UNIVERSITY OF OKLAHOMA

GRADUATE COLLEGE

GENERAL SUPERVISED LEARNING FRAMEWORK FOR OPEN WORLD CLASSIFICATION

A DISSERTATION

SUBMITTED TO THE GRADUATE FACULTY

in partial fulfillment of the requirements for the

Degree of

DOCTOR OF PHILOSOPHY

By

SAI KRISHNA THEJA BHAVARAJU
Norman, Oklahoma
2020

GENERAL SUPERVISED LEARNING FRAMEWORK
FOR OPEN WORLD CLASSIFICATION

A DISSERTATION APPROVED FOR THE
SCHOOL OF INDUSTRIAL AND SYSTEMS ENGINEERING

BY THE COMMITTEE CONSISTING OF

Dr. Charles Nicholson, Chair

Dr. Christan Grant

Dr. Andres Gonzalez

Dr. Shivakumar Raman

Dr. Theodore Trafalis

Acknowledgements

Firstly, I would like to express my sincere gratitude to my advisor Dr. Charles Nicholson for the continuous support, for his patience, motivation, and immense knowledge. His guidance helped me in all the time of research and writing of this thesis. I could not have imagined having a better advisor and mentor for my Ph.D study. Besides my advisor, I would like to thank the rest of my thesis committee: Dr. Christan Grant, Dr. Andres Gonzalez, Dr. Shivakumar Raman, and Dr. Theodore Trafalis for their insightful suggestions and encouragement. I would like to specially thank Dr. Saptarshi Mandal for all the intellectual discussions and brain storming sessions we had. Last but not the least, I would like to thank my family: my parents for supporting me and keeping me motivated throughout my journey. This would not have been possible without their moral support. Also, I thank my friends and my roommates for their continued help and encouragement.

Contents

List of Tables

Chapter 1

Introduction

1.1 Understanding the open world classification problem

In traditional supervised learning for classification, models are trained based on data sets that contain examples of all classes to be identified. That is, in so called *closed world* problems, all classes are known in advance. Once the model is trained, it can used to predict or otherwise discriminate between these same classes in new data. However, there are problems where this condition does not hold. That is, in *open world* problems, the training data is incomplete and the new data for which the model has been developed may contain classes which the model was not trained on. This problem has its origins in the field of computer vision for recognition in which a target image class or set of classes should be recognized among known and unknown image classes. Indeed, much of the research on open world problems is concentrated within the domain of computer vision and referred to as open set recognition (OSR) or open world recognition

(OWR). In [1], one of the earliest works in OSR, the authors formally defined terms such as open space and openness that correspond to the problem of open set recognition, and also show how this problem setting is different from general data modeling task. In [2], the authors extend the concept of OSR to OWR and identify the necessary tasks for an effective system, i.e., the system should be able to detect unknown classes, label unknown points, and update the model. The open world supervised learning scenario is not limited to computer vision, but applies to numerous domains associated with traditional classification problems, i.e., open world classification is applicable to any classification problem without a guarantee on the exhaustiveness of the training classes. Indeed, it is plausible that one not know a priori if a multi-class problem is open or closed. Given this broad spectrum for application, the open world condition applies to problems with vastly different data types, sources, and characteristics which includes image, text, sensor data, or more traditional structured data types.

The present work addresses this with a general framework that tackles this broad perspective. For open world classification (OWC), ideally a model is trained on a finite set of known classes and when applied to new data, it is able to accurately address four tasks: (i) label all known classes, (ii) identify the instances associated with the new, unknown classes, (iii) create new, distinct classes for these instances, and (iv) update itself without losing predictive performance power on the known classes and be able to consistently classify the unknown classes in new data. These tasks are similar to those from [2], however, unlike their work, here we are only interested in an approach which accomplishes these goals automatically and without human intervention. Furthermore, OWC should not be limited to only specific machine learning (ML) techniques, since depending on the domain and data type, modelers often use different type of clas-

sifiers for the problem at hand. The present study addresses these goals with a framework that applies irrespective of the type of data or nature of the classifier.

1.2 Contributions of this work

The present work develops a general framework that tackles the problem of open world classification (OWC), by addressing the following four tasks: (i) label all known classes, (ii) identify the instances associated with the new, unknown classes, (iii) create new, distinct classes for these instances, and (iv) update itself without losing predictive performance power on the known classes and be able to consistently classify the unknown classes in new data. However, the first task is not the focus of this paper and the remaining three tasks can be broadly classified into two categories namely, identification and categorization. While the task of identification involves identifying data instances associated with the new or unknown classes, categorization deals with the discovery of the unknown classes and also categorizing the identified instances into their respective classes, without human intervention. The contributions of this paper can be attributed to these two tasks and are as follows:

Identification of unknown data Most of the works in the literature address the open world problem in the domain of computer vision. Furthermore, the solutions proposed to this problem are very algorithmic specific and are tied to a specific machine learning algorithm, which restricts other researchers from using classifiers of their interest. To address this issue, we approach this problem with a two stage process by separating the training phase from the identification phase. In Chapter 3, we developed an algorithm that identifies the data instances from

open world data that belong to unknown classes. This algorithm can be used in conjunction with any classifier and also the type of the data.

Categorization of unknown data Most of the works in the literature, propose solutions to the identification task as a complete solution to the open world problem. One the other hand, identifying the data belonging to unknown classes as the only solution, it is important to discover the number of new classes and also to categorize the instances accordingly. Furthermore, the tasks of class discovery and categorization should not include any human intervention. In Chapter 4, we developed an algorithm that discovers the unknown classes and also categorize the instances into their respective classes, without any human intervention.

Chapter 2

Literature review

2.1 Introduction

In the previous chapter, four tasks corresponding to OWC have been stated, of which the first task is not the focus of this paper, assuming a high quality classifier already exists. However, the remaining three tasks can be broadly classified into two categories namely, identification and categorization. While identification deals with identifying instances associated with the new or unknown classes, categorization involves categorizing these instances into new, distinct classes. Furthermore, the task of updating the model with an additional capability of classifying the new classes, is also considered a part of the categorization phase. The works available in the literature are grouped into these two categories and presented in this section.

2.2 Identification of unknown classes

As specified in the previous section, OSR based works, constitute major portion of literature related to this task and we make an effort to present some of the noticeable works in this section.

Open set recognition as defined in [1] provides a mathematical foundation and basic definitions for open world problems. Furthermore, in the same work, the authors modify a support vector machine to design a 1-vs-set machine to address OSR. [3] extended this to deal with multiple target classes and formulated a Weibull-Calibrated SVM in conjunction with a technique based on compact abating probability to identify the data instances belonging to unknown classes. Similarly, [4, 5] have modified a traditional SVM classifier to adapt to the open world setting. Alternatively, there are works [2, 6, 7] in the literature based on the Nearest Class Mean (NCM) classifier to solve the problem of open set recognition. Similarly, [8] developed an open set classifier that is an extension to the nearest neighbor classifier for OSR. [9]

Apart from considering the traditional ML models for OSR, there are works that address this problem from a deep learning perspective. One of the early works corresponding to this progression is done by [10], where the authors replaced the SoftMax layer of a neural network with an OpenMax layer to adapt to the problem of open set recognition. Similarly, [11, 12] developed methods to handle OSR by making algorithmic modifications to the output layer of a deep neural network. On the other hand, [13] tried to solve the problem of open set recognition by developing a neural network based representation of the data samples. [14] introduced a modified version of an autoencoder to solve for OSR. In their next work, Multi task learning based CNN for OSR was developed by [15].

Applications of OSR in various fields such as malware detection, face recognition, text classification etc., were done by [11, 16–20]. While most of the deep learning based works discussed so far are discriminative approaches, there are works in the literature that address this problem in a generative perspective as well. Similar to the OpenMax classifier discussed previously, [21] developed a generative openmax classifier to identify data points from unknown classes. Conversely, [22] proposed a new data augmentation technique called counterfactual image generation (OSRCI) in conjunction with generated adversarial network (GAN) to address the problem of OSR. Other works in the field of OSR include [23–30]. A more detailed survey about the various works in this field is provided by [31].

2.3 Categorization of unknown classes

All the works in the domain of open set recognition that have been discussed so far mainly focuses on just the identification of the data instances coming from unknown classes; they do not provide any solutions as to how the identified data can be organized into their respective classes. However, we have identified few works that try to solve this problem. In [2], the authors tried to categorize the data instances by human intervention, where the identified data instances were manually labelled and used to further update the model. In [32], the authors developed a neural network based model called Pairwise Classification Network (PCN) to identify if two images belong to the same class or a different class, along with another deep learning model called Open Classification Network (OCN) for open image classification. The predictions made by PCN are used as a distance function for hierarchical clustering to cluster the data points. In [33], on the other hand, tried to solve this problem from a generative modeling perspective where

they implement modified Hierarchical Dirichlet Process (HDP) to model both the training and test data, and also to come up with an estimate as to how many clusters are possibly present in the test data. They claim that this estimate can be further used as a prior to clustering algorithms such as K-means, etc. Considering these works, the solution provided by [2] does not satisfy our need completely, as we would want a technique that automatically identifies the clusters in the data without any human intervention. On the other hand, [32] and [33] also provide an estimation as opposed to accurate value about the possible number of clusters in the data. On the whole, we can understand that the categorization of the identified data instances is still an unsolved problem, and there is a need for a solution that can automatically and accurately organize the identified instances.

2.4 Other related work

We briefly mention areas within supervised learning that have some conceptual similarities to open world classification, yet are distinct from and pose entirely different challenges than open world problems. *Semi-supervised learning* train models on relatively small subsets of labeled data and a large set of unlabeled data [34]. *Few shots learning* [35–40], *one shot learning* [41, 42], and *zero shot learning* [43–45] are extreme forms of semi-supervised learning. In the latter case, while models are trained to make predictions on data that belong to classes not seen during training, the algorithm is provided semantic or attribute information regarding the unseen classes [31]. *Domain adaption* machine learning is designed for scenarios in which the training data and the test data have different data distributions [46–50]. In *concept drift* problems [51–53], the data distribution changes dynamically over time and the ML model must update itself. However,

in all of these domains, all classes are known in advance of model training. This includes the number of classes that may be encountered as well examples of all classes and/or information about all classes. In the open world classification problem we address, we assume no information about the quantity or quality of any unknown classes in the test data.

2.5 Gaps and research objectives

While there are much fewer works that try to provide solution to the problem of open world classification, there are certain drawbacks associated with them.

- Most of the works discussed above are applied especially to the domain of computer vision.
- All the works make algorithmic changes to the existing machine learning models to come with a classifier that can handle the open world problem.
- To the best of our knowledge, there are no signs of works that can be applied to any given type of data.
- There are no techniques available in the literature that can accurately categorize the data instances into their categories.

Taking the above mentioned drawbacks into consideration, the following research objectives have been formulated, and will be addressed in the present work.

- *Research objective 1: To develop a methodology that can handle the open world problem for any type of data.*

- *Research objective 2: To come up with a technique that can automatically categorize the data instances from unknown classes without human intervention*

Chapter 3

Identification of data that belong to unknown classes

3.1 Background and methodology

Let $\mathcal{K}$ denote the set of all known classes and $\mathcal{U}$ denote the set of unknown classes. Let $\mathcal{T}$ denote the training data containing instances of $k = |\mathcal{K}|$ known classes and $\mathcal{O}$ denote the open world test data which may contain up to $k + u$ classes, where $u = |\mathcal{U}|$. Let n_{T} denote the number of observations in $\mathcal{T}$ and n_{O} denote the number of observations in $\mathcal{O}$. For a given observation i, let $c_i \in \mathcal{K} \cup \mathcal{U}$ denote the true class. Both $\mathcal{T}$ and $\mathcal{O}$ have the same number of features, m. Let $\mathcal{C}$ be any classifier, trained on $\mathcal{T}$, which maps the m-dimensional training or test data to k probability values, $0 \leq p_j \leq 1 \ \forall j \in \mathcal{K}$. Additionally, $\sum_{j \in \mathcal{K}} p_j = 1$. If X denotes the input data, then $\mathcal{C} : X \to \mathbb{R}^k$, whereas the predicted class is given by $\hat{c} = \arg\max_{j \in \mathcal{K}} p_j$.

There are basically two key issues to be addressed in the context of open world problem, of which the first motive is to identify the data instances belonging to

the unknown classes (identification) and the second motive is to categorize these identified data points into their respective categories (categorization).

The task of identifying unknown classes is necessary when a trained classifier is applied to the open world test data. Hence, it sufficient to have an identification method that occurs during the application phase. As mentioned in the previous section, a lot of works in the field of open world problem have been limited to computer vision and are mostly algorithmic modifications to existing machine learning models. Furthermore, these works try to integrate the task of identification with the actual model training, making it a single process.

However, to provide a general approach to identify instances in $\mathcal{O}$ associated with classes in $\mathcal{U}$, we separate model training or algorithm specifications from the identification task. This is accomplished by performing anomaly detection on the k-dimensional probability space generated by $\mathcal{C}(\mathcal{T})$ and $\mathcal{C}(\mathcal{O})$. The identification task is now independent of the type of probabilistic classifier and makes no assumptions on the original input data.

The probability distribution generated by $\mathcal{C}(\mathcal{T})$ is a function of the quality of the classifier and the separability of the classes in the training data. However, the k-dimensional probability space is independent of the type of data associated with any problem domain. For perfect probabilistic classifiers, not only is the predicted class correct for any observation i, i.e., $\hat{c}_i = c_i$, but also each p_j is a k-dimensional binary vector. If such a classifier were applied to an observation associated with class $j \notin \mathcal{K}$, the k-dimensional space would be inadequate to represent the entity. Nonetheless, the observation would be mapped as to the space. The projection of such a point would likely be a non-binary vector of probabilities. For less than perfect classifiers, we operate under the hypothesis that the k-dimensional probability distribution for a known class is detectably

distinct from that of the distribution for an unknown class.

There are a variety of metrics to measure dissimiliarity between probability distributions including Kullback–Leibler (KL) divergence [54] and Jensen-Shannon distance (JSD) [55]. For two probability distributions P and Q on the same probability space X, the KL divergence D_{KL} is computed as

$$D_{\mathrm{KL}}\left(P \| Q\right) = \sum_{x \in X} P(x) \log \frac{P(x)}{Q(x)}$$

The JSD is a symmetric dissimilarity measurement based on the asymmetric KL divergence. The JSD between two probability distributions P and Q is

$$D_{\mathrm{JSD}}\left(P \| Q\right) = \sqrt{\frac{1}{2} D_{\mathrm{KL}}\left(P \| M\right) + \frac{1}{2} D_{\mathrm{KL}}\left(Q \| M\right)}$$

where $M = \frac{1}{2}\left(P + Q\right)$.

3.2 Algorithm for anomaly detection using Jensen-Shannon distance

The algorithm for identification of unknown classes based on JSD follows. First, the predicted probabilities from the trained classifier are determined. Specifically, $\mathbf{P}_{\mathrm{train}}$ is an $n \times k$ probability matrix generated from the application of $\mathcal{C}$ on $\mathcal{T}$. For each class $j \in \mathcal{K}$, compute the centroid, $\tilde{c}_j$, for the set of observations $C_j = \{i \in \mathbf{P}_{\mathrm{train}} : c_i = j\}$ of the predicted probability distributions from $\mathbf{P}_{\mathrm{train}}$. Next, compute the D_{JSD} between every observation $i \in C_j$ and each $\tilde{c}_j$ and determine the corresponding mean, μ_j, and standard deviation, σ_j, of the dissimilarity values between each point and each centroid.

Generate the prediction probabilities for every data point in O using $\mathcal{C}$ and store in $\mathbf{P}_{\text{test}}$. The JSD for every point $i \in \mathbf{P}_{\text{test}}$ to the previously determined centroids is computed, $D_{\text{JSD}}(i, \tilde{c}_j)\ \forall j \in \mathcal{K}$. These values are then mean-centered and scaled using μ_j and σ_j.

For each class $j \in \mathcal{K}$, if the value of the test point i from $\tilde{c}_j$ is greater than critical value (τ), the data point is identified as anomaly and is stored in $\mathcal{F}$. The value of τ is determined using Chebyshev's inequality [56] based on a confidence parameter (α). Since we are unaware of the underlying distribution from which the distances are generated, using Chebyshev's inequality will provide generalization to the method as it can be used for any data set irrespective of its underlying distribution.

Algorithm 1 Identify outsiders (Jensen-Shannon Distance)

Input:

training data $\mathcal{T}$ with n observations
open world data $\mathcal{O}$
probabilistic classifier $\mathcal{C}$
confidence parameter α

Output:

$\mathcal{F}$: flagged data

1: $\mathbf{P}_{\text{train}} \leftarrow \mathcal{C}(\mathcal{T})$
2: Compute centroids $\tilde{c}_j\ \forall j \in \mathcal{K}$ from $\mathbf{P}_{train}$
3: Compute $D_{\text{JSD}}(i, \tilde{c}_j)\ \forall i \in \mathbf{P}_{train}$ and $\mu_j, \sigma_j\ \forall j \in \mathcal{K}$
4: $\mathbf{P}_{\text{test}} \leftarrow \mathcal{C}(\mathcal{O})$
5: For each $j \in \mathcal{K}$, compute $D_{\text{JSD}}(i, \tilde{c}_j)\ \forall i \in \mathbf{P}_{test}$ and mean-center and scale using μ_j and σ_j
6: Determine τ using Chebyshev's inequality for a confidence parameter α
7: For each $i \in \mathbf{P}_{\text{test}}$, if the scaled value of $D_{\text{JSD}}(i, \tilde{c}_j) \geq\ \tau,\ \forall j \in \mathcal{K}$, identify i as an anomaly and store it in $\mathcal{F}$.

Chapter 4

Categorization of data that belong to unknown classes

4.1 Background and methodology

Once data instances are identified as not belonging to the set of known classes, the next step is to categorize the instances into new classes. This is the most critical part of the analysis since incorrect categorization of instances can mislead the model while making predictions.

The prediction probabilities associated with the data instances are the key drivers for this analysis as they offer generalization to the methodology. When classifier $\mathcal{C}$ is applied to observations belonging classes in $\mathcal{U}$, the resulting k-dimensional space is generally inadequate. The distribution of prediction probabilities corresponding to observations that belong to class $j \in \mathcal{K}$ are ideally skewed towards class j in such a way to as exceed some confidence threshold. This is impossible for classes in $\mathcal{U}$. The error distribution for an observation belonging to class $i \in \mathcal{U}$ may either be non-informative or contain inherent patterns

since the open world classifier is incomplete with respect to classes in $\mathcal{U}$. We hypothesize that the error is informative and additionally there is a distinguishable *residual signature* for different unknown classes.

4.2 Algorithm for detecting unknown classes using association rule mining (ARM)

ARM is a rule-based machine learning method to discover interesting relations between variables in a large databases [9] and is considered great tool for decision making in various fields such as, market basket analysis [57,58], medical diagnosis [59, 60], Bioinformatics [61–63], Multimedia [64, 65], computer network security [66, 67], census data [68], remote sensing data [69, 70].

The algorithm for categorization of identified data instances based on ARM is provided in Algorithm 2. The required inputs for this algorithm include the sets $\mathcal{T}$, $\mathcal{F}$, $\mathcal{O}$, and the classifier $\mathcal{C}$. Furthermore, ω, η, β, and γ are user-defined parameters that are described below.

This is an iterative algorithm where the unknown classes are discovered based on the residual signatures that exist within the flagged instances. Furthermore, for every unknown class discovered, all the instances associated with that particular unknown class are identified and removed from the flagged set $\mathcal{F}$. This process is continued until the stopping criteria is satisfied. The stopping condition for this algorithm is when there are less than ω fraction of data points left in $\mathcal{F}$.

The first step in the methodology is to find the prediction probabilities of all the identified data instances. Specifically, $\mathbf{P}_{\text{identified}}$ is an $m \times k$ probability

matrix generated from the application of $\mathcal{C}$ on $\mathcal{F}$, where $|\mathcal{F}| = m$. An example is depicted in Table 4.1. In this table, index 1 is associated with the first flagged observation for which classes c_1 and c_2 have the H highest probabilities across the residual signature. Similarly, the highest H probabilities for every data point in $\mathbf{P}_{\text{identified}}$ are determined and the corresponding training class labels are stored as items in a set $\mathcal{I}$. Table 4.2 depicts an example set $\mathcal{I}$ for H=2 from the data shown in Table 4.1. Apply the Apriori algorithm [71] on $\mathcal{I}$ to generate frequent itemsets of length H based on a user-defined value of support (S) and confidence (C). Support and confidence are ARM-specific parameter values. Determine the itemset s from $\mathcal{I}$, with the highest support value and identify all the instances from $\mathcal{F}$ with s as their highest H probabilities and store them in $\mathcal{B}$. If the number of data points in $\mathcal{B}$ is at least η percentage of $\mathcal{F}$, proceed forward. Update $\mathcal{F}$ by removing $\mathcal{B}$ from $\mathcal{F}$. Next step is to label all the instances in $\mathcal{B}$ with a new class $k + i$. Initially i is to set to be equal to 0 and is incremented by 1 for every unknown class discovered. Update $\mathcal{T}$ by appending $\mathcal{B}$ to $\mathcal{T}$. Retrain the classifier $\mathcal{C}$ on $\mathcal{T}$. Make predictions on $\mathcal{F}$ using $\mathcal{C}$ and determine all the instances that are predicted as $k + i$ with a prediction confidence (prediction probability) value of at least γ and store them in $\mathcal{P}$. Update $\mathcal{F}$ and $\mathcal{T}$ by removing $\mathcal{P}$ from $\mathcal{F}$ and appending $\mathcal{P}$ to $\mathcal{T}$. This process is repeated until the number of data points in $\mathcal{P}$ is less than $\beta\ |\mathcal{F}|$. The output of this algorithm is an updated classifier ($\mathcal{C}$) that is capable of classifying both the known and the identified unknown classes.

Algorithm 2 Residual signature based categorization of identified instances

Input:

training data $\mathcal{T}$ with n observations
identified data $\mathcal{F}$ with m observations
open world data $\mathcal{O}$
probabilistic classifier $\mathcal{C}$
maximum probability parameter H
ω, η, and β: termination parameters
γ: prediction probability parameter

Output:

Updated probabilistic classifier $\mathcal{C}$

$L \leftarrow |\mathcal{F}|$
$i \leftarrow 0$
while $|\mathcal{F}| \geq \omega(L)$ **do**
    $\mathbf{P}_{\text{identified}} \leftarrow \mathcal{C}(\mathcal{F})$
    create $\mathcal{I}$ from $\mathbf{P}_{\text{identified}}$ based on H
    $\mathcal{B} \leftarrow$ instances in $\mathcal{F}$ associated with the itemset with the highest support value.
    if $|\mathcal{B}| < \eta|\mathcal{F}|$ **then**
     return $\mathcal{C}$
    end if
    $i \leftarrow i + 1$
    $\mathcal{F} \leftarrow \mathcal{F} \setminus \mathcal{B}$
    create new label $k + i$ for all observations in $\mathcal{B}$
    $\mathcal{T} \leftarrow \mathcal{T} \cup \mathcal{B}$
    retrain classifier $\mathcal{C}$ on $\mathcal{T}$
    repeat
        $\mathcal{P} \leftarrow$ instances in $\mathcal{F}$ that are predicted as $k + i$ with a prediction probability of at least γ
        $\mathcal{T} \leftarrow \mathcal{T} \cup \mathcal{P}$
        retrain classifier $\mathcal{C}$ on $\mathcal{T}$
        $\mathcal{F} \leftarrow \mathcal{F} \setminus \mathcal{P}$
    until $|\mathcal{P}| < \beta|\mathcal{F}|$
end while
return $\mathcal{C}$

Table 4.1: $\mathbf{P}_{\text{identified}}$

index	c_1	c_2	c_3	$\ldots$	c_k
1	**0.40**	**0.50**	0.08		
2	**0.70**	0.07	**0.20**		
3	**0.60**	0.09	**0.30**		
$\vdots$				$\ddots$	
m					

Table 4.2: $\mathcal{I}$ for H=2

index	itemset
1	$\{c_1, c_2\}$
2	$\{c_1, c_3\}$
$\vdots$	
m	$\{c_1, c_3\}$

Chapter 5

Experiments and results

5.1 Introduction

To analyze the performance of the proposed methodology, four different experiments with the number of unknown classes ranging from 0 to 3 are conducted. A wide variety of data types from different domains are considered. Table 5.1 lists all the experiments conducted and the associated data sets.

Table 5.1: Data description

Experiment	Data	$\lvert\mathcal{K}\rvert$	$\lvert\mathcal{U}\rvert$	n_{T}	n_{O}
1	Code commit messages	5	0	1418	1351
2	Traditional numeric data	6	1	5416	2690
3	Human activity recognition	6	2	2898	2839
4	Hand written digits	10	3	22504	6000

Certain factors are same across all experiments. Random Forest is used as the classifier and Table 5.2 lists the accuracy, precision, recall and f1-score for the performance of the classifier. on the training data, prior to implementing the proposed framework. The level of confidence for the Chebyshev's inequality (α)

is considered to be 0.1 and the distance metric used is Jensen-Shannon Distance.

Table 5.2: Performance of $\mathcal{C}$ in every experiment

Performance metric	1	2	3	4
Accuracy	0.98	0.95	0.94	0.97
Precision	0.98	0.95	0.94	0.97
Recall	0.98	0.95	0.94	0.97
F_1-score	0.98	0.95	0.94	0.97

5.2 Data

5.2.1 Code commit messages

CCM data for experiment 1 is derived from different open source projects available on GitHub such as messages that belong to different categories such as bug fixing, design improvement, adding new features, improving non functional requirements and no category. The dataset contains 5 different classes with a total of 3377 instances.

5.2.2 Traditional numeric data

TND [72] used for experiment 2 is derived from Geo-spatial data having six different classes corresponding to the land cover such as impervious, farm, forest, grass, orchard and water. The features present in the dataset are the maximum NDVI (normalized difference vegetation index) values derived from the time-series of satellite images. Each observation in the dataset is represented using 29 attributes and there are a total of 10546 observations.

5.2.3 Human activity recognition data

HAR is a sensor data [73] corresponding to different human activities such as walking, walking upstairs, walking downstairs, sitting, standing and laying. This dataset contains 10299 observations and 561 features.

5.2.4 Hand written digits data

This dataset [74] contains images corresponding to the hand written digits 0 to 9. There are a total of 60000 observations each represented by a 784 dimensional feature vector.

5.3 Analysis on text data

As the first step in the process, Algorithm 1 is applied on the open world data containing 1351 instances, out of which 52 have been identified, which is less than 4% of the total number of observations in $\mathcal{O}$. Based on the Chebyshev's inequality, there is a chance for 10% of the known data to be identified as belonging to unknown classes. Furthermore, presence of an unknown class would result in higher percentage of identified instances. However, in this scenario, we do not see a strong evidence to suspect the presence of an unknown class. Thus, we can terminate the algorithm here and not proceed forward to the second stage of the methodology.

5.4 Analysis on Crowd sourcing data

The training data considered for this experiment contains classes *1, 2, 3, 4* and *5* and the open world data contains a new class *0* along with the known classes.

Table 5.3: Classwise data distribution of $\mathcal{O}$ and $\mathcal{F}$ for Text data

Class	class information	$\lvert\mathcal{O}\rvert$	$\lvert\mathcal{F}\rvert$
1	bug fixing	417	19
2	no category	117	1
3	design improvement	181	8
4	adding new features	229	5
5	non functional requirements	407	19

From Table 5.4, it is evident that 1150 out of 2690 instances have been identified by Algorithm 1. Furthermore, about 88% of the identified data instances belong to the unknown class. As we have a large number of instances identified in the first step, we proceed forward to the categorization stage of the framework. Algorithm 2 is applied on $\mathcal{F}$ using the parameter setting provided in Table 5.5 and it successfully discovered the unknown class. This evident from Table 5.9, as we see that about 88% of the data instances belonging to class *0* are classified correctly and the majority of the remaining instances are misclassified as class *1*, as class *0* (farm) is more similar to class *1* (forest). The overall accuracy of the classifier increases from 50% to 90% and the information regarding other performance metrics is provided in Table 5.10. This experiment demonstrates the successful working of the proposed methodology for the case when a single unknown class is present in the open world data.

Table 5.4: Classwise data distribution of $\mathcal{O}$ and $\mathcal{F}$ for TND

Class	class information	$\lvert\mathcal{O}\rvert$	$\lvert\mathcal{F}\rvert$	Percentage (%)
0	Farm	1224	1014	88.17
1	Forest	1230	91	7.90
2	Grass	126	15	1.73
3	Impervious	70	20	1.30
4	Orchard	8	4	0.52
5	Water	32	6	0.34

Table 5.5: Parameter settings for experiments 2,3 and 4

Parameter	Experiments			
	1	2	3	4
Chebyshev's parameter (τ)	3.16	3.16	3.16	3.16
Maximum probability limit (H)	2	3	3	4
Support (S)	0.1	0.1	0.1	0.1
Confidence (C)	0.1	0.1	0.1	0.1
Termination parameter (ω)	0.25	0.25	0.25	0.25
Termination parameter (β)	0.1	0.1	0.1	0.1
Termination parameter (η)	0.1	0.1	0.1	0.1
Prediction confidence (γ)	0.6	0.6	0.6	0.4

Table 5.6: Confusion matrix for performance of $\mathcal{C}$ on $\mathcal{O}$ for crowd sourcing data

True class	Predicted class					
	0	1	2	3	4	5
0	0	1087	11	123	0	3
1	0	1126	0	4	0	0
2	0	6	57	7	0	0
3	0	13	1	112	0	0
4	0	7	0	0	1	0
5	0	4	0	3	0	25

5.5 Analysis on HAR data

The open world data contains data from all the classes out of which classes *2* and *6* were randomly selected to be considered as the unknown classes and the remaining as the known classes. The distribution of the data before and after the identification stage is displayed in Table 5.11. We see that 1990 out of 2839 instances present in $\mathcal{O}$ are flagged, out of which about 96% of the them belong to the unknown classes. We proceed forward and apply Algorithm 2 based on the parameters specified in Table 5.5. This algorithm successfully discovered the two unknown classes and assign the identified instances into two new classes.

Table 5.7: Confusion matrix for the final performance of $\mathcal{C}$ on $\mathcal{O}$ for TND

	Predicted class					
True class	0	1	2	3	4	5
0	1082	109	1	29	0	3
1	72	1054	0	4	0	0
2	13	1	53	3	0	0
3	19	1	0	106	0	0
4	3	4	0	0	1	0
5	2	2	0	3	0	25

Table 5.8: Initial and final performance of $\mathcal{C}$ on $\mathcal{O}$ for TND

Performance metric	Initial	Final
Accuracy	0.51	0.90
Precision	0.27	0.90
Recall	0.51	0.90
F_1-score	0.35	0.90

We further evaluate the quality of the categorization by providing a confusion matrix which is displayed in Table 5.13. Based on the results from this table it is evident that nearly 90% of observations belonging to class *2* are classified correctly and the majority of the remaining instances are misclassified as class *3* as class *2* (walking upstairs) exhibit more similarity to class *3* (walking downstairs) compared to other classes. Similarly, 97.5% of observations belonging to class *6* are classified correctly and the major portion of the remaining instances are misclassified as class *4* as class *6* (laying) is more similar to class *4* (sitting). The overall accuracy of the classifier increases from 24% to 93% and the information regarding other performance metrics is provided in Table 5.10. This experiment demonstrates the successful working of the proposed on data containing two unknown classes.

Table 5.9: Confusion matrix for performance of $\mathcal{C}_{new}$ on $\mathcal{O}$ for crowd sourcing data

		Predicted					
		0	1	2	3	4	5
True	0	1082	109	1	29	0	3
	1	72	1054	0	4	0	0
	2	13	1	53	3	0	0
	3	19	1	0	106	0	0
	4	3	4	0	0	1	0
	5	2	2	0	3	0	25

Table 5.10: Initial and final performance of $\mathcal{C}$ on $\mathcal{O}$ for TND

Performance metric	Initial	Final
Accuracy	0.51	0.90
Precision	0.27	0.90
Recall	0.51	0.90
F_1-score	0.35	0.90

5.6 Analysis on MNIST data

The open world data contains observations belonging to all the digits *0* to *9*, out which digits *1,5* and *8* are considered as the unknown classes and the remaining digits as known classes. The distribution of the open world data and the flagged data is displayed in Table 5.15. From the table, it is evident that about 86% of the instances in $\mathcal{O}$ are flagged as belonging unknown classes. Furthermore, more than 80% of the identified instances in $\mathcal{F}$ belong to the unknown classes. Since the criteria to proceed forward is met, we apply Algorithm 2 on $\mathcal{F}$ using the parameters specified in Table 5.5. From the results, all the three unknown classes are discovered and the quality of the categorization is displayed in Table 5.17. About 79% of observations belonging to class *1*, 8% of the observations belonging to class *5* and 73% of the observations belonging to class *8* are classified

Table 5.11: Classwise data distribution of $\mathcal{F}$ for HAR data

Class	Class information	$\lvert\mathcal{O}\rvert$	$\lvert\mathcal{F}\rvert$	Percentage (%)
1	Walking	196	27	1.35
2	Walking upstairs	917	748	37.58
3	Walking downstairs	149	10	0.50
4	Sitting	175	13	0.65
5	Standing	211	30	1.50
6	Laying	1191	1162	58.39

Table 5.12: Confusion matrix for performance of $\mathcal{C}$ on O for HAR data

	Predicted class					
True class	1	2	3	4	5	6
1	194	0	1	1	0	0
2	209	0	605	0	103	0
3	8	0	140	0	1	0
4	0	0	0	163	12	0
5	0	0	7	9	195	0
6	1049	0	3	99	40	0

correctly. The performance of the latest classifier on class *5* is poor compared to other unknown classes due to the lack of a common pattern across majority of the observations belonging to class *5*. Furthermore, only few instances possess a similarity pattern and are identified by the Algorithm 2. Hence, the classifier has insufficient observations to learn from and resulted in performing poorly on that particular class. However, it is interesting to note that, for class *1*, the majority of the misclassified instances are predicted as class *7*, and the reason is that digit *1* possess similar pixel arrangement with digit *7* and this similarity resulted in the misclassification. Similarly, digits *5* and *8* are more similar to digit *3* and the results explain the same as the majority of the instances that belong to digits *5* and *8* are misclassified as digit *3*. Furthermore, the results from this experiment support our hypothesize that the error is informative and

Table 5.13: Confusion matrix for the final performance of $\mathcal{C}$ on $\mathcal{O}$ for HAR data

	Predicted class					
True class	1	2	3	4	5	6
1	167	21	0	0	0	8
2	2	822	92	0	1	0
3	7	9	133	0	0	0
4	0	0	0	162	13	0
5	0	7	0	10	194	0
6	0	1	0	34	2	1154

Table 5.14: Initial and final performance of $\mathcal{C}$ on $\mathcal{O}$ for HAR data

Performance metric	Initial	Final
Accuracy	0.24	0.93
Precision	0.09	0.94
Recall	0.24	0.93
F_1-score	0.12	0.93

Table 5.15: Classwise data distribution of $\mathcal{O}$ for MNIST data

Class	$\lvert\mathcal{O}\rvert$	$\lvert\mathcal{F}\rvert$	Percentage (%)
0	435	59	1.71
1	1109	1104	32.09
2	396	78	2.26
3	433	81	2.35
4	425	98	2.84
5	892	790	22.96
6	422	76	2.20
7	454	79	2.29
8	999	988	28.72
9	435	87	2.52

that the *residual signature* helps in distinguishing the unknown classes. The final performance of the classifier increases from 48% to 75% and other performance metrics are displayed in Table 5.16.

Table 5.16: Initial and final performance of $\mathcal{C}$ on $\mathcal{O}$ for MNIST data

Performance metric	Initial	Final
Accuracy	0.48	0.75
Precision	0.28	0.73
Recall	0.48	0.75
F_1-score	0.34	0.72

Table 5.17: Confusion matrix for the final performance of $\mathcal{C}$ on $\mathcal{O}$ for MNIST data

	Predicted class									
True class	0	1	2	3	4	5	6	7	8	9
0	410	0	1	0	1	0	0	0	23	0
1	0	873	10	2	4	6	4	1	207	2
2	0	6	363	2	3	0	0	2	20	0
3	0	2	5	400	1	6	0	5	12	2
4	3	2	1	0	394	14	2	2	0	7
5	13	6	9	387	34	70	28	7	284	54
6	1	0	0	0	3	0	403	0	15	0
7	0	9	6	1	2	22	0	410	0	4
8	5	24	39	67	17	31	10	6	735	65
9	0	1	1	5	7	28	0	3	3	387

Chapter 6

Sensitivity analysis

6.1 Introduction

Given the presence of different parameters that govern the functioning of the methodology, it is imperative to analyze the impact of change in these parameter values on the performance of the methodology. The following are the parameters that we considered for the sensitivity analysis.

- Number of known classes ($|\mathcal{K}|$)
- Chebyshev's parameter (τ)
- Maximum probability parameter (H)
- Quality of the classifier

To maintain consistency, we conduct all the analyses on the MNIST data as it is one of the standard data sets available.

6.2 Sensitivity analysis: Number of known classes

Purpose The goal of this analysis is to analyze the impact of number of known classes on the overall performance of the methodology.

Hypothesis As the number of known classes increases, the dimensionality of the probabilistic space increases and provides more detail for categorization.

Experimental setup Three experiments, by varying the number of known classes at 2,5 and 8 are conducted. Furthermore, data corresponding to digits *8* and *9* are considered as the unknown classes for all the experiments. Figures 1 and 2 are the images corresponding to digits 8 and 9 respectively. Also, random forest (RF) is the classifier used for all the experiments conducted. The parameter settings associated with Algorithm 1 and Algorithm 2 and the details about each experiment is provided in Table 6.1.

Table 6.1: Sensitivity analysis 1: Experimental setup

	Experiments		
Parameter	1	2	3
Number of unknown classes	2	2	2
Number of known classes	2	5	8
Classifier	RF	RF	RF
Chebyshev's parameter (τ)	3.16	3.16	3.16
Maximum probability limit (H)	1	3	4
Termination parameter (ω)	0.25	0.25	0.25
Termination parameter (β)	0.1	0.1	0.1
Termination parameter (η)	0.1	0.1	0.1
Prediction confidence (γ)	0.6	0.6	0.4

Table 6.2: Sensitivity analysis-1 (Experiment-1): Classwise data distribution of $\mathcal{O}$ for MNIST data

Class	$\lvert\mathcal{O}\rvert$	$\lvert\mathcal{F}\rvert$
0	937	49
1	1063	38
8	1005	982
9	995	972

6.2.1 Experiment-1

The training data considered for this experiment contains classes *0* and *1* and the open world data contains two new classes *8* and *9* along with the known classes.

The first step is the identification stage where we apply Algorithm 1 on $\mathcal{O}$ to identify the data instances belonging to the unknown classes. The results of the identification stage indicate that about 50% of the instances in $\mathcal{O}$ are identified (flagged) as potentially not belonging to the known classes. From Table 6.2, we further evaluate the quality of the identified instances by analyzing the distribution of the flagged instances in $\mathcal{F}$, and we see that about 96% of the flagged instances belong to the unknown classes. Since we have more than 10% of the instances as flagged, we move forward to the next step which is the categorization of flagged instances. Algorithm 2 is applied on $\mathcal{F}$ and the results from Table 6.4 indicate that Algorithm 2 discovered a single unknown class as opposed to two unknown classes. Furthermore, we also see that about 98% of the instances belonging to class *8* are correctly classified whereas instances that belong to *9* are completely misclassified. Table 6.3 provides a comparison of performance between the old and new classifier and we see improvement in the overall performance of the classifier.

Table 6.3: Sensitivity analysis-1 (Experiment-1):Initial and final performance of $\mathcal{C}$ on $\mathcal{O}$ for MNIST data

Performance metric	Initial	Final
Accuracy	0.49	0.73
Precision	0.25	0.61
Recall	0.49	0.73
F_1-score	0.33	0.65

Table 6.4: Sensitivity analysis-1 (Experiment-1):Confusion matrix for the final performance of $\mathcal{C}$ on $\mathcal{O}$ for MNIST data

	Predicted class			
True class	0	1	8	9
0	928	0	9	0
1	0	1024	39	0
8	7	9	989	0
9	14	4	977	0

6.2.2 Experiment -2

In the second experiment, we increase the number of training classes to 5, but the number of unknown classes remain the same. The training data $\mathcal{T}$ includes images corresponding to digits *0,1,2,3* and *4* and $\mathcal{O}$ contains data that belong to digits *0,1,2,3,4,8* and *9*. The same procedure similar to the first experiment is followed and the results corresponding to the identification phase are displayed in Table 6.5. From the table, we see that about 46% of instances that are identified belong to the unknown classes. From Table 6.2, we further evaluate the quality of the identified instances by analyzing the distribution of the flagged instances in $\mathcal{F}$, and we see that about 88% of the flagged instances belong to the unknown classes. Since we have more than 10% of the instances as flagged, we move forward to the next step which is the categorization of flagged instances. Algorithm 2 is applied on $\mathcal{F}$ and the results from Table 6.4 indicate that Algorithm 2 discovered

Table 6.5: Sensitivity analysis-1 (Experiment-2): Classwise data distribution of $\mathcal{O}$ for MNIST data

Class	$\lvert\mathcal{O}\rvert$	$\lvert\mathcal{F}\rvert$
0	403	51
1	430	40
2	387	55
3	399	55
4	381	62
8	1005	989
9	995	595

both the unknown classes successfully. However, we see that 64% of class *8* are correctly classified, while the remaining 36% of them are spread across the other classes, of which majority of them are wrongly classified as class *9*. Similarly, for class *9*, 73% of them are correctly classified and the majority of them misclassified as class *4*. The overall performance of the final classifier is provided in Table 6.3 and we see a considerable improvement in the performance.

Table 6.6: Sensitivity analysis-1 (Experiment-2): Initial and final performance of $\mathcal{C}$ on $\mathcal{O}$ for MNIST data

Performance metric	Initial	Final
Accuracy	0.49	0.81
Precision	0.30	0.83
Recall	0.0.49	0.81
F_1-score	0.0.36	0.81

6.2.3 Experiment-3

In the final experiment, we increase the number of training classes to 8 and the number of unknown classes remain the same. The training data $\mathcal{T}$ includes images corresponding to digits *0,1,2,3,4,5,6,7* and $\mathcal{O}$ contains data that belong to

Table 6.7: Sensitivity analysis-1 (Experiment-2): Confusion matrix for the final performance of $\mathcal{C}$ on $\mathcal{O}$ for MNIST data

	Predicted class						
True class	0	1	2	3	4	8	9
1	0	421	2	2	2	3	0
2	3	0	346	0	5	33	0
3	0	0	2	351	0	36	10
4	1	1	1	0	360	0	18
8	6	36	22	27	21	641	252
9	9	7	10	17	218	3	731

all the digits *0* to *9*. The result of the identification phase is displayed in Table 6.8 and we can see that more than 50% of instances that are identified belong to the unknown classes. From Table 6.2, we further evaluate the quality of the identified instances by analyzing the distribution of the flagged instances in $\mathcal{F}$, and we see that about 84% of the flagged instances belong to the unknown classes. Since we have more than 10% of the instances as flagged, we move forward to the next step which is the categorization of flagged instances. Algorithm 2 is applied on $\mathcal{F}$ and the results from Table 6.4 indicate that Algorithm 2 discovered both the unknown classes successfully. However, we see that 74% of class *8* are correctly classified, while the remaining 26% of them are spread across the other classes, of which majority of them are wrongly classified as class *3*. Similarly, for class *9*, 79% of them are correctly classified and the majority of the remaining instances are misclassified as class *4*. The overall performance of the final classifier is provided in Table 6.3 and we see a considerable improvement in the performance.

Conclusion The main motive of analysis is to determine the effect of the number of training classes on the proposed framework. In this regard, we carried out different experiments with varying the number of training classes ranging from 2

Table 6.8: Sensitivity analysis-1 (Experiment-3): Classwise data distribution of $\mathcal{O}$ and $\mathcal{F}$ for MNIST data

Class	$\lvert\mathcal{O}\rvert$	$\lvert\mathcal{F}\rvert$
0	247	41
1	283	20
2	236	65
3	268	63
4	239	54
5	230	70
6	231	45
7	266	45
8	1005	1001
9	995	963

Table 6.9: Sensitivity analysis-1 (Experiment-3): Initial and final performance of $\mathcal{C}$ on $\mathcal{O}$ for MNIST data

Performance metric	Initial	Final
Accuracy	0.48	0.86
Precision	0.30	0.88
Recall	0.48	0.86
F_1-score	0. 36	0.87

to 8. Results strongly suggest that as the number of known classes increases, the ability to identify and discriminate the unknown classes also improves. However, it is interesting to note from Table 6.11 that, in all the experiments conducted, the performance of the methodology corresponding to the identification task remains consistent. The only difference in the performance is noticed with the categorization task. In the first experiment, there are insufficient number of known classes to capture the similarity patterns existing in the data and both the unknown classes exhibited similarity to the same known class. Increase in the number of training classes, resulted in a unique similarity pattern to each unknown class and resulted in better categorization in the second and third experiments. However,

Table 6.10: Sensitivity analysis-1 (Experiment-3): Confusion matrix for the final performance of $\mathcal{C}$ on $\mathcal{O}$ for MNIST data

	Predicted class									
True class	0	1	2	3	4	5	6	7	8	9
1	0	280	1	1	1	0	0	0	0	0
2	3	2	221	1	3	0	2	2	2	0
3	0	0	4	243	0	0	1	6	8	6
4	0	1	0	0	227	0	0	1	0	10
5	2	1	0	2	0	212	3	0	2	8
6	3	0	0	0	0	1	227	0	0	0
7	1	1	7	0	2	1	0	251	0	3
8	3	21	37	89	14	29	7	5	743	57
9	9	3	13	12	121	5	0	27	18	787

we see some discrepancy in the prediction distribution for the unknown classes and the overall misclassification rate associated with the categorization of the unknown classes decreases as seen in Table 6.11. Furthermore, from the same table, we see that the overall performance of the final classifier also increases as the number of known classes increases.

Table 6.11: Sensitivity analysis 1: Combined results of all experiments

	Experiments		
Parameter	1	2	3
Percentage of data identified	50	46	50
Number of unknown classes discovered	1	2	2
Percentage misclassified for class *8*	2	36	26
Percentage misclassified for class *9*	100	27	21
Final F_1-score	0.65	0.81	0.87

6.3 Sensitivity analysis: Chebyshev's parameter

Purpose One of the critical parameters associated with the framework is the Chebyshev's parameter (τ) and it governs the way the methodology works and plays a major role in the identification phase. It is imperative to choose an appropriate value as any fallacious results that occur in this stage will subsequently lead to undesirable results in the second stage and ultimately lead to overall failure of the framework. Hence, it is imperative to assign a proper value to τ to achieve desired results.

Hypothesis The higher the value of τ, the fewer will be the number of instances flagged. On the other hand, a very low value of τ might cause more instances from the known classes be flagged which is again not desirable. The methodology fails for extreme values of the Chebyshev's parameter τ.

Experimental setup To maintain consistency, all the experiments are conducted on the MNIST data, where the training data $\mathcal{O}$ includes data corresponding to digits *0* to *7* and the open world data $\mathcal{O}$ contains data that belongs to all the digits *0* to *9*. The only varying parameter for this analysis is τ and is set to 1.4, 2.0 and 4.47 for the first, second and third experiment respectively. The details about the other parameters associated with the methodology is provided in Table 6.12.

Table 6.12: Sensitivity analysis 2: Experimental setup

Parameter	Experiments		
	1	2	3
Classifier	RF	RF	RF
Chebyshev's parameter (τ)	1.414	2	4.47
Maximum probability limit (H)	4	4	4
Termination parameter (ω)	0.25	0.25	0.25
Termination parameter (β)	0.1	0.1	0.1
Termination parameter (η)	0.1	0.1	0.1
Prediction confidence (γ)	0.4	0.4	0.4

6.3.1 Experiment-1

The first step is the identification stage where we apply Algorithm 1 on $\mathcal{O}$, using the Chebyshev's parameter (τ) that is set to a value of 1.414. The results of the identification stage indicate that about 65% of the instances in $\mathcal{O}$ are identified (flagged) as belonging to the unknown classes. From Table 6.13, we analyze the distribution of the flagged instances in $\mathcal{F}$, and we see that about 76% of them belong to the unknown classes. Since we have more than 10% of the instances as flagged, we move forward to the next step which is the categorization of flagged instances. Algorithm 2 is applied on $\mathcal{F}$ and the results from Table 6.14 indicate that Algorithm 2 successfully discovered both the unknown classes. Furthermore, we also see that about 53% of the instances belonging to class *8* are correctly classified while the remaining are spread across the other classes, of which majority of them are wrongly classified as class *3*. Similarly, for class *9*, 84% of them are correctly classified and the majority of the remaining instances are misclassified as class *4*. The overall accuracy of the classifier increases from 48% to 82% and the information regarding other performance metrics is provided in Table 6.15.

Table 6.13: Sensitivity analysis-2 (Experiment-1): Classwise data distribution of $\mathcal{O}$ and $\mathcal{F}$ for MNIST data

Class	$\lvert\mathcal{O}\rvert$	$\lvert\mathcal{F}\rvert$
0	247	68
1	283	34
2	236	93
3	268	99
4	239	82
5	230	108
6	231	69
7	266	66
8	1005	1002
9	995	994

6.3.2 Experiment -2

The first step is the identification stage where we apply Algorithm 1 on $\mathcal{O}$, using the Chebyshev's parameter (τ) that is set to a value of 2. The results of the identification stage indicate that about 62% of the instances in $\mathcal{O}$ are identified (flagged) as belonging to the unknown classes. From Table 6.16, we analyze the distribution of the flagged instances in $\mathcal{F}$, and we see that about 80% of them belong to the unknown classes. Since we have more than 10% of the instances as flagged, we move forward to the next step which is the categorization of flagged instances. Algorithm 2 is applied on $\mathcal{F}$ and the results from Table 6.17 indicate that Algorithm 2 successfully discovered both the unknown classes. Furthermore, we also see that about 53% of the instances belonging to class *8* are correctly classified while the remaining are spread across the other classes, of which majority of them are wrongly classified as class *3*. Similarly, for class *9*, 84% of them are correctly classified and the majority of the remaining instances are misclassified as class *4*. The overall accuracy of the classifier increases from 48% to 82% and the information regarding other performance metrics is provided in Table 6.18.

Table 6.14: Sensitivity analysis-2 (Experiment-1):Confusion matrix for the final performance of $\mathcal{C}$ on $\mathcal{O}$ for MNIST data

	Predicted class									
True class	0	1	2	3	4	5	6	7	8	9
0	246	0	0	0	0	0	1	0	0	0
1	0	280	1	1	1	0	0	0	0	0
2	1	2	223	1	5	0	1	1	2	0
3	0	0	5	243	0	1	1	6	10	2
4	0	1	0	0	224	0	0	1	1	12
5	2	1	0	2	0	211	3	0	4	7
6	3	0	0	0	0	0	228	0	0	0
7	1	1	7	0	1	1	0	249	0	6
8	6	54	67	138	16	79	14	5	530	96
9	7	4	15	12	74	5	0	24	16	838

Table 6.15: Sensitivity analysis-2 (Experiment-1): Initial and final performance of $\mathcal{C}$ on $\mathcal{O}$ for MNIST data

Performance metric	Initial	Final
Accuracy	0.48	0.82
Precision	0.30	0.84
Recall	0.48	0.82
F_1-score	0.36	0.81

Table 6.16: Sensitivity analysis-2 (Experiment-2): Classwise data distribution of $\mathcal{O}$ and $\mathcal{F}$ for MNIST data

Class	$\lvert\mathcal{O}\rvert$	$\lvert\mathcal{F}\rvert$
0	247	51
1	283	29
2	236	76
3	268	87
4	239	66
5	230	90
6	231	55
7	266	55
8	1005	1002
9	995	991

Table 6.17: Sensitivity analysis-2 (Experiment-2): Confusion matrix for the final performance of $\mathcal{C}$ on $\mathcal{O}$ for MNIST data

	Predicted class									
True class	0	1	2	3	4	5	6	7	8	9
0	246	0	0	0	0	0	1	0	0	0
1	0	280	2	1	0	0	0	0	0	0
2	1	2	223	1	5	0	1	1	2	0
3	0	0	5	243	0	1	1	6	10	2
4	0	1	0	0	224	0	0	1	1	12
5	2	1	0	2	0	211	3	0	4	7
6	3	0	0	0	0	0	228	0	0	0
7	1	1	7	0	1	1	0	249	0	6
8	6	54	67	138	16	79	14	5	530	96
9	7	4	15	12	74	5	0	24	16	838

Table 6.18: Sensitivity analysis-2 (Experiment-2): Initial and final performance of $\mathcal{C}$ on $\mathcal{O}$ for MNIST data

Performance metric	Initial	Final
Accuracy	0.48	0.82
Precision	0.30	0.84
Recall	0.48	0.82
F_1-score	0.36	0.81

6.3.3 Experiment -3

Algorithm 1 is applied on $\mathcal{O}$, using the Chebyshev's parameter (τ) that is set to a value of 4.47. The results of the identification stage indicate that about 59% of the instances in $\mathcal{O}$ are identified (flagged) as belonging to the unknown classes. From Table 6.19, we analyze the distribution of the flagged instances in $\mathcal{F}$, and we see that about 84% of them belong to the unknown classes. Since we have more than 10% of the instances as flagged, we move forward to the next step which is the categorization of flagged instances. Algorithm 2 is applied on $\mathcal{F}$ and the results from Table 6.17 indicate that Algorithm 2 successfully discovered both

Table 6.19: Sensitivity analysis-2 (Experiment-3): Classwise data distribution of $\mathcal{O}$ and $\mathcal{F}$ for MNIST data

Class	$\lvert\mathcal{O}\rvert$	$\lvert\mathcal{F}\rvert$
0	247	42
1	283	20
2	236	57
3	268	60
4	239	54
5	230	66
6	231	42
7	266	42
8	1005	1002
9	995	967

the unknown classes. Furthermore, we also see that about 53% of the instances belonging to class *8* are correctly classified while the remaining are spread across the other classes, of which majority of them are wrongly classified as class *3*. Similarly, for class *9*, 85% of them are correctly classified and the majority of the remaining instances are misclassified as class *4*. The overall accuracy of the classifier increases from 48% to 82% and the information regarding other performance metrics is provided in Table 6.21.

Conclusion The main motive of this analysis to determine the effect of Chebyshev's parameter (τ) on the performance of the methodology. We hypothesized that the number of instances flagged decreases as the value of τ increases and the results from Table 6.22 support our hypothesis. Furthermore, it is interesting to note that decrease in the number of flagged instances can be mainly attributed to the instances from the known classes while there is a negligible decrease for the unknown classes. However, from Table 6.22, we can see that the overall performance of the final classifier in all the three experiments remains consistent. This

Table 6.20: Sensitivity analysis-2 (Experiment-3): Confusion matrix for the final performance of $\mathcal{C}$ on $\mathcal{O}$ for MNIST data

	Predicted class									
True class	0	1	2	3	4	5	6	7	8	9
0	246	0	0	0	0	0	1	0	0	0
1	0	281	1	1	0	0	0	0	0	0
2	2	2	224	0	3	0	0	2	3	0
3	0	0	6	244	1	2	0	5	9	1
4	0	1	0	0	225	0	0	1	1	11
5	2	1	0	2	0	215	3	0	1	6
6	3	0	0	0	0	0	228	0	0	0
7	1	1	7	1	1	0	0	250	0	5
8	5	66	67	128	22	73	11	6	530	97
9	8	5	14	14	90	6	0	21	16	821

Table 6.21: Sensitivity analysis-2 (Experiment-3): Initial and final performance of $\mathcal{C}$ on $\mathcal{O}$ for MNIST data

Performance metric	Initial	Final
Accuracy	0.48	0.82
Precision	0.30	0.84
Recall	0.48	0.82
F_1-score	0.36	0.81

implies that even when there is difference in the percentage of flagged instances, the overall performance of the framework across all experiments remains consistent. This characteristic of the methodology is desirable as the results show that the framework is robust to the false positives, i.e., the instances that actually belong to the known classes but are identified by the methodology as potentially belonging to the unknown classes.

Table 6.22: Sensitivity analysis 2: Combined results of all experiments

Parameter	Experiments		
	1	2	3
Chebyshev's parameter (τ)	1.414	2	4.47
Percentage of data identified	65	62	59
Percentage of instances in $\mathcal{F}$ that belong to unknown classes	50	50	49
Number of unknown classes discovered	2	2	2
Percentage correctly classified for class *8*	53	53	53
Percentage correctly classified for class *9*	84	84	85
Final F_1-score	0.81	0.81	0.81

6.4 Sensitivity analysis: Maximum probability parameter (H)

Purpose Maximum probability parameter (H) is one of the parameters that can have considerable effect on the performance of the framework. This parameter governs the categorization step of the methodology and decides the size of the itemsets that are generated by Algorithm 2. In this analysis, we determined the performance of the framework by conducting experiments for different values of H.

Hypothesis A very low value of H does not provide enough information to capture the residual patterns existing within the data. On the other hand, a very high value of H will result in a common residual signature across all the unknown classes and makes it difficult to distinguish one another.

Experimental setup To maintain consistency, in all the experiments the training data $\mathcal{T}$ includes the classes *0* to *7* and the open world data $\mathcal{O}$ contains data that belongs to all the digits *0* to *9*. Three experiments with different values of

H, 1, 4 and 7 are conducted. The details about the parameter settings associated with other parameters are provided in Table 6.23. Since the parameter H is a part of Algorithm 2, we apply Algorithm 1 on $\mathcal{O}$ to determine the set of flagged instances $\mathcal{F}$ and use the same flagged set for all the experiments conducted in this analysis. Th results of identification stage is provided in Table 6.24. From the results, we see that about 59% of the instances in $\mathcal{O}$ are identified (flagged) as belonging to the unknown classes. Furthermore, 83% of the flagged instances belong to the unknown classes.

Table 6.23: Sensitivity analysis 3: Experimental setup

Parameter	Experiments		
	1	2	3
Classifier	RF	RF	RF
Chebyshev's parameter (τ)	3.16	3.16	3.16
Maximum probability limit (H)	1	4	7
Termination parameter (ω)	0.25	0.25	0.25
Termination parameter (β)	0.1	0.1	0.1
Termination parameter (η)	0.1	0.1	0.1
Prediction confidence (γ)	0.4	0.4	0.4

6.4.1 Experiment-1

For this experiment, the value of H is set equal to 1. Algorithm 2 is applied on the data and the results for the categorization phase are provided in Table 6.25. From this table, we see that 3 unknown classes are identified, while the actual number of unknown classes present is 2. Furthermore, 37% of the instances belonging to class *8* are classified correctly, 39% of them are misclassified as class *9* and 13% of them are misclassified as belonging to the third unknown class. On the other hand, 93% of the instances belonging to class *9* are correctly classified and

Table 6.24: Sensitivity analysis 3: Classwise data distribution of $\mathcal{O}$ and $\mathcal{F}$ for MNIST data

Class	$\lvert\mathcal{O}\rvert$	$\lvert\mathcal{F}\rvert$
0	368	63
1	415	33
2	360	96
3	409	100
4	361	73
5	335	105
6	362	69
7	390	75
8	1490	1486
9	1510	1443

the remaining are misclassified as other classes, of which only 10 instances are classified as belonging to the *new* class. However, from Table 6.26, we see an improvement in the overall performance of the final classifier compared to the initial classifier.

Table 6.25: Sensitivity analysis-3 (Experiment-1): Confusion matrix for performance of $\mathcal{C}_{new}$ on $\mathcal{O}$ for MNIST data

	Predicted class										
True class	0	1	2	3	4	5	6	7	8	9	new
0	365	1	0	0	0	0	2	0	0	0	0
1	0	441	1	1	0	0	0	0	1	1	0
2	3	2	339	0	0	0	1	5	2	7	1
3	0	0	3	310	0	0	1	5	85	2	3
4	0	2	0	0	289	0	0	1	1	68	0
5	2	2	0	1	0	230	5	0	2	4	89
6	3	1	0	0	0	1	355	0	0	0	2
7	1	1	6	0	0	0	0	375	1	6	0
8	6	58	65	4	1	5	13	4	552	581	201
9	10	5	7	4	6	0	0	39	29	1400	10

Since the value of H was considered to be 1, only itemsets with size 1 are returned and the categorization of the instances happen based on these itemsets.

Table 6.26: Sensitivity analysis-3 (Experiment-1): Initial and final performance of $\mathcal{C}$ on $\mathcal{O}$ for MNIST data

Performance metric	Initial	Final
Accuracy	0.48	0.77
Precision	0.30	0.83
Recall	0.0.48	0.77
F_1-score	0.0.36	0.77

The results corresponding to the performance of the final classifier is provided in table. From table conf, we can see that, for this particular value of H, 3 new classes were identified by the framework as opposed to 2 unknown classes. This can be considered as the failure of the framework and is due to the very low value being set to the parameter H.

6.4.2 Experiment-2

For this experiment, the value of H is set equal to 4. Algorithm 2 is applied on the data and the results for the categorization phase are provided in Table 6.27. From this table, we see that exactly two unknown classes are discovered. Furthermore, 66% of the instances belonging to class *8* are classified correctly and majority of the remaining instances are misclassified as belonging to class *3* and class *9*. On the other hand, 86% of the instances belonging to class *9* are correctly classified and majority of the remaining instances are misclassified as class *4*. The overall accuracy of the classifier increases from 48% to 86% and the information regarding other performance metrics is provided in Table 6.28.

Table 6.27: Sensitivity analysis-3 (Experiment-2):Confusion matrix for the final performance of $\mathcal{C}$ on $\mathcal{O}$ for MNIST data

	Predicted class									
True class	0	1	2	3	4	5	6	7	8	9
0	365	0	0	1	0	0	2	0	0	0
1	0	412	2	1	1	0	0	0	1	0
2	3	2	341	0	2	0	1	4	7	0
3	0	0	5	369	0	2	1	7	16	9
4	0	2	0	0	342	0	2	1	4	10
5	2	1	0	4	0	303	6	0	5	14
6	2	1	0	0	0	2	357	0	0	0
7	0	1	9	0	3	0	0	372	1	4
8	9	58	72	134	20	74	17	4	986	116
9	15	6	16	18	93	6	1	31	18	1306

Table 6.28: Sensitivity analysis-3 (Experiment-2): Initial and final performance of $\mathcal{C}$ on $\mathcal{O}$ for MNIST data

Performance metric	Initial	Final
Accuracy	0.48	0.86
Precision	0.30	0.87
Recall	0.48	0.86
F_1-score	0.36	0.86

6.4.3 Experiment-3

For this experiment, the value of H is set equal to 7. Algorithm 2 is applied on the data and the results for the categorization phase are provided in Table 6.27. From this table, we see that exactly two unknown classes are discovered. However, only 13% of the instances belonging to class *8* are correctly classified correctly and 77% of the instances are misclassified as belonging to class *9*. On the other hand, 90% of the instances belonging to class *9* are correctly classified and majority of the remaining instances are misclassified as class *4* and class *7*. The overall accuracy of the classifier increases from 48% to 70% and the information

regarding other performance metrics is provided in Table 6.30.

Table 6.29: Sensitivity analysis-3 (Experiment-3): Confusion matrix for the final performance of $\mathcal{C}$ on $\mathcal{O}$ for MNIST data

	Predicted class									
True class	0	1	2	3	4	5	6	7	8	9
0	333	0	1	1	0	0	0	0	0	33
1	0	399	2	0	0	0	0	0	14	0
2	3	2	304	0	1	0	1	1	26	22
3	0	0	2	368	0	3	1	3	13	19
4	0	0	0	0	307	0	0	0	26	28
5	2	2	1	2	2	287	2	0	14	23
6	2	1	0	0	0	2	357	0	0	0
7	0	1	6	0	0	0	0	349	17	17
8	3	21	28	50	4	21	7	0	202	1154
9	1	4	1	17	24	5	0	28	73	1357

Table 6.30: Sensitivity analysis-3 (Experiment-3): Initial and final performance of $\mathcal{C}$ on $\mathcal{O}$ for MNIST data

Performance metric	Initial	Final
Accuracy	0.48	0.70
Precision	0.30	0.71
Recall	0.48	0.71
F_1-score	0.36	0.69

Conclusion Based on the experiments conducted, we see that a very low value or a very high value of parameter H resulted in the failure of the methodology. Furthermore, a very low value of H implies that we trying to categorize the flagged instances based on a single similarity pattern. As a result, very little information is used to categorize the instances as we are considering the similarity of identified instances with just one of the training classes. On the other hand, when the value of H is set to a high value, more information than required is used for

categorization. This information is no more informative as we are considering the similarity of the flagged instances with all the training classes. This induces noise into the process and the itemsets generated are associated to instances from multiple unknown classes as opposed to a particular unknown class. This eventually leads to incorrect categorization of the models and is evident from the results displayed in Table 6.36. Furthermore, we see a poor performance from the classifiers when H is set to extreme values. Hence, a moderate value of H is required to obtain desired results, and as a rule of thumb, we recommend this value to be equal to $|\mathcal{K}|/2$.

Table 6.31: Sensitivity analysis 3: Combined results of all experiments

	Experiments		
Parameter	1	2	3
Maximum probability limit (H)	1	4	7
Number of unknown classes discovered	3	2	2
Percentage correctly classified for class *8*	37	66	13
Percentage correctly classified for class *9*	93	86	90
F_1-score	0.77	0.86	0.69

6.5 Sensitivity analysis: Quality of classifier

Purpose This analysis does not correspond to any parameter associated with the methodology but relates to one of the characteristics of the classifier itself. For the purpose of analysis, we examine the effect of quality of the classifier on the overall performance of the framework. This is an important analysis to conduct as we encounter classifiers with varied performance depending on the problem.

Hypothesis A high quality classifier has better understanding of the patterns that exist within the training data and makes predictions on the known classes that are confident and accurate. The same classifier supplied with instances belonging to unknown classes makes predictions that are inaccurate and not confident. This discrepancy in predictions helps us to identify the unknown classes. On the other hand, a low quality classifier is inaccurate and not confident in making predictions on both the known classes as well as the unknown classes. This similarity in the performance, makes it difficult to distinguish between known and unknown classes. Hence, the performance of the framework improves with the increase in the quality of the classifier.

Experimental setup To maintain consistency, all the experiments are conducted on the MNIST data, where the training data $\mathcal{O}$ includes data corresponding to digits *0* to *7* and the open world data $\mathcal{O}$ contains data that belongs to all the digits *0* to *9*. Three experiments by varying the quality of the classifier are conducted. The details about the experimental settings are provided in Table 6.32.

Table 6.32: Sensitivity analysis 4: Experimental setup

Parameter	Experiments		
	1	2	3
Classifier	RF	RF	RF
Quality (Accuracy %)	High	Moderate	Low
Chebyshev's parameter (τ)	3.16	3.16	3.16
Maximum probability limit (H)	4	4	4
Termination parameter (ω)	0.25	0.25	0.25
Termination parameter (β)	0.1	0.1	0.1
Termination parameter (η)	0.1	0.1	0.1
Prediction confidence (γ)	0.4	0.4	0.4

Experiment -1

In the first experiment, we consider the classifier with a very high accuracy of 99%. The first step is the identification stage where we apply Algorithm 1 on $\mathcal{O}$, using the Chebyshev's parameter (τ) that is set to a value of 3.16. The results of the identification stage indicate that about 59% of the instances in $\mathcal{O}$ are identified (flagged) as belonging to the unknown classes. From Table 6.33, we analyze the distribution of the flagged instances in $\mathcal{F}$, and we see that about 83% of them belong to the unknown classes. Since we have more than 10% of the instances as flagged, we move forward to the next step, which is the categorization of flagged instances. Algorithm 2 is applied on $\mathcal{F}$ and the results from Table 6.34 indicate that Algorithm 2 successfully discovered both the unknown classes. Furthermore, we also see that about 74% of the instances belonging to class *8* are correctly classified while the remaining are spread across the other classes, of which majority of them are wrongly classified as class *3*. Similarly, for class *9*, 79% of them are correctly classified and the majority of the remaining instances are misclassified as class *4*. The overall accuracy of the classifier increases from 48% to 86% and the information regarding other performance metrics is provided in Table 6.35.

Experiment -2

In this experiment, we consider a classifier with a lower performance (Accuracy: 70%) compared to the classifier in the previous experiment. We apply Algorithm 1 on $\mathcal{O}$, using the Chebyshev's parameter (τ) that is set to a value of 3.16. The result of the identification stage is that none of the instances are flagged as belonging to the unknown classes. Since we do not have any data to proceed

Table 6.33: Sensitivity analysis-4 (Experiment-1): Classwise data distribution of $\mathcal{O}$ and $\mathcal{F}$ for MNIST data

Class	$\lvert\mathcal{O}\rvert$	$\lvert\mathcal{F}\rvert$
0	247	41
1	283	20
2	236	65
3	268	63
4	239	54
5	230	70
6	231	42
7	266	45
8	1005	1001
9	995	963

further, we terminate the algorithm here.

6.5.1 Experiment-3

For this experiment, we consider a classifier with a relatively low performance (Accuracy: 51%) compared to the previous two experiments. Similar to the previous experiment, none of the instances were identified for a critical value of 3.16 after the identification stage.

Conclusion The results from this analysis support our hypothesis that the quality of the classifier plays a crucial on the performance of the framework. The framework performed successfully when used in conjunction with a high quality classifier as supported by the results from Experiment 6.5. For Experiments 6.5 and 6.5.1, we see that the framework terminated in the first stage of the process itself as Algorithm 1 failed to identify instances that belong to the unknown classes.

Table 6.34: Sensitivity analysis-4 (Experiment-1):Confusion matrix for the final performance of $\mathcal{C}$ on $\mathcal{O}$ for MNIST data

	Predicted class									
True class	0	1	2	3	4	5	6	7	8	9
0	246	0	0	0	0	0	1	0	0	0
1	0	280	1	1	1	0	0	0	0	0
2	3	2	221	1	3	0	2	2	2	0
3	0	0	4	243	0	0	1	6	8	6
4	0	1	0	0	227	0	0	1	0	10
5	2	1	0	2	0	212	3	0	2	8
6	3	0	0	0	0	1	227	0	0	0
7	1	1	7	0	2	1	0	251	0	3
8	3	21	37	89	14	29	7	5	743	57
9	9	3	13	12	121	5	0	27	18	787

Table 6.35: Sensitivity analysis-4 (Experiment-1): Initial and final performance of $\mathcal{C}$ on $\mathcal{O}$ for MNIST data

Performance metric	Initial	Final
Accuracy	0.48	0.86
Precision	0.30	0.88
Recall	0.48	0.86
F_1-score	0.36	0.87

Table 6.36: Sensitivity analysis 4: Combined results of all experiments

	Experiments		
Parameter	1	2	3
Quality (Accuracy %)	High	Moderate	Low
Percentage of identified instances	59	0	0
Number of unknown classes discovered	2	-	-
F_1-score	0.86	-	-

Chapter 7

Case study - Social media analytics for community resilience

7.1 Introduction

Social media platforms such as Facebook and Twitter have become prevalent communication tools in modern society. These platforms provide a mechanism for collecting dynamic data on human behavior and sentiment. Such data has proven useful to study a variety of activity including crime prediction [75], disease outbreak [76], stock market prices [77], and political election results [78], among other things.

Recent studies consider the use of social media during natural disasters [79], studying mainly either the mood of the population or the various reactions of the public during a specific incident. Furthermore, most of the works that utilize social media data to support emergency management mainly rely on Twitter

data for analysis [80]. One of the first works in social media data analysis during disasters was in 2008 after the wildfires in South Carolina [81]. Since then, many case studies have been related to the Haiti earthquakes [82, 83] Hurricane Irene [84, 85], or Hurricane Sandy [86–89]. A summary of the ongoing research in the area of emergency management using Twitter as a source of data is provided in [90]. Most of the related analysis, especially those occurring in the United States, rely heavily on data collected from Twitter.

Twitter is a micro-blogging service in which, collectively, users broadcast hundreds of millions of brief messages daily [91]. One major characteristic of Twitter is that the messaging service is conducted in real-time and the day and time that the message is sent is recorded. Twitter messages, known as tweets, can also be labeled with keywords using the hashtag symbol (#) to allow messages to be categorized. The twitter feed (i.e., on-going stream of tweets sent to users) can be filtered based on these labels. Additionally, if the user creating a tweet has permitted location identification services from Twitter, the data include automatic geo-location coordinates embedded within the tweet. The real-time nature of this social media platform, the ability to search for specific keyword labels, and the ability to filter by date, time, and location facilitates data collection regarding how the engaged population react to major events.

In research related to the use of social media during natural disasters, several studies measure user sentiment. For example, [92] and [87] classify the tweet text as expressing Positive, Neutral or Negative emotions (also referred to as the sentiment polarity). [84] conducted a demographic analysis of the sentiment using the tweets corresponding to Hurricane Irene. A binary (positive or negative) or three-way (positive, negative, neutral) classification of sentiment are common levels of granularity used in sentiment analysis [86, 93–97]. On the other

hand, [98] performed a fine-grained sentiment analysis on the disaster-related tweets. The tweets were classified into 7 categories which include anger, disgust, fear, happiness, sadness, and surprise. [99] proposed a big data framework to analyze the user sentiment by applying various text mining and machine learning techniques on disaster related tweets. Apart from sentiment analysis on natural hazard related tweets, researchers have analyzed tweet sentiment during other types of emergencies such as the 2013 Boston Marathon bombing [100], the 2017 Las Vegas shooting [101], the Syrian refugee crisis [102] and the Ebola disease outbreak [103].

Works that use social media to tackle various disaster related tasks have proliferated in the recent times [104–107]. In research related to the use of social media during natural disasters, several studies measure user sentiment. For example, [92] and [87] classify the tweet text as expressing Positive, Neutral or Negative emotions (also referred to as the sentiment polarity). [84] conducted a demographic analysis of the sentiment using the tweets corresponding to Hurricane Irene. A binary (positive or negative) or three-way (positive, negative, neutral) classification of sentiment are common levels of granularity used in sentiment analysis [86, 93–97]. On the other hand, [98] performed a fine-grained sentiment analysis on the disaster-related tweets. The tweets were classified into 7 categories which include anger, disgust, fear, happiness, sadness, and surprise. [99] proposed a big data framework to analyze the user sentiment by applying various text mining and machine learning techniques on disaster related tweets. Apart from sentiment analysis on natural hazard related tweets, researchers have analyzed tweet sentiment during other types of emergencies such as the 2013 Boston Marathon bombing [100], the 2017 Las Vegas shooting [101], the Syrian refugee crisis [102] and the Ebola disease outbreak [103]. With an intention to pro-

vide valuable crisis response information to the humanitarian organizations, [108] have released a dataset that contains tweets belonging to various actionable categories. [109] used machine learning models to classify disaster related tweets into various categories of disaster management phases like mitigation, preparedness, emergency response and recovery. [110] classified tweets during Haiti disaster into multiple informative categories. [111] classified disaster related tweets that come under the category of situational awareness. A hybrid model combining both machine learning and rule based methods was proposed by [112] to classify disaster related tweets which can help the emergency responders identify people who are at risk. [113] used Convolutional Neural Networks to classify informative tweets from non-informative tweets. A summarization of research in the field of crisis management using twitter as the main source of information is provided by [90, 114].

These works explain the importance of social media analytics in effective disaster management and ameliorating the community resilience. The basic idea is to extract useful information from disaster related tweets by classifying them into various informative categories. One of the key aspects associated with developing the classification models for tweet classification is the preparation of training data. Literature suggests works that deal with different classification types such as informative vs non-informative, sentiment analysis, multi-class classification involving multiple informative classes, etc. Furthermore, in each of the these tasks, the decision about the type of classification or the categories for tweet classification are decided by the researcher and the training data is prepared accordingly. After the data preparation phase, machine learning models are trained to classify the tweets into different categories, to use them at the time of hazards that occur in the future. Since, the choice of tweet categories is manual and lim-

ited, it is possible for the classifiers to encounter tweets from categories that are not present in the training data. This opens up the need to address the problem of open world classification in social media analytics for disaster management. In Section 7.2, we demonstrate the occurrence OWC on Twitter analysis and also the performance of the proposed framework on a sample real world dataset.

7.2 Application of the proposed framework on disaster related Twitter data

We apply the proposed framework on Crisis MMD dataset [115], which comprises of human-annotated tweets collected during different major disasters. The dataset under consideration contains 10,347 tweets corresponding to different humanitarian categories such as affected individuals, infrastructure and utility damage, injured or dead people and, rescue volunteering or donation effort.

For the experiment, "rescue volunteering or donation effort" class is considered to be the unknown class while the other classes are the known classes. In order to be consistent with the testing and the sensitivity analysis, we use random forest for the classification using the Tf-idf based word representation. The distribution of the open world data and the identified instances is displayed in Table 7.1. From this table, we see that about 30% of the instances in $\mathcal{O}$ are flagged as belonging to the unknown class. Furthermore, 77% of the flagged instances belong to the unknown class. Since, we have more than 10% of the instances as flagged in the first stage,we proceed to the second stage based on the parameter settings displayed in Table 7.2. The results indicate that the algorithm automatically terminated after discovering the presence of a single unknown class in

Table 7.1: Class wise data distribution of $\mathcal{O}$ and $\mathcal{F}$ for Twitter data

Class	class information	$\lvert\mathcal{O}\rvert$	$\lvert\mathcal{F}\rvert$
0	affected individuals	193	83
1	infrastructure and utility damage	479	57
2	injured or dead people	196	53
3	rescue volunteering or donation effort	1977	641

the open world data. Furthermore, Table 7.4 depicts that 82% of the instances that belong to class *3* are classified correctly and the majority of the remaining instances are misclassified as class *1*. Before applying our framework, the distribution of the predictions made by the initial classifier on the open world data is provided in Table 7.3. This table shows, that majority of the instances in $\mathcal{O}$ that belong to the unknown class are misclassified as class *1*. This shows that class *3* possess semantic similarity with class *1* in the vector space, resulting in the misclassification as class *1*, before and after the application of the framework. From Tables 7.4 and 7.6,we see that in presence of of an unknown class, the performance of the classifier on the known classes is not degraded much, but the framework automatically discovered the single unknown class and correctly classified 1630 instances out of 1977 instances totally present. This case study demonstrates the possibility of open world problems in social media analytics and also the applicability of the proposed framework to such real world applications.

Table 7.2: Parameters setting for Algorithm 2 - Twitter data

Parameter	Value
Maximum probability limit (H)	3
Termination parameter (ω)	0.25
Termination parameter (β)	0.1
Prediction confidence (γ)	0.6

Table 7.3: Confusion matrix for the initial performance of $\mathcal{C}$ on $\mathcal{O}$ for Twitter data

	Predicted class			
True class	0	1	2	3
0	49	128	16	0
1	7	467	5	0
2	3	36	157	0
3	252	1631	94	0

Table 7.4: Confusion matrix for the final performance of $\mathcal{C}$ on $\mathcal{O}$ for Twitter data

	Predicted class			
True class	0	1	2	3
0	23	50	9	111
1	4	383	2	90
2	0	20	150	26
3	19	321	7	1630

Table 7.5: Initial and final performance of $\mathcal{C}$ on $\mathcal{O}$ for Twitter data

Performance metric	Initial	Final
Accuracy	0.24	0.77
Precision	0.08	0.79
Recall	0.24	0.77
F_1-score	0.11	0.76

Table 7.6: Initial and final performance of $\mathcal{C}$ on known classes for Twitter data

Performance metric	Initial	Final
Accuracy	0.82	0.69
Precision	0.83	0.85
Recall	0.82	0.69
F_1-score	0.79	0.72

Chapter 8

Summary

8.1 Conclusion

To date, the problem of open world classification has been addressed primarily in the domain of computer vision. Most of the works that are available in the literature try to solve this problem especially for the image data. The main goal of this work is to be able to generalize the problem of open world classification without confining it to a particular domain or the data being worked upon. To the best of our knowledge, there is no work in the literature that solves the problem of open world classification, irrespective of the nature of the data and also the classifier in use. We address this issue by developing a framework that is based on the projection of data on to a probabilistic space as opposed to working with the original feature space. Hence, this projection provides the method to generalize to any type of data. Also, the idea of projecting the data onto a probabilistic space makes the methodology independent of the classifier, as it is possible to generate the probabilistic space using any classifier. Furthermore, most of the techniques available in the literature that address this issue are either data spe-

cific or algorithmic specific. Considering this issue, we propose a methodology that acts like an accessory to any classifier to solve the problem of open world classification.

It is evident from the literature review that most of the works in this field try to build models that focus mainly on the identification phase. This is just a partial solution and is complete only when the identified data points are organized into their respective classes. While there are a few works in the literature that address the problem of discovering the unknown classes, their solutions are not accurate as they just provide an estimate as to how many unknown classes can be present in the open world data. To address this issue, in this paper, we propose a novel way of categorizing the data instances belonging to unknown classes based on the idea of residual signature. Each unknown class can possess a unique signature in the probabilistic space and this idea is leveraged to group the data. The experiments prove the successful working of the proposed methodology for different number of unknown classes and also for different types of data. The case study presented in this paper demonstrates the applicability of the proposed framework to social media analytics to improve community resilience.

8.2 Limitations

Ideally, we would expect Algorithm 1 to flag only those instances that belong to the unknown classes. However, we see some of the observations from the known classes also being flagged by the algorithm. From the results in Section 6.5, we see that the framework does not perform as expected when used in conjunction with a low quality classifier. Another limitation is with regards to Algorithm 2 where the performance of the framework is relatively poor when there are fewer number

of known classes present in the data. This is specifically true when the number of unknown classes is greater than the number of known classes. Furthermore, when few instances are associated with a particular itemset during the categorization phase, causes imbalance in the dataset which makes the task of training the machine learning model difficult.

In Algorithm 2, the class labels can be replaced with the actual probability values to generate the itemsets. Techniques such as [35–40], *one shot learning* [41, 42] can be applied when dealing with imbalanced data sets.

8.3 Future work

This framework is based on the assumption that the open world data is already available at once, which is basically referred to as the offline setting. One of the future works could be based on adapting this framework to the online setting, where a single or relatively small number of instances are provided to the framework at a given instance of time.

Bibliography

[1] Walter J Scheirer, Anderson de Rezende Rocha, Archana Sapkota, and Terrance E Boult. Toward open set recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(7):1757–1772, 2013.

[2] Abhijit Bendale and Terrance Boult. Towards open world recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1893–1902, 2015.

[3] Walter J Scheirer, Lalit P Jain, and Terrance E Boult. Probability models for open set recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(11):2317–2324, 2014.

[4] Lalit P Jain, Walter J Scheirer, and Terrance E Boult. Multi-class open set recognition using probability of inclusion. In *European Conference on Computer Vision*, pages 393–409. Springer, 2014.

[5] Matthew D Scherreik and Brian D Rigling. Open set recognition for automatic target classification with rejection. *IEEE Transactions on Aerospace and Electronic Systems*, 52(2):632–642, 2016.

[6] Thomas Mensink, Jakob Verbeek, Florent Perronnin, and Gabriela Csurka. Distance-based image classification: Generalizing to new classes at near-

zero cost. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(11):2624–2637, 2013.

[7] Marko Ristin, Matthieu Guillaumin, Juergen Gall, and Luc Van Gool. Incremental learning of ncm forests for large-scale image classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3654–3661, 2014.

[8] Pedro R Mendes Júnior, Roberto M De Souza, Rafael de O Werneck, Bernardo V Stein, Daniel V Pazinato, Waldir R de Almeida, Otávio AB Penatti, Ricardo da S Torres, and Anderson Rocha. Nearest neighbors distance ratio open-set classifier. *Machine Learning*, 106(3):359–386, 2017.

[9] Rakesh Agrawal, Tomasz Imieliński, and Arun Swami. Mining association rules between sets of items in large databases. In *Proceedings of the 1993 ACM SIGMOD international conference on Management of data*, pages 207–216, 1993.

[10] Abhijit Bendale and Terrance E Boult. Towards open set deep networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1563–1572, 2016.

[11] Lei Shu, Hu Xu, and Bing Liu. Doc: Deep open classification of text documents. *arXiv preprint arXiv:1709.08716*, 2017.

[12] Navid Kardan and Kenneth O Stanley. Mitigating fooling with competitive overcomplete output layer neural networks. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 518–525. IEEE, 2017.

[13] Mehadi Hassen and Philip K Chan. Learning a neural-network-based representation for open set recognition. *arXiv preprint arXiv:1802.04365*, 2018.

[14] Poojan Oza and Vishal M Patel. C2ae: Class conditioned auto-encoder for open-set recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2307–2316, 2019.

[15] Poojan Oza and Vishal M Patel. Deep cnn-based multi-task learning for open-set recognition. *arXiv preprint arXiv:1903.03161*, 2019.

[16] Konrad Rieck, Philipp Trinius, Carsten Willems, and Thorsten Holz. Automatic analysis of malware behavior using machine learning. *Journal of Computer Security*, 19(4):639–668, 2011.

[17] Manuel Gunther, Steve Cruz, Ethan M Rudd, and Terrance E Boult. Toward open-set face recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 71–80, 2017.

[18] Hakan Cevikalp, Bill Triggs, and Vojtech Franc. Face and landmark detection by using cascade of classifiers. In *2013 10th IEEE International Conference and Workshops on Automatic Face and Gesture Recognition (FG)*, pages 1–7. IEEE, 2013.

[19] Geli Fei and Bing Liu. Breaking the closed world assumption in text classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 506–514, 2016.

[20] Sridhama Prakhya, Vinodini Venkataram, and Jugal Kalita. Open set text classification using cnns. In *Proceedings of the 14th International Conference on Natural Language Processing (ICON-2017)*, pages 466–475, 2017.

[21] ZongYuan Ge, Sergey Demyanov, Zetao Chen, and Rahil Garnavi. Generative openmax for multi-class open set classification. *arXiv preprint arXiv:1707.07418*, 2017.

[22] Lawrence Neal, Matthew Olson, Xiaoli Fern, Weng-Keen Wong, and Fuxin Li. Open set learning with counterfactual images. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 613–628, 2018.

[23] Hakan Cevikalp. Best fitting hyperplanes for classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6):1076–1088, 2016.

[24] He Zhang and Vishal M Patel. Sparse representation-based open set recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(8):1690–1696, 2016.

[25] Edoardo Vignotto and Sebastian Engelke. Extreme value theory for open set classification–gpd and gev classifiers. *arXiv preprint arXiv:1808.09902*, 2018.

[26] Andras Rozsa, Manuel Günther, and Terrance E Boult. Adversarial robustness: Softmax versus openmax. *arXiv preprint arXiv:1708.01697*, 2017.

[27] Douglas O Cardoso, João Gama, and Felipe MG França. Weightless neural networks for open set recognition. *Machine Learning*, 106(9-10):1547–1567, 2017.

[28] Ryota Yoshihashi, Wen Shao, Rei Kawakami, Shaodi You, Makoto Iida, and Takeshi Naemura. Classification-reconstruction learning for open-set recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4016–4025, 2019.

[29] Ethan M Rudd, Lalit P Jain, Walter J Scheirer, and Terrance E Boult. The extreme value machine. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(3):762–768, 2017.

[30] Inhyuk Jo, Jungtaek Kim, Hyohyeong Kang, Yong-Deok Kim, and Seungjin Choi. Open set recognition by regularising classifier with fake data generated by generative adversarial networks. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2686–2690. IEEE, 2018.

[31] Chuanxing Geng, Sheng-jun Huang, and Songcan Chen. Recent advances in open set recognition: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[32] Lei Shu, Hu Xu, and Bing Liu. Unseen class discovery in open-world classification. *arXiv preprint arXiv:1801.05609*, 2018.

[33] Chuanxing Geng and Songcan Chen. Collective decision for open set recognition. *arXiv preprint arXiv:1806.11258*, 2018.

[34] Olivier Chapelle, Bernhard Scholkopf, and Alexander Zien. Semi-supervised learning (chapelle, o. et al., eds.; 2006)[book reviews]. *IEEE Transactions on Neural Networks*, 20(3):542–542, 2009.

[35] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *Advances in neural information processing systems*, pages 4077–4087, 2017.

[36] Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. 2016.

[37] Victor Garcia and Joan Bruna. Few-shot learning with graph neural networks. *arXiv preprint arXiv:1711.04043*, 2017.

[38] Brenden M Lake, Russ R Salakhutdinov, and Josh Tenenbaum. One-shot learning by inverting a compositional causal process. In *Advances in neural information processing systems*, pages 2526–2534, 2013.

[39] Luca Bertinetto, João F Henriques, Jack Valmadre, Philip Torr, and Andrea Vedaldi. Learning feed-forward one-shot learners. In *Advances in neural information processing systems*, pages 523–531, 2016.

[40] Yanwei Fu, Timothy M Hospedales, Tao Xiang, and Shaogang Gong. Learning multimodal latent attributes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(2):303–316, 2013.

[41] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. In *Advances in neural information processing systems*, pages 3630–3638, 2016.

[42] Li Fei-Fei, Rob Fergus, and Pietro Perona. One-shot learning of object categories. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(4):594–611, 2006.

[43] Mark Palatucci, Dean Pomerleau, Geoffrey E Hinton, and Tom M Mitchell. Zero-shot learning with semantic output codes. In *Advances in neural information processing systems*, pages 1410–1418, 2009.

[44] Christoph H Lampert, Hannes Nickisch, and Stefan Harmeling. Learning to detect unseen object classes by between-class attribute transfer. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 951–958. IEEE, 2009.

[45] Yanwei Fu, Tao Xiang, Yu-Gang Jiang, Xiangyang Xue, Leonid Sigal, and Shaogang Gong. Recent advances in zero-shot recognition: Toward data-efficient understanding of visual content. *IEEE Signal Processing Magazine*, 35(1):112–125, 2018.

[46] Hal Daume III and Daniel Marcu. Domain adaptation for statistical classifiers. *Journal of Artificial Intelligence research*, 26:101–126, 2006.

[47] Shai Ben-David, John Blitzer, Koby Crammer, and Fernando Pereira. Analysis of representations for domain adaptation. In *Advances in neural information processing systems*, pages 137–144, 2007.

[48] Hal Daumé III. Frustratingly easy domain adaptation. *arXiv preprint arXiv:0907.1815*, 2009.

[49] Vishal M Patel, Raghuraman Gopalan, Ruonan Li, and Rama Chellappa. Visual domain adaptation: A survey of recent advances. *IEEE signal processing magazine*, 32(3):53–69, 2015.

[50] Makoto Yamada, Leonid Sigal, and Yi Chang. Domain adaptation for structured regression. *International journal of computer vision*, 109(1-2):126–145, 2014.

[51] Gerhard Widmer and Miroslav Kubat. Learning in the presence of concept drift and hidden contexts. *Machine learning*, 23(1):69–101, 1996.

[52] Indrė Žliobaitė. Learning under concept drift: an overview. *arXiv preprint arXiv:1010.4784*, 2010.

[53] João Gama, Indrė Žliobaitė, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia. A survey on concept drift adaptation. *ACM computing surveys (CSUR)*, 46(4):1–37, 2014.

[54] Solomon Kullback and Richard A Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951.

[55] Jianhua Lin. Divergence measures based on the shannon entropy. *IEEE Transactions on Information Theory*, 37(1):145–151, 1991.

[56] John G Saw, Mark CK Yang, and Tse Chin Mo. Chebyshev inequality with estimated mean and variance. *The American Statistician*, 38(2):130–132, 1984.

[57] Yen-Liang Chen, Kwei Tang, Ren-Jie Shen, and Ya-Han Hu. Market basket analysis in a multiple store environment. *Decision support systems*, 40(2):339–354, 2005.

[58] Gary J Russell and Ann Petersen. Analysis of cross category dependence in market basket selection. *Journal of Retailing*, 76(3):367–392, 2000.

[59] Dragan Gamberger, Nada Lavrac, and Viktor Jovanoski. High confidence association rules for medical diagnosis. 1999.

[60] Jesmin Nahar, Tasadduq Imam, Kevin S Tickle, and Yi-Ping Phoebe Chen. Association rule mining to detect factors which contribute to heart disease in males and females. *Expert Systems with Applications*, 40(4):1086–1093, 2013.

[61] Nitin Gupta, Nitin Mangal, Kamal Tiwari, and Pabitra Mitra. Mining quantitative association rules in protein sequences. In *Data Mining*, pages 273–281. Springer, 2006.

[62] Kwong-Sak Leung, Ka-Chun Wong, Tak-Ming Chan, Man-Hon Wong, Kin-Hong Lee, Chi-Kong Lau, and Stephen KW Tsui. Discovering protein–dna binding sequence patterns using association rule mining. *Nucleic acids research*, 38(19):6324–6337, 2010.

[63] Sung Hee Park, José A Reyes, David R Gilbert, Ji Woong Kim, and Sangsoo Kim. Prediction of protein-protein interaction types using association rule based classification. *BMC bioinformatics*, 10(1):36, 2009.

[64] Osmar R Zaiane, Jiawei Han, and Hua Zhu. Mining recurrent items in multimedia with progressive resolution refinement. In *Proceedings of 16th International Conference on Data Engineering (Cat. No. 00CB37073)*, pages 461–470. IEEE, 2000.

[65] Mei-Ling Shyu, Shu-Ching Chen, and Rangasami L Kashyap. Generalized affinity-based association rule mining for multimedia database queries. *Knowledge and Information Systems*, 3(3):319–337, 2001.

[66] James J Treinen and Ramakrishna Thurimella. A framework for the application of association rule mining in large intrusion detection infrastructures. In *International Workshop on Recent Advances in Intrusion Detection*, pages 1–18. Springer, 2006.

[67] Shingo Mabu, Ci Chen, Nannan Lu, Kaoru Shimada, and Kotaro Hirasawa. An intrusion-detection model based on fuzzy class-association-rule mining using genetic network programming. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 41(1):130–139, 2010.

[68] Donato Malerba, Floriana Esposito, Francesca A Lisi, and Annalisa Appice. Mining spatial association rules in census data. *Research in Official Statistics. v5 i1*, pages 19–44, 2003.

[69] Qin Ding, Qiang Ding, and William Perrizo. Association rule mining on remotely sensed images using p-trees. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 66–79. Springer, 2002.

[70] Umamaheshwaran Rajasekar and Qihao Weng. Application of association rule mining for exploring the relationship between urban land surface temperature and biophysical/social parameters. *Photogrammetric Engineering & Remote Sensing*, 75(4):385–396, 2009.

[71] Rakesh Agrawal, Ramakrishnan Srikant, et al. Fast algorithms for mining association rules. In *Proc. 20th int. conf. very large data bases, VLDB*, volume 1215, pages 487–499, 1994.

[72] Brian A Johnson and Kotaro Iizuka. Integrating openstreetmap crowdsourced data and landsat time-series imagery for rapid land use/land cover

(lulc) mapping: Case study of the laguna de bay area of the philippines. *Applied Geography*, 67:140–149, 2016.

[73] Davide Anguita, Alessandro Ghio, Luca Oneto, Xavier Parra, and Jorge Luis Reyes-Ortiz. A public domain dataset for human activity recognition using smartphones. In *Esann*, volume 3, page 3, 2013.

[74] Yann LeCun, Corinna Cortes, and CJ Burges. Mnist handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, 2, 2010.

[75] X. Wang, M. Gerber, and D. Brown. Automatic crime prediction using events extracted from twitter posts. In *Proceedings of the 2012 5th International Conference on Social Computing, Behavioral-Cultural Modeling and Prediction*, pages 231–238, 2012.

[76] J. Ritterman, M. Osborne, and E. Klein. Using prediction markets and twitter to predict a swine flu pandemic. In *Proceedings of the 1st International Workshop on Mining Social Media*, volume 9, pages 9–17, 2009.

[77] J. Si, A. Mukherjee, B. Liu, Q. Li, H. Li, and X. Deng. Exploiting topic based twitter sentiment for stock prediction. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistic*, pages 24–29, 2013.

[78] A. Tumasjan, T. Sprenger, P. Sandner, and I. Welpe. Predicting elections with Twitter: What 140 characters reveal about political sentiment. In *Proceedings of the 4th International Conference of Weblogs and Social Media*, volume 10, pages 178–185, 2013.

[79] Mark T Riccardi. The power of crowdsourcing in disaster response operations. *International Journal of Disaster Risk Reduction*, 20:123–128, 2016.

[80] Christian Reuter, Gerhard Backfried, Marc-André Kaufhold, and Fabian Spahr. ISCRAM turns 15: A trend analysis of social media papers 2004-2017. In *Proceedings of the 15th ISCRAM conference*, 2018.

[81] Jeannette Sutton, Leysia Palen, and Irina Shklovski. Backchannels on the front lines: Emergent uses of social media in the 2007 southern California wildfires. In *Proceedings of the 5th International ISCRAM Conference*, pages 624–632, Washington, DC, United States, May 2008.

[82] Sidharth Muralidharan, Leslie Rasmussen, Daniel Patterson, and Jae-Hwa Shin. Hope for Haiti: An analysis of Facebook and Twitter usage during the earthquake relief efforts. *Public Relations Review*, 37(2):175–177, 2011.

[83] C. Caragea, N. McNeese, A. Jaiswal, G. Traylor, H-W Kim, P. Mitrat, D. Wu, A. Tapia, L. Giles, B. Jansen, and J. Yen. Classifying text messages for the Haiti earthquake. In *Proceedings of the 8th International ISCRAM Conference*, 2011.

[84] Benjamin Mandel, Aron Culotta, John Boulahanis, Danielle Stark, Bonnie Lewis, and Jeremy Rodrigue. A demographic analysis of online sentiment during hurricane irene. In *Proceedings of the Second Workshop on Language in Social Media*, pages 27–36. Association for Computational Linguistics, 2012.

[85] Karen Freberg, Kristin Saling, Kathleen G. Vidoloff, and Gina Eosco. Using value modeling to evaluate social media messages: The case of Hurricane Irene. *Public Relations Review*, 39(3):185–192, 2013.

[86] Han Dong, Milton Halem, and Shujia Zhou. Social media data analytics applied to hurricane sandy. In *2013 International Conference on Social Computing*, pages 963–966. IEEE, 2013.

[87] Cornelia Caragea, Anna Squicciarini, Sam Stehle, Kishore Neppalli, and Andrea Tapia. Mapping moods: Geo-mapped sentiment analysis during Hurricane Sandy. In *Proceedings of the 11th International ISCRAM Conference*, University Park, PA, United States, May 2014.

[88] Y. Kryvasheyeu, H. Chen, E. Moro, P. Van Hentenryck, and M. Cebrian. Performance of social network sensors during Hurricane Sandy. *PLoS ONE*, 10(2):1–19, 2015.

[89] Mehdi Jamali, Ali Nejat, Souparno Ghosh, Fang Jin, and Guofeng Cao. Social media data and post-disaster recovery. *International Journal of Information Management*, 44:25–37, 2019.

[90] Maria Martinez-Rojas, Maria del Carmen Pardo-Ferreira, and Juan Carlos Rubio-Romero. Twitter as a tool for the management and analysis of emergency situations: A systematic literature review. *International Journal of Information Management*, 43:196–208, 2018.

[91] Internet live stats. Twitter usage statistics. `http://www.internetlivestats.com/twitter-statistics/`, 2018. Accessed: 2018-09-18.

[92] Ahmed Nagy and Jeannie Stamberger. Crowd sentiment detection during disasters and crises. In *Proceedings of the 9th International ISCRAM Conference*, pages 1–9, Vancouver, Canada, April 2012.

[93] A. Go, R. Bhayani, and L. Huang. Twitter sentiment classification using distant supervision. Technical report, Stanford University, 2009.

[94] A. Agarwal, B. Xie, I. Vovsha, O. Rambow, and R. Passonneau. Sentiment analysis of Twitter data. In *Proceedings of the ACL 2011 Workshop on Languages in Social Media*, pages 30–38, 2011.

[95] F. Cruz, J. Troyano, F. Enríquez, F. Ortega, and C. Vallejo. 'Long autonomy or long delay?' The importance of domain in opinion mining. *Expert Systems with Applications*, 40(8):3174–3184, 2013.

[96] S.-T. Li and F.-C. Tsai. A fuzzy conceptualization model for text mining with application in opinion polarity classification. *Knowledge-Based Systems*, 39:23–33, 2013.

[97] Venkata K. Neppalli, Cornelia Caragea, Anna Squicciarini, Andrea Tapia, and Sam Stehle. Sentiment analysis during Hurricane Sandy in emergency response. *International Journal of Disaster Risk Reduction*, 21:213–222, 2017.

[98] Axel Schulz, Tung Dang Thanh, Heiko Paulheim, and Immanuel Schweizer. A fine-grained sentiment analysis approach for detecting crisis related microposts. In *ISCRAM*, 2013.

[99] Rexiline J Ragini, Rubesh P.M Anand, and Vidhyacharanand Bhaskar. Big data analytics for disaster response and recovery through sentiment. *International Journal of Information Management*, 42:13–24, 2018.

[100] Jaeung Lee, Basma Abdul Rehman, Manish Agrawal, and H Raghav Rao. Sentiment analysis of twitter users over time: the case of the boston bombing tragedy. In *Workshop on E-Business*, pages 1–14. Springer, 2015.

[101] Neha Singh, Nirmalya Roy, and Aryya Gangopadhyay. Analyzing the sentiment of crowd for improving the emergency response services. In *2018 IEEE International Conference on Smart Computing (SMARTCOMP)*, pages 1–8. IEEE, 2018.

[102] Nazan Öztürk and Serkan Ayvaz. Sentiment analysis on twitter: A text mining approach to the syrian refugee crisis. *Telematics and Informatics*, 35(1):136–147, 2018.

[103] Yafeng Lu, Xia Hu, Feng Wang, Shamanth Kumar, Huan Liu, and Ross Maciejewski. Visualizing social media sentiment in disaster scenarios. In *Proceedings of the 24th International Conference on World Wide Web*, pages 1211–1215. ACM, 2015.

[104] Mark T Riccardi. The power of crowdsourcing in disaster response operations. *International Journal of Disaster Risk Reduction*, 20:123–128, 2016.

[105] Muhammad Imran, Shady Elbassuoni, Carlos Castillo, Fernando Diaz, and Patrick Meier. Practical extraction of disaster-relevant information from social media. In *Proceedings of the 22nd International Conference on World Wide Web*, pages 1021–1024. ACM, 2013.

[106] France Cheong and Christopher Cheong. Social media data mining: A social network analysis of tweets during the 2010-2011 australian floods. *PACIS*, 11:46–46, 2011.

[107] Shamanth Kumar, Geoffrey Barbier, Mohammad Ali Abbasi, and Huan Liu. Tweettracker: An analysis tool for humanitarian and disaster relief. In *Fifth international AAAI conference on weblogs and social media*, 2011.

[108] Firoj Alam, Ferda Ofli, and Muhammad Imran. Crisismmd: Multimodal twitter datasets from natural disasters. In *Twelfth International AAAI Conference on Web and Social Media*, 2018.

[109] Qunying Huang and Yu Xiao. Geographic situational awareness: mining tweets for disaster preparedness, emergency response, impact, and recovery. *ISPRS International Journal of Geo-Information*, 4(3):1549–1568, 2015.

[110] 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. Classifying text messages for the haiti earthquake. In *Proceedings of the 8th international conference on information systems for crisis response and management (ISCRAM2011)*. Citeseer, 2011.

[111] Sudha Verma, Sarah Vieweg, William J Corvey, Leysia Palen, James H Martin, Martha Palmer, Aaron Schram, and Kenneth M Anderson. Natural language processing to the rescue? extracting" situational awareness" tweets during mass emergency. In *Fifth International AAAI Conference on Weblogs and Social Media*, 2011.

[112] J Rexiline Ragini, PM Rubesh Anand, and Vidhyacharan Bhaskar. Mining crisis information: A strategic approach for detection of people at risk through social media analysis. *International journal of disaster risk reduction*, 27:556–566, 2018.

[113] Cornelia Caragea, Adrian Silvescu, and Andrea H Tapia. Identifying informative messages in disaster events using convolutional neural networks. In *International Conference on Information Systems for Crisis Response and Management*, pages 137–147, 2016.

[114] Sergio Luna and Michael J Pennock. Social media applications and emergency management: A literature review and research agenda. *International journal of disaster risk reduction*, 28:565–577, 2018.

[115] Firoj Alam, Ferda Ofli, and Muhammad Imran. Crisismmd: Multimodal twitter datasets from natural disasters. In *Proceedings of the 12th International AAAI Conference on Web and Social Media (ICWSM)*, June 2018.