DAILY LIFE ACTIVITY ANOMALY DETECTION

Activities

1. Sleep
2. Medicine
3. Eating

Anomalies

1. Start Time of the Activity
2. Duration of the Activity
3. Location of the Activity
4. Sequence of activities Performed

Datasets

CASAS DATASET HHXXX TYPE [<http://casas.wsu.edu/datasets/>]

Framework:

Training Part (75-25 Train-Test-Split) :

First, data is processed into different forms using scripts (Github)

1. **Start-Time and Duration:**

Using Clustering methods, we form different clusters and take average of each cluster and store them. These averages act as a normal behavior of our timings which we use in Fitness functions.

1. **Sequence of Activities:**

First, we give a one letter abbreviation to all activities like Sleep (S), Eat (E), Medicine (M), Work (W). Then, we concatenate them in the order of activities performed. For example: ‘SEMWEMS’. We do this for all days and store them in a file. (Sequence.py script)

1. **Location**: For each activity, we find all the locations and filter them using a threshold count. Then for each location, we store with their percentage of occurrence.

For example: Bedroom 45%, Living Room 30%, Dining 25%

*Note: Trained and tested on both Filled and Unfilled Data.*

Anomaly detection is done in 2 stages-

**Stage 1: High level anomaly detection**: This includes fitness score (start-time and duration anomaly), Sequence score (sequence anomaly) and Location anomaly score.

1. **Fitness Score**: Calculates similarity between two timings

**Fitness Function**

To evaluate difference between two timing

*Paper 1 Fitness Function*

Text, letter

Description automatically generated

*Our Modified Function:*

n)

n)

(We calculate fitness score of each activity and sum them up to get fitness score of whole day).

1. **Sequence score:** Calculates the percentage value of matched activity sequence.

Extracting Sequences of required activities in the form of a String.

For example: SEMSWEMS

S – Sleep

E – Eat

M – Medicine

W – Work

Comparing Sequence:

Using **pairwise2** function from Python **BIO** library

Input: [ List of Previous Sequences, Current Day Sequence ]

Output: Percentage value of best matched sequence

1. **Location Anomaly Score**: 0 if no location anomaly. Increases when activity is performed in a wrong location. Depends on the weightage of activity and location.

**Stage 2: Low level anomaly detection**: Deep inspection of each anomaly

1. Start Time and Duration Anomaly: We use fuzzy modelling to calculate the abnormality level of the anomaly.

**Fuzzy Modeling**

*Using Trapezoidal Membership Function* *Diagram

Description automatically generated with medium confidence*

*pf = 0.4* *(percentage factor)*

*b = min(cluster)*

*c = max(cluster)*

*x = c – b (difference)*

*a = b – (x\*pf)*

*d = c + (x\*pf)*

Chart, histogram

Description automatically generated

Normal : Timing is Normal

Abnormal- : Timing is less than Normal

Abnormal+ : Timing is more than Normal

1. Sequence anomaly: Visual representation to show changes and sequence and missing of activities.

Text

Description automatically generated

1. Location Anomaly: Find out what is causing location anomaly.

ADAPTIVE LEARNING:

Keep adding new points into the clusters and removing the old ones will change the average of the clusters, updating them to their normal behavioral changes.

This document is not for any explanation purpose. Just to revise what we have done till yet.