Software Architecture for Human-Robot Dialog Systems

Jonathan Placatka

Background

A Lightweight Algorithm for Social Robot Turn Taking

- uses standard signal processing techniques to allow a robot to recognize starts, stops, and pauses in conversation
- was used in a conversational robot for self-reflection
- does not incorporate speech understanding
- human-robot dialog on systems with low computing power

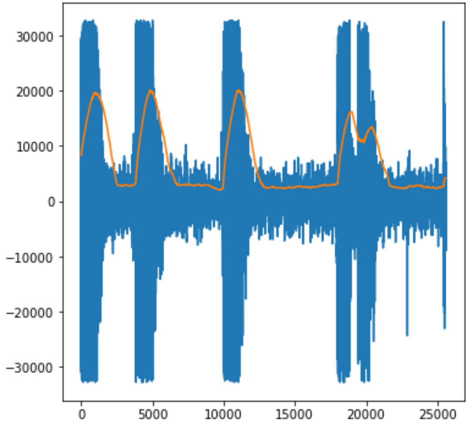
Understanding Lorena’s Algorithm

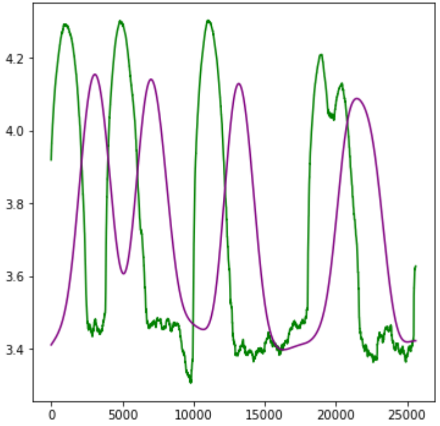
```
data = csvread("44.1KHzPCMdatafile.csv");

% down sample PCM data, 44.1 kHz to 544.4Hz
reducedData = resample(data, 1,81);

% get RMS values and use the kernel to smooth the data
RMSData=zeros(size(reducedData,1),1);

for i = 137:size(RMSData,1) % .25 sec is 136 samples
  RMSData(i) = rms(reducedData(i-136:i));
end

RMSData = log10(RMSData+1); % match perception

% use 1 second Gaussian filter
RMSDataFiltered = conv(RMSData, GaussianFilter, "same");

% get derivative of RMS data
Derivative = conv(RMSDataFiltered, [1 -1], "same");

% set treshold. RMS treshold based on "silence" at start
DerivativeThresh = .004;  % chosen via trial and error
RMSThresh = max(reducedData(1:250))*1.5;

%test for status
for i = 1: size(reducedData,1)
    if (status ~= talking && ...
            Derivative(i) > DerivativeThresh)
        status = talking;
    end
    if (status == talking && ...
            Derivative(i) < -DerivativeThresh && ...
            RMSDataFiltered(i) < RMSThresh)
        PauseStart = i;
        status = smallPause;
    end
    if (status == smallPause && ...
            (i - PauseStart) > 1088) %pause > 2 sec
        status = stop;
    end
    detection(i) = status;
end
```

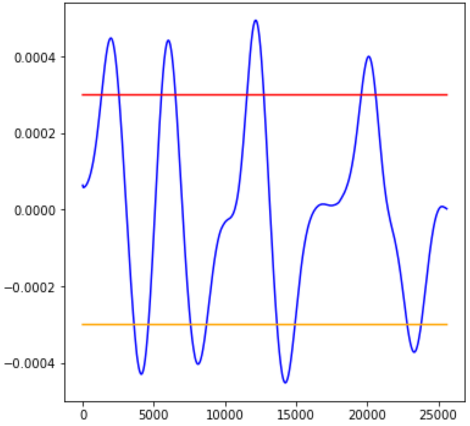
- required background knowledge not in my degree
- self-learning for signal processing techniques (RMS, fourier analysis, convolutions)
- coding in Java: dealing with live audio data, stream processing, moving windows, latency

Algorithm overview

RMS -> Logarithm -> Gaussian Smooth -> Derivative -> Check Thresholds

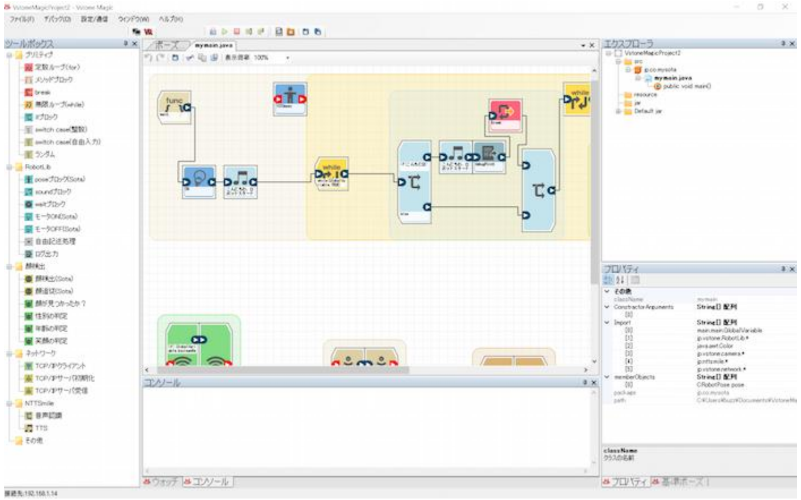






Sota

- designed for social applications
- small, inexpensive



Running Lorena’s Algorithm on Sota

Sota challenges:

- Custom linux distribution
- Different audio drivers
- Very low processing power: dual-core Intel Atom CPU @ 500 MHz
- Documentation in Japanese

Challenges

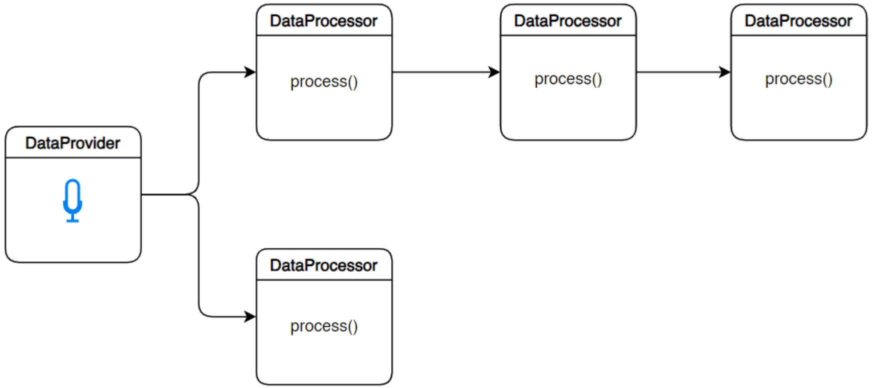
- Working with real time audio data is difficult
- Building a dialog system is time consuming
- Difficult to modify/experiment
- Want to build upon existing system to add new features

Approach

- “I don’t want to do that again!”
- **event-driven pipeline approach**
- solve local problems individually, chain together

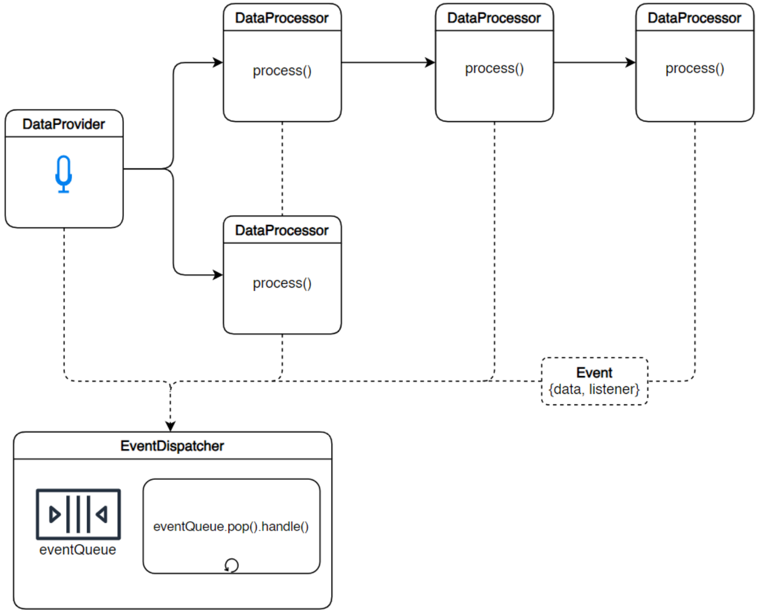
Pipeline Architecture for Dialog Systems

- Idea: break up processing tasks into modules
- Each processor does a small, targeted job
- 2 types of modules: DataProvider, DataProcessor



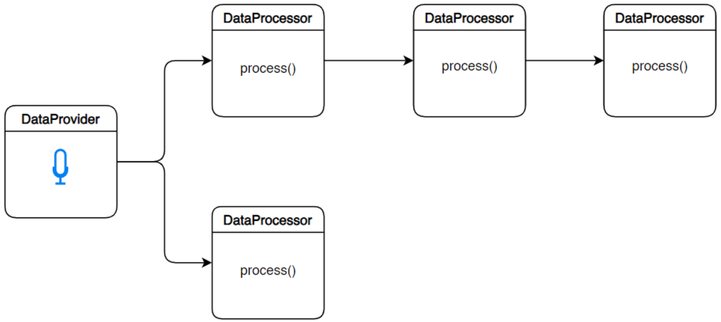
Event-Based Dialog Architecture

- I created an event system to centrally manage the flow of data
- User defines the chain of DataProviders/DataProcessors
- Modules generate events which are dispatched by the event system in order



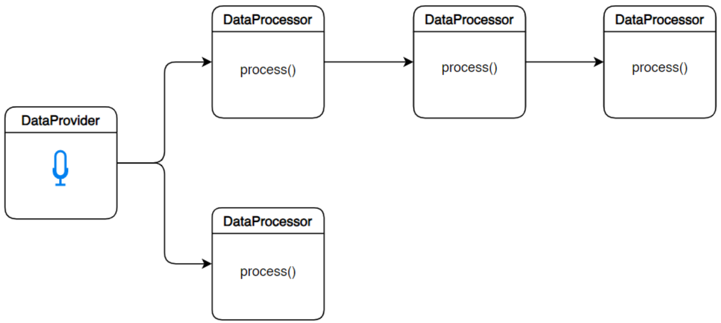
Benefits

- Hides complexity
- Modular, easy to modify system and prototype new ideas
- Easy to create new modules
- Supports concurrency
- Not limited to audio data

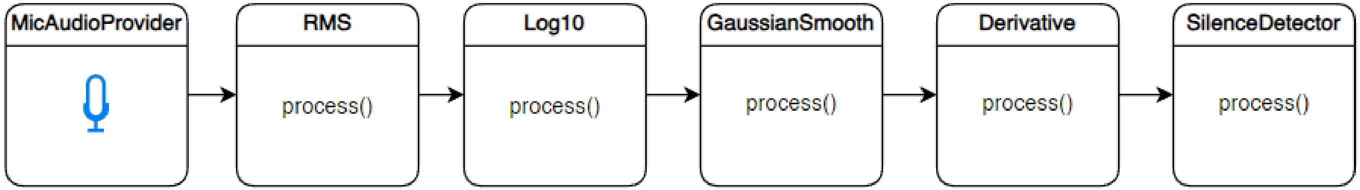


Litmus test

- Version 1 - raw java: 2+ months
- Version 2 - my architecture: 4 hours



Example: Turn-Taking Algorithm



```
EventDispatcher dispatcher = new EventDispatcher();

DataProvider provider = new MicAudioProvider(sampleRate:4000, bufferSize:2048);

RMS rms = new RMS(windowSize:1000);
provider.addListener(rms);

Log10 log = new Log10();
rms.addListener(log);

GaussianSmooth g = new GaussianSmooth(sigma:1, size:4000);
log.addListener(g);

Derivative d = new Derivative();
g.addListener(d);

SilenceDetector s = new SilenceDetector(pauseLength:8000, startupLength:8000, threshold:0.0003);
d.addListener(s);

s.addListener(new SotaOutputController());

provider.start();
dispatcher.run();
```

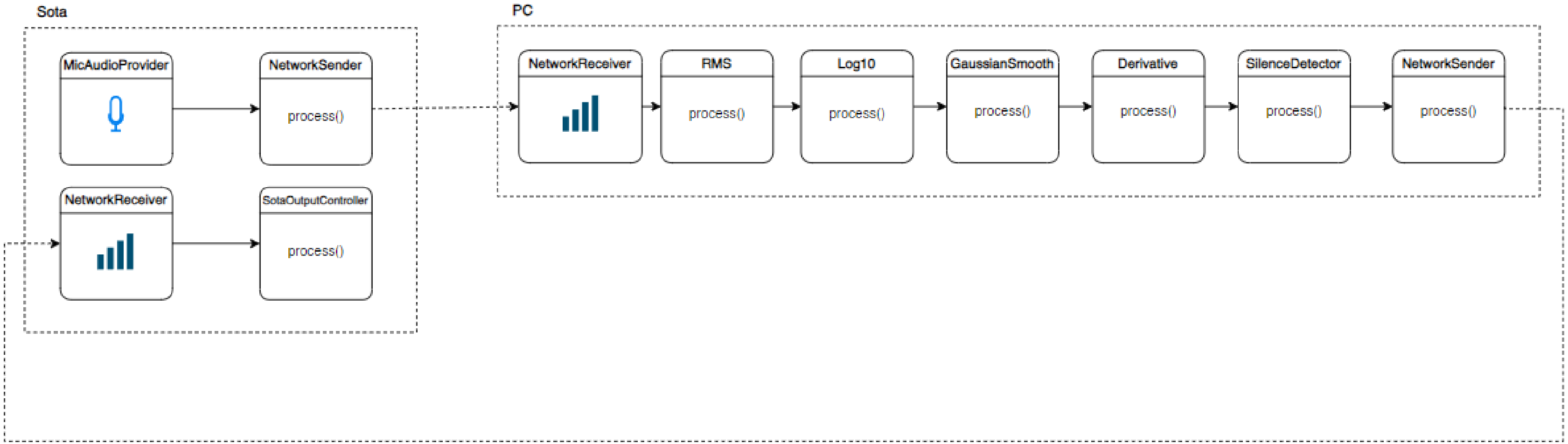
Example: DataProcessor

```java
public class Log10 extends DataProcessor {

    @Override
    protected Data process(Data input, EventGenerator sender) {
        DoubleData doubleInput = (DoubleData)input;
        double[] data = doubleInput.data;
        double[] output = new double[data.length];

        for(int i = 0; i < data.length; i++) {
            output[i] = Math.log10(data[i]);
        }

        return new DoubleData(output);
    }
}
```

Demo

Networking

- Overcame issue of low processing power on Sota by adding networking support



Future Work

Can create new modules to run alongside turn-taking algorithm:

- Prosody
- Tone
- Speech Understanding

Use library of modules to build and prototype dialog systems

Questions