| 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: 2015-03-18** |

## Abstract

This report includes a mini project performed within the course Intelligent Systems, DVA406.   
The project chosen is a Sneeze Detector.

BLA BLA BLA

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 5

5.4. Workflow 5

6. Method 6

6.1. Extract Features Design 6

6.2. CBR System Design 7

6.3. Feature Vector 9

6.4. Similarity Functions and Weight values 10

6.5. Case Base Library maintenance 11

6.6. System Performance Evaluation 11

6.7. Feature Vector Evaluation 11

7. Results and analysis (possibly with suggestion for improvement) 12

8. Conclusion 12

9. References 12

## Introduction

In the course DVA406 Intelligent System a mini-project is included as part of the

examination. The project should let you define your problem, find a solution and solve it.

The project chosen is: “Bless You” – a CBR-based Sneeze Detector.

## Related work

Här skriver vi individuellt 2-3 st summeringar var, från artiklar relevanta för kursinnehållet, addera referenser för artiklarna

## Problem formulation

### Background

The current trend to analyze big data is a way to get early indications of events in the society. One such event is the outbreak of an influenza. It is imaginable that using sneeze detectors could be used to get an early indication of such an outbreak.

### Problem High Level Description

A microphone, placed in e.g. a public library, keeps listening to the sound in the library.

When it detects that someone sneezes a counter is incremented. A supervisory system is able to read the sneeze count at cyclic intervals. The read counter values can be used to detect if a flu is in progress.

## Approach

The experimental “Bless You” system prototype features:

* Sound input in the form of .wav-files.
* A library of sound files of known sneezes used for CBR.
* Detection of sneezes according to a feature detection: such as FFT, Crest Factor, and Wave-lets.
* Response is just a detection sneeze being TRUE or FALSE and an incremented count.

### Block Diagram, CBR System

In the figure below:

* “New Problem” = a new sneeze sound file.
* “Case Library” = sneeze features

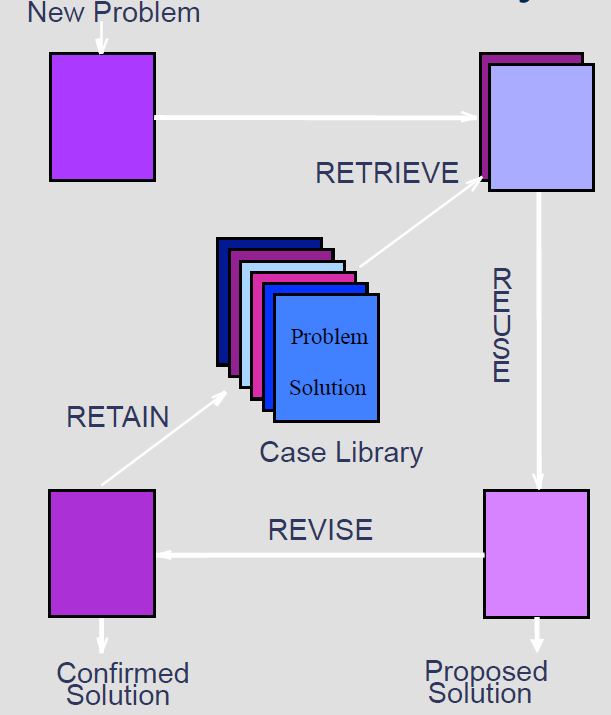


Figure x: CBR System overview

### Block Diagram, Feature Detector

The audio files in the library are analyzed for usable features.

**Sneeze Feature Extractor**:

This routine reads the sound files and extracts a relevant feature using e.g. FFT, Crest Factor or any other means.

Sound files

**Sneeze Feature Case Library**:

A database with cases containing the feature types and their corresponding feature values.

Figure X: fdadsaf

### Block Diagram, CBR evaluation

An audio file is evaluated by CBR.

**Sneeze Feature Case Library**:

A database with extracted features of the reference sneezes.

**“Bless You”**:

This program reads the selected case and compares the feature set to the Case Library and a match is search for.

The Case Library is updated if required.

The count is incremented at match.

Sneeze audio file

Report on match

Figure X: sadfdsf

### Workflow

Overview of the complete system, it consists of two programs working in collaboration via sets of files. Sound files in .wav-formats are used for feature extraction, the result is stored as feature files in .ftr-format. These files are used by the CBR System case library together with a sound file to evaluate if it contains a sneeze or not

Extract Features

.wav files

.ftr files

CBR System

Result report

Sound file to be evaluated

Operator Interaction

.ftr for evaluated file

Figure X:asdfdsf

## Method

### Extract Features Design

Operations to be performed on each .wav-file.

Command line program: BlessYou.exe

Usage:

BlessYou P1 [P2]

where

P1 = name of text file with names of all .wav-files to be examined

P2 = path to directory for created .ftr-files (optional)

Inputs: .wav-files, 16bit, 44kHz

Output: created .ftr-files.

Format of list file used as P1: one line per .wav-file:

line = <marker for type of sound> TAB [<path>]<filename of .wav-file>

<path> = <absolute path> | <relative path to directory of the list file itself>

<marker for type of sound> = ‘0’ if not a sneeze sound

‘1’ if a sneeze sound

‘?’ if unknown contents.

Flow of computations for each case, i.e. .wav-file.

1. Read .wav file contents (16 bit PCM, 44 kHz)
2. If stereo: calculate sample as average of left and right sample.
3. Normalize: search for largest sample, scale all samples so that the largest sample is 100% of 16 bits.
4. Search for start of possible sneeze: search for a sample with an absolute magnitude of at least 50%. TBA
5. Evaluate length of suspected sneeze, check for a low level 20% TBA after at least 0.3 seconds.
6. Normalize to standard length, L, e.g. 1 second TBA, split into N (e.g. 10/seconds TBA) equal time interval, indexed as t = [0, N-1]
7. The range of samples to be evaluated are now, evaluated time is: 3 seconds TBA
   * 1. 20% before TBA
     2. 80% after TBA
8. Now the feature extraction can be made, calculated as a number, e.g. 32 TBA, of float values or vector of float values, e.g.:
   1. Peak values
   2. Mean values
   3. RMS values
   4. Peak to peak values
   5. CF (Crest Factor) values
   6. PZ (Passage through Zero> values
   7. FFT values (vector)
   8. Wavelet values (vector)
9. Suggested Feature Vector: [EOlsson76 p.29, equation 2.33]  
   FV = [Peak(x), Mean(x), RMS(x), CF(x)]  
   where x is a vector of time-based sample intervals .

10. Save the extracted feature values to a file with the same file name as the main file name of the .wav-file. TBA

### CBR System Design

Input: path to directory of .ftr-files.

Input Path and file name of sound file to be evaluated:

Output: report on evaluated file being a sneeze or not.

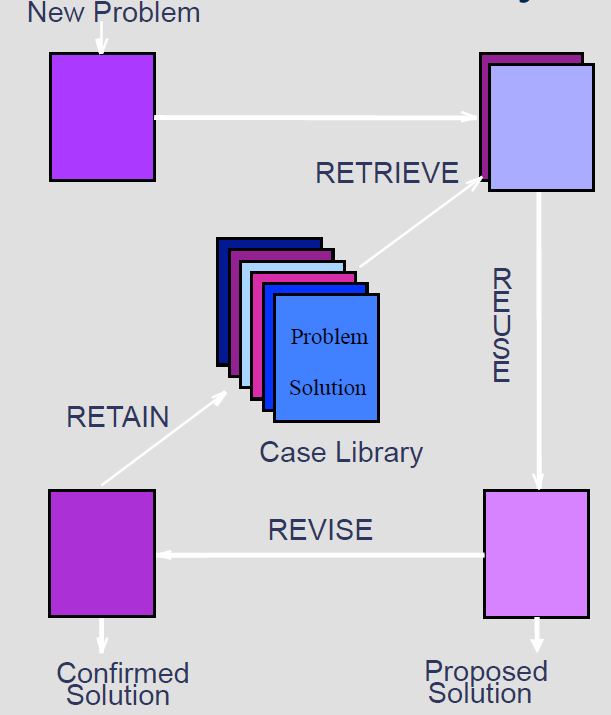


Figure X: CBR System

Notes on figure:

**New Problem:**

New .wav-file is read and the features extracted and stored in a .ftr-file.

**Retrieve:**

Match the current .ftr-file against the cases in the library using the similarity measurement “Nearest Neighbor” and create a list with the best matches using Similarity Function (SF) according to below

**Reuse:**

Inspect the five best matches and use a majority vote to determine whether it is a proposed sneeze or not. Present the result to the user.

**Revise:**

**Revideras används vid maintenance**

When running the program in maintenance mode, the program iterates over all cases in the case library as well as the new case. Statistics are gathered and afterwards the worst case is identified as:

First priority: the case that have participated in voting and voted max wrong every time

Second priority: case that never participated in voting and has lowest SF value

Third priority: case that has the lowest SF value

The outcome of the operation can be seen in the following figure:

sadfdsf

If the proposed solution does not match the confirmed solution it needs to revised and hopefully improve the detection system in the process. When an incorrect solution is proposed, the triggering input file is to be added to the case library. This will assist in the classification of similar cases in the future. The weight values for the feature extraction systems may also be modified to help future classification.

**Retain:**

**Revideras används vid maintenance**

Retain all cases but the worst case from the Revise operation. This case will be removed from the case library.

**Proposed Solution:**

Propose the answer from the reuse operation.

**Confirmed Solution:**

??

### Feature Vector

The feature type vector contains a set of different features, e.g.:   
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

The 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