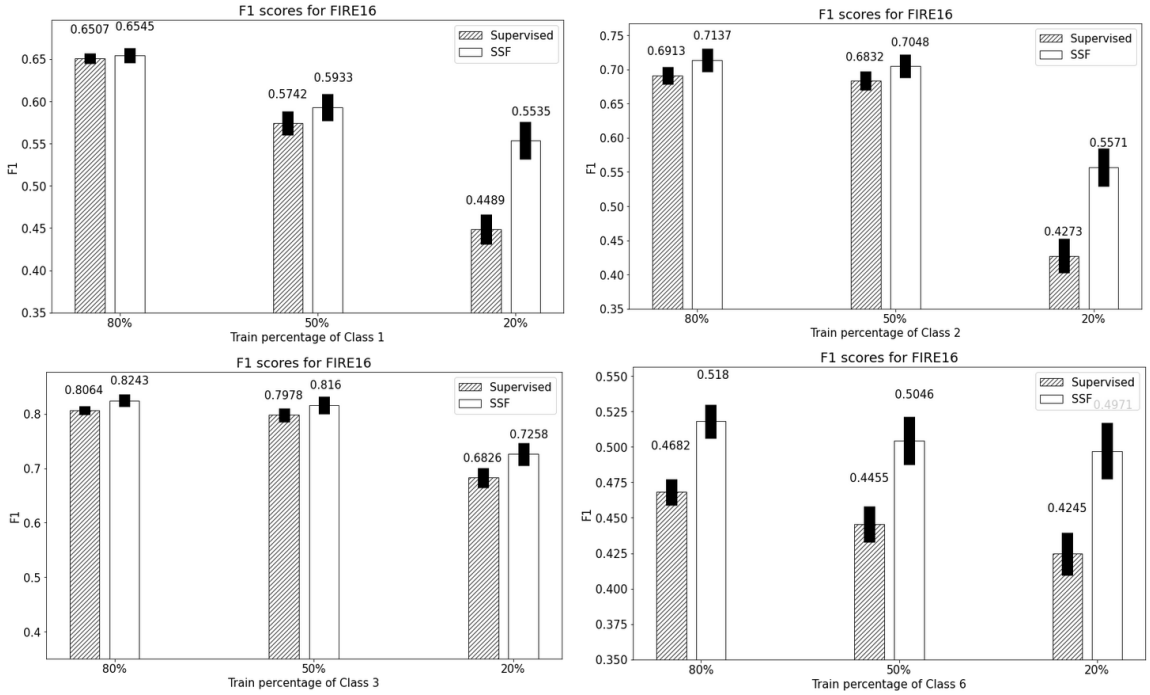

Figure 7: Performance improvement of resource-constrained classes with 80%, 50% and 20% of training data for FIRE16 (Black rectangle represents SD).

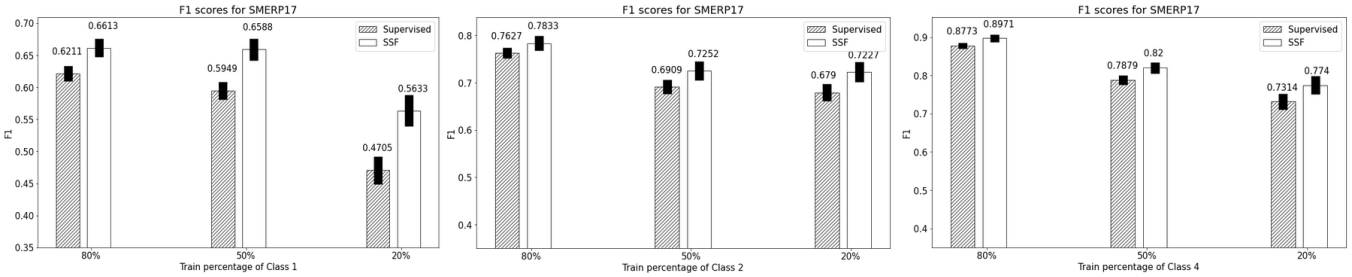

Figure 8: Performance improvement of resource-constrained classes with 80%, 50% and 20% of training data for SMERP17.

help in planning the rescue and relief operations, and thereby being able to reach out to more number of affected people in a timely manner, and helping them in fighting the adverse conditions. Noticeably, we see that the Recall scores for the resource-constrained classes improved by a large margin. In the datasets considered, the available resource-constrained classes happened to be the classes related to "Medical resources required", "Requirement/availability of resources at specific locations" and "Resources required." Identification of more number of short-texts from these classes from the massive stream of short-texts would directly help the people in the affected regions.

7 CONCLUSION AND FUTURE WORK

In this work, we studied the usefulness of semi-supervised learning to classify short-texts generated during ongoing disasters to predefined granular classes. We applied the proposed self-learning based semi-supervised learning framework in three disaster-related datasets to show it's effectiveness over supervised algorithms and other semi-supervised learning based baselines. Specifically, under the assumption that the distribution of classes in the real world is imbalanced and a sufficient number of samples might not be available, our proposed framework outperforms supervised and other baselines significantly for resource-constrained classes by adaptively allowing more samples to be selected for such classes. By incrementally incorporating unlabelled data, we can effectively manage rescue operations during ongoing disasters. Our study also provides useful insights into model performance when labelled data is limited and imbalanced. While Precision of resource-constrained classes was moderate, Recall score is deficient potentially missing to retrieve much crucial situational information. More exploration is needed from the effects of feature generation techniques such as embedding based techniques, to gain better insights on the effects of selecting unlabeled data. We plan to explore such effects in our future work.

ACKNOWLEDGMENTS

This work is supported by the Vivesvaraya PhD Scheme for Electronics and IT, Ministry of Electronics and Information Technology (MeitY), Government of India under Grant No.:
EE/2016-17/034/MLA/MZAK/0235.

REFERENCES

[1] Cornelia Caragea, Nathan McNeese, Anuj Jaiswal, Greg Traylor, Hyun-Woo Kim, Prasenjit Mitra, Dinghao Wu, Andrea H Tapia, Lee Giles, Bernard J Jansen, et al. 2011. Classifying text messages for the Haiti earthquake. In *Proceedings of the 8th international conference on information systems for crisis response and management (ISCRAM2011)*. Citeseer.

[2] Cornelia Caragea, Adrian Silvescu, and Andrea H Tapia. 2016. Identifying Informative Messages in Disasters using Convolutional Neural Networks.. In *ISCRAM*.

[3] Oduwa Edo-Osagie, Gillian Smith, Iain Lake, Obaghe Edeghere, and Beatriz De La Iglesia. 2019. Twitter mining using semi-supervised classification for relevance filtering in syndromic surveillance. *PloS one* 14, 7 (2019).

[4] Samujjwal Ghosh and Maunendra Sankar Desarkar. 2018. Class specific TF-IDF boosting for short-text classification: Application to short-texts generated during disasters. In *Companion Proceedings of the The Web Conference 2018*. 1629–1637.

[5] Saptarshi Ghosh and Kripabandhu Ghosh. 2016. Overview of the FIRE 2016 Microblog track: Information Extraction from Microblogs Posted during Disasters.. In *FIRE (Working Notes)*. 56–61.

[6] Alec Go, Richa Bhayani, and Lei Huang. 2009. Twitter sentiment classification using distant supervision. *CS224N project report, Stanford* 1, 12 (2009), 2009.

[7] Muhammad Imran, Carlos Castillo, Fernando Diaz, and Sarah Vieweg. 2015. Processing social media messages in mass emergency: A survey. *ACM Computing Surveys (CSUR)* 47, 4 (2015), 1–38.

[8] Vivian Lay Shan Lee, Keng Hoon Gan, Tien Ping Tan, and Rosni Abdullah. 2019. Semi-supervised Learning for Sentiment Classification using Small Number of Labeled Data. *Procedia Computer Science* 161 (2019), 577–584.

[9] Hongmin Li, Nicolais Guevara, Nic Herndon, Doina Caragea, Kishore Neppalli, Cornelia Caragea, Anna Cinzia Squicciarini, and Andrea H Tapia. 2015. Twitter Mining for Disaster Response: A Domain Adaptation Approach.. In *ISCRAM*.

[10] Dina Fine Maron. 2013. How Social Media Is Changing Disaster Response. *Scientific American* (2013). https://www.scientificamerican.com/article/how-social-media-is-changing-disaster-response/

[11] Hossein Mobahi, Mehrdad Farajtabar, and Peter L. Bartlett. 2020. Self-Distillation Amplifies Regularization in Hilbert Space. arXiv:cs.LG/2002.05715

[12] Ken Moule. 2012. Situation awareness for disaster management in the information age. *Global GBM* (2012).

[13] Alexandra Olteanu, Carlos Castillo, Fernando Diaz, and Sarah Vieweg. 2014. Crisislex: A lexicon for collecting and filtering microblogged communications in crises. In *Eighth International AAAI Conference on Weblogs and Social Media*.

[14] Neha Pandey and S Natarajan. 2016. How social media can contribute during disaster events? Case study of Chennai floods 2015. In *2016 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*. IEEE, 1352–1356.

[15] V Pekar, J Binner, and H Najafi. 2016. Detecting Mass Emergency Events on Social Media: One Classification Problem or Many?. In *Proceedings of the International Conference on Data Mining (DMIN)*. The Steering Committee of The World Congress in Computer Science, Computer …, 31.

[16] J Platt. 1999. Probabilistic Outputs for SVMs and Comparisons to Regularized Likehood Methods, Advances in Large Margin Classifiers.

[17] Takeshi Sakaki, Makoto Okazaki, and Yutaka Matsuo. 2010. Earthquake shakes Twitter users: real-time event detection by social sensors. In *Proceedings of the 19th international conference on World wide web*. ACM, 851–860.

[18] Tanmoy Chakraborty Debasis Ganguly Gareth Jones Marie-Francine Moens Saptarshi Ghosh, Kripabandhu Ghosh. 2017. First International Workshop on Exploitation of Social Media for Emergency Reliefand Preparedness (SMERP). In *39th European Conference on IR Research, ECIR*. LNCS 10193. https://doi.org/10.1007/978-3-319-56608-5

[19] Isaac Triguero, Salvador García, and Francisco Herrera. 2015. Self-labeled techniques for semi-supervised learning: taxonomy, software and empirical study. *Knowledge and Information systems* 42, 2 (2015), 245–284.

[20] David Yarowsky. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In *33rd annual meeting of the association for computational linguistics*. 189–196.

[21] Shanshan Zhang and Slobodan Vucetic. 2016. Semi-supervised discovery of informative tweets during the emerging disasters. *arXiv preprint arXiv:1610.03750* (2016).