MÄLARDALENS HÖGSKOLA (MDH)

Intelligent Systems
DVA 406, vt15

Project:
Bless You!
- a CBR based sneeze detector

Göran Forsström
gfm10001@student.mdh.se

Simon Palmér
Spr10002@student.mdh.se

Niclas Säll
sns10001@student.mdh.se

Date: 2015-03-29

1. Abstract

This report accounts for 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 solutions of previous, similar problems. A CBR system makes use of a case library which stores previous cases and uses them in the evaluation and classification of new problems. Hence, this kind of system is 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 similarity 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

2. 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.

3. Related work

3.1. 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]

3.2. 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]

3.3. Hybrid Data Mining and Case-Based Reasoning User Modeling System Architecture

Discussed by Niclas

This article describes an application of CBR as an evaluation method within a larger system. It shows how CBR can be used in applications where a learning process is vital for the performance of the whole system.

Today, when big data is a reality, there is a problem that knowledge bases simply contain too much data. In order to draw correct conclusions from a data set, i.e. filter out irrelevant data, good algorithms are needed to make the system perform. This article describes a hybrid architecture to model a user oriented knowledge base, i.e. adapted to fit individual users.

The first step is to construct a model where the user can influence the model by interaction, so user preferences can be included in the model. Secondly the relevant database is vectorized, to fit in the third step which is the classification of the vectors for the CBR evaluation, i.e. build up the case library. In order to make the vectorization efficient a Support Vector Machine is implemented. This can still be a tedious work, so a Sequential Minimal Optimization algorithm is implemented to further increase efficiency.

In order to make the following CBR process more efficient a Self-Organizing Map (SOM) is planned to be implemented. The SOM steps are: establish a feature-map, optimize the feature map to improve the performance. In our project we have the same steps.

The article CBR part is relevant to see how a CBR system can be integrated into a large complex system. We have used this part.

In our project we do not have the size and complexity of the database, nor do we have any user specific knowledge base requirements, leading to the need of dynamic knowledge base and features. Our feature map (cases in case library) are randomly selected from the database and the features are statically chosen. In the maintenance phase we can optimize the library.

[Isa/Blanchfield/Yuan08]

3.4. Case-Based Reasoning: TBA

Discussed by Niclas

The Fourier Transform (FT) is a method for transforming a continuous periodic function from a time domain into a frequency domain. The paper starts discussing sinusoids for building up a function and how this function can be decomposed in to the individual sinusoids building up the complete function. Furthermore, a sinusoid can be built up of complex numbers and exponentials instead of sinus and cosines terms, leading to that a periodic function can be described accordingly. Using these facts a function can be expressed both in the time domain and the frequency domain. These functions are analogue but for computer algorithms they need to be discrete, Discrete Fourier Transform (DFT) deals with this.

By sampling a continuous signal at discrete time intervals (sampling period) the Fourier transform can be implemented as a DFT. The sampling frequency (1 / sampling period) must be at least twice the highest frequency of the signal to be investigated, according to the Nyquist Theorem. The DFT calculation complexity is $O(N^2)$.

In order to reduce the complexity the Fast Fourier Transform (FFT) was invented. It has calculation complexity of $O(N \log_2 N)$.

The FFT algorithm has to have a sample length that is a power of 2, if not so originally, it can be padded t correct length.

The Fourier transform can be used in many different applications such as Polynomial Multiplication, Sequential Retrieval and Filtering. This paper gives a good overview of the Fourier Transform, how it is applied and interpreted, without going in to the specific mathematical details. It is very easy to read and to get the big picture as well as some details, and applications. It built a good foundation for implementing FFT in the project.
[Shatkay95]

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.

Discussed by Göran

In this paper, Jing Lin analyses the problem to decode machine failure sound recordings disturbed by heavy noise. The type of noise can be so heavy that not even experienced humans can recognize the present type of machine fault. Lin argues that the wavelet technique can purify the sound if the wavelet used is similar to the machine sound searched for - in this case an impulse. As the Morlet wavelet (based on an exponentially decaying cosine signal) has the characteristics of an impulse, it is considered as a good candidate even if it does not have the orthogonal property which makes it more difficult to handle theoretically. The Morlet wavelet is applied to automobile engine problems and makes it possible to filter the noise so that a human listening to the output now can identify the malfunction.
The idea to use a wavelet similar to a sneeze was considered in our project but never implemented due to shortage of resources. According to the paper we would probably ended up with the same problem as Lin: the difficulty to set proper thresholds, a task that requires both knowledge and experience.
[J.Lin01]

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

Discussed by Göran

In this paper I. Watson discusses whether Case Based Reasoning is a *methodology* or a *technology* and decides on the alternative "*methodology*" as CBR describes **what** the CBR reasoner does instead of **how** it is done. This matches the Peter Checkland quote that a methodology is "*a set of principles that guides actions*". Watson continues to describe a set of 4 CBR-examples that are guided by these principles in different ways using different techniques: Nearest Neighbor, Induction, Fuzzy Logic and Database SQL-statements. Watson then concludes that all the examples use the same methodology but with different techniques and thus CBR is a methodology.
Thus in this project we could have had a much broader view on the techniques used than just Nearest Neighbor - an interesting idea that the lack of time did not permit us to expand on.
[Watson99]

4. Problem formulation

4.1. 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 a public place, e.g. a 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.

4.2. Problem High Level Description

Create a system that can:

- Input sound 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.

5. Approach

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

Approaches:

- 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 name lists in text files so that it is 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: 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 of unknown status.
- Learning: simulated maintenance phase where the case library is updated with the result from the analysis of a new set of sample files with known status.

6. Method

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

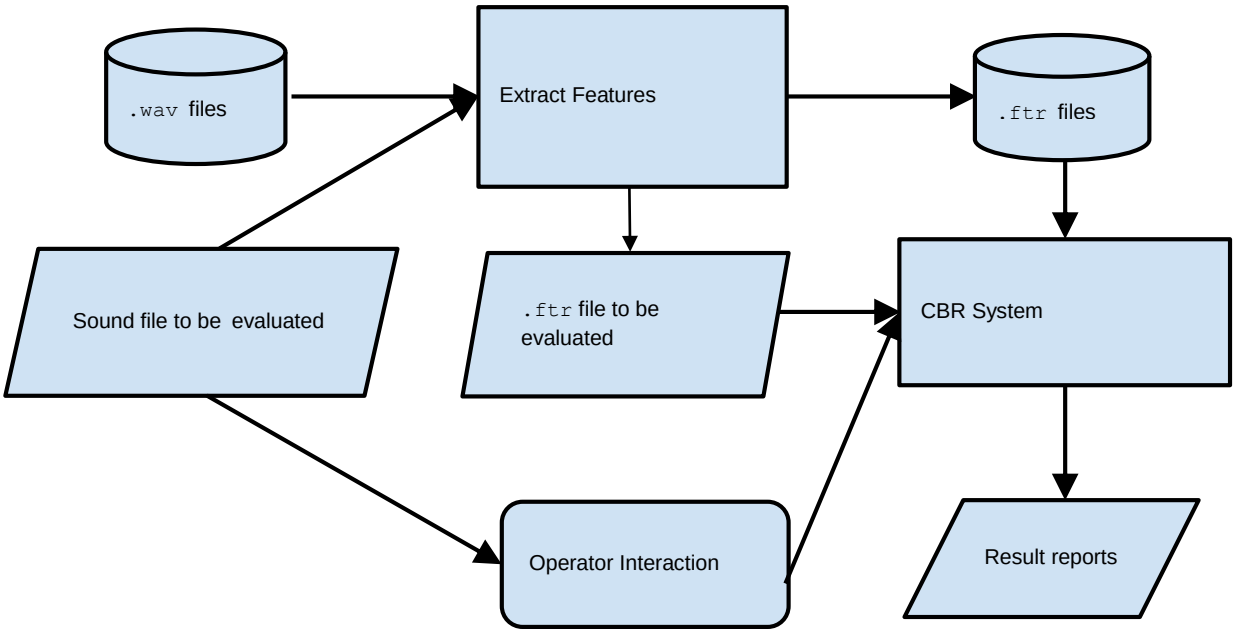

Figure 1: program structure

6.1. Program parameters

Usage:

```
BlessYou P1 P2 [P3]
```

where

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

`P2` = File name for new problem | "all": all files in Case Library run in sequence

`P3` = 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.
```

6.2. Extract Features Design

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

1. Read `.wav` file contents (16 bit PCM, 44.1 kHz, Stereo/Mono)
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% after at least a time defined in `C_TRIGGER_OFF_DURATION_IN_MILLI_SECS`, e.g. 1000 msec.

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]
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.

6.3. CBR System Design

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

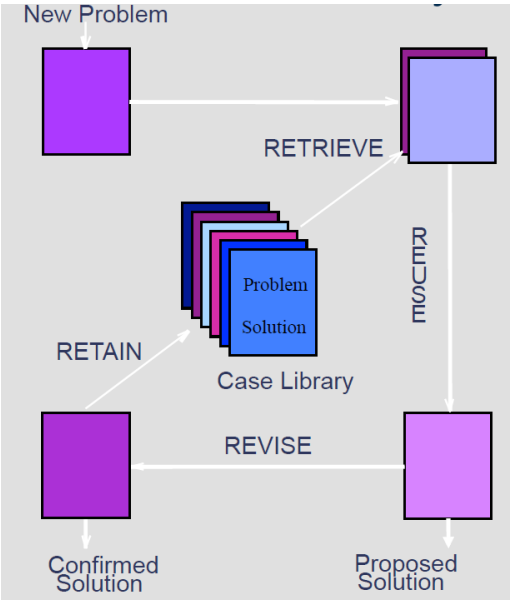

Figure 2: CBR System [Funk15]

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 the new problem 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 and evaluate it for sneeze or not according to Retrieve and Reuse phases above.
3. Find the worst case by:
   i. 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.
   ii. If no such case exist: Select the case which has never participated in voting and has the lowest SF value
   iii. 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.

6.4. 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]) $\text{rms} = \sqrt{\frac{1}{n}(x_1^2 + \cdots + x_n^2)}$	
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	fft14(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	fft12(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 1: Feature Types.

6.5. 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]:

$$SF(N,R) = \sum_{k=1}^{n} w_k * f_k(N,R)$$

Where

w, weights $\sum_{k=1}^{n} w_k = 1$

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:

$f_k(N,R) = \frac{1}{1+d_k(N,R)}$ (0.0 = no similarity; 1.0 = full similarity)

Where

$$d_k(N,R) = \sum_{i=1}^{p} |n_i - r_i|$$

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

6.6. 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 at the evaluation.

7. Results and analysis

With a case library of 50 sneeze sound files and 50 none-sneeze sound files randomly chosen among a total of about 160 sound samples, the result is a detection rate of approximately 87 %.
However, after maintaining the library by running the maintenance function, where the 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)
```

To present the system a simple GUI has been developed for demonstration purposes.

8. 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 base for the detection is to get a statistical measure of the sneeze frequency – not to be able to properly detect each and every sneeze!
Before actually deploying a system as “Bless You”, thoughts should be given to the matter of personal integrity. Calculating statistics for too small populations, say a city suburb, and then make it possible to notice that this part of a city has worse health than other parts could be a break of the integrity of this population. And even worse: make health insurance companies raise the fees for inhabitants in such an area!
Another obstacle for such a system as “Bless You” is that it easily could be suspected of illegal surveillance and thus introduce public indignity.

Suggested future development:

1. The system can be optimized further by adjusting the weight values per feature type.
2. Weights can be added also for the intervals when calculating the individual feature type distance.

3. The FFT feature can be change to use 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 in real time.
7. Add features, e.g. wavelets

9. References

Ref.	Document Title
EOlsson76	*Fault Diagnosis of Industrial Machines using Sensor Signals and CASE-based Reasoning* E. Olsson Doctoral Dissertation nr. 76, MdH Västerås, Sweden 2009
Funk15	*DVA406, Case-Based Reasoning, Lecture 6* P. Funk MdH Västerås, Sweden 2015
Aamodt/Plaza94	*Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches* A.Aamodt & E. Plaza 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 S. Ontañón and A. Ram 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
Shatkay95	*The Fourier Transform – A Primer* H. Shatkay http://www.phys.hawaii.edu/~jgl/p274/fourier_intro_Shatkay.pdf
Isa/Blanchfield/Yuan08	*A Hybrid Data Mining and Case-Based Reasoning User Modeling System Architecture* D. Isa, P. Blanchfield & C. Z. Yuan http://www.iaeng.org/publication/WCE2008/WCE2008_pp76-79.pdf