| Mälardalens högskola (mdh) |
| --- |
| Intelligent Systems  DVA 406 |
| Project:  Bless You!  -a CBR based sneeze detector  DRAFT 0.1 |
| **Time: Spring 2015**  Göran Forsström  [gfm10001@student.mdh.se](mailto:gfm10001@student.mdh.se)  Simon Palmér  Spr10002@student.mdh.se  Niclas Säll  sns10001@student.mdh.se |
| Teacher: Sara Afshar  sara.afshar@mdh.se  **Date: 2015-03-27\_NS** |

## Abstract

TBA

Contents

1. Abstract 1

2. Introduction 3

3. Related work 3

4. Problem formulation 3

4.1. Background 3

4.2. Problem High Level Description 3

5. Approach 3

5.1. Block Diagram, CBR System 3

5.2. Block Diagram, Feature Detector 4

5.3. Block Diagram, CBR evaluation 4

5.4. Workflow 4

6. Method 4

6.1. Extract Features Design 4

6.2. CBR System Design 4

6.3. Feature Vectors 4

6.4. Similarity Functions and Weight values 6

6.5. Case Base Library maintenance 6

6.6. System Performance Evaluation 7

7. Results and analysis 7

8. Conclusion 8

9. References 8

## Introduction

TBA

## Related work

TBA

## Problem formulation

### Background

TBA

### Problem High Level Description

TBA

## Approach

TBA

### Block Diagram, CBR System

Figure x: CBR System overview

### Block Diagram, Feature Detector

TBA

### Block Diagram, CBR evaluation

TBA

### Workflow

TBA

## Method

### Extract Features Design

TBA

### CBR System Design

The CBR system is detailed as per the figure below. The state diagram follows normal conventions for a CBR system.

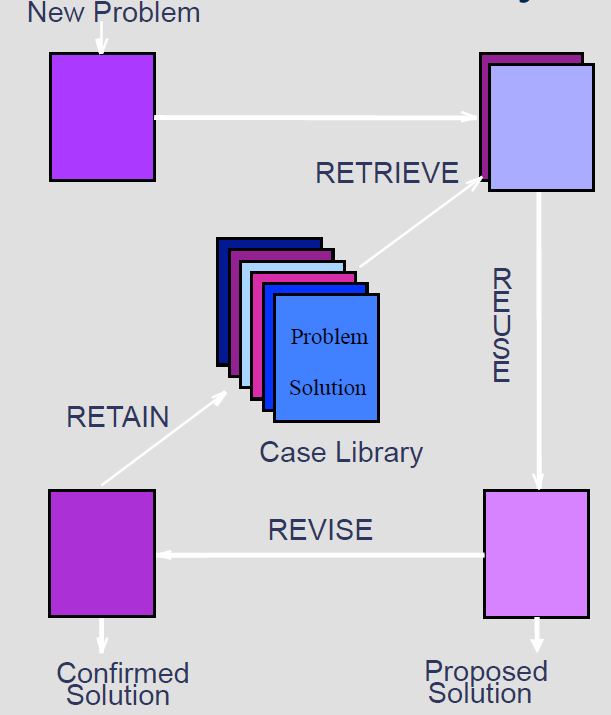


Figure X: CBR System

Notes on figure:

**New Problem:**New .wav-file is read and the feature types are extracted.

**Retrieve:**Match the new case against the cases in the library using the similarity function (SF) according to paragraph 6.4 below and create a list containing the five best matches

**Reuse:**Inspect the five best matches in terms of sneeze or not and use a majority vote to determine whether it is a proposed sneeze or not. This evaluation is the same as is done in the k-NN-method (k-Nearest Neighbors).

**Proposed Solution:**Present the result from the reuse phase to the user.

This concludes the phase when a new problem shall be classified as a proposed sneeze or not.

**Revise:**When running the program in maintenance mode, the revise phase iterates over all cases in the case library as well as a new case. The detailed process used for maintenance is:

1. Add a new, random case of known status to the library
2. Calculate the SF value for every case
3. Find the worst case by:
   * 1. Select the case which has participated in voting but voted wrong every time, if there are multiple such cases choose the case with the lowest SF value.
     2. If no such case exist: Select the case which has never participated in voting and has the lowest SF value
     3. If no case has been selected at this point, select the case with lowest SF value
4. Remove the worst case from the case library
5. Repeat 1-4 for every candidate case available to the CBR system.

The reason for this phase is to improve the case library to achieve a better hit rate.

**Retain:**In this phase all cases but the worst case from the Revise operation are retained. This case will be removed from the case library.

### Feature Vectors

The cases contains feature type vectors, which holds a set of different features. The following feature types have been selected, the table also describe how calculations are performed on the sample array (sArr), per interval (curr interval).

|  |  |  |
| --- | --- | --- |
| **Feature type** | **Calculation** | **Comment** |
| Peak | max ( | sArr[curr interval] | ) |  |
| Peak to Peak | max ( sArr[curr interval]) –  min (sArr[curr Interval] |  |
| Average | average ( | sArr[curr interval] | ) |  |
| RMS | rms( sArr[curr interval] )  rms = Root Mean Square value |  |
| Crest Factor | cf( | sArr[curr interval] | )  cf is calculated as Peak / rms |  |
| Passing through zero | pz( | sArr[curr interval] | )  pz is calculated as number of times zero is passed within an interval |  |
| FFT16 | fft16( sArr[curr interval])  Current interval is calculated using fixed number of samples in this case 65536. FFT is calculated as the energy average value in the frequency interval 1 – 5 kHz | The frequency interval can be modified for optimization purposes |
| FFT14 | fft16( sArr[curr interval])  Current interval is calculated using fixed number of samples in this case 16384. FFT is calculated as the energy average value in the frequency interval 1 – 5 kHz | The frequency interval can be modified for optimization purposes |
| FFT12 | fft16( sArr[curr interval])  Current interval is calculated using fixed number of samples in this case 4096. FFT is calculated as the energy average value in the frequency interval 1 – 5 kHz | The frequency interval can be modified for optimization purposes |

Table X: Feature types.

Replaced acc to above FTV = [PeakFeatureValueVector[i], AverageFeatureValueVector [i], Peak2PeakFeatureValueVector [i], RMSFeatureValueVector [i], CFFeatureValueVector [i], PZFeatureValueVector [i], FFT16FeatureValueVector[i], FFT14FeatureValueVector[i], FFT12FeatureValueVector[i]],

which characterizes the case, where i is an interval index according to above.

**Calculation of Feature Value vector**

Input: Sound sample: double sampleArr[]

Size of Sound sample: int size

time interval index: int timeIntervalIx

time interval size: int timeIntervalSize

The current interval in sampleArr under evaluation is timeIntervalIx \* timeIntervalSize ... (timeIntervalIx + 1) \* timeIntervalSize -1. The distance formula is according to:

The calculation of feature values, ni in interval (i) is depending on the feature type and is calculated as:

PeakFeatureValueVector: ni = max ( | sampleArr[current interval] | )

AverageFeatureValueVector ni = average ( | sampleArr[current interval] | )

Peak2PeakFeatureValueVector ni = max ( sampleArr[curentInterval]) – min (sampleArr[currentInterval]

RMSFeatureValueVector ni = rms( sampleArr[current interval] )

Root Mean Square value

CFFeatureValueVector ni = cf( | sampleArr[current interval] | )

cf is calulated as Pmax /rms value.

PZFeatureValueVector ni = pz( | sampleArr[current interval] | ) pz is calulated as number of times zero is passed within an interval

FFT16FeatureValueVector ni = fft16( sampleArr[current interval])

Current interval is calculated using fixed number of´ samples in this case 65536.

FFT is calculated as the energy average value

in the frequency interval 1 – 5 kHz

FFT14FeatureValueVector ni = fft14( sampleArr[current interval])

Current interval is calculated using fixed number of´ samples ,in this case 16384.

FFT is calculated as the energy average value

in the frequency interval 1 – 5 kHz

FFT12FeatureValueVector ni = fft12( sampleArr[current interval])

Current interval is calculated using fixed number of´ samples ,in this case 4096.

FFT is calculated as the energy average value

in the frequency interval 1 – 5 kHz

### Similarity Functions and Weight values

To compare the cases in order to find the best match a similarity function (SF) is defined according to [E. Olsson76 p.32, equation 2.34]:

where:

*w, weights* = 1

*N,* is the new case

*R,* is the retrieved case from case library

*n,* is the number of feature types in each case

*k,* is the current feature type

*f,* is the similarity function for feature type k in cases *N* and *R* it is defined as:

Where:

*i,* is the sound sample interval

*p,* is the number of intervals

*n,* is the feature value in interval *i* for the new case

*r,* is the feature value in interval *i* for the retrieved case

### Case Base Library maintenance

The purpose of maintenance is to continuously update the library so that the cases stored are the most useful. Keeping too many cases in the library will make the matching slow. One way to prune obsolete cases, i.e. case that seldom are matched, another way is to merge similar cases into a single case.

Details TBA

### System Performance Evaluation

Initially the system is used to evaluate different methods to extract features. Use the details below to evaluate how well a specific system works.

A specific system is:

1. A selected Feature Vector
2. A selected Similarity Function.

a. Set up a specific system with a multiple case files that are marked as containing sneeze or not.

b. Set up methods to extract features and compare them.

c. Execute the system:

1. For each case file (Fx) in the set of all case files.
2. Set up the CASE library for the rest of the case files but file (Fx).
3. Evaluate the selected case file (Fx) against the system.
4. Generate report:  
   file (Fx) is in fact a sneeze/none-sneeze:
   * + 1. file (Fx) is detected as sneeze with a probability of p1%
       2. file (Fx) is detected as none-sneeze with a probability of p2%

d. Calculate performance values for the current system.

1. Calculate best limits for p1 and p2 to evaluate:
2. VTS % of correctly detected sneezes (True Sneezes) - search Max
3. VFS % of incorrectly detected sneezes (False Sneezes) - search Min
4. VTN % of correctly detected none-sneezes (True Sneezes) - search Max
5. VFN % of incorrectly detected none-sneezes (False None-Sneezes) - search Min

e. Calculate a single performance value for the current system.

1. VTOT = w1 (VTS + VTN)/2 - w2 (VFS +VFN )/2

Suggestion:

1. 1. w1 = 75% w2 = 25%

## Results and analysis

The result is a detection rate of approximately XX % when analyzing a new sound file, with a case library of 50 sneeze sound files and 50 none-sneeze sound files randomly chosen among the total about 160 sound samples. However, after maintaining the library by running the maintenance function where remaining 60 sound files are used to optimize the case library the detection rate is increased to XX %.

(possibly with suggestion for improvement)

Suggested improvements:

1. The system can be optimized further by adjusting the weight values per feature type
2. Weights can be added for the intervals when calculating the individual feature type distance
3. The FFT feature can be change to more or less number of samples
4. The FFT frequency band can be adjusted.
5. Introduce noise. The new samples that are introduced for testing do not include any noise, which makes the evaluation simpler.
6. Use a microphone to continuously listen and evaluate if sneezes occur.

## Conclusion

The program manages to do a correct evaluation in about XX % of the cases which is better than

## References

Lägg till referenser