| Mälardalens högskola (mdh) |
| --- |
| Intelligent Systems  DVA 406, vt15 |
| Project:  Bless You!   * a CBR based sneeze detector |
| Göran Forsström  [gfm10001@student.mdh.se](mailto:gfm10001@student.mdh.se)  Simon Palmér  Spr10002@student.mdh.se  Niclas Säll  sns10001@student.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.

Case Based Reasoning (CBR) is a process of solving problems based on the solution of previous, similar problems. A CBR system make use of a case library which stores previous cases and uses them in the evaluation and classification of new problems. Hence, these kinds of systems are easy to maintain and update. By combining a CBR system with a sound analyzer it is possible to create a system that can learn to recognize different types of sounds and classify them accordingly. By extracting the distinct features of each sound and measuring its similarity to the new problem case it is possible to produce a raw value of how likely it is that the sound belongs to a given group. Using this method we have been able to create a software program capable of recognizing a human sneeze with a high level of reliability, able to correctly classify the given samples with over 90% hit rate.

Contents

1. Abstract 1

2. Introduction 3

3. Related work 3

3.1. Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches 3

3.2. Case-Based Reasoning and User-Generated AI for Real-Time Strategy Games 3

3.3. Case-Based Reasoning: TBA 4

3.4. Case-Based Reasoning: TBA 4

3.5. Feature extraction of machine sound using wavelet and its application in fault diagnosis. Proceedings of European Conference on Case-Based Reasoning pages 686-701, 2004. NTDE International, 34:25-30, 2001. 4

3.6. Case-based reasoning is a methodology not a technology 4

4. Problem formulation 4

4.1. Background 4

4.2. Problem High Level Description 4

5. Approach 4

6. Method 5

6.1. Program parameters 5

6.2. Extract Features Design 6

6.3. CBR System Design 6

6.4. Feature Vectors 8

6.5. Similarity Functions and Weight values 8

6.6. Case Base Library maintenance and optimization 9

7. Results and analysis 9

8. Conclusion 9

9. References 10

## Introduction

In the course DVA406 Intelligent System a mini-project is included as part of the examination. In the project you define a problem, find a solution for it and solve it.

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

## Related work

### Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches

**Discussed by Simon**

In this report, Agnar Aamodt and Enric Plaza explains and discuss the basic workings and history of the case based reasoning system (CBR). They explain how Case-based reasoning is a fairly new approach to problem solving and learning that has gained a lot of attention over the last few years. Originating in the US, the basic idea and underlying theories have spread to other continents, and we are now within a period of highly active research in case-based reasoning in Europe, as well. Over the last few years, case-based reasoning (CBR) has grown from a rather specific and isolated research area to a field of widespread interest. Activities are rapidly growing - as seen by the increased rate of research papers, availability of commercial products, and also reports on applications in regular use. The paper future goes on and explains in details how a CBR systems operates and what components it has. The paper provided important details and insights for this project, explaining how the different components of a CRB operates and interact with each other, assisting in solving many of the issues that came up during the design and development process.

[Aamodt/Plaza94]

### Case-Based Reasoning and User-Generated AI for Real-Time Strategy Games

**Discussed by Simon**

In the report, Santiago Ontañón and Ashwin Ram discuss modern approaches to the use of CBR in computer games. Over the last thirty years computer games have become much more complex, offering incredibly realistic simulations of the real world. As the realism of the virtual worlds that these games emulate improves, players also expect the characters inhabiting these worlds to behave in a more realistic way. Thus, game developers are increasingly focusing on developing the intelligence of these characters. However, creating (AI) for modern computer games is both a theoretical and engineering challenge. For this reason, it is hard for end-users to customize the AI of games in the same way they currently customize graphics, sound, maps or avatars. The paper goes on to discuss how game developers may make use of a CBR system in their game in order to solve these issues and improve the quality of their game. While this paper was not as useful to the project in terms of content as some of the others, it did provide great insights in different areas of application for a CBR system. The paper also discuss the inherent problems with using a CBR inside a very broad domain, making it harder for the system to find an optimal solution to the presented problem.

[Ontañón/Ram11]

### Case-Based Reasoning: TBA

**Discussed by Niclas**

In this report, Agnar Aamodt and Enric Plaza explains and discuss

[Aamodt/Plaza94]

### Case-Based Reasoning: TBA

**Discussed by Niclas**

In the report, Santiago Ontañón and Ashwin Ram discuss

[Ontañón/Ram11]

### Feature extraction of machine sound using wavelet and its application in fault diagnosis. Proceedings of European Conference on Case-Based Reasoning pages 686-701, 2004. NTDE International, 34:25-30, 2001.

**Discussed by Göran**

In this report, Jing Lin explains and discuss

[J.Lin01]

### Case-based reasoning is a methodology not a technology

**Discussed by Göran**

In the report, Santiago Ontañón and Ashwin Ram discuss

[Watson99]

## 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 sneeze detectors could be used to get an early indication of such an outbreak.

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.

### Problem High Level Description

Create a system that can:

* Input sound input data.
* Extract sound features and place them in a case library.
* Compare a new sound with the cases in the library and evaluate if it is a match or not.
* Maintain the library by updating it with new cases that gives better performance.

## Approach

The approach to analyze and solve the problem was to create an experimental “Bless You” system prototype that contains the basic CBR functions:

* As the intended system is a server function a simple command line program was suitable as an experimental platform.
* The program output is documented as report files and console printouts.
* A set of random sneezes were collected from the internet as well as sneeze-similar sounds such as coughs together with random sounds.
* The found sounds were captured and edited into standard type of .wav-files: PCM, 16bit, 44.1 KHz, 1 or 2 channels.
* As the time to analyze the sound files was thought to possibly be quite long, a set of cached data files (.ftr-files) was introduced to optimize performance.
* The program is controlled by command line parameters and file lists in text files so that it was easy to experiment with different sets of sound files.
* The program is able to run in two modes depending of the parameter setup:
  + Build a case library, then in a loop: extract 1 file from the library and evaluate performance, repeat for each entry in the case library and calculate average performance.
  + Build a case library, then evaluate a single selected file.
* Learning: simulated maintenance phase where the case library is updated from the result of the analyze of a new set of sample files with known status.

## Method

The method used is implemented in a program structured as below:

Extract Features

.wav files

CBR System

Result report

Sound file to be evaluated

Operator Interaction

.ftr file to be evaluated file

.ftr files

Figure 1: program structure

### Program parameters

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)

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

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

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

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

‘1’ if a sneeze sound

‘?’ if unknown contents.

### Extract Features Design

Flow of operations 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 set to a predefined value, defined in C\_MAX\_POSSIBEL\_VALUE, e.g. 100000
4. Search for start of possible sneeze: search for a sample with an absolute magnitude of at least C\_TRIGGER\_LEVEL\_IN\_PERCENT, e.g. 50%.
5. Evaluate length of suspected sneeze, check for a low level defined in C\_TRIGGER\_OFF\_LEVEL\_IN\_PERCENT, e.g. 10% TBA after at least a time defined in C\_TRIGGER\_OFF\_DURATION\_IN\_MILLI\_SECS, e.g. 1000 ms.
6. Split into N = C\_NR\_OF\_INTERVALS, e.g. 10, equal time interval, indexed as t = [0, N-1]
7. Add a prefetching length, defined in C\_TRIGGER\_PREFETCH\_IN\_MILLI\_SECS of samples before the trigger.
8. Now the feature extraction can be made, the result is stored in a vector of float values, one value per interval. There is one vector per feature type, see table Feature Types! This is based on the suggested Feature Vector in: [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 .
9. If there is performance issues in extracting the features, future optimization is possible by caching the vectors for each sound file in a .ftr-file with the same file name as the main file name of the .wav-file.

### CBR System Design

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

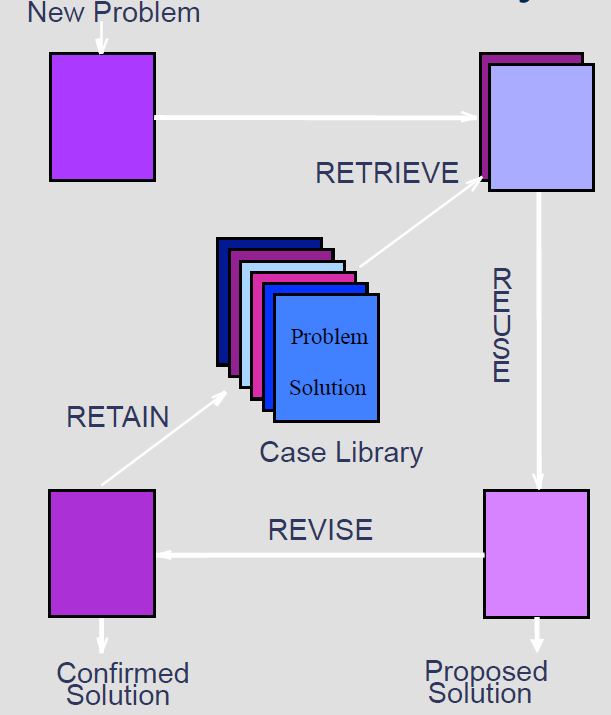


Figure 2: CBR System

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 6.5 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. At Retain then the best cases are kept and the worst removed.

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 |  |
| 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.

### 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 and optimization

Maintaining a high quality and relevant case library is critical for the performance and efficiency of any CBR system. Having too many cases will result in slow evaluation and system slowdown. Having too few cases can lead to insufficient data resulting in a high error ratio. In order to keep the case library in optimal condition cases that are no longer relevant or inconsistent should be removed while new cases with better consistence and contribution should be added. Hence, cases that are rarely used in evaluation or causes incorrect evaluations are prime candidates for removal.

When evaluating the case library the system strive to maintain the current ratio of sneeze/non-sneeze cases as well as the total volume of the library. As such, if we add a non-sneeze case we also make sure that we remove a non-sneeze case and vice versa. This is the process used for evaluation.

## Results and analysis

With a case library of 50 sneeze sound files and 50 none-sneeze sound files randomly chosen among the total about 160 sound samples, the result is a detection rate of approximately 87 %.

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 91 %.

Details before maintenance (87%)

Number of correct SNEEZE guesses: 48

Number of correct NONE SNEEZES guesses: 39

Number of incorrect SNEEZE guesses: 2

Number of incorrect NONE SNEEZES guesses: 11

Details after maintenance (91%)

Number of correct SNEEZE guesses: 49 (+1)

Number of correct NONE SNEEZES guesses: 42 (+3)

Number of incorrect SNEEZE guesses: 2

Number of incorrect NONE SNEEZES guesses: 7 (-4)

## Conclusion

The program manages to do a correct evaluation in about 90 % of the cases which is better than we had expected when starting the project.

In fact, for the intended application, this is quite good enough as the bases for the detection is to get a statistical measure of the sneeze frequency – not to be able to properly detect each and every sneeze!

Suggested future development:

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.

## References

|  |  |  |
| --- | --- | --- |
| **Ref.** | | **Document Title** |
| EOlsson76 | Fault Diagnosis of Industrial Machines using Sensor Signals and CASE-based Reasoning  Doctoral Dissertation nr. 76, MdH Västerås, Sweden 2009 | |
| Aamodt/Plaza94 | Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches  <https://www.idi.ntnu.no/emner/tdt4173/papers/Aamodt_1994_Case.pdf> | |
| Ontañón/Ram11 | Case-Based Reasoning and User-Generated AI for Real-Time Strategy Games  <http://www.cc.gatech.edu/faculty/ashwin/papers/er-11-02.pdf> | |
| Lin01 | Feature extraction of machine sound using wavelet and its application in fault diagnosis  J.Lin  Proceedings of European Conference on Case-Based Reasoning pages 686-701, 2004 + NTDE International, 34:25-30, 2001. | |
| Watson99 | Case-based reasoning is a methodology not a technology  I. Watson  <http://link.springer.com/chapter/10.1007%2F978-1-4471-0835-1_15#page-1> | |
|  |  | |
|  |  | |