Computer Science Faculty Research & Creative Works

Computer Science

01 Nov 2019

A Deep Learning Approach for Tweet Classification and Rescue Scheduling for Effective Disaster Management

Md. Yasin Kabir

Sanjay Kumar Madria
Missouri University of Science and Technology, madrias@mst.edu

Recommended Citation

A Deep Learning Approach for Tweet Classification and Rescue Scheduling for Effective Disaster Management

Md. Yasin Kabir
Department of Computer Science
Missouri University of Science and Technology, USA
mkdv6@mst.edu

Sanjay Madria
Department of Computer Science
Missouri University of Science and Technology, USA
madrias@mst.edu

ABSTRACT

Every activity in disaster management demands accurate and up-to-date information to allow a quick, easy, and cost-effective response to reduce the possible loss of lives and properties. It is a challenging and complex task to acquire information from different regions of a disaster-affected area in a timely fashion. The extensive spread and reach of social media and networks such as Twitter allow people to share information in real-time. However, gathering of valuable information requires a series of operations such as (1) processing each tweet for the text classification, (2) possible location determination of people needing help based on tweets, and (3) priority calculations of rescue tasks based on the classification of tweets. These are three primary challenges in developing an effective rescue scheduling operation using social media data. In this paper, first, we propose a deep learning model combining attention based Bi-directional Long Short-Term Memory (BLSTM) and Convolutional Neural Network (CNN) to classify the tweets. Next, we perform feature engineering to create an auxiliary feature map which dramatically increases the model accuracy. In our experiments using data from Hurricanes Harvey and Irma, it is observed that our proposed approach performs better compared to other classification methods based on Precision, Recall, F1-score, and Accuracy, and is highly effective to determine the priority of a tweet. Furthermore, to evaluate the effectiveness and robustness of the proposed classification model a merged dataset comprises of 4 different datasets from CrisisNLP and another 15 different disasters data from CrisisLex are used. Finally, we develop an adaptive multi-task hybrid scheduling algorithm considering resource constraints to perform an effective rescue scheduling operation considering different rescue priorities.

CCS CONCEPTS

• **Human-centered computing** → **Social media**; • **Computing methodologies** → **Neural networks**; • **Theory of computation** → Scheduling algorithms.

KEYWORDS

Deep Learning, Neural Network, Social Media, Disaster management, Rescue Scheduling, Priority Determination.

1 INTRODUCTION

Social media such as Twitter and Facebook experiencing mass adaptation and exponential growth. The roles of social media extended but not limited to health and disease analysis and propagation detection [24], Quantifying controversial information [8], and disaster crisis management [1, 31]. Natural disasters frequently disrupt regular communication due to the damaged infrastructures [26] which lead to an outflow of information. A report on Hurricane Sandy [4] shows that people were using social media more frequently to communicate. People were seeking help quickly and promptly as they strive to contact friends and family in and out of the disaster area, looking for information regarding transport, shelter, and food. Hence, The huge flow of information over social media can be beneficial in managing a natural disaster more effectively. During Hurricane Sandy, Twitter proved its usefulness, and at the time of Hurricane Harvey and Irma, again Twitter played a crucial role in the rescue, donation, and recovery. Figure 1 represents two tweets seeking rescue during Hurricane Harvey. People also tweeted similarly at the time of Hurricane Irma. However, while the use of social network seems appealing, still most of the applications are lacking features and fall short in their usability [18].

> #houstonflood please help my friend 11711 wood shadow Houston TX 77013 she stuck with 5 kids
> 7:06 AM - 27 Aug 2017
>
> Water rising quick please help ASAP please!! 820 smith street, port Arthur Texas,77640
> 11:41 PM - 29 Aug 2017
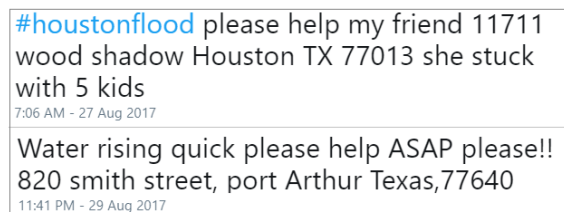
Figure 1: Examples of rescue requesting tweets

Institutional and Volunteer rescue efforts save a lot of lives during a crisis. However, those rescue missions are not well-organized and structured due to uncertainty. Individual volunteers have time constraints and lack of resources. Moreover, some rescue missions might need extra precaution, advanced equipment, and medical facilities. Besides that, due to the variety of help requesting tweets, some of those tweets might be out of sight. Hence, an automated system is essential to understand the context of the tweets, classify

the specific tweets for rescue, prioritize those tweets based on context, and then schedule rescue missions and allocate necessary resources accordingly. Our primary contributions in this paper are:

- Developing a multi-headed binary classifier to classify the tweets into six different classes using deep learning where a single tweet can belong to multiple classes. We use a unique machine learning pipeline with a set of punctuation-based auxiliary features which are specifically correlated with the disaster-related tweets.
- Evaluating and comparing the proposed model with different machine learning models and diverse datasets.
- We formally introduce a method for priority determination of each rescue request which plays a crucial role in maintaining fairness in the rescue scheduling.
- We propose a resource constraint and burst time adaptive rescue scheduling algorithm with multi-tasking and priority balancing to perform improved rescue operations.

2 RELATED WORK

2.1 Social Media for Disaster Management

Most of the prior research work research works using social media and networks for disaster management are focused on assessing the disaster situation, and a little, if any, is focused on their use in rescue mission and planning. Authors in [31] proposed a rescue scheduling algorithm on Hurricane Harvey which connects the victims with the scattered volunteers. A heuristic multi-agent reinforcement learning scheduling algorithm, named as ResQ [21], utilizes reinforcement learning to coordinates the volunteers and the victims during a disaster. [32] proposed a system that uses machine learning mechanism to extract the data that is generated by Twitter messages during a crisis. Authors in [7] presented how social media communication was used during the catastrophic Haiti earthquake. They adapted the method of crowd-sourcing for designing coordination protocols and mechanisms in order to create coordination between the organizations and their relief activities. [27] analyzed the extensive use of Twitter data in case of mass convergence or disaster situation such as the Southern California Wildfire. After several devastating incidents, a few disaster management applications such as Ushahidi [22] have been developed.

2.2 Tweets Classification

The basic approach for tweet classification is to extract features from the text. Support Vector Machines (SVM) is one of the widely used entity in classification, which draws a linear separator plane among the classes [2]. To perform the classification, K-Nearest Neighbor Classifier [10] offers proximity-based classifier, and uses distance measurement among the words.

The idea of the deep neural network for natural language processing first used in [16] uses a multitask learning model using the neural network. [5] proposed a deep neural network consisting of 29 layers for natural language processing. [13] and [20] showed that combining with external pre-learned word vectors such as GloVe [25], a neural network can be trained better for the disaster datasets. Our proposed deep learning model took inspiration form their work. However, those works did not consider any auxiliary features or attention layer. As a tweet has character length restriction, attention

layer with domain-specific engineered auxiliary features can be highly influential. In this work, we create a set of auxiliary features and use an attention based deep neural network to classify the tweets into 6 different classes where each class represents a binary output label, and a single tweet can belong to multiple classes.

2.3 Scheduling Algorithms

The scheduling algorithms intend to optimize the time and the use of resources among different parties employing certain constraints. The primary purpose of a scheduling algorithm is to ensure fairness among the participants while maximizing resource utilization. First-Come-First-Served (FCFS) algorithm can not provide fairness when someone cannot wait to use the resource or when someone needs a priority based on a situation. [17] worked with fixed-priority scheduling to consider the complexity of determining whether a set of periodic real-time tasks can be scheduled on $m > 1$. [12] proposed fixed-priority scheduling using a fixed-relative deadline. After a certain period of time, a task became suspended upon failure and the resource became available. [11] presents a scheduling algorithm for emergency medical rescue conflict monitoring and dispatch scheduling based on the hybrid estimation and intent inference. [30] took a heuristic approach for solving the rescue unit assignment and scheduling problem under the resource constraints. In [31], the authors discuss the utilization of the public resources for disaster rescue with the priority based scheduling policy. The authors present a discussion about the fairness and importance of priority based on rescue scheduling. However, there is no formulation to determine the priority scores of rescue scheduling tasks. In this paper, we formally define a method to determine the priority score of rescue tasks and propose a multi-task hybrid scheduling policy using priority, based on certain criteria to develop an effective and efficient rescue scheduling algorithm.

3 TWEETS CLASSIFICATION AND INFORMATION RETRIEVAL

Twitter data from two different natural disasters (Hurricane Harvey and Hurricane Irma) were collected for this work. We collected these tweets from August 26 to August 31, 2017 and September 10 to September 17, 2017, respectively. We use Twitter Stream API to collect the tweets along with various meta-information such as user information, geo-location, tags, entities, etc. The pre-processing step involves discarding non-English tweets, filtering noises and duplicates, removing special characters, stop-words, and jargons.

According to the FEMA, WHO, and NCDP, the "vulnerable populations" or "at-risk individuals" includes children, senior citizens, pregnant women, disabled, sick or injured persons. A tweet classifier is developed using the neural network to identify whether a tweet falls into one or more classes from six different classes (Rescue needed, DECW, Water needed, Injured, Sick, Flood). DECW stands for Disabled, Elderly, Children and Women. Those six classes help in determining the rescue situation, and their priorities along with the resources needed or requested in a tweet. We use the label Water_needed as a request for drinkable water identified as a vital resource during any disaster or emergency by the Centers for Disease Control and Prevention (CDC).

3.1 Deep Neural Network

The proposed deep learning model comprises 7 primary components. Figure 2 depicts the fundamental system architecture of the model.

Input layer: Pre-processed tweets fed to the input layer which is connected with the embedding layer.

Embedding layer: This layer encodes the input into real-valued vectors using lookup tables. In this work, we used a pretrained word vectors named Crisis [13] and GloVe [25] which generates a feature word vectors using co-occurrences based statistical model. Embedding applied to the words aids to map all tokenized words in every tweet to their respective word vector tables. To unify the feature vector matrix, appropriate padding is added.

BLSTM layer: The Long-Short Term Memory (LSTM) is a specialized version of Recurrent Neural Network (RNN) that is capable of learning long term dependencies. While LSTM can only see and learn from past input data, Bidirectional LSTM (BLSTM) runs input in both forward and backward direction. This bidirectional feature of BLSTM is critical for the various applications involved with understanding complex language [28].

The input gate i_t, forget gate f_t, output gate o_t, and cell state activation c_t of the implemented LSTM version in this work can be defined by the equations (1)-(5) where σ represents the logistic sigmoid function, h represents the respective hidden vectors, and W is the weight matrix. A detailed explanation of each equation and more about LSTM are available on [9].

$$i_t = \sigma \left(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i \right) \quad (1)$$

$$f_t = \sigma \left(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f \right) \quad (2)$$

$$o_t = \sigma \left(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_t + b_o \right) \quad (3)$$

$$c_t = f_t c_{t-1} + i_t \tanh \left(W_{xc}x_t + W_{hc}h_{t-1} + b_c \right) \quad (4)$$

$$h_t = o_t \tanh \left(c_t \right) \quad (5)$$

Attention layer: Every word in a sentence does not contribute equally to represent the semantic meaning and the primary concept of attention [19] originated from this observation. We use a word-level deterministic, differentiable attention mechanism to identify the words with the closer semantic relationship in a tweet. Equation 6 represents the attention score ei, e of each word t in a sentence i, where g is an activation function. More information on the attention mechanism is available on [15].

$$e_{i,j} = g \left(W h_t c \right) \quad (6)$$

Auxiliary features input: A tweet can only contain 280 characters (previously 140) which forces a user to express emotions in a different way compared to a traditional English sentence. People use extra punctuations and emoticons to intensify the meaning of a tweet. We also observed (e.g. Figure 1) greater use of numeric characters in a rescue seeking tweet due to the fact that people try to share location in the tweets In this work, we perform feature engineering to obtain a set of specific auxiliary features that can assist the classification model to learn better. A list of extracted auxiliary features that shows noticeable influence during the model evaluation is given in Table 1. The well-known Natural Language Toolkit (NLTK) is used to extract those features.
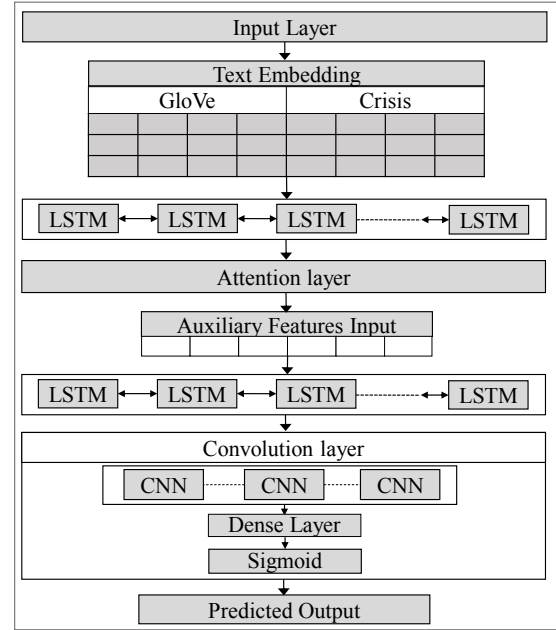

Figure 2: System architecture of the proposed model

Table 1: Auxiliary Features

polarity, subjectivity, sentiment, wordsVsLength, exclamationMarks, digitVsLength, punctuationVsLength, nounsVsWwords, sadVsWords, capitalsVsWords, uniqueWords, numberOfTags.

Convolution layer The convolution layer performs a matrix-vector operation in the sentence-level representation sequence. Let us assume that $H \in \mathbb{R}^{d*w}$ be the weight matrix, and the feature mapping done as $c \in \mathbb{R}^{l-w+1}$. The i-th element of the feature map can be defined as:

$$c_i = \sigma \left(\sum \left(C\left[*, i : i + w \right] oH \right) + b \right) \quad (7)$$

In sentence-level representation, $C[*, i : i + w]$ is the i-th to i+w-th column vector. The word vectors pass through the convolution layers [29] where all the input information merged together to produce a features map. The Rectified Linear Unit (ReLU) used as the activation to deal with the non-linearity in the convolution layer and generate a rectified feature map. Finally, the dense layers are activated for generating the outputs.

Output layer The activation function *sigmoid* is used in the dense layer as we want to perform multi headed binary classification. The model produces binary values for all six target output classes. Detailed information on model hyperparameters and evaluation results is given in section 5.1.

3.2 Location extraction

Due to the privacy policy of Twitter, most of the tweets do not contain any location information. In those cases, we try to extract location using user profile meta information and the location information provided in the tweeted text. Combining the Stanford

Named Entity Recognizer (NER) [6] and Google map API, an application is built for extracting location.

4 RESCUE SCHEDULING

4.1 Problem Specification

Let us assume that the number of rescue teams be m with n pending rescue tasks. Let the processing time of rescue task j by team i be t_{ij}, where $1 \leq i \leq m, 1 \leq j \leq n$. Based on a typical disaster situation, we consider that the number of rescue tasks is greater than or equal to the number of rescue teams ($n \geq m$). The problem is to organize and assign the tasks to rescue teams in such a way that the amount of waiting time for each rescue mission is minimized. However, due to the inconsistent nature of the rescue tasks and the location of the incidents, the formulation of this problem faces the following major challenges.

(1) Depending on the capabilities and resources every rescue team may not capable of processing each task.
(2) It is difficult to precisely estimate the required time t_{ij} for a task due to the uncertainty of the environment and location of an incident.
(3) Tasks might have different priorities based on the people needed to execute them and their physical condition. The environmental condition of a person such as surrounded by flood water, or fire should also be taken into account while determining priority.

Along with the above challenges, we also consider the following restrictions and conditions to formulate the problem effectively.

- We imposed a time t_j for a task j, where t_j denotes the required time for a rescue team i to move from initial rescue center to the place of incident. The time for moving from the location of a task j to another task j' is represented by $t_{jj'}$.
- Every team requires a preparation time before leaving for a scheduled rescue job from their respective rescue management station. The preparation time is denoted as t_i for every team for a specific task j. Also, after a certain period, every team might require a resting time of t_{ir} before the next task.

Considering the above sequence of times (t_{ij}, t_j, $t_{jj'}$, t_i, and t_{ir}), we can estimate a probable time for a rescue mission. Although the time can be changed based on the situation, we consider some constant time variable considering the distance of a task location and the probable situation of the environment around the incident.

4.2 Priority determination

A significant step for the rescue scheduling algorithm is determining the priority of rescue tasks. We use the output labels of tweet classifier (Section 3.1) and assign a weight for each label to determine the priority of that tweet. Assume the assigned weights for different labels of the tweets is represented by a vector $w_j = [w_1, w_2, ...w_n]$. A feature vector $\alpha_i = [\alpha_1, \alpha_2, ...\alpha_m]$ also used which denotes the weight of other considerable variables such as the number of victims, real-time environmental conditions and future weather forecasts of a specific location. Equation 8 represents the formula to estimate the priority for a rescue task. The base priority value of a tweet is 1 where the maximum priority score can be 10.

$$f_p = \sum_{i=1}^{m} \alpha_i + \sum_{j=1}^{n} w_j \qquad (8)$$

4.3 Rescue scheduling algorithms

General scheduling algorithms are not applicable in disaster rescue scenario as those algorithms might be unfair due to different situations, physical conditions, and the critical importance of human life. A priority-based scheduling algorithm might provide a better solution where we need to consider and determine the priority continuously. In a disaster scenario, priorities can change with time and environmental conditions. Hence, We develop an effective rescue scheduling algorithm considering priority, environmental severity, and processing time of every single task. We like to define the terms which we use to represent our algorithms.

- Tasks: A task is the combination of one or more valid rescue requests by an individual or multiple people. A list of valid requests forms a sequence of tasks which demands to be scheduled appropriately.
- Processors: The number of rescue units which can complete a given task is the processors. A processor is responsible to execute a given task, release the resources upon completion, and get back to the initial state to execute a new task.
- Arrival Time: The time of receiving a valid rescue request represents the arrival time for a specific task. In our rescue scheduling system, arrival is the time-stamp of a tweet.
- Burst Time: The probable time required to complete a task by a processor can be defined as burst time. The burst time is realistically represents the service time of a processor for a rescue mission. In a disaster scenario, estimating appropriate burst time is very challenging. Similar tasks might take different times to complete under separate circumstances. To address this issue, first, we assume a probable burst time based on the rescue operations in previous disasters. After the completion of a few rescue missions, the burst time of the future mission is determined using the actual completion time of those missions. To predict the future burst time, we use the exponential averaging method. Given n tasks (taskSeq[1...n]) and burst time for tasks t_i, the predicted burst time for the next task $taskSeq_n + 1$ will be:

$$BT_{n+1} = \alpha T_n + (1 - \alpha)BT_n \qquad (9)$$

In the above equation, α is a constant factor ranging ($0 <= \alpha <= 1$). The value that can predict the best possible burst time will be assigned as α. The variable BT_n denotes the predicted or assumed burst time for the task n, and T_n represents the actual burst time needed for completing task n.

Three different scheduling algorithms are implemented for the experiments. All of those algorithms are implemented using multiple processors as it is expected to have more than one rescue unit in an emergency rescue situation. Although we emphasize on Multi-task Hybrid Scheduling algorithm, however, we study fundamental rescue algorithms to understand the limitations of these established methods. This study also indicates the necessity of a novel adaptive Hybrid Scheduling algorithm for a disaster scenario.

4.3.1 First-Come-First-Serve (FCFS). In FCFS scheduling system, the task requests are sequentially processed in the order of the arrival time. A sequence of tasks list (taskSeq) with the requests arrival time (arrivalTime) and probable burst time (burstTime) is fed to the algorithm as input. The algorithm returns the scheduled tasks sequence with the possible start time. However, estimate burst time can change and needed to update while the processor is processing a task. While FCFS is a simplest scheduling algorithm, it has two major concerns which need some attention.

- In a disaster scenario, every rescue request is not similarly critical. FCFS fails to consider the tasks which have an urgency of completion.
- FCFS is a non-preemptive scheduling algorithm which is responsible for the short jobs to wait longer based on the sequence order.

4.3.2 Priority Scheduling. In a disaster scenario, conducting rescue missions based on priority can be crucial. There can be rescue requests which can wait longer, and might not be critical like other requests. A priority-based scheduling algorithm is more appropriate considering those facts. The algorithm executes the task using an ordered queue with high to low priority. A priority queue based scheduling algorithm is demonstrated in the Algorithm 1.

4.3.3 Multi-Task Hybrid Scheduling. The incidents at the end of the priority queue need to wait longer when there is a large scale disaster because of plenty of rescue requests. Assume there are some tasks which need to wait longer for rescue due to lower priority. Suppose some of those tasks are located in an area where the disaster situation is worsening by time. The severity can increase fast at those places. A priority balancing scheduling policy might be helpful in such a scenario. It may need more information and human input to decide how and when to increase the priority of a task before it enters into critical condition. To solve this dilemma, we introduce a priority balancing module which re-calculate the priority score after the completion of each rescue mission.

Instead of a single rescue task in a mission, a rescue team can execute multiple tasks depending on available resources. For example, in a flood situation, several individuals can be rescued in the same boat and transferred to a shelter together. We illustrate this idea along with priority balancing in Algorithm 2. A processor can be assigned for multiple tasks in a single rescue mission if it has available resources. We use a 2 miles radius area for this purpose. A processor looks for other available tasks which are within 2 miles radius of the assigned event. It will incorporate multiple tasks as long as the processor has adequate resources and executes those tasks sequentially using priorities. Comparative performance evaluation of the algorithms is present in Section 5.2. In Section 5.3 we describe and demonstrate the Multi-Task Hybrid Scheduling algorithm using a real-world disaster scenario.

5 EXPERIMENTAL RESULTS

5.1 Tweets Classifier Evaluation

The primary goal of the tweet classification is to identify the people who need help and determine a priority score for each tweet based on the classified labels. To accomplish this goal, 4900 tweets were manually labeled into six different binary classes from 68,574

Algorithm 1 Priority scheduling with multi-processors

Input: processorNo, taskSeq[1...n], arrivalTime[$at_1...at_n$], burstTime[$bt_1....bt_n$], tasksPriority[1...n];

Output: scheduleSeq[$task_i...task_n$], startTime[$st_1...st_n$], turnAroundTime, avgWaitingTime, avgTurnAroundTime;

    Initialization: All the processors K are released and ready to begin a task.

    Initialize, scheduleSeq, startTime, and turnAroundTime as list; currTime = 0, waitingTime = 0, totalTurnAroundTime = 0;

    Sort the taskSeq, arrivalTime, burstTime using taskPriority and assign the tasks in priority queue P_{queue};

1: **if** (new task request) **then**
2:     update P_{queue}, taskSeq, arrivalTime, burstTime, number of tasks n;
3: **end if**
4: **for** i = 1 to n **do**
5:     select task i to be processed;
6:     dequeue the root element from P_{queue}
7:     scheduleSeq.append(i);
8:     K^* are the available processors to process task i;
9:     **if** ($K^* \neq \emptyset$) **then**
10:      assign current task to K;
11:      **if** (currTime<arrivalTime[i]) **then**
12:       currTime = arrivalTime[i];
13:      **end if**
14:      startTime.append(currTime);
15:      waitingTime = waitingTime + (currTime-arrivalTime[i]);
16:      completionTime = currTime + burstTime[i];
17:      currentTrunAroundTime = completionTime - arrivalTime[i];
18:      totalTurnAroundTime = totalTurnAroundTime + currentTrunAroundTime;
19:      turnAroundTime.append(currentTrunAroundTime);
20:      release K;
21:     **else**
22:      return to if
23:     **end if**
24: **end for**
25: calculate avgWaitingTime, avgTurnAroundTime;

---

preprocessed tweets on Hurricane Harvey and Irma. We evaluate the proposed classification model on this labeled dataset and compared it with the well-established Logistic Regression (LR), Support Vector Machine (SVM) and fundamental CNN model. Moreover, in order to fully understand the effectiveness of our approach, we evaluate our model on several past disaster datasets obtained from CrisisNLP [13] and CrisisLex [23]. We use the same datasets and data settings of Nguyen et al.[20] and compare the output of our proposed model with the stated results of LR, SVM, and CNN in the same paper. Further, to evaluate the robustness of our proposed technique, we merged 15 different disasters data from CrisisLex [23] and perform a binary classification which identifies the tweets relevant to a particular disaster.

5.1.1 Model Parameters. A set of optimal parameters is crucial to achieve desired performance results. We perform rigorous parameter tuning and select an optimal set that is used in all the

Algorithm 2 Multi-tasks Hybrid Scheduling

Input: processorNo,   taskSeq[1...n],   tasksPriority[1...n],
arrivalTime[$at_1...at_n$],   burstTime[$bt_1....bt_n$],   tasksloca-
tion[1...n], disRadius;

Output: scheduleSeq[$task_i...task_n$], startTime[$st_1...st_n$],
turnAroundTime, avgWaitingTime, avgTurnAroundTime;

Initialization: All the processors K are released and ready to
begin a task.

Initialize, scheduleSeq, startTime, and turnAroundTime as; cur-
rTime = 0, waitingTime = 0, totalTurnAroundTime = 0;

Sort the variables in descending order using taskPriority. Re-
sort the values in ascending order using burstTime and arrival-
Time for same taskPriority tasks. Assign the tasks in priority
queue P_{queue};

1: **if** (new task request) **then**
2:    update P_{queue}, and resort taskSeq, arrivalTime, burstTime,
      number of tasks n;
3: **end if**
4: **for** i = 1 to n **do**
5:    select task i to be processed;
6:    dequeue the root element from P_{queue}
7:    scheduleSeq.append(i);
8:    K^* are the available processors to process task i;
9:    **if** ($K^* \neq \emptyset$ and available K is capable of addressing task i)
      then
10:       assign current task to K;
11:       **for** $m = i + 1$ to n **do**
12:          calculate the distance d of taskSeq[m] from current task
             using tasksLocation[m];
13:          **if** (d<disRadius and K has the extra resources to com-
             plete taskSeq[m] after current task) **then**
14:             add taskSeq[m] with the current task queue and cre-
                ate a sub-scheduling for those tasks;
15:             dequeue the taskSeq[m] and update P_{queue};
16:          **end if**
17:       **end for**
18:       estimate startTime, waitingTime, totalTurnAroundTime
          following the similar process of algorithm 2.
19:       release K;
20:    **else**
21:       return to if
22:    **end if**
23: **end for**
24: calculate avgWaitingTime, avgTurnAroundTime;

---

experiments. We use the same parameter for better evaluation and
model reproducibility. The popular evaluation metrics such as pre-
cision, recall, F1-score, accuracy , and AUC score is used to validate
and compare the experimental result of the models.

*5.1.2 **Evaluation on Hurricane Harvey and Hurricane Irma
data**.* We use 4900 manually labeled tweets for this evaluation
where 3920 tweets (80%) used for training and the rest of the 20%
tweets used for testing. In the evaluation tables, we denote our
model as CNN_{AAf}, which stands for CNN with Attention and
Auxiliary features. We compare our model (CNN_{AAf}) with LR,
SVM, and CNN without attention and auxiliary features. Our model

Table 2: Hyperparameter values

Hyperparameter	Value/Description
Text embedding	Dimension: 300
BLSTM Layer	2 layers; 300 hidden units in each (Forward and Backward)
Conv1D Layer	3 layers; 300 convolution filters
Dense Layer	3 layers; First 2 layers have 150 and 75 units respectively and the last one is output (Dense)
Drop-out rate	Word Embedding: 0.3; Dense layer: 0.2 each;
Activation func-tion	Conv1D, BLSTM, Dense: ReLU; Output Dense layer: Sigmoid;
Adam optimizer	Learning rate = 0.0001; $beta_1$=0.9;
Epochs and batch	Epochs = 10 to 25; batch size = 128;

Table 3: Classifier evaluation (Hurricane Harvery and Irma)

Model	Precision	Recall	F1-score	Accuracy
LR	55.8	93.0	69.7	84.5
SVM	65.1	85.4	73.9	88.5
CNN	61.6	90.8	73.4	87.5
CNN_{AAf}	81.7	93.4	87.2	93.7

Table 4: Evaluation metrics for individual classes

Class	Precision	Recall	F1-score	Accuracy
Help	87.9	97.7	91.2	94.9
Flood	78.2	94.1	85.3	91.3
Water Needed	87.5	71.4	78.7	98.0
DCEW	93.7	73.2	82.3	98.5
Weighted Avg	81.7	93.4	87.2	93.7

outperformed all other models by more than 5% inaccuracy metrics.
In terms of precision, the proposed model performed surprisingly
well and outperformed the closed result of SVM by around 25%.

Table 3 represents the full evaluation results for the different
classifiers. Table 4 represents the evaluation metrics for individual
classes (Hurricane Harvey and Irma) using CNN_{AAf} model. The
distributions of the six classes in the data are Help - 29.1%, Flood -
26.3%, Water Needed - 4.9%, DCEW - 4.1%, Injured - 0.3%, Sick - 0.3%.
However, we discarded labels Injured and Sick due to lack of enough
data instances for training and testing so that it cannot influence the
metrics of the model. As there are few true positive instances, those
two labels achieve a higher rate of Accuracy although the model is
not identifying true positive instances. We can also observe a better
precision and accuracy for labels Water Needed and DCEW. This is
happening as there are also a few true positive instances. However,
still, the model has performed well for the recall and F1-score as the
words found in the tweets for those labels have fewer variations.

*5.1.3 **Evaluation on CrisisNLP and CrisisLex datasets**.* We
use the same datasets and class distributions consisting of Nepal
Earthquake, California Earthquake, Typhoon Hagupit, and Cyclone
PAM which is described in [13]. In that paper, the authors evaluate
the models on event data, out-of-event data and a combination

Table 5: Classifier evaluation AUC scores (CrisisNLP)

Disaster Name	LR	SVM	CNN_I	CNN_{AAf}
Nepal Earthquake	82.6	83.6	84.8	87.5
California Earthquake	75.5	74.7	78.3	83.6
Typhoon Hagupit	75.9	77.64	85.8	88.3
Cyclone PAM	90.6	90.74	92.6	92.6

of both datasets. In table 5, we represent the results on the combination of both datasets. Clearly, our proposed CNN_{AAf} model outperformed all other models in term of AUC score which the authors also used in the referenced paper. Auxiliary features have a high impact to better understand the semantic meaning of the tweets which is reflected on the AUC score.

We consider 15 different natural disaster datasets from CrisisLex [23]. After removing null values and preprocessing the merged datasets contains 13738 data instances. We use around 75% data for training (9268) and validation (1030) and 25% data for testing (3440). The comparative evaluation result using sklearn metrics is presented in table 6. It is observable that the domain-specific auxiliary features along with attention layer is highly beneficial for understanding and identifying crisis tweets. Our proposed approach can be used on a diverse set of datasets with good outcome and this might play a crucial role to develop quick response application on the disaster domain. More information on the 15 datasets is available in the extended version of our paper [14].

Table 6: Classifier evaluation (CrisisLex)

Model	Precision	Recall	F1-score	Accuracy
LR	85.8	71.1	77.8	85.8
SVM	90.9	74.7	82.1	73.2
CNN	93.4	76.3	84.2	76.4
CNN_{AAf}	93.6	93.7	93.4	93.6

5.2 Computational experiment on scheduling algorithms

A computational experiment has been performed on the proposed algorithm in Section 4. For the purpose of evaluation and comparison, a data-set consisting of hurricane Harvey tweets between 27th August 2017 and 31st August 2017 have been used. To identify the rescue seeking tweets, the proposed tweet classification model is used. We processed the identified tweets to extract and determine the required information for the scheduling algorithm such as location, possible service time (burst time), and priority using the described process in Section 3. The priority of each tweet was determined on a scale of 10 using four classes (Flood, Water Needed, DCEW, and Sick or Injured), labeled by the classifier following Equation 8. The weights for those classes were assigned as 1.5, 1.5, 2 and 2.5, respectively. For the environmental feature vector, we use a random distribution between 0.5 to 2.5. However, automatic weight determination still remains an open problem for the research. Next, the probable service time was estimated for each of the rescue tasks. We use the normal distribution of average service time as 54 minutes which is described in [31]. Finally, after all the processing, 174 rescue seeking tweets were found from
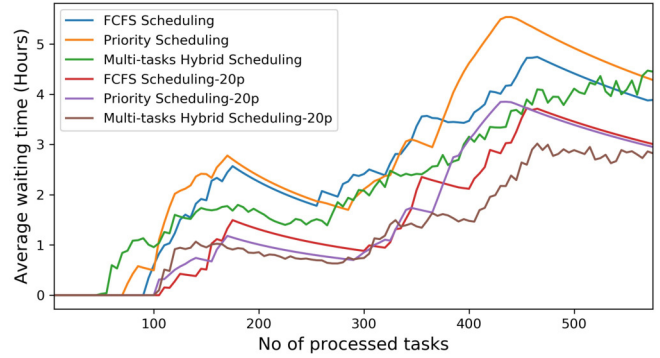
Figure 3: Average waiting time using 10 and 20 processors

around 72 hours data frame. This sample size is relatively small and distributed over a longer period. Hence, we performed upsampling using resample and linear interpolation methods from python pandas library and created a dataset containing 550 rescue tasks to evaluate the rescue algorithms.

The algorithms were implemented using the multiprocessing system. We use the number of rescue units (processors) as 10 and 20 to evaluate the performance of the scheduling algorithms. In Multi-task hybrid scheduling algorithm, the traveling time from one rescue location to another also considered while combining multiple tasks. Eventually, this estimation reduces the processing time for those tasks. Table 7 describes the summary of the three algorithms. In the table, 10p and 20p represent the number of processors used to execute those algorithms. The average waiting times are lowest in case of Multi-task hybrid scheduling algorithm. The average waiting time (hours) with the number of processed tasks is represented in Figure 3. The experimental results can be summarized as follows.

Table 7: Average waiting time summary

Algorithms	Max avg WT		Mean avg WT	
	10p	20p	10p	20p
FCFS	4.74	3.73	2.53	1.61
Priority	5.54	3.85	2.81	1.63
Multi-tasks Hybrid	4.47	3.02	2.24	1.31

- FCFS scheduling algorithm performs better comparing to Priority scheduling algorithm. However, in a disaster scenario, FCFS is not a fair policy to distribute the resources and rescue mission. Priority scheduling has a longer average waiting time because the lower priority tasks are waiting longer in the queue.
- Multi-tasks hybrid scheduling beats all other algorithm with respect to average waiting time. This algorithm is more practical for effective rescue scheduling and resource allocation as it consider resource constraints. It allows completing small tasks together of a nearest distance. Furthermore, it can be utilized to transfer the required resources (such as water, medicine) to the different locations while optimizing the average waiting time. However, the maximum average waiting time for this algorithm can be high for a task with less

priority and larger processing time. It can happen when the location of a mission is far away with a low priority score.

5.3 Experimental analysis on real-world disaster scenario

A sample data-set is processed from the tweets during Hurricane Harvey to demonstrate Multi-task Hybrid Scheduling algorithm. An area of 20 square miles radius at Port Arthur, Texas has been selected for performing rescue operations. Figure 4 represents the geographical locations of the victims (Red icons) and the hyphothetical rescue operation base (Home icon) in the Port Arthur, Texas during hurricane Harvey. The ArcGIS javascript API [3] is used to create Figure 4 and 5. To demonstrate the algorithm, we assume that there is a rescue operation base at Tyrrell Elementary School, Port Arthur, TX. The experimental process can be summarized by the following steps:

- First, We have selected the rescue seeking tweets and extracted the location using the Stanford Named Entity Recognizer (NER) [6] and Google map API.
- Second, we extracted 10 tweets which were arrived first and located around 20 miles radius of the rescue operation base after 12pm of 30th August 2017.
- Third, the priority score, probable burst time and distance metrics have been calculated for each of the 10 rescue tasks.
- Finally, the Multi-task Hybrid Scheduling algorithm created the rescue schedule. We have simulated the experiment using 2 and 4 rescue units and two different distributions of the possible burst time. First, we assumed the required burst time to be 54 minutes for each task based on the paper on hurricane Harvey rescue by Yang et al. [31]. Further, we use a random completion time for the first 5 tasks and predict the burst time of future rescue missions using equation 9.
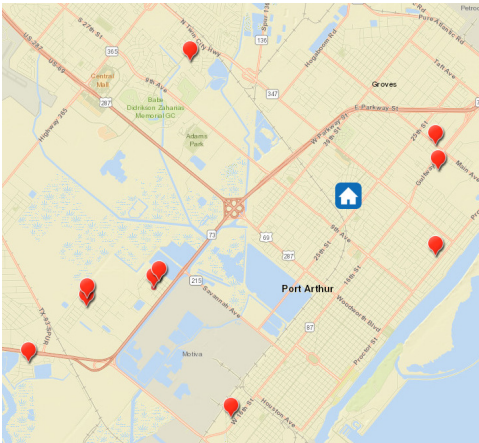

Figure 4: The positions of the victims and operation base.

Tables 8 and 9 represent example data sample of tweet labels and environmental features for priority calculation using Equation 8. We have used demo weights for the labels and environmental features as (Flood - 1.5, Water Needed - 1.5, DCEW - 2, Sick or Injured - 2.5, Storm - 1, Road Damaged - 1, forecasted storm - 0.5,

Table 8: Classified tweet labels for priority determination

id	Flood	Water Needed	DCEW	Sick or Injured
1	1	1	0	1
2	1	0	0	0
3	0	1	1	0
4	1	0	0	1
5	0	0	0	0

Table 9: Environmental features example

id	Current		Forecasted	
	Storm	Road Damaged	Storm	Flood
1	0	1	1	0
2	0	0	1	0
3	0	1	0	1
4	0	1	0	0
5	1	0	0	0

forecasted flood - 0.5) respectively. We used experimental weights because determining the weights for those labels and features requires domain expert and extensive study. An appropriate authority or domain expert will be able to input precise weight values for the labels and environmental features considering the situation during an actual disaster. In the tables, *id* represent the respective tweet which is later refers to the same numbered *taskSeq* in Table 10. The calculated priorities also presented in Table 10 as Priority Score.

Table 10 represents some columns of the processed sample data set of Port Arthur for the rescue scheduling. In the table, the burstTime is represented in minutes and distanceFromBase is measured in miles. To use Multi-task Hybrid Scheduling algorithm on the data, we need to assume some parameters. We consider the starting time of rescue mission as 14:00, the speed of the used vehicles or boats to rescue is 20MPH, and after the completion of each rescue mission a rescue unit requires 30 minutes as a preparation time before next task.

Table 10: Real-world data sample for simulation

taskSeq	Arrival Time	Burst Time	Priority Score	Distance from Base
1	12:13	54	7	5.1
2	12:45	54	2	5.0
3	12:58	54	5	6.9
4	14:07	54	5	7.0
5	14:46	54	1	3.9
6	15:23	75	2	4.5
7	16:10	70	8	1.9
8	16:52	30	7	7.7
9	17:30	35	5	1.8
10	18:05	45	6	2.0

The Multi-task Hybrid Scheduling algorithm can be demonstrated on the data in the table 10 as follows. We use 2 rescue units to illustrate the algorithm.

(1) The Start time of the rescue operation is 14:00. So, there will be 3 tasks in the queue at the time of the first iteration. The algorithm will first sort the tasks based on the priority score. Hence, the sorted sequence will be 1 >= 3 >= 2.

(2) The location of the highest priority task (taskSeq 1) will be the point of interest. The algorithm will consider a perimeter of 2 square miles of that point and check if any other rescue task is there which can be combined. We can observe that taskSeq 1,2, and 3 are within 2 miles radius. If a rescue unit contains enough resource for running those 3 operations sequentially, it will combine those tasks and rescue the people in a single go without coming back and forth to the base.

(3) The algorithm will further create a sub-schedule of 3 tasks assigned to rescue unit 1. Task 1 has the highest priority and hence, the rescue unit will first go to location 1. From Figure 5, we can observe that tasks 1 and 3 are in a close distance. However, task 2 has a higher priority. As the algorithm emphasizes the priority score most, it will schedule task 2 before task 3. The rescue unit will assist the people in location 2 and then come back to location 3. Finally, it will come back to the base after the completion of all 3 tasks.

(4) If there are multiple tasks with the high priority ($priority >= 7$), separate rescue unit will be assigned despite of there occurrence in a close proximity. In our experimental setup if two tasks with high priority are within 2 miles radius, the algorithm assigns two separate units for those two tasks. However, if there is only one rescue unit available, the algorithm will follow the above approach. Multiple tasks with the same priority will be sorted based on burst time and arrival time, respectively. Multiple tasks with same priority and burst time will be sorted using arrival time.

(5) Based on the conditions, taskSeq 1, 2 and 3 will be assigned to rescue unit 1. Rescue Unit 2 will take care of taskSeq 4 which arrives at 14:07. The algorithm will wait until the completion of a task, after which a rescue unit became available.

(6) The taskSeq 4 will complete first and rescue unit 2 will become available around 16:13. The algorithm will iterate again and sort the remaining tasks. At this point, the queue contains 3 tasks (taskSeq 5,6 and 7).

(7) Employing the conditions, the sorted order for the tasks will be 7 >= 6 >= 5. The taskSeq 7 has a high priority and there are no other victims nearby. Hence, rescue unit 2 will be assigned to complete task 7.

(8) Rescue unit 1 will be available again at 17:55. The algorithm will continue iterating until all of the 10 tasks are completed.

Table 11 represents some output values and rescue schedule for the data illustrated in Table 10. The column *StartTime* represents the scheduled time for the respective task. *RouteDistance* denotes the actual one-way path that a rescue unit needs to travel for a particular rescue mission. When multiple tasks are group together for a single mission the *RouteDistance* became the path between previous task and current task. For example, in Table 10, the distance of rescue location of taskSeq 3 from base is 6.9 miles. However, as tasks 1,2 and 3 grouped together the distance between the previous task 1 and task 3 became 2 miles. *RouteDuration* is the rounded time in minutes to travel the specific *RouteDistance*. In our experiment,

Table 11: Rescue scheduling output table of Multi-tasks Hybrid Scheduling algorithm using 2 rescue units

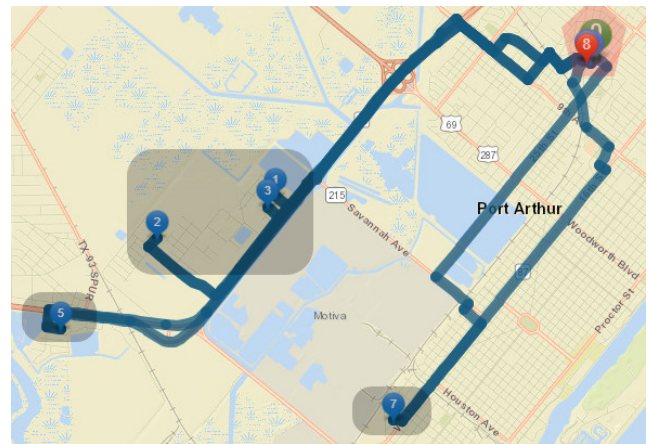| taskSeq	Start Time	Route Distance	Route Duration	Waiting Time	TAround Time	Unit
1	14:00	5.1	15	122	176	1
3	15:09	2.0	06	137	191	1
2	16:09	2.2	07	211	265	1
4	14:07	7.0	21	21	75	2
7	16:13	1.9	06	09	79	2
8	17:55	7.7	23	86	116	1
10	18:05	2.0	06	06	51	2
9	19:32	1.8	06	128	163	2
6	19:41	4.5	14	272	347	1
5	20:49	3.9	12	375	429	2

Figure 5: Route of Rescue Unit 1 by rescue order

we assume that a rescue unit needs 3 minutes to travel a mile. *WaitingTime* is the subtraction of *StartTime* and *ArrivalTime* with the addition of required travel time (*RouteDuration*) for a rescue location. The turnaround time is represented by *TAroundTime* in the table which is the summation of *WaitingTime* and *BurstTime*. The rightmost column in the table represents the assigned rescue unit for a task. After returning from a rescue mission to the base, a rescue unit requires a preparation time to become available for the next mission. In the experiment above, the rescue unit 1 reached at the base at 17:25 after completing the first rescue mission of task 1,2 and 3. It became available at 17:55 after necessary preparations.

The routes of the rescue missions assigned to rescue unit 1 presented in table 11 are illustrated in Figure 5. The red pentagon shadow area denotes the rescue operation base. The black shadowed rectangular shapes represent the rescue mission. Location points 1, 2, 3, 5 and 7 denote the taskSeq 1,3, 2, 8 and 6, respectively. Pointers 1,2, and 3 are inscribed in the same box as those tasks were combined together and performed in a single mission. The rescue unit 1 will start from the base (0) and travel to point 1, 2 and 3 to rescue victims and complete the tasks 1,3 and 2 in the first rescue mission. It will return back to base which is denoted by blue pointer (4) below the red pointer indicating 8. The unit will again

travel to location 5, return to the base (6) and complete the taskSeq 8. Finally, the location of the third rescue mission pointed by 7 and the missions will be completed by rescue unit 1 after reaching to the base (point 8).

We have also conducted the same experiment with 4 rescue units. The average waiting time and turnaround time reduced dramatically in this scenario. In the first experiment with 2 rescue units, the average waiting time and turnaround time is around 137 minutes and 189 minutes respectively. With 4 rescue units, waiting time and turnaround time came down to 49 minutes and 102 minutes. With the low number of rescue units, the tasks with low priority need to wait longer which increase the average waiting time. From table 10 and table 11, we can observe that taskSeq 5 arrived at 14:46 with a priority score of 1. Due to the very low priority, task 5 scheduled last at time 20:49 with a waiting time of 375 minutes. However, tasks with higher priority such as 1, 7 and 8 had to wait a fairly lower amount of time.

6 CONCLUSION AND FUTURE WORK

In this paper, we utilized social media (Twitter) for disaster management applications such as categorizing, identifying, and prioritizing users who need help and developed an algorithm for rescue scheduling. We introduced a novel approach for an effective rescue scheduling algorithm. First, we developed a tweet classifier using deep learning with attention layer and auxiliary features. The classifier labels every tweet into six different classes. Those labels allow us to identify the necessary information to assist the person/people in the tweet and estimate a priority score for that task. Second, we developed a multi-task hybrid scheduling algorithm and conducted the experiments using real disasters data for evaluating the efficiency of the algorithm. In the future, we would like to work on precise location determination and optimal estimation of the required time for a rescue mission. In addition, we are developing a fully-featured web application for deploying on the real-time disaster to evaluate the effectiveness of our work in disaster management.

ACKNOWLEDGMENTS

REFERENCES

[1] David E Alexander. 2014. Social media in disaster risk reduction and crisis management. *Science and engineering ethics* 20, 3 (2014), 717–733.
[2] Mehdi Allahyari, Seyedamin Pouriyeh, Mehdi Assefi, Saied Safaei, Elizabeth D Trippe, Juan B Gutierrez, and Krys Kochut. 2017. A brief survey of text mining: Classification, clustering and extraction techniques. *arXiv preprint arXiv:1707.02919* (2017).
[3] ArcGIS. 2019. *ArcGIS for Developers.* Retrieved March 7, 2019 from https://developers.arcgis.com/javascript/
[4] Drake Baer. 2012. As Sandy became# Sandy, emergency services got social. *Fast Company* 9 (2012).
[5] Alexis Conneau, Holger Schwenk, Loïc Barrault, and Yann Lecun. 2016. Very deep convolutional networks for text classification. *arXiv preprint arXiv:1606.01781* (2016).
[6] Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd annual meeting on association for computational linguistics.* Association for Computational Linguistics, 363–370.
[7] Huiji Gao, Geoffrey Barbier, Rebecca Goolsby, and Daniel Zeng. 2011. *Harnessing the crowdsourcing power of social media for disaster relief.* Technical Report. Arizona State Univ Tempe.
[8] Kiran Garimella, Gianmarco De Francisci Morales, Aristides Gionis, and Michael Mathioudakis. 2018. Quantifying controversy on social media. *ACM Transactions on Social Computing* 1, 1 (2018), 3.
[9] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. 2013. Speech recognition with deep recurrent neural networks. In *2013 IEEE international conference on acoustics, speech and signal processing.* IEEE, 6645–6649.
[10] Eui-Hong Sam Han, George Karypis, and Vipin Kumar. 2001. Text categorization using weight adjusted k-nearest neighbor classification. In *Pacific-asia conference on knowledge discovery and data mining.* Springer, 53–65.
[11] Bin Hu, Fang Pan, and Lei Wang. 2019. A scheduling algorithm for medical emergency rescue aircraft trajectory based on hybrid estimation and intent inference. *Journal of Combinatorial Optimization* 37, 1 (2019), 40–61.
[12] Wen-Hung Huang and Jian-Jia Chen. 2016. Self-suspension real-time tasks under fixed-relative-deadline fixed-priority scheduling. In *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2016.* IEEE, 1078–1083.
[13] Muhammad Imran, Prasenjit Mitra, and Carlos Castillo. 2016. Twitter as a lifeline: Human-annotated twitter corpora for NLP of crisis-related messages. *arXiv preprint arXiv:1605.05894* (2016).
[14] Md Kabir, Sanjay Madria, et al. 2019. A Deep Learning Approach for Tweet Classification and Rescue Scheduling for Effective Disaster Management. *arXiv preprint arXiv:1908.01456* (2019).
[15] Akshi Kumar, Saurabh Raj Sangwan, Anshika Arora, Anand Nayyar, Mohamed Abdel-Basset, et al. 2019. Sarcasm Detection Using Soft Attention-Based Bidirectional Long Short-Term Memory Model With Convolution Network. *IEEE Access* 7 (2019), 23319–23328.
[16] Mirella Lapata, Phil Blunsom, and Alexander Koller. 2017. Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, Vol. 1.
[17] Joseph Y-T Leung and Jennifer Whitehead. 1982. On the complexity of fixed-priority scheduling of periodic, real-time tasks. *Performance evaluation* 2, 4 (1982), 237–250.
[18] Bruce R Lindsay. 2011. Social media and disasters: Current uses, future options, and policy considerations.
[19] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025* (2015).
[20] Dat Tien Nguyen, Kamela Ali Al Mannai, Shafiq Joty, Hassan Sajjad, Muhammad Imran, and Prasenjit Mitra. 2016. Rapid classification of crisis-related data on social networks using convolutional neural networks. *arXiv preprint arXiv:1608.03902* (2016).
[21] Long Nguyen, Zhou Yang, Jiazhen Zhu, Jia Li, and Fang Jin. 2018. Coordinating Disaster Emergency Response with Heuristic Reinforcement Learning. *arXiv preprint arXiv:1811.05010* (2018).
[22] Ory Okolloh. 2009. Ushahidi, or "testimony": Web 2.0 tools for crowdsourcing crisis information. *Participatory learning and action* 59, 1 (2009), 65–70.
[23] Alexandra Olteanu, Sarah Vieweg, and Carlos Castillo. 2015. What to expect when the unexpected happens: Social media communications across crises. In *Proceedings of the 18th ACM conference on computer supported cooperative work & social computing.* ACM, 994–1009.
[24] Mina Park, Yao Sun, and Margaret L McLaughlin. 2017. Social media propagation of content promoting risky health behavior. *Cyberpsychology, Behavior, and Social Networking* 20, 5 (2017), 278–285.
[25] Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP).* 1532–1543.
[26] Irina Shklovski, Moira Burke, Sara Kiesler, and Robert Kraut. 2010. Technology adoption and use in the aftermath of Hurricane Katrina in New Orleans. *american Behavioral scientist* 53, 8 (2010), 1228–1246.
[27] Kate Starbird and Leysia Palen. 2010. *Pass it on?: Retweeting in mass emergency.* International Community on Information Systems for Crisis Response and âĂę.
[28] Shuohang Wang and Jing Jiang. 2015. Learning natural language inference with LSTM. *arXiv preprint arXiv:1512.08849* (2015).
[29] Xingyou Wang, Weijie Jiang, and Zhiyong Luo. 2016. Combination of convolutional and recurrent neural network for sentiment analysis of short texts. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers.* 2428–2437.
[30] Felix Wex, Guido Schryen, Stefan Feuerriegel, and Dirk Neumann. 2014. Emergency response in natural disaster management: Allocation and scheduling of rescue units. *European Journal of Operational Research* 235, 3 (2014), 697–708.
[31] Zhou Yang, Long Hoang Nguyen, Joshua Stuve, Guofeng Cao, and Fang Jin. 2017. Harvey flooding rescue in social media. In *Big Data (Big Data), 2017 IEEE International Conference on.* IEEE, 2177–2185.
[32] Jie Yin, Sarvnaz Karimi, Andrew Lampert, Mark Cameron, Bella Robinson, and Robert Power. 2015. Using social media to enhance emergency situation awareness. In *Twenty-Fourth International Joint Conference on Artificial Intelligence.*