Project: Connected Emergency Response **Learning Team-Suite (CERLT-S)**

Subject Area: Connected Emergency Response Learning Tools (CERLT)

Authored on: 14/09/2021, 15/09/2021, 16/09/2021, 17/09/2021 and still work in progress as solution finding was more important than coding a small part of the CERLT-S

Team: AOEC

Team members:

1. K.S.Venkatram 2. Abhiram and 3. Aakkash K V

For: SAAI Factory Hackathon

Submission: Green Globe Codification to help connected emergency response at sites

Help learn from Sense and Respond Drills, Evacuations and Analysis to improve the use of art/art form/art work/allied innovation in assistants that are part of an A-Z (CERC) assistant portfolio

Table of contents **Page No**

1. Inspiration

We at AOEC find that "connected emergency response" can help many occupants where different LifeScore abilities are considered to help prepare for, sensitize, strategize and respond to swiftly save and protect life.

We at AOEC have hosted a proof of concept URL to develop this further. As a part of this ...

The Connected Emergency Response Learning Team-Suite is a framework of CERC Tools that use machine learning for different assistants to enable CERC Sense & Response systems, Social Accountability and a Bio-centrism to sensitize occupants or responders to mitigate risk, emerge & procreate.

The project will showcase Connected Emergency Response Learning for a Connected Emergency Response Centre and its A-Z (CERC) assistant framework.

Responsive & Sustainable development is termed as development that meets the needs for life score codification, risk mitigation and disaster management for trends seen or futuristically possible.

Connected Emergency Response Centres need to be designed, developed and incorporated in buildings / facilities (called as sites) to help resultant involvement or swift action to save and protect life.

About AOEC

AOEC stands for Akaash Open Enterprise Centre (a Gap analysis and problem solving consultancy) with a team comprising of myself (K.S.Venkatram), Abhiram (Technical consultant and Operations Advisor) and Aakkash K V (BTECH Automotive Engineering).

2. Problem solving (background)

The Connected Emergency Response Centre framework will need to deploy sense and respond assistants that help use the LifeScores of sites and occupants to procreate & improve the use of art/art form/art work/allied innovation in A-Z assistants that help occupants or responders swiftly act to protect and save life.

The issue being, that occupants at sites differ in their abilities to act at the time of a disaster, threat and/or accelerated risk.

However a CERC department & staff at a site can help design/implement/deploy assistants in a knowledgeable, sufficient, timely and trend sensitive manner to remain Responsive and Socially Accountable to prepare for, address LifeScore differences, sensitize, gather enquiries, resolve queries, requests or issues.

It is also a possible global endeavour and “feeling accountable” vision to transcend the issues of risk mitigation and disaster management that is adept for the ensuing climate change possible in the times to come.

Social accountability for connected emergency response is today more a global risk-mitigator. Can SA8000 be revisited?

A new SA8000-CERC with Social accountability to provide an auditable, voluntary standard, based on CCMA and Connected Emergency Response, to incorporate sense & respond solutions for

risk mitigation & disaster management, where the role of the solutions is to sensitize & empower human resources to identify, prepare for and understand a needful response to protect welfare, life & investments.

The Learning from Sense and Respond assistants

The problem on hand is to learn from each assistant's Sense and Respond experience to identify/improve the "trainable qualified-product-experiences" and the "trainable qualified-product-information" for the assistant.

A. Trainable qualified-product-experiences for an assistant are:

A.1. Evaluation of Critical Path Method for Emergency Management

[1] Relevant [2] Good [3] Adverse impact [4] Not applicable

A.2. Evaluation of Critical Path Method for Behavioral Health

[1] Relevant [2] Good [3] Adverse impact [4] Not applicable

A.3. Evaluation of Critical Path Method for Public Health

[1] Relevant [2] Good [3] Adverse impact [4] Not applicable

A.4. Evaluation of Critical Path Method for First Responders

[1] Relevant [2] Good [3] Adverse impact [4] Not applicable

A.5. Evaluation of Critical Path Method for Ambulatory Care

[1] Relevant [2] Good [3] Adverse impact [4] Not applicable

A.6. Evaluation of mitigating or managing LifeScore dynamics

[1] Relevant [2] Good [3] Adverse impact [4] Not applicable

B. Trainable qualified-product-information for an assistant are:

B.1. Evaluation of Real Time Score for

[A] Guidelines for Connected Emergency Response

[1] Relevant [2] Good [3] Adverse impact [4] Not applicable

[B] Impact reduction for Connected Emergency Response

[1] Relevant [2] Good [3] Adverse impact [4] Not applicable

[C] Positive health and wellness

[1] Relevant [2] Good [3] Adverse impact [4] Not applicable

[D] Better chances of survival for Connected Emergency Response

[1] Relevant [2] Good [3] Adverse impact [4] Not applicable

B.2. Evaluation of Interactive factors that help

[A] Remembering the Sense & Respond Intent/System for CERC

[1] Relevant [2] Good [3] Adverse impact [4] Not applicable

[B] Making sense of the Sense & Respond Intent/System for CERC

[1] Relevant [2] Good [3] Adverse impact [4] Not applicable

[C] Understanding the Sense & Respond Intent/System for CERC

[1] Relevant [2] Good [3] Adverse impact [4] Not applicable

[D] Application of the Sense & Respond Intent/System for CERC

[1] Relevant [2] Good [3] Adverse impact [4] Not applicable

B.3 Evaluation of Process-oriented factors that help the

[A] Anytime need to use this assistant / innovation for CERC

[1] Relevant [2] Good [3] Adverse impact [4] Not applicable

[B] Anywhere use of this assistant / innovation for CERC

[1] Relevant [2] Good [3] Adverse impact [4] Not applicable

[C] Anyhow use of this assistant / innovation for CERC

[1] Relevant [2] Good [3] Adverse impact [4] Not applicable

[D] Zero-unplanned effort use of this assistant / innovation for CERC

[1] Relevant [2] Good [3] Adverse impact [4] Not applicable

B.4. Evaluation of Performance factors that help the

[A] Social Performance / Trust Level for the Occupants

[1] Relevant [2] Good [3] Adverse impact [4] Not applicable

[B] Social Performance / Trust Level for the CERC team

[1] Relevant [2] Good [3] Adverse impact [4] Not applicable

[C] Social Performance / Trust Level for First Responders / Special-assistance Responders

[1] Relevant [2] Good [3] Adverse impact [4] Not applicable

[D] Social Performance / Trust Level for Construction & Building experts / associated governing authorities

[1] Relevant [2] Good [3] Adverse impact [4] Not applicable

B.5. Evaluation of Environment factors that help

[A] Site specific A-Z Portfolio for CERC

[1] Relevant [2] Good [3] Adverse impact [4] Not applicable

[B] Timeline for responsiveness and Deployment for CERC

[1] Relevant [2] Good [3] Adverse impact [4] Not applicable

[C] Strategy for sensors, systems, processes, services or remedial steps for CERC

[1] Relevant [2] Good [3] Adverse impact [4] Not applicable

[**D] Develop responsiveness via a Design-Bid-Build option, or a Design-Build option or a Construction Management** option

[1] Relevant [2] Good [3] Adverse impact [4] Not applicable

3. What it does (Solution and Approach)

The CERLT-S and its tools implement / improve Bio-centrism for Connected Emergency Response by

[a] Creative Adversial Network solutions (with Immersive & Perceptive Time Series Forecasting) for the Real Time Score, Interactive factors

[b] Generative Adversial Network solutions (with Objective Reality Recommendation engine) for the Process-oriented factors, Performance factors

[c] Convolutional Network solutions (with Strategic Connect Feature extraction) for Green Globe responsiveness

[d] Future CERC solutions (with Classification or Supervised Learning) for the Environment factors

4. Inference

The solution involves implementation / improvement of Bio-centrism for Connected Emergency Response.

For a case study in this hackathon we consider an assistant for an Emergency Exit/Exit/associated stairway**, where** LifeScore dynamics of the ability of occupants could relate to “not being to run

steadily or fast, not being able to use, assist or clasp with hands firmly, not being able to walk down steps/not being able to climb steps easily, not being well to accomplish emergency response, needing to be assisted in mobility, being pregnant, needing to carry a baby, or child or known aged person”. We term this as **Equity Level in Biocentrism.**

The lack of Biocentrism in the Emergency Exit/Exit/associated stairway could be addressed via Green Globe or LifeScore codification, a Response strategist and Made-to-assist codes that need to be incorporated in the assistant for these pre-requisites and Equity level.

5. Methodology

In the solution,

1. The Green Globe codifications in the repository are clustered using a combination of

(a) **Text-analytics** of “text fields” with select assistant names / descriptions,

(b) **“trainable qualified-product-experiences” for the assistants,**

(c) “trainable qualified-product-information” for the assistants and

(d) **a categorization variable** that categorizes the nature of sense and respond assistance, that is whether **Visual, Auditory and/or Tactile or Some other Sense and Respond experience.**

2. The Text-analytics technique is based on **Word2Vector**

3. The clustering technique is based on **DBSCAN**

4. The **Cosine similarity algorithm** is used to classify sense and respond experiences to fit within one of the buckets created (where this is based on text categorization)

5. Keras is used to create a CNN (Convolutional Neural Network) for object recognition, which we call as Strategic Connect Feature extraction from a Visual depiction/sign/art work that helps occupants or responders during a drill, evacuation or for risk mitigation

6. sklearn.neighbors is used to find Visual depiction/sign/art work recommendations for certain prerequisites, thinking expected, application and LifeScore interlinks

7. sklearn.linear_model, sklearn. model_selection and sklearn. metrics – to import Logistic Regression, train_test_split and accuracy_score for Response strategies for assistance related to LifeScore abilities of occupants

6. How we will build it

We at AOEC are developing the idea using the Python & Anaconda framework and different libraries for Neural networks, data analysis, array processing, Natural language processing, Text-analytics & clustering, visualizing of clusters, **sense and respond assistance** description similarity

For object recognition (like the running man with the Emergency exit sign), we will use Keras to create a CNN (Convolutional Neural Network) for object recognition, which we call as Strategic Connect Feature extraction from a Visual depiction/sign/art work that helps occupants or responders during a drill, or evacuation or for risk mitigation.

For our promo, we will use CIFAR-10 for this task. CIFAR-10 stands for Canadian Institute for Advanced Research.

For our promo, we will use a Content based recommendation engine to recommend Visual CERC codification for the Emergency exit assistant.

The running man illustration will have associated accentuated illustrations that occupants will need to be instructed about . The meaning for this accentuation is that the exit is Useful for a particular age group, has self-help information that can be reviewed as possible, has added-help information for old, sick or differently able, the exit is sensitive to the ability of the occupants & has CERC incorporation, the exit expects Self-organization interaction to address issues when people rush, the exit is Internet integrated to sense & respond via connectivity/addon(s(to help occupants/security/facility staff/CERC staff utilize the exit in a planned way during a drill, evacuation or connected emergency response.

We will use a Visual CERC data set for the Emergency exit assistant, which we assume is available in the /data folder. The dataset we assume will contain the following fields:

1. Pre-requisites actualized

[a] Useful for a particular age group (RMUA 000),

[b] Useful for any age group (RMUA 001),

[c] Has self-help information (RMUA 010),

[d] Has added-help information for old, sick or differently able (RMUA 100)

2. Thinking expected

[a] LifeScore Sensitized thinking expected (TE 0001),

[b] Remedial thinking expected (TE 0010),

[c] Self-organization for emergency response expected (TE 0100),

[d] Is Internet Interfaced (TE 1000)

3. Application

[a] Preparedness (A 00001),

[b] Mitigation (A 00010),

[c] Response (A 00100),

[d] Recovery (A 01000),

[e} CERC (A 10000)

4. LifeScore Interlinks for the occupants

[a] From same floor (L 0001),

[b] From same block (L 0010),

[c] From same site (L 0100),

[d] From same **Equity Level in Biocentrism** (based on similarity of sense and respond assistance) (L 1000)

[4] Response-strategist

[a] vital-mindfulness-pack (R 0000001)

[b] vital-mindfulness-guide (R 0000010)

[c] vital-mindfulness-forum (R 0000100)

[d] vital-mindfulness-pack and vital-mindfulness-guide (R 0000011)

[e] vital-mindfulness-pack and vital-mindfulness-forum (R 0000101)

[f] vital-mindfulness-guide and vital-mindfulness-forum (R 0000110)

[g] vital-mindfulness-pack and vital-mindfulness-guide and vital-mindfulness-forum (R 0000111)

[5] Made-to-assist-codes

[a] Emergency Management (M 00001)

[b] Behavioral Health (M 00010)

[c] Public Health (M 00100)

[d] First Responders (M 01000)

[e] Ambulatory Care (M 10000)

[f] .... combinations

For our promo, we will use Immersive & Perceptive Time Series Forecasting for the Real Time Score, Interactive factors

Real Time Score for

[A] Guidelines for Connected Emergency Response

[1] Relevant (0001)

[2] Good (0010)

[3] Adverse impact (0100)

[4] Not applicable (1000)

[B] Impact reduction for Connected Emergency Response

[1] Relevant (0001)

[2] Good (0010)

[3] Adverse impact (0100)

[4] Not applicable **(1000)**

[C] Positive health and wellness

[1] Relevant (0001)

[2] Good (0010)

[3] Adverse impact (0100)

[4] Not applicable (1000)

[D] Better chances of survival for Connected Emergency Response

[1] Relevant (0001)

[2] Good (0010)

[3] Adverse impact (0100)

[4] Not applicable (1000)

Interactive factors that help

[A] Remembering the Sense & Respond Intent/System for CERC

[1] Relevant (0001)

[2] Good (0010)

[3] Adverse impact (0100)

[4] Not applicable (1000)

[B] Making sense of the Sense & Respond Intent/System for CERC

[1] Relevant (0001)

[2] Good (0010)

[3] Adverse impact (0100)

[4] Not applicable (1000)

[C] Understanding the Sense & Respond Intent/System for CERC

[1] Relevant (0001)

[2] Good (0010)

[3] Adverse impact (0100)

[4] Not applicable (1000)

[D] Application of the Sense & Respond Intent/System for CERC

[1] Relevant (0001)

[2] Good (0010)

[3] Adverse impact (0100)

[4] Not applicable (1000)

By Time Series Forecasting for the Real Time Score data, Interactive factors data, we can forecast the adverse impact, or usefulness or relevance of the Visual CERC art/art form/art work/allied innovation for the Emergency exit assistant.

We build an ARIMA model for the Time Series for the Real Time Score, Interactive factors, where AR = Auto Regressive term, I = Differencing (due to de-trending) and MA = moving average term

For our promo, we will use Classification based Recommendation / Learning for the Environment factors

The Environment factors for Biocentrism include

[A] Site specific A-Z Portfolio for CERC

[B] Timeline for responsiveness and Deployment for CERC

[C] Strategy for sensors, systems, processes, services or remedial steps for CERC

[D] Develop responsiveness via a Design-Bid-Build option, or a Design-Build option or a Construction Management option

For the Equity Level needed at a site for Connected Emergency Response, the Biocentrism in the Emergency Exit/Exit/associated stairway could be in-context related to any of the following:

1. Critical Path Method for Emergency Management

2. Critical Path Method for Behavioral Health

3. Critical Path Method for Public Health

4. Critical Path Method for First Responders

5. Critical Path Method for Ambulatory Care

6. Mitigating or managing LifeScore dynamics

For our promo, we will focus on **Mitigating or managing LifeScore dynamics** by implementing what we call as zero-unplanned effort to use this emergency response assistance.

We know zero-unplanned effort is important as LifeScore dynamics of the ability of occupants could relate to “not being to run

steadily or fast, not being able to use, assist or clasp with hands firmly, not being able to walk down steps/not being able to climb steps easily, not being well to accomplish emergency response, needing to be assisted in mobility, being pregnant, needing to carry a baby, or child or known aged person".

For problem solving, we host a Response-strategist that reads the form filled by an occupant/occupant group for a Site_CERC_Assistant, to recommend sense and respond experiences or assistants to accomplish zero-unplanned effort to use the Site_CERC_Assistant.

The form could include data such as

Site name

Location

Block or Building

Flat, or Facility or Associated Occupancy

Occupant / Occupant Group

LifeScore classifier of ability

Site_CERC_Assistant code (like for the Emergency Exit it is Site_CERC_Assistant_E_1)

Response strategy that is enabled on submission of the form, where the strategy could include

[a] vital-mindfulness-pack (numeric value 1)

[b] vital-mindfulness-guide (numeric value 2)

[c] vital-mindfulness-forum (numeric value 4)

[d] vital-mindfulness-pack and vital-mindfulness-guide (numeric value 3)

[e] vital-mindfulness-pack and vital-mindfulness-forum (numeric value 5)

[f] vital-mindfulness-guide and vital-mindfulness-forum (numeric value 6)

[g] vital-mindfulness-pack and vital-mindfulness-guide and vital-mindfulness-forum (numeric value 7)

For the promo, we could say that the Response strategy for the Emergency Exit could initially include vital-mindfulness-guide and vital-mindfulness-forum, where the vital-mindfulness-guide will guide the occupants on intrinsic preparedness, mitigation, sense & respond inter-relationships so they can use the Emergency Exit in a planned and less difficult manner, and where the vital-mindfulness-forum enables/ involves occupants of the site to interact and evaluate their problems and solutions via a “Seamless forum”, so a mindful approach is adopted for all inter-relationships that affect occupants in using an Emergency Exit.

For any vital mindfulness, the data as a LifeScore classifier based strategy could be trained or checked for accuracy using logistic regression.

<u>For relating to the LifeScore for occupants</u>

Today occupants are known to need intervention or remedial healthcare at different times. For a site to be able to provide SMART intervention / interaction for its occupants, we use a LifeScore to report the interrelated Bio-informatic considerations that decide what the occupant needs or expects.

- To understand interrelated considerations, we state that each occupant has a (A) different ability, (B) different liability factor/weakness factor/stress level factor/behavioral problem influencer and (C) Emotional Quotient/Intelligence Quotient that decides what each of them can do in a disaster drill or disaster response. The Green Globe codifications ensure that occupants can be evaluated for these influencers without exposing their vulnerability to the unsecure world. *The LifeScore can develop learning pathways and decision trees for different forms of Art + AI/ML enabled visualization, reality and immersive drills or experiences that are visual, tactile, auditory etc.*
- The Green Globe codification defines a cumulative LifeScore for each occupant on the basis of certain factors such as
- (1) **Physical ability** to personally respond or mitigate risk (where the person may be Healthy, Weak, Sick or with Adversity)
- (2) **Mental ability** to personally respond or mitigate risk (where the person may be Healthy, Afflicted, Sick or Differently able)
- (3) **Acclimatized ability** to personally respond or mitigate risk (where the person is Knowledgeable; Partially knowledgeable or is Not acclimatized to mitigate risk)
- (4) **Liability to respond or mitigate risk** of occupants present (where the person may be Experienced; have Intermediate level onus; or Cannot help others)
- (5) **Health parameterization** to help a Disaster management team know beforehand about whether the occupant has a life changing condition (such as Evaluated and informed; Not evaluated or not informed; or Selective understanding about condition), has different life support requirements (Physical help needed for

response; Companionship needed for response; or Handicapped so additional help needed), uses different wear-ons that accentuate behavioral or stress vulnerability (Artificial limbs or prosthetics; Aids for hearing or speaking; Pacemaker for the heart; Chrono-immunizers; Cancer care accessories etc)

The details of the libraries follow:

Specific libraries to load data, perform computation and display output are

(a) Pandas – Data acquisition library

(b) numpy – Array processing library

(c) nltk.data and nltk.corpus – Natural language processing library

(d) gensim and gensim.models – for text analytics and clustering, where the Word2Vector function is used

(e) gensim.models.keyedvectors – to import keyed vectors

(f) matplotlib – for visualizing clusters

(g) sklearn.cluster – to import DBSCAN for clustering

(h) sklearn.metrics.pairwise – to import cosine-similarity to find out sense and respond assistance description similarity

(i) keras.datasets – to import the CIFAR-10 dataset

(j) keras – to create a Convolutional Neural Network

(k) scipy.misc - to import image functions

(l) sklearn.neighbors – to import Nearest neighbors

(m) statsmodels.tsa.stattools – to import adfuller

(n) statsmodels.tsa.arima_model – to import ARIMA

(o) sklearn.linear_model – to import Logistic Regression

(p) sklearn. model_selection – to import train_test_split

(q) sklearn. metrics – to import accuracy_score

(r) imblearn. over_sampling – to import SMOTE

Work in progress

Code snippets in the basic proof of concept for a CERC tool that clusters / trains sense and respond assistance instances at a site (step wise)

(1) To import libraries and functions

(2) To load data

(3) For filtering of requests based on assistant groups for “sense and respond assistance categorization” (where there are multiple assistant groups and one CERC Hub category, it is noted that the CERC Hub category is a proof of concept that proposes to help Connected Emergency Response problem solving and adept solution finding

(4) Text analytics to create the training data for the machine learning algorithm

(5) Running of the clustering function

(6) Assigning of a new sense and respond assistance request to a correct bucket based on the cosine-similarity function

Code snippets in the basic proof of concept for a CERC tool that creates a CNN for object recognition that can help sense and respond assistance at a site (step wise)

(1) To import libraries and functions

(2) To load data

(3) To view some images from this data

(4) To convert the class to a hot encoding matrix

(5) To create the model using convolutional layers and max pooling

(6) To flatten the output

(7) To compile the model

(8) To print the summary of the CNN

(9) To fit the model

Code snippets in the basic proof of concept for a CERC tool that uses a recommendation engine to recommend Visual CERC codification for the Emergency exit assistant **at a site (step wise)**

(1) To import libraries and functions

(2) To read Visual CERC data for the Emergency exit assistant

(3) To check initial data

(4) To apply any algorithm to convert categorical variable to numeric values

(5) To do the same for all other attributes

(6) To create nearest neighbour object

(7) To fit model

(8) To define sense and respond assistance requirement in terms of LifeScore interlinks, **Response-strategist and/or Made-to-assist-codes . In this promo we look at LifeScore interlinks**

(9) To check the most suitable additions to the Visual illustration problem solving, like **Pre-requisites, Thinking expected, Application**

Code snippets in the basic proof of concept for a CERC tool that uses Immersive & Perceptive Time Series Forecasting for the Real Time Score, Interactive factors (step wise)

(*) To do the analysis first plot the Real time score data, Interactive factors data for the Emergency exit assistant

(1) To import libraries and functions

(2) To get and print test result

(3) To apply test on Real time score data, Interactive factors data for the Emergency exit assistant

(4) If the Test Statistics is higher than critical value meaning raw time series is not stationary, so apply transformations and again check results

(6) Regenerate **plot of the transformed** Real time score data, Interactive factors data for the Emergency exit assistant

(7) Get a new time series with a difference of consecutive values

(8) Plot the new data

(9) If any record has a NaN remove that value and apply Dickey-Fuller test, check Test Statistics to see how close is it to the critical value to report confidence about the time series being stationary

(10) Apply aggregation, smoothing and regression-fitting to make Real time score data, Interactive factors data more stationary

Apply moving average technique of drills or evacuations or assistance for the past 12 values (meaning 12 months or 1 year) representing the yearly average at that point. If we feel that new values are more important than the old ones, we can use the weighted moving average technique.

(11) Generate a difference series

(12) Check the result of the test

Check Test Statistics to see how close is it to the critical value to report confidence about the time series being stationary

(13) Apply ARIMA model for time series forecasting

Note on the ARIMA model

AR (Number of Auto regressive terms): This represents the lag of the dependent variable

I (Number of differences): Number of non-seasonal differences

MA (Moving average): Lagged errors

(13.1) Import the library needed for the ARIMA model

(13.2) Make the model using AR only at first

(13.3) Calculate RSS for the model

(13.4) Make the model using MA

(13.5) Check the error

(13.6) Check the RSS value to identify the difference, if needed consider both AR and MA

(13.7) Check the error

(13.8) Check the RSS value of the above three models

(13.9) Check the best model, take it to the original form to get the current output predictions

(13.9.1)To do this create a new time series of values generated from the model

(13.9.2) Convert differencing to the log scale

(13.9.3) Add differences to the base numbers by using the cumsum() function

(13.9.4) Add values to the first base number

(13.9.5) As we are working on log, take the exponent value

(13.9.6) Plot the result

(13.9.7) Check the overall error

Thereon create models with different settings and compare RMSE

Code snippets in the basic proof of concept for a CERC tool that uses Classification based Recommendation / Learning for the Environment factors

(1) To import libraries and functions

(2) Read **Equity Level needed at a site for Connected Emergency Response for definite effort or ease in using** the Emergency Exit/Exit/associated stairway

(3) Show all the columns

(4) Check the head of the dataframe

(5) Use relevant attributes to build the classification model

(6) Convert values to numeric if needed

(7) Check the shape of the data

(8) Extract features from the dataset

(9) Extract class

(10) Split the data in training and test set

(11) Create object of logistic regression

(12) Use SMOTE algorithm for over sampling

(13) Fit the model

(14) Predict using the logistic regression model

(15) Calculate accuracy for equity level

Work in progress

7. Challenges we ran into

There are many needs for occupants and responders to act swiftly to help protect and save life/investment at the time of a disaster, threat and/or accelerated risk.

So we need to categorize sense and respond assistance based on LifeScores of sites/occupants, need for disaster readiness, mitigation, responsiveness and recovery via anytime, anywhere, anyhow, zero unplanned effort and emergent assistance, impact reduction, automation and control systems technique, where we review a real-world example for the same, that is the **assistant for an** Emergency Exit/Exit/associated stairway.

8. Accomplishments that we're proud of

Application of real-world illustrations for an Emergency Exit/ Exit/associated stairway assistant in a Connected Emergency Response Learning Team Suite (promo) that we intend to design further.

9. What we learned (Conclusion)

Machine Learning Algorithms help us use past understanding or today's details to ideate and enable solutions for corresponding or standardized resolution, where machine learning can quicken problem solving and solution finding.

10. Future Scope

Building more scope, intelligence and functionality in CERC(s) and Hub analytics to design more sense & respond assistance, intelligence and ensure continual improvement in disaster readiness, mitigation, responsiveness and recovery via A-Z (CERC) assistance, impact reduction, automation and machine learning for

1. Emergency Management
2. Behavioral Health
3. Public Health
4. First Responders
5. Ambulatory Care
6. Connected Emergency Response Analysis for A-Z assistants

11. What's next for Connected Emergency Response Learning Team-Suite (CERLT-S)

We will take the next steps in **designing a more multi-purpose Connected Emergency Response Learning Team Suite (CERLT-S)**.

We will **use and elevate this fundamental concept in a Connected Emergency Response Centre**, in a **solution deployment level,** that helps sites and occupants mitigate the current “complexity/ risk/crisis” in sensing and responding to disasters, threats and/or accelerated risks.

12.A Code Snippet Details (only a basic proof of concept for a CERC tool that clusters / trains sense and respond assistance instances at a site)

12.A.1 To import libraries and functions

```
import os

import pandas as pd

import numpy as np

from numpy import array

from IPython.display import display

#For natural language processing ability

import nltk.data

from nltk.corpus import stopwords

#gensim libraries

import gensim

from gensim.models import word2vec

from gensim.models.keyedvectors import KeyedVectors
```

```
#to visualize the clusters
import matplotlib.pyplot as plt
import matplotlib.cm as cm

#for clustering
from sklearn.cluster import DBSCAN
import sklearn.metrics as metrics

#to compute service request description similarity
from sklearn.metrics.pairwise import cosine_similarity
```

12.A.2 To load data

The algorithm will be based on the source of data for the CERC tools / CERLT-S, where this can be a spreadsheet, a .CSV dump of sense and respond assistance, or direct customized access to a CERLT-S database.

The interest is to load the data into the program as an array of strings.

The structure of the .CSV data file for example is

SRA_number	**sense_and_respond_assistance**	**CERC_group**
SRA30000	*Site LifeScore*	*CERC_Hub*
SRA32000	*Site SA8000-CERC*	*CERC_Hub*
SRA34000	*Site* Made-to-assist codes	*CERC_Hub*
SRA36000	*Site* Occupant LifeScores	*CER_ Hub*
SRA40000	Emergency Exit	Site_CERC_Assistant_E1
SRA41000	Exit	Site_CERC_Assistant_E2
SRA42000	Associated stairway	Site_CERC_Assistant_S
SRA50000	Emergency Management	Site_CERC_CPM_1

SRA51000	Behavioral Health	Site_CERC_CPM_2
SRA52000	Public Health	Site_CERC_CPM_3
SRA60000	First Responders	Site_CERC_CPM_4
SRA61000	Ambulatory Care	Site_CERC_CPM_5
SRA62000	Connected Emergency Response Analysis	Site_CERC_CPM_6
...	...	...

Code snippet

#data file that contains old sense and respond assistance details

f = 'old_sense_and_respond_assistance.xlsx'

data_1 = pd.read_excel (f, sheet_name='SRA',
converters={'sense_and_respond_assistance':str})

12.A.3 For filtering of requests based on groups for "sense and respond assistance"

The algorithm will be based on identifying the information from the data set to make sense and respond assistance categorization or clustering easy. For this example, we will use the CERC group or department to be the driver element for the clustering. The details are as follows

{'*CERC_Hub'*,

'Site_CERC_Assistant_E1',

'Site_CERC_Assistant_E2',

'Site_CERC_Assistant_S',

'Site_CERC_CPM_1',

'Site_CERC_CPM_2',

'Site_CERC_CPM_3',

'Site_CERC_CPM_4',

'Site_CERC_CPM_5',

'Site_CERC_CPM_6'}

Code snippet

```
assignment_group_subset =  {

'CERC_Hub',

'Site_CERC_Assistant_E1',

'Site_CERC_Assistant_E2',

'Site_CERC_Assistant_S',

'Site_CERC_CPM_1',

'Site_CERC_CPM_2',

'Site_CERC_CPM_3',

'Site_CERC_CPM_4',

'Site_CERC_CPM_5',

'Site_CERC_CPM_6'}

}

data_1 = data_1[data_1.assignment_group.isin(assignment_group_subset)]
```

12.A.4 Text analytics to create the training data for the machine learning algorithm

The algorithm

1. Create training data by averaging vectors for the words in the Sense and respond assistance (SRA)

2. Calculate the average feature vector for each element and return a 2D numpy array

3. This array is the training data for running cluster functions

Code snippet

```
#Load Google's pre-trained Word2Vec model known to contain 300 dimensioned vectors for
# 3 million words and phrases, this is still in a point of (work in progress) evaluation

model_google3M = gensim.models.KeyedVectors.load_word2vec_format('./GoogleNews-vectors-negative300.bin', binary=True)
```

```
#Create training data by averaging vectors for words in the sense_and_respond_assistance
#column

def createFeatureVec(words, model, num_features):


   #convert Index2word list to a set for speedy execution

   index2word_set = set (model.wv.index2word)


   #loop over each word in the sense_and_respond_assistance

   #if it is in the model's vocabulary, add its feature vector to the total

   for word in words:

      if word in index2word_set:

         nwords  = nwords + 1.

         featureVec = np.add ( feastureVec, model[word])


   #divide the result by the number of words to get the average

   featureVec = np.divide(featureVec, nwords)

   return featureVec


def getAvgFeatureVecs(vShortDescription_s, model, num_features):

   #for the given set of vShortDescription calculate the average feature vector for each list of
#words and return a 2D numpy array

   counter = 0

   #preallocate a 2D numpy array for speed in execution

   vShortDescriptionVecs = np.zeros((len(vShortDescription_s), num_features), dbtpe =
'float32')
```

```
    for vShortDescription in vShortDescription_s:
        vShortDescriptionVecs[int(counter)] = createFeatureVec(vShortDescription, model,
num_features)
        counter = counter + 1.


    return vShortDescriptionVecs
 clustering_vec = getAvgFeatureVecs(data_1['sense_and_respond_assistance'],
model_google3M, 300)
```

12.A.5 Running of the clustering function

The algorithm uses DBSCAN for clustering, which uses a high-density clustering approach. The positions of the vectors created in 12.4 are checked and high-density areas are taken as a new cluster, where low density areas separate clusters

Code snippets

```
#clustering using DBSCAN
db = DBSCAN(eps=0.3, min_samples = 10).fit(clustering_vec)
core_samples_mask = np.zeros_like(db.labels_, dtype=bool)
core_samples_mask[db.core_sample_indices_] = True
labels = db.labels_
```

Visualizing the Clustering output

```
#plot result
unique_labels = set (labels)
colors = [plt.cm.Spectral(each)
        for each in np.linpace(0,1,len(unique_labels))]
for k, col in zip (unique_labels, colors):
        if k == -1:
        #use black for noise aspect
                col = [0,0,0,1]
```

class_member_mask = (labels == k)

xy = clustering_vec[class_member_mask & core_samples_mask]

plt.plot(xy.iloc[:,0], xy.iloc[:,1], ‘o’, markerfacecolor = tuple(col), markeredgecolor = ‘k’, makersize = 14)

xy = clustering_vec[class_member_mask & core_samples_mask]

plt.plot(xy.iloc[:,0], xy.iloc[:,1], ‘o’, markerfacecolor = tuple(col), markeredgecolor = ‘k’, makersize = 1)

plt.title(‘CERLT-S Estimated number of clusters: %d’ %n_clusters_)

plt.show()

12.A.6 Assigning of a new sense and respond assistance to a correct bucket based on the cosine-similarity function

The Algorithm used

1. Create the vector from the description text of the Sense and respond assistance (SRA) using the Word2Vec function

2. Calculate the similarity score for the vector using the cosine_similarity function

3. Find the cluster that the Sense and respond assistance is assigned to where this is done based on the maximum similarity score and averaged across all Sense and respond assistance in the cluster

4. If no matching cluster is found for a Sense and respond assistance vector then the Sense and respond assistance has no training detail in the repository and hence is unassigned for any clustering

Code snippets

#Function assigns a new Sense and respond assistance to previously grouped Sense and respond assistance clusters

def newSRAClusterer(newSRAText):

#vectorize SRAText

newSRAVector = getAvgFeatureVecs (newSRAText, model_google3M, 300)

#Build the data frame with Sense and respond assitance meta data and similarity scores

```
data_8 = pd.concat ([pd.DataFrame(labels),
data_6_1[['SRA_number',sense_and_respond_assistance',

'CERC_group']]], axis=1)

data_8.rename (columns = {0:'Cluster'},inplace = True)

data_8['similarityScore'] = cosine_similarity(data_6_1.iloc[:,26:326], newSRAVector)

# Find the cluster that the Sense and respond assistance is assigned to where this is done

# based on the maximum similarity score and averaged across all sense and respond

# assistance in the cluster

similarityScoreMean = data_8.groupby ('Cluster')['similarityScore'].mean().max()

newSRACluster = data_8.groupby ('Cluster')['similarityScore'].mean().idxmax()

if similarityScoreMean >= 0.7

#this threshold needs to be tuned to ensure noise element is not incorrectly assigned a
clustered bucket

print ('The Sense and respond assistance is assigned to the cluster', newSRACluster)

print ('The Sense and respond assistance similarity to the assigned cluster:',
round(similarityScoreMean,2))

else:

print ('This Sense and respond assistance is unlike any detail in the training repository and is
not assigned to any cluster')

return similarityScoreMean, newSRACluster
```

12.B Code Snippet Details (only a basic proof of concept for a CERC tool creates a CNN for object recognition that can help sense and respond assistance at a site)

12.B.1 To import libraries and functions

```
from keras.datasets import cifar10

from keras.layers import dropout

from keras.layers.convolutional import MaxPooling2D

from keras.layers import Flatten

from keras.constraints import maxnorm

from scipy.misc import toimage

import matplotlib.gridspec as gridspec

import matplotlib.pyplot as plt

from keras.utils import np_utils

from keras.models import Sequential

from keras.layers import Dense

from keras.optimizers import SGD

from keras.layers.convolutional import Conv2D

from keras import backend as K

K.set_image_dim_ordering('th')
```

12.B.2 Loading the data

```
(x_train,y_train), (x_test, y_test) = cifar10.load_data()
```

12.B.3 See examples from this data

```
fig = plt.figure()
gs = gridspec.GridSpec(4, 4, wspace = 0.0)
ax = [plt.subplot(gs[i]) for i in range(4*4)]
for i in range(16)::
  ax[i].imshow(toimage(x_train[i]))
plt.show()
```

12.B.4 Covert the class to one hot encoding matrix

```
y_train_onehot = np_utils.to_categorical (y_train)
y_test_onehot = np_utils.to_categorical (y_test)
```

12.B.5 Use simple CNN architecture

```
#Create model
#Sequential model is selected to get a stack of layers
num_classes = 10
model = Sequential()
# First convolution layer
model.add (Conv2D(32, (3, 3), padding = 'same', input_shape=(3, 32, 32), activation='relu'))
# Second convolution layer
model.add (Conv2D(32, (3, 3), padding = 'same', activation='relu',))
# Pooling
model.add(MaxPooling2D(pool_size=(2, 2)))
# Flatten the output
model.add(Flatten())
```

```
model.add(Dense(512, activation = ‘relu’))

# Output class

model.add(Dense(num_classes, activation = ‘softmax’))

# Compile model

epochs = 50

lrate = 0.05

sgd = SGD(lr=lrate, momentum = 0.8, decay = lrate/epochs, nesterov = False)

# Compile the model

model.compile (loss=’categorical_crossentropy’, optimizer = sgd, metrics=[‘accuracy’])

# Print the summary of the CNN

print(model.summary())
```

12.B.6 Fitting the model

```
model.fit(x_train, y_train_onehot, validation_data=(x_test, y_test_onehot), epochs=250,
batch_size = 100)
```

12.B.7 Final evaluation of the model

```
loss, accuracy = model.evaluate(x_test, y_test_onehot, verbose=0)

print(“Model accuracy = {%.4f}” %format(accuracy))
```

12.C Code snippets in the basic proof of concept for a CERC tool that uses a recommendation engine to recommend Visual CERC codification for the Emergency exit assistant at a site (step wise)

12.C.1 To import libraries and functions

```
import pandas

import numpy

import sklearn

from sklearn.neighbors import NearestNeighbors
```

12.C.2 To read Visual CERC data for the Emergency exit assistant

```
visual_cerc_emergencyexit_data = pandas.read_csv ('./data/
visual_cerc_emergencyexit_data.csv')
```

12.C.3 To check initial data

```
visual_cerc_emergencyexit_data.head()
```

12.C.4 To apply any algorithm to convert categorical variable to numeric values

```
# Convert prerequisites

visual_cerc_emergencyexit_data.code[visual_cerc_emergencyexit_data ['prerequisites'] ==
'Useful for a particular age group', 'prerequisites'] = 0

visual_cerc_emergencyexit_data.code[visual_cerc_emergencyexit_data ['prerequisites'] ==
'Useful for any age group', 'prerequisites'] = 1

visual_cerc_emergencyexit_data.code[visual_cerc_emergencyexit_data ['prerequisites'] ==
'Has self-help information', 'prerequisites'] = 2

visual_cerc_emergencyexit_data.code[visual_cerc_emergencyexit_data ['prerequisites'] ==
'Has added-help information for old, sick or differently able'', 'prerequisites'] = 4
```

12.C.5 To do the same for all other attributes

Similarly we need to convert **Thinking expected**

Similarly we need to convert **Application**

Similarly we need to convert **LifeScore Interlinks for the occupants**

12.C.6 To create nearest neighbour object

nn1 = NearestNeighbors (n_neighbors=1)

12.C.7 To fit model

nn1.fit (visual_cerc_emergencyexit_data.code[:, 'prerequisites':' Application']])

12.C.8 To define sense and respond assistance requirement in terms of LifeScore interlinks, **Response-strategist and/or Made-to-assist-codes . In this promo we look at LifeScore interlinks**

requirement = [4,1,16,1]

12.C.9 To check the most suitable additions to the Visual illustration problem solving, like **Response-strategist and/or Made-to-assist-codes for required Pre-requisites, Thinking expected, Application and LifeScore interlinks**

print (nn1.kneighbors (requirement))

12.D Code snippets in the basic proof of concept for a CERC tool that uses Immersive & Perceptive Time Series Forecasting for the Real Time Score, Interactive factors (step wise)

(*) To do the analysis first plot the Real time score data, Interactive factors data for the Emergency exit assistant

For our promo

We will use the **Impact reduction for Connected Emergency Response** column of the Real time score for Visual CERC art/art form/art work/allied innovation

visual_cerc_emergencyexit_realtimescore = pandas.read_csv ('./data/ visual_cerc_emergencyexit_ realtimescore.csv', parse_dates=['Month'], index_col=1)

visual_cerc_emergencyexit_realtimescore.head()

visual_cerc_emergencyexit_realtimescore.plot()

(1) To import libraries and functions

from statsmodels.tsa.stattools import adfuller

(2) To get and print test result

```
def test_timeseries_stationary (ts) :
    ts = ts ('Real time score')
    print ('Dickey-Fuller Test:')
    dickey_fuller_test = adfuller (ts, autolag = 'AIC')
    dickey_fuller_output = pandas.Series (dickey_fuller_test[0:4], index = ['Test Statistic', 'p-value', 'Number of Lags Used', 'Observations Used'])
    for key,value in dickey_fuller_test[4].items():dickey_fuller_output['Critical Value (%s)'%key] = value
    print (dickey_fuller_output)
```

(3) To apply test on Real time score data, Interactive factors data for the Emergency exit assistant

test_timeseries_stationary (visual_cerc_emergencyexit_realtimescore)

(4) If the Test Statistics is higher than critical value meaning raw time series is not stationary, so apply transformations and again check results. Take the log of the existing data

For this promo, the column we are detailing code for is **"Impact reduction for Connected Emergency Response" where the evaluation could result in any of the following values**

[1] Relevant (0001) (numeric value = 1)

[2] Good (0010) (numeric value = 2)

[3] Adverse impact (0100) (numeric value = 4)

[4] Not applicable **(1000)** (numeric value = 8)

visual_cerc_emergencyexit_realtimescore_log_value = numpy.log
((visual_cerc_emergencyexit_realtimescore)

(6) Regenerate **plot of the transformed** Real time score data, Interactive factors data for the Emergency exit assistant

visual_cerc_emergencyexit_realtimescore_log_value.plot()

(7) Get a new time series with a difference of consecutive values

visual_cerc_emergencyexit_realtimescore_log_value_diff =
visual_cerc_emergencyexit_realtimescore_log_value -
visual_cerc_emergencyexit_realtimescore_log_value.shift()

(8) Plot the new data

visual_cerc_emergencyexit_realtimescore_log_value_diff.plot()

(9) If any record has a NaN remove that value and apply Dickey-Fuller test, check Test Statistics to see how close is it to the critical value to report confidence about the time series being stationary

visual_cerc_emergencyexit_realtimescore_log_value_diff.dropna (inplace=True)

test_timeseries_stationary (visual_cerc_emergencyexit_realtimescore_log_value_diff)

(10) Apply aggregation, smoothing and regression-fitting to make Real time score data, Interactive factors data more stationary

Apply moving average technique of drills or evacuations or assistance for the past 12 values (meaning 12 months or 1 year) representing the yearly average at that point. If we feel that new values are more important than the old ones, we can use the weighted moving average technique.

exponential_weighted_average = pandas.ewma
(visual_cerc_emergencyexit_realtimescore_log_value, halflife=12)

plt.plot (visual_cerc_emergencyexit_realtimescore_log_value)

plt.plot (exponential_weighted_average, color ='Green')

(11) Generate a difference series

exponential_weighted_average_diff = visual_cerc_emergencyexit_realtimescore_log_value - exponential_weighted_average

(12) Check the result of the test

test_timeseries_stationary (exponential_weighted_average_diff)

Check Test Statistics to see how close is it to the critical value to report confidence about the time series being stationary

(13) Apply ARIMA model for time series forecasting

Note on the ARIMA model

AR (Number of Auto regressive terms): This represents the lag of the dependent variable

I (Number of differences): Number of non-seasonal differences

MA (Moving average): Lagged errors

(13.1) Import the library needed for the ARIMA model

from statsmodels.tsa.arima_model import ARIMA

(13.2) Make the model using AR only at first

Auto Regression Model, MA = 0 in this promo

ARIMA_model_1 = ARIMA (visual_cerc_emergencyexit_realtimescore_log_value, order (2,1,0))

results_AR_1 = ARIMA_model_1.fit (disp=-1)

plt.plot (visual_cerc_emergencyexit_realtimescore_log_value_diff)

plt.plot (results_AR_1.fittedvalues, color ='Green')

plt.title ('AR Model')

(13.3) Calculate RSS for the model

print ('Residual Sum of the Square: %.6f' %sum((results_AR_1.fittedvalues-visual_cerc_emergencyexit_realtimescore_log_value_diff['Real time score'])**2))

(13.4) Make the model using MA

Moving Average Model, MA = 1 in this promo

ARIMA_model_2 = ARIMA (visual_cerc_emergencyexit_realtimescore_log_value, order (0,1,1))

results_MA_2 = ARIMA_model_2.fit (disp=-1)

plt.plot (visual_cerc_emergencyexit_realtimescore_log_value_diff)

plt.plot (results_MA_2.fittedvalues, color ='Green')

plt.title ('MA Model')

(13.5) Check the error

print ('Residual Sum of the Square: %.6f' %sum((results_MA_2.fittedvalues-visual_cerc_emergencyexit_realtimescore_log_value_diff['Real time score'])**2))

(13.6) Check the RSS value to identify the difference, if needed consider both AR and MA

ARIMA Model, AR+MA in this promo

ARIMA_model_3 = ARIMA (visual_cerc_emergencyexit_realtimescore_log_value, order (2,1,1))

results_ARIMA_3 = ARIMA_model_3.fit (disp=-1)

plt.plot (visual_cerc_emergencyexit_realtimescore_log_value_diff)

plt.plot (results_ARIMA_3.fittedvalues, color ='Green')

plt.title ('ARIMA Model (AR+MA)')

(13.7) Check the error

print ('Residual Sum of the Square: %.6f' %sum((results_ARIMA_3.fittedvalues-visual_cerc_emergencyexit_realtimescore_log_value_diff['Real time score'])**2))

(13.8) Check the RSS value of the above three models

(13.9) Check the best model, take it to the original form to get the current output predictions

(13.9.1)To do this create a new time series of values generated from the model

ARIMA_diff_values = pandas.Series (results_ARIMA_3.fittedvalues, copy=True)

(13.9.2) To convert differencing to the log scale, first (13.9.3)

(13.9.3) Add differences to the base numbers by using the cumsum() function

ARIMA_diff_values_cumsum = ARIMA_diff_values.cumsum()

(13.9.4) Add values to the first base number

ARIMA_log = pandas.Series
(float(visual_cerc_emergencyexit_realtimescore_log_value.ix[0]), index = visual_cerc_emergencyexit_realtimescore_log_value.index)

ARIMA_log = ARIMA_log.add (ARIMA_diff_values_cumsum, fill_value=0)

(13.9.5) As we are working on log, take the exponent value

ARIMA_Result = numpy.exp (ARIMA_log)

(13.9.6) Plot the result

plt.plot (visual_cerc_emergencyexit_realtimescore_log_value_diff)

plt.plot (ARIMA_Result)

plt.title ('ARIMA Model')

(13.9.7) Check the overall error

print ('Root Mean Squared Error: %.6f' %numpy.sqrt(sum((ARIMA_Result - visual_cerc_emergencyexit_realtimescore ['Real time score'])**2)/len(visual_cerc_emergencyexit_realtimescore ['Real time score'])))

Thereon create models with different settings and compare RMSE

12.E Code snippets in the basic proof of concept for a CERC tool that uses Classification based Recommendation / Learning for the Environment factors

(1) To import libraries and functions

import pandas

import sklearn

from sklearn.linear_model import Logistic Regression

from sklearn. model_selection import train_test_split

from sklearn. metrics import accuracy_score

from imblearn. over_sampling import SMOTE

(2) Read **Equity Level needed at a site for Connected Emergency Response for definite effort or ease in using** the Emergency Exit/Exit/associated stairway

equity_level_needed_data = pandas.read_csv ('./data/equity_level_needed_data.csv', sep=';')

(3) Show all the columns

Pandas.set_option ('display.max_columns', None)

(4) Check the head of the dataframe

equity_level_needed_data.head()

(5) Use relevant attributes to build the classification model

equity_level_needed_data = equity_level_needed_data[['Site name', 'Location', 'Block or Building', 'Flat, or Facility or Associated Occupancy', 'Occupant / Occupant Group', 'LifeScore classifier of ability', 'Response strategy']]

(6) Convert values to numeric if needed

(7) Check the shape of the data

equity_level_needed_data.shape()

(8) Extract features from the dataset

```
X = equity_level_needed_data.iLoc[:, :6]

X = pandas.get_dummies (equity_level_needed_data.iLoc[:, :6]).values

(9) Extract class

Y = equity_level_needed_data.iLoc[:, :7].values

(10) Split the data in training and test set

X_train, X_test, Y_train, Y_test = train_test_split(X,Y, test_size=0.10)

(11) Create object of logistic regression

logistic_regression = LogisticRegression()

(12) Use SMOTE algorithm for over sampling

smote_object = SMOTE (random_state=4)

X_res, y_res = smote_object.fit_sample (X_train, Y_train)

(13) Fit the model

logistic_regression.fit (X_res, y_res)

(14) Predict using the logistic regression model

Y_pred = logistic_regression.predict (X_test)

(15) Calculate accuracy for equity level

accuracy_score (Y_test, Y_pred)
```