The 13th Korea-Japan Workshop on Ling and Lg Processing
Waseda University, 2012/12/01

Towards User Generated Speech Databases in Language Education

Gábor PINTÉR

Kobe University
School of Languages and Communication

g-pinter@pearl.kobe-u.ac.jp
www.pinlab.info/talks/20121201-kj13/

The Slides

- find the slides at:

www.pinlab.info/talks/
20121201-kj13

Roadmap

0. Self Introduction
1. Background & Problems
    1. ASR + CALL = CAPT
    2. Problems & limitations
    3. Just like Moodle?
    4. Existing solutions
2. Yet Another Project
    1. Project overview
    2. Bolts and nuts
    3. Data
    4. What comes next?

Section 0

Self Introduction

From Hungary

Hungary

population
10 million

area
93,000 km2

language
Hungarian (92%)

src: en.wikipedia.org

Education & Work

Undergrad.	Japanese, English Linguistics, generative phonology & syntax
MA	phonology, optimality theory
PhD	perceptual phonology, phonotactics
Work	Advanced Media: Speech Recognition
	Research & Development
Teaching	Osaka U of Foreign Languages, Kobe University, Kansai University

Research Interest

Linguistics

psycho-linguistics
phonology
phonetics

Education

Pronunciation
Hungarian
English

speech recognition

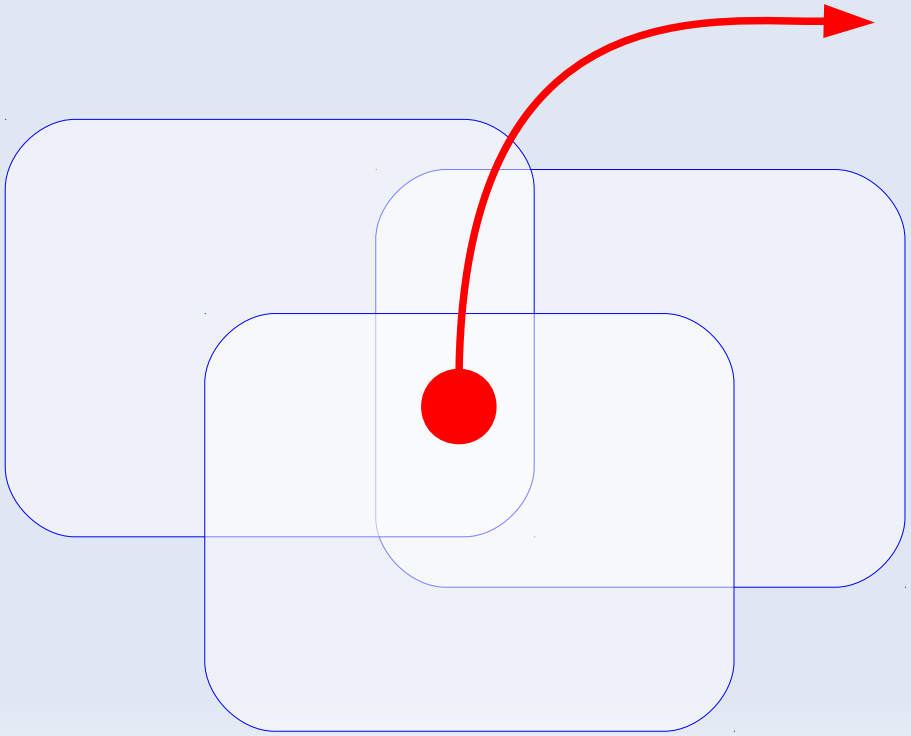
programming

Artificial Intelligence

Research Interest

Intersection of domains

Purpose	education	pronunciation training
Theory	linguistics	L1/L2 perception
Method	AI	Automatic Speech Recognition (ASR)

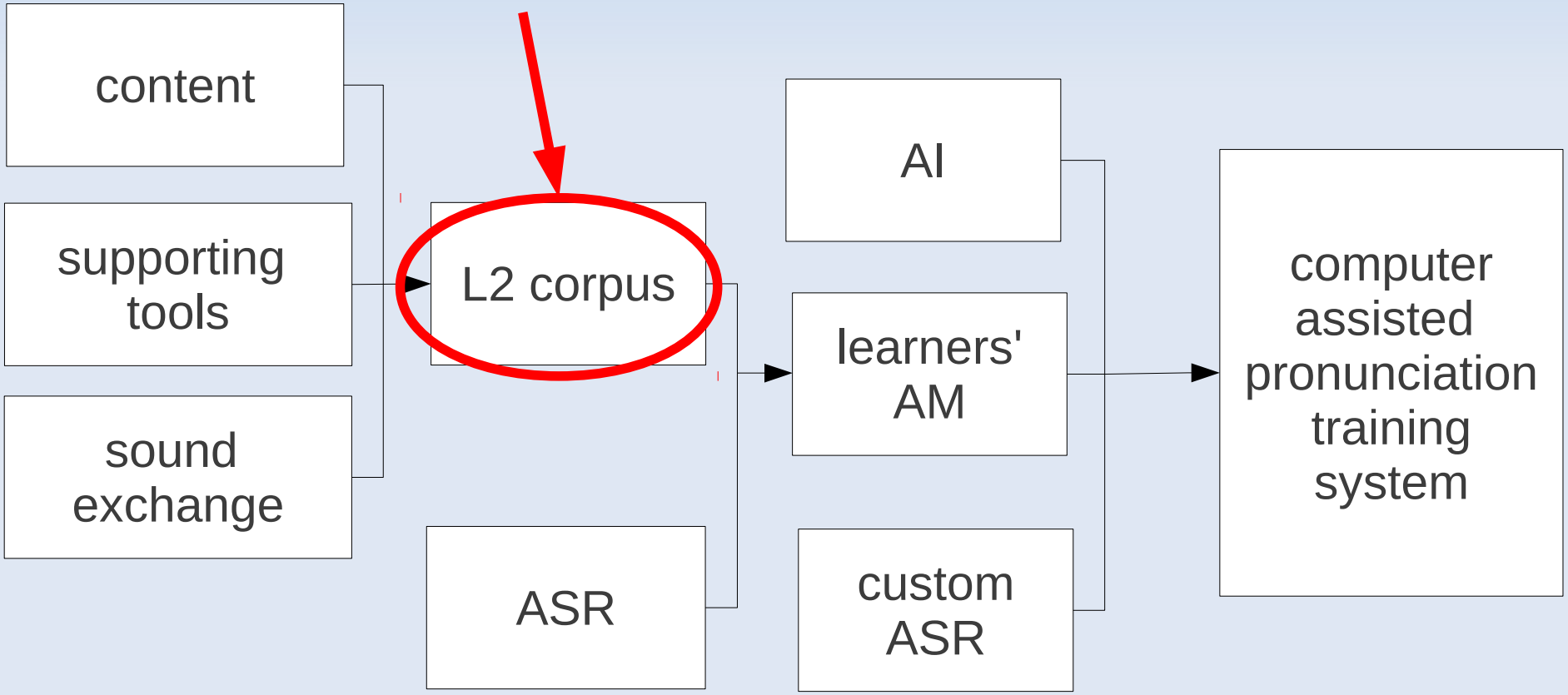


CAPT

- **C**omputer
- **A**ssisted
- **P**ronunciation
- **T**raining

Goal

- develop pronunciation training solutions
- to help create projects with similar goals



Roadmap

0. Self Introduction
1. Background & Problems
    1. ASR + CALL = CAPT
    2. Problems & limitations
    3. Just like Moodle?
    4. Existing solutions
2. Yet Another Project
    1. Project overview
    2. Bolts and nuts
    3. Data
    4. What comes next?

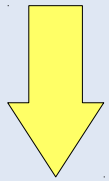
Section 1.1

ASR + CALL = CAPT

ASR meets CALL

- traditional pronunciation training
  - demanding: requires individual attention
  - unfit for large classes
  - unfit for self-study (unlike multiple choice exercises)
  - teachers are often skeptical

    → pronunciation training is often neglected



high expectations for
Computer-**A**ssisted **L**anguage **L**earning

AmiVoice CALL Lite - pronunciation -

メインメニューに戻る

発音評定

Standard Edition LOAD

単語-Level1

02. map

map

和訳

総合評価

★ ★ ★ ★ ★

○ 語尾の「p」の後に、日本語の母音の「う」の音が入っています。

○ 「æ」の発音が、日本語の「あ」の発音になっています。

map

TERCHER

USER

PITCH CURVE

発話開始

戻る 次へ

再生

speed

同時に再生

再生

speed

表示切替

いまいち・・・。　ひとつずつ問題点をクリアしましょう。

過去の練習履歴

CAPT

Computer **A**ssisted **P**ronunciation **T**raining

The Big Players in Japan

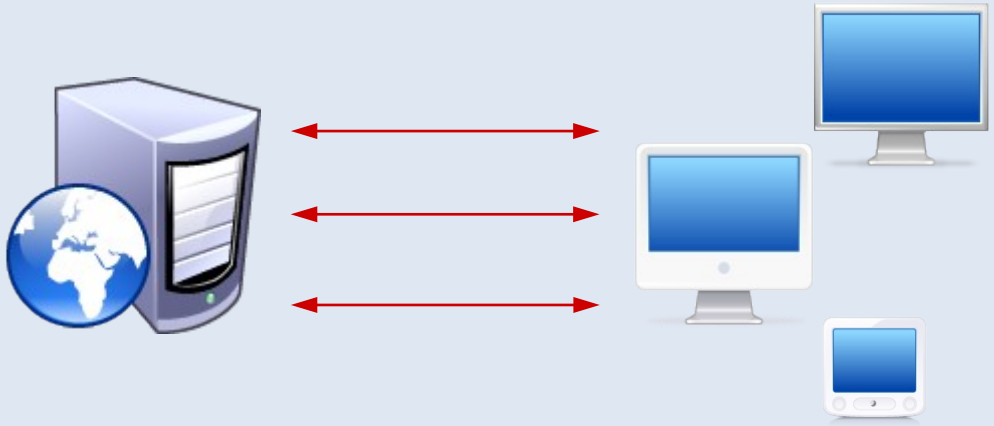
Company	Solution	License (yen)
ATR	ATR CALL	n/a
Advanced Media	Ami Voice Call	15,000
PronTest	発音検定	30,000

CAPT Product Types

- **standalone software**
- relatively cheap
- installed on a desktop
- single user

- **system level solution**
- expensive
- server-client typology
- tie-in sale of hardware

standalone desktop



server **clients**

Structure Design

えーと ...
mappu?

speech data

Say “map” !

You said “mappu”!
Please try again!

trained models

MAPPU

MAP

ASR

?

MAPPU

CAPT System

Section 1.2

Problems
with
Commercial Solutions

Commercial Solutions

- not so popular ↔ contrary to the expectations
- reasons
  1. price → prohibitive
  2. functionality → limited
  3. content → limited
  4. domains → CAPT / ASR mismatch

1. Prices

- prohibitive prices
  (1) classroom use of individual licenses
  15,000 x 40students = 600,000yen

  (2) custom system (+content)
  starts from 1-2 million yen

- not an option for individual teachers
- in many cases even institutes can't afford it

1. Prices

- Models (AM, LM) need to be trained on real data
- huge amount of labeled data is needed

Target Domain	Training Data
1 speaker fixed set of words	1 hour
speaker independent dictation	10,000 hours

- training time:
  - ranges between a few days → a month

high demands on human / computational resources

2. Customization

(1) contents

- each problem needs **(a)** data **(b)** tuning
- contents are hard-wired → can't be changed on the fly

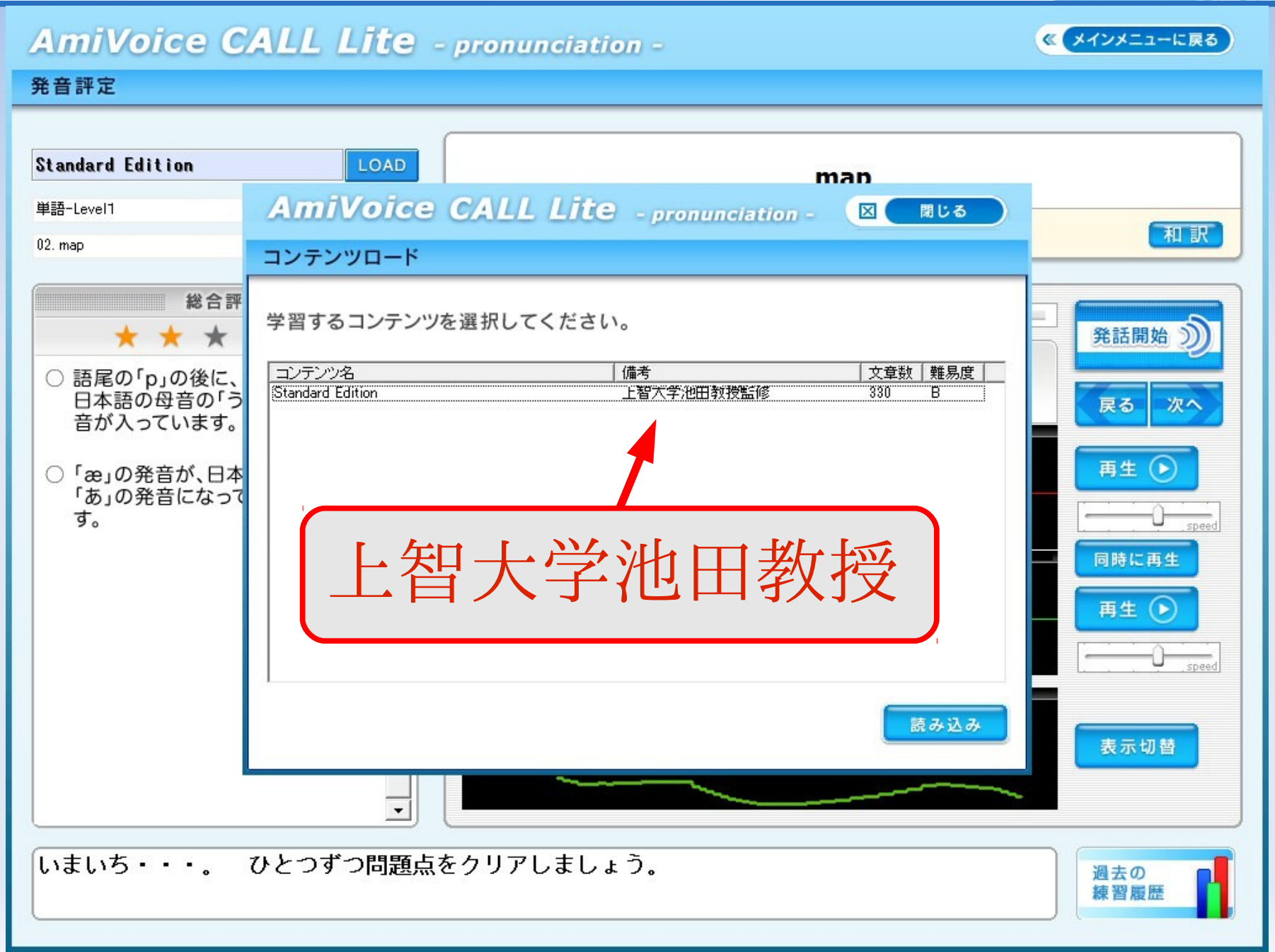
  'light' ↔ 'right' *Boring!*

  'climb' ↔ 'crime' *can I use this instead? → unlikely*

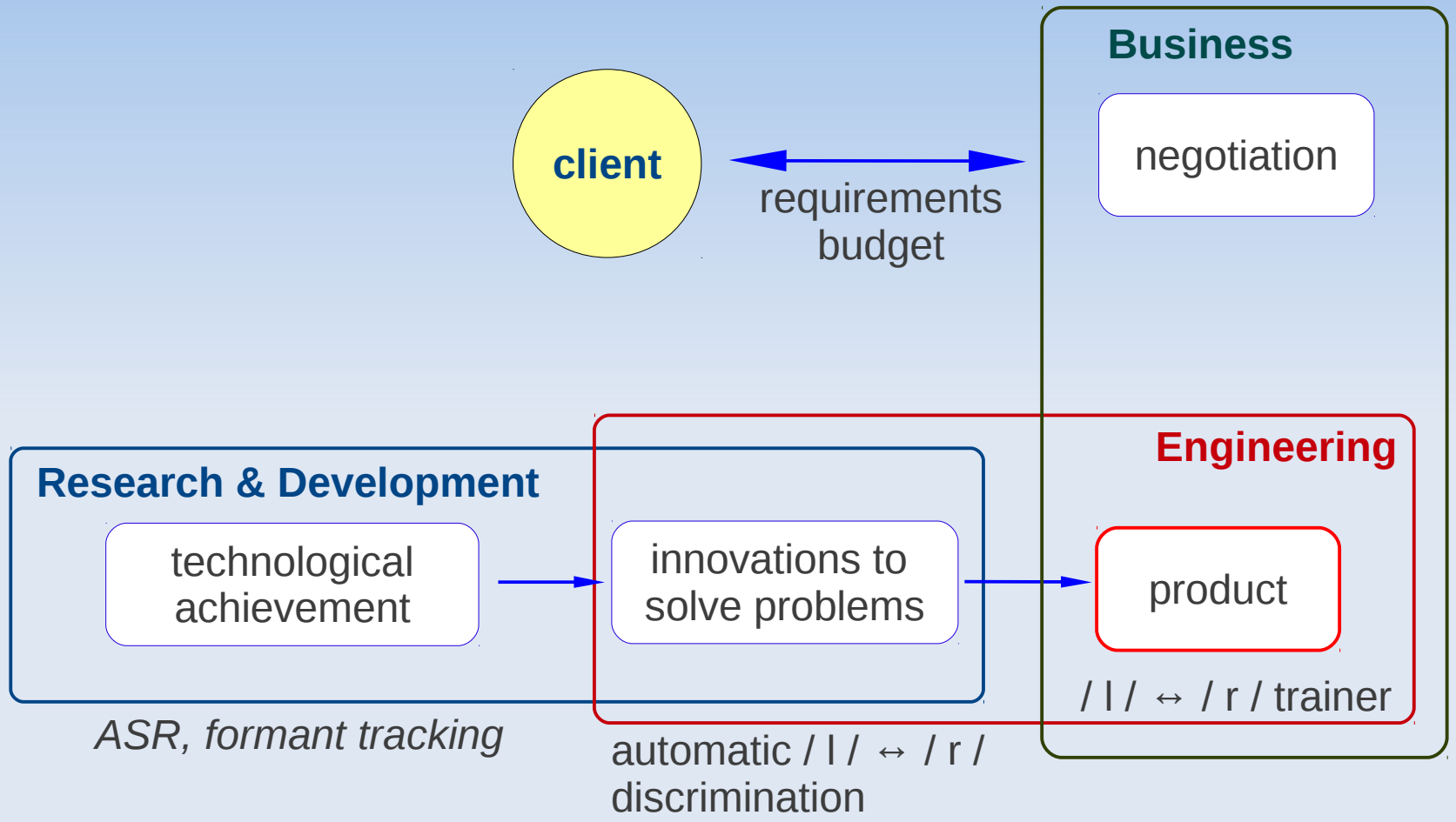
(2) software

- commercial CAPT systems are closed source
- not extensible by the user
- customization through vendors (vendor lock)

Custom Content



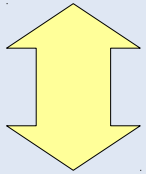
Life-cycle of a Product



- research is expensive, innovative solutions are preferred
- changing *anything* is difficult

3. Problem Domains

- **ASR** is designed to..
  - handle native speakers
  - detect **what** they speak
    → tolerance for variation



- **CAPT** is designed to
  - handle non-native speakers
  - detect **how** they speak
    → has to detect wrong type of variations
    → has to detect error types

Inherent Controversy

	ASR	CAPT
designed for:	native	non-native
recognize:	what ppl speak	how ppl speak

- limitations of design → performance barriers
- CAPT systems
  - mistakes have to be predefined + trained
  - must be tuned separately for each L2
    → separate tuning for Japanese, Spanish... speakers of English

4. Functionality

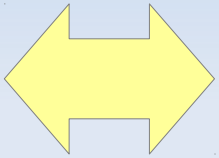
- CAPT relies on traditional ASR
- technological constrains
  - segment recognition (e.g., / l / ↔ / r / detection)
    → inherent to ASR technology
  - prosody recognition (e.g., stress / intonation detection)
    → inherently absent in ASR (cf. preprocessing)

Recognition of..	Innovation Cost	Implementation
segments	low	common
intonation	high	rare / experimental

Conflicting interests

corporate interest

- selling technology (even if not needed)
- minimizing research costs (reuse available components)
- restricting development to a number of platforms (Windows only)
- ensuring customer loyalty (by vendor lock)



educational interest

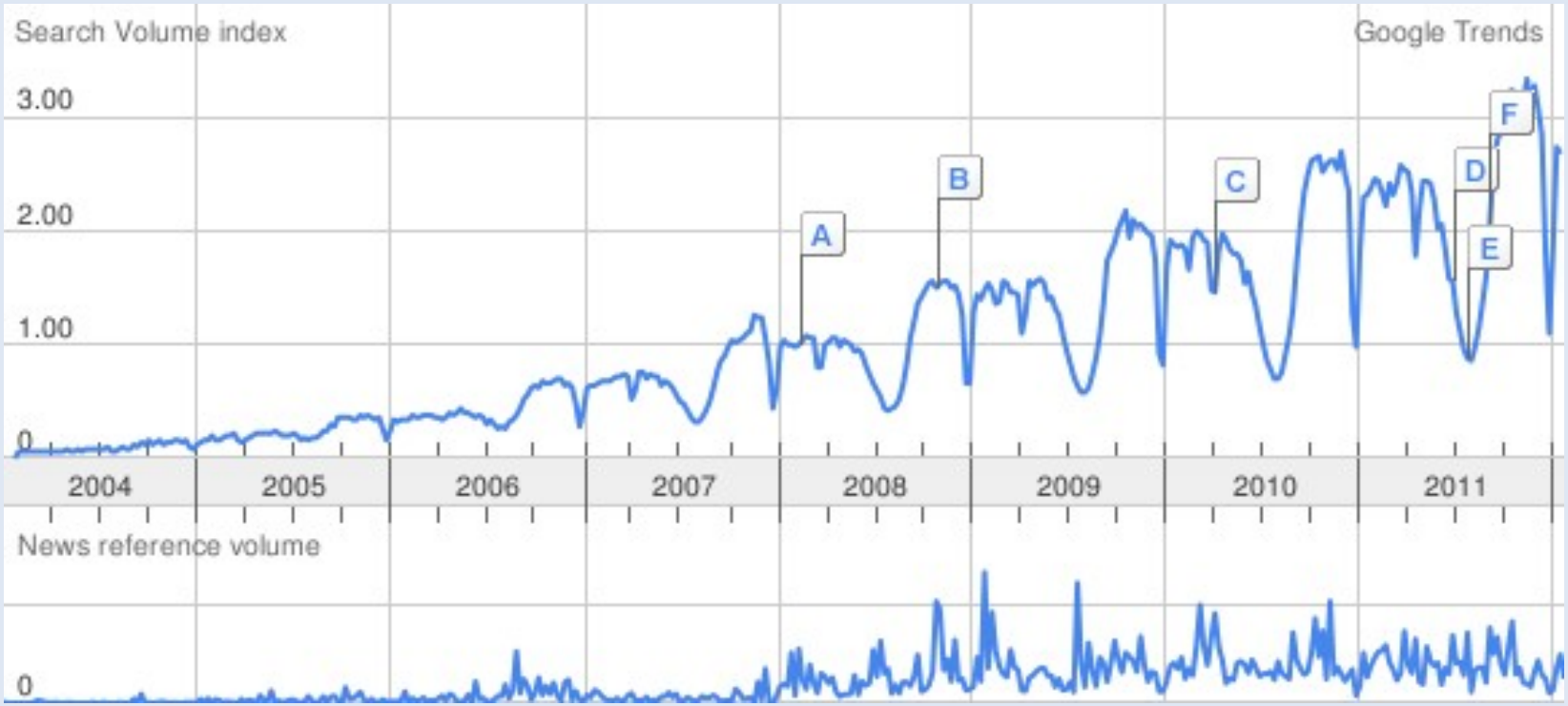
- low prices
- extensibility of
  - functionality
  - contents
- integration with other systems
- availability across several platforms

Section 1.3

Just like **Moodle**?

LMS

- Moodle
  - most popular Learning Management System (LMS)
  - free and open source
  - widely successful



Moodle vs CAPT

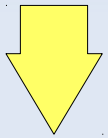
	Moodle	CAPT
publishing contents		
creating assignments	*with plugins*	
collecting student data		*in some solutions*
giving feedback		
authentication authorization		

Open source CAPT?

- Why not imitate Moodle?
- How about a free and open source ASR for pronunciation training?
- limitations
  1. platform
  2. modality
  3. community

1. Platforms

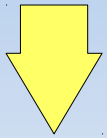
- Moodle lives in the browser
  → massively cross-platform
  even your cell phones browsers!
- developing CAPT <u>for the browser</u>?



- ASR is CPU & memory hungry intensive
  → Do we want to run 100 FFTs / sec in a browser?
- Maintenance
  → JavaScript ASR? *cf. existing frameworks: HTK, Sphinx, Julius →compiled languages*

1. Platforms

- develop native application?



- for which platform?
  → increasing number of platforms

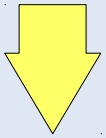
Microsoft	Win7 vs. Win8 vs. WinRT
Google	Android phone vs tablet
Apple	iOS vs. MacOS
Linux	< a complete mess >

- limited human resources

2. Modality

- **text** → easy (works out of the box)
- **sound** → complicated

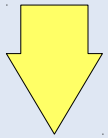
  Ever had a malfunctioning sound on your PC?
  Ever had malfunctioning text? (besides char encoding)
- **recording** → hardware / platform specific
  - browsers / HTML / JavaScript → no recording
  - sound drivers
    - → native code
    - → virtual machine (JVM, Flash)
  - error prone



huge **disadvantage** for sound-based systems

3. Developer base

- **Moodle** → wide developer base
  - common problem: content management
  - common solution: Linux + Apache + MySQL + PHP (**LAMP**)
    → easy to get involvement in the development
- **ASR / CAPT** → narrow developer base
  special mix of expertise required
  - linguistics: phonetics, phonology
  - digital signal processing
  - statistics
  - programming



potential developer community is rather thin

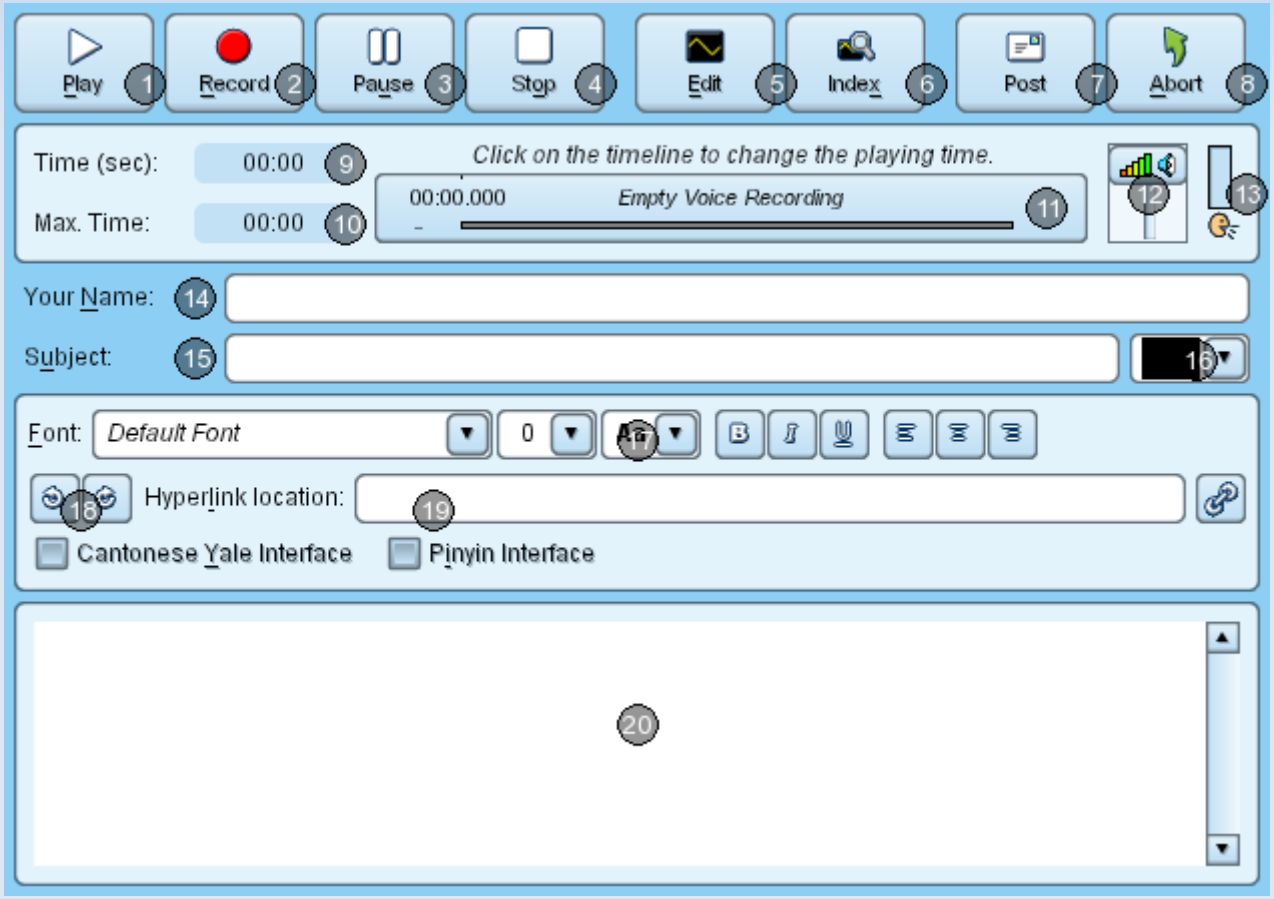
Section 1.4

Existing Solutions

The Gong Project

- great software by

  Department of Computer Science and Engineering
  Hong Kong University of Science and Technology



- server-client sound exchange
- various activities
- Moodle plugin

Roadmap

0. Self Introduction
1. Background & Problems
    1. ASR + CALL = CAPT
    2. Problems & limitations
    3. Just like Moodle?
    4. Existing solutions
2. Yet Another Project
    1. Project overview
    2. Bolts and nuts
    3. Data
    4. What comes next?

Section 2.1

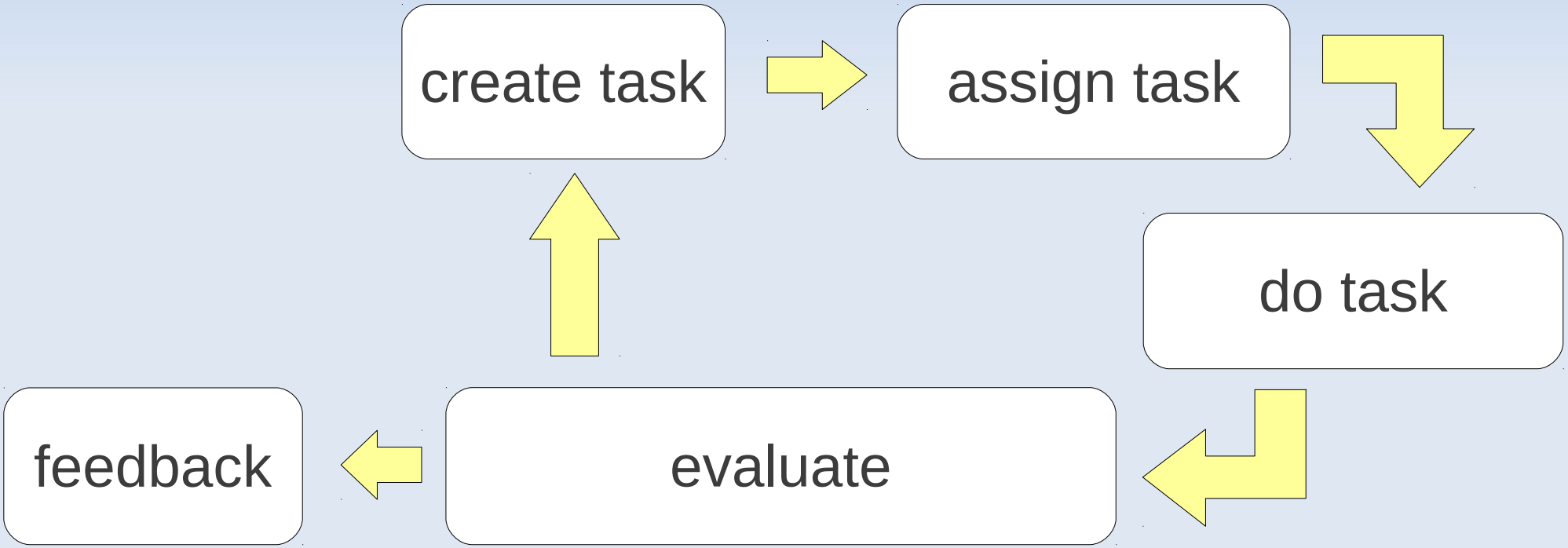
Project Overview

Idea

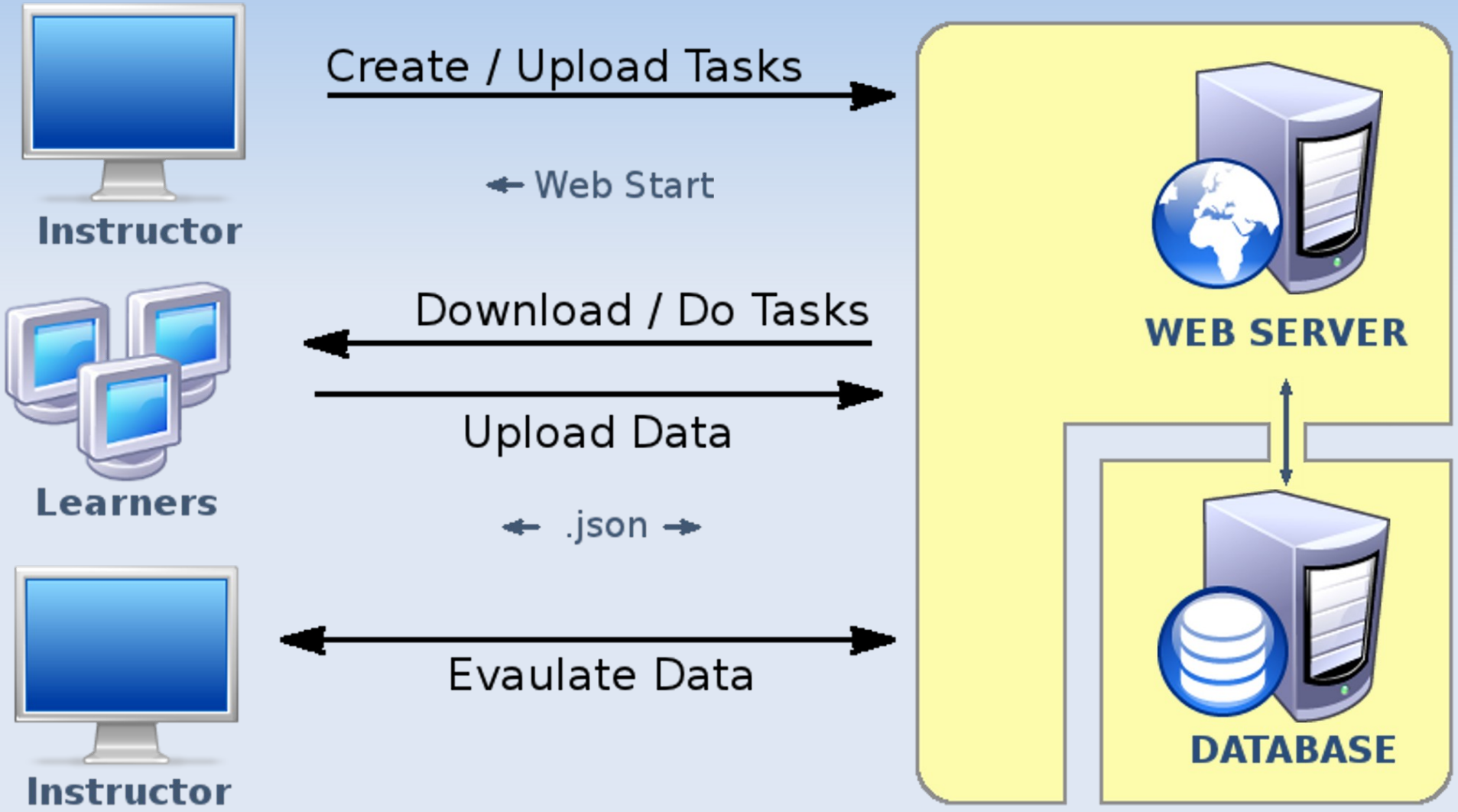
- for academic/educational use
- user-oriented, user-friendly
- sound exchange + evaluation system
- can be used without any ASR
- customizable / extandable / reusable
  → modular
  → open source
- ASR can be added later!

Framework Design

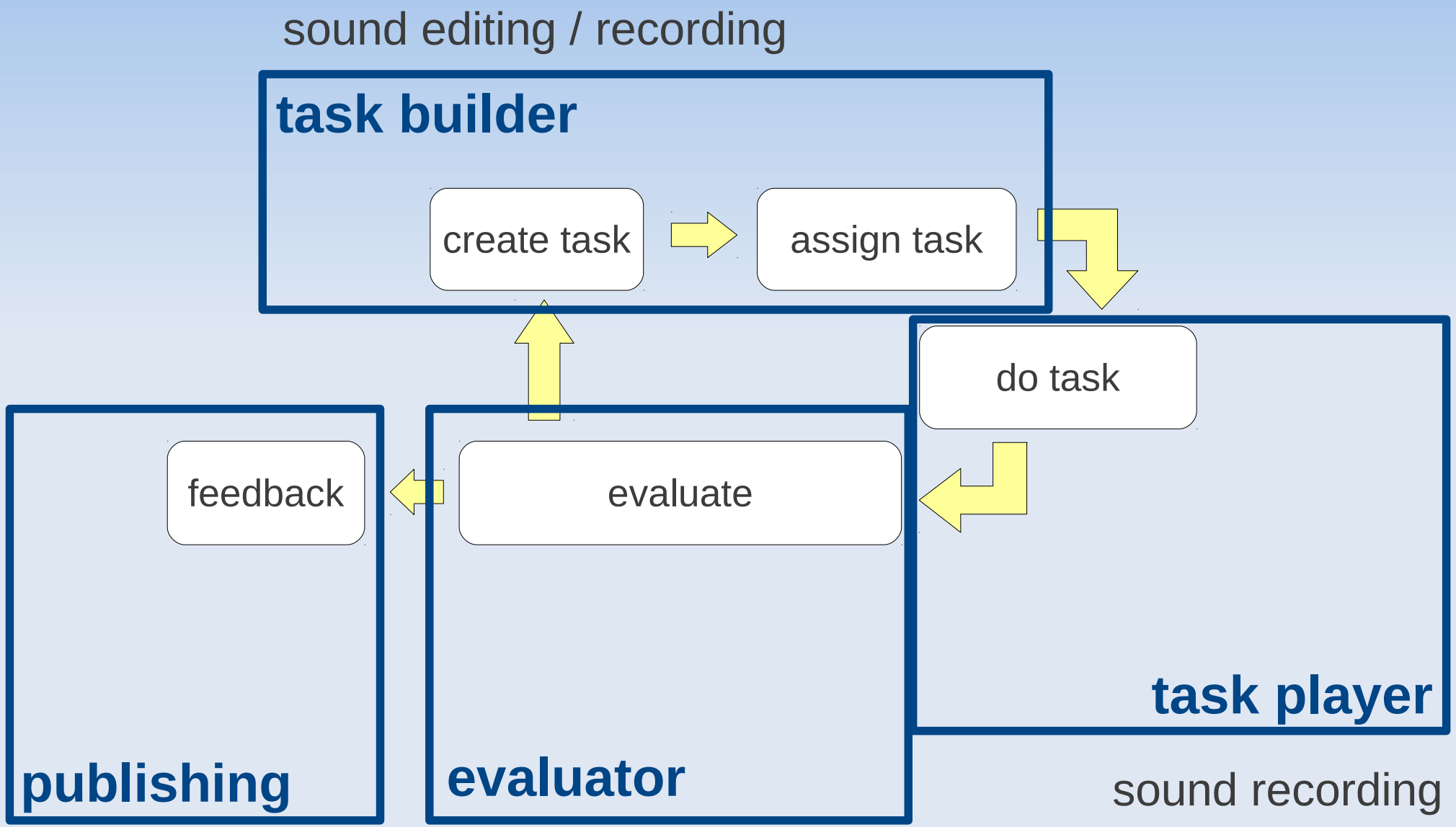
- user generated content (cf. web 2.0)
- main focus on **functionality:** student-instructor interaction



Dataflow


Instructor
Create / Upload Tasks
Web Start
Learners
Download / Do Tasks
Upload Data
.json
Instructor
Evaulate Data
WEB SERVER
DATABASE

Components


sound editing / recording
task builder
create task
assign task
do task
feedback
evaluate
task player
publishing
evaluator
sound recording
availability
handling big data

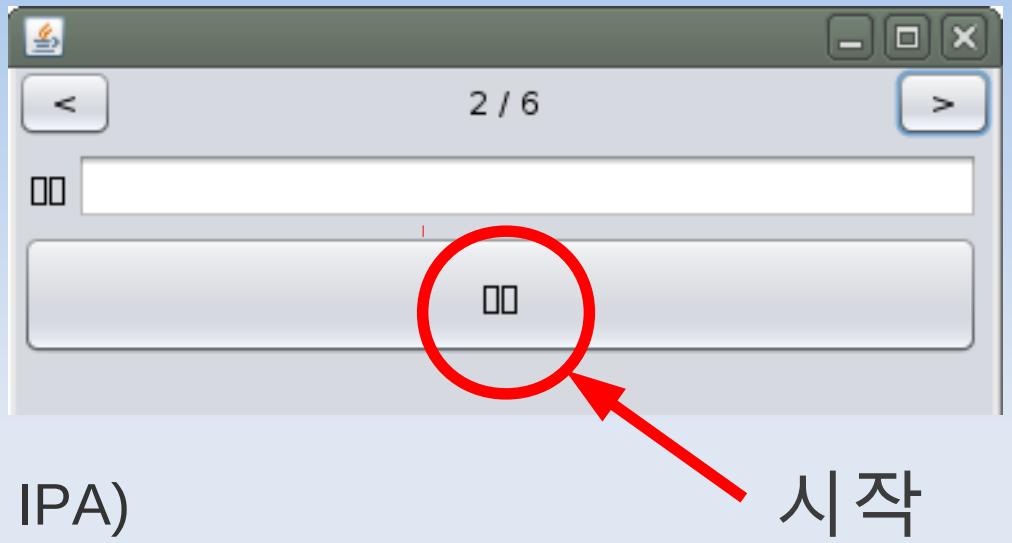
Section 2.2

Bolts & Nuts

Task Player

Technical details

- Java + **Swing**
  → runs on desktop
- alternative: **SWT**
  → better widgets
  → can't embed fonts! (e.g., IPA)



- Sound: **OpenAL** (LWJGL wrapper)
  → Sun's reference implementation is too simple
- deployment: **Java Web Start**
  → runs on desktop: where JRE is available

Server Side

Web server

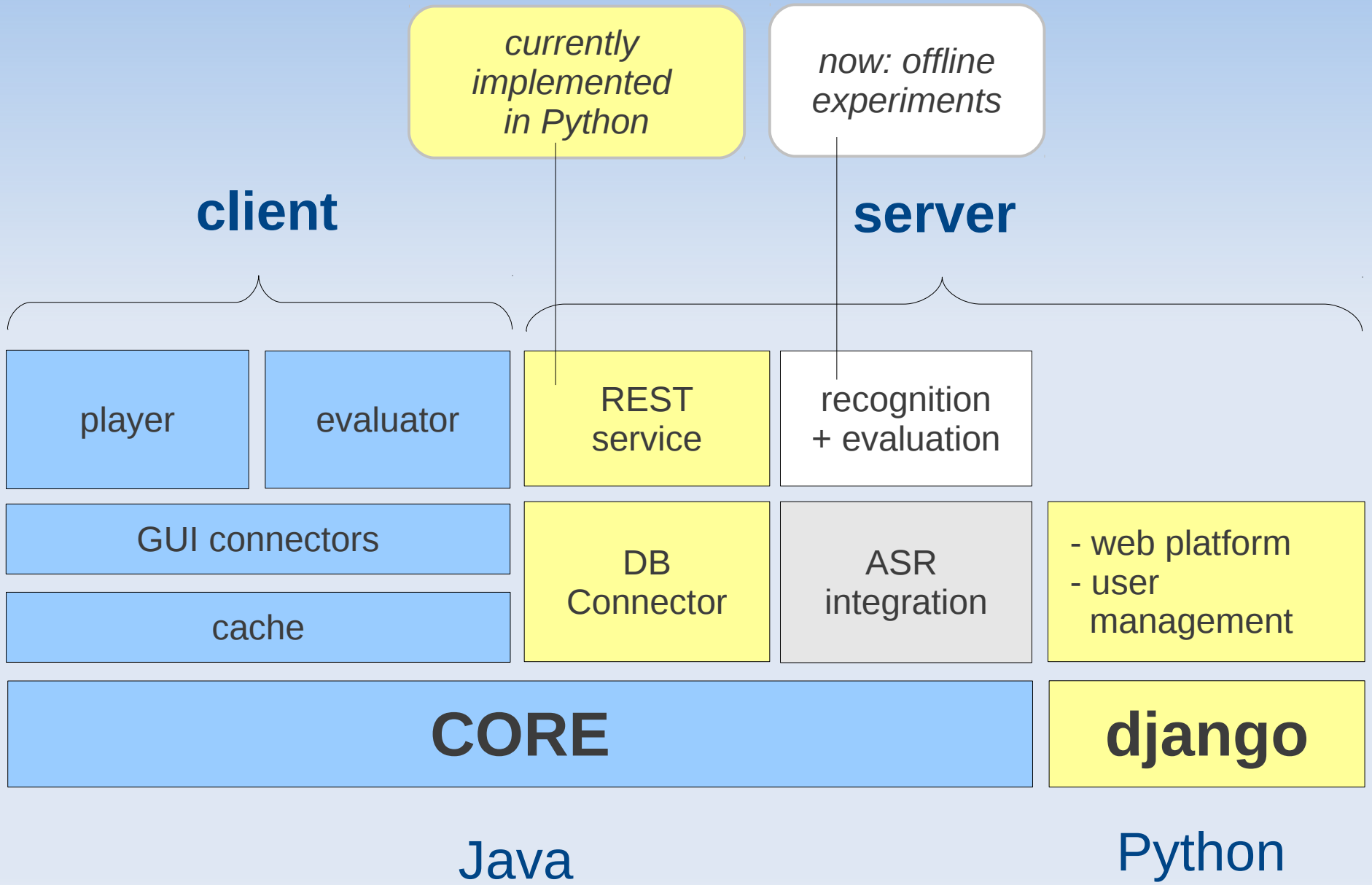
- **django™**
- functions as a CMS
- user authentication
- non-speech MySQL backend
- speech RESTful services
- Python → Jython (plan)
- Jython + Sphinx modules → ASR functionality

.json

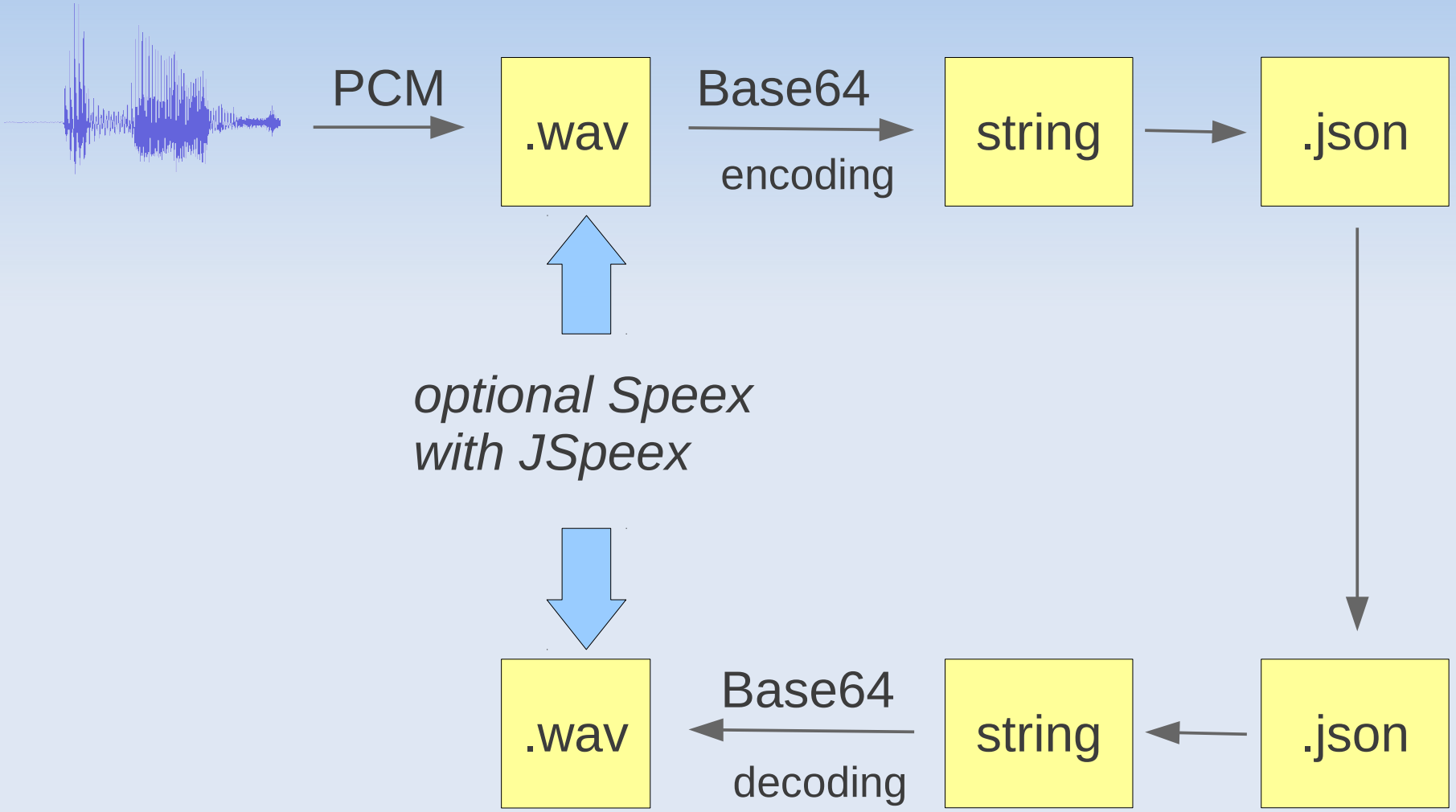
DB server

- **MongoDB™**
- NoSQL → fits development phase
- .json format storage
- data:
  - student response (mainly sound)
  - evaluation

Code base


currently implemented in Python
now: offline experiments
client
server
player
evaluator
REST service
recognition + evaluation
GUI connectors
cache
DB Connector
ASR integration
- web platform
- user management
CORE
django
Java
Python

Transferring Sound



Section 2.3

Results

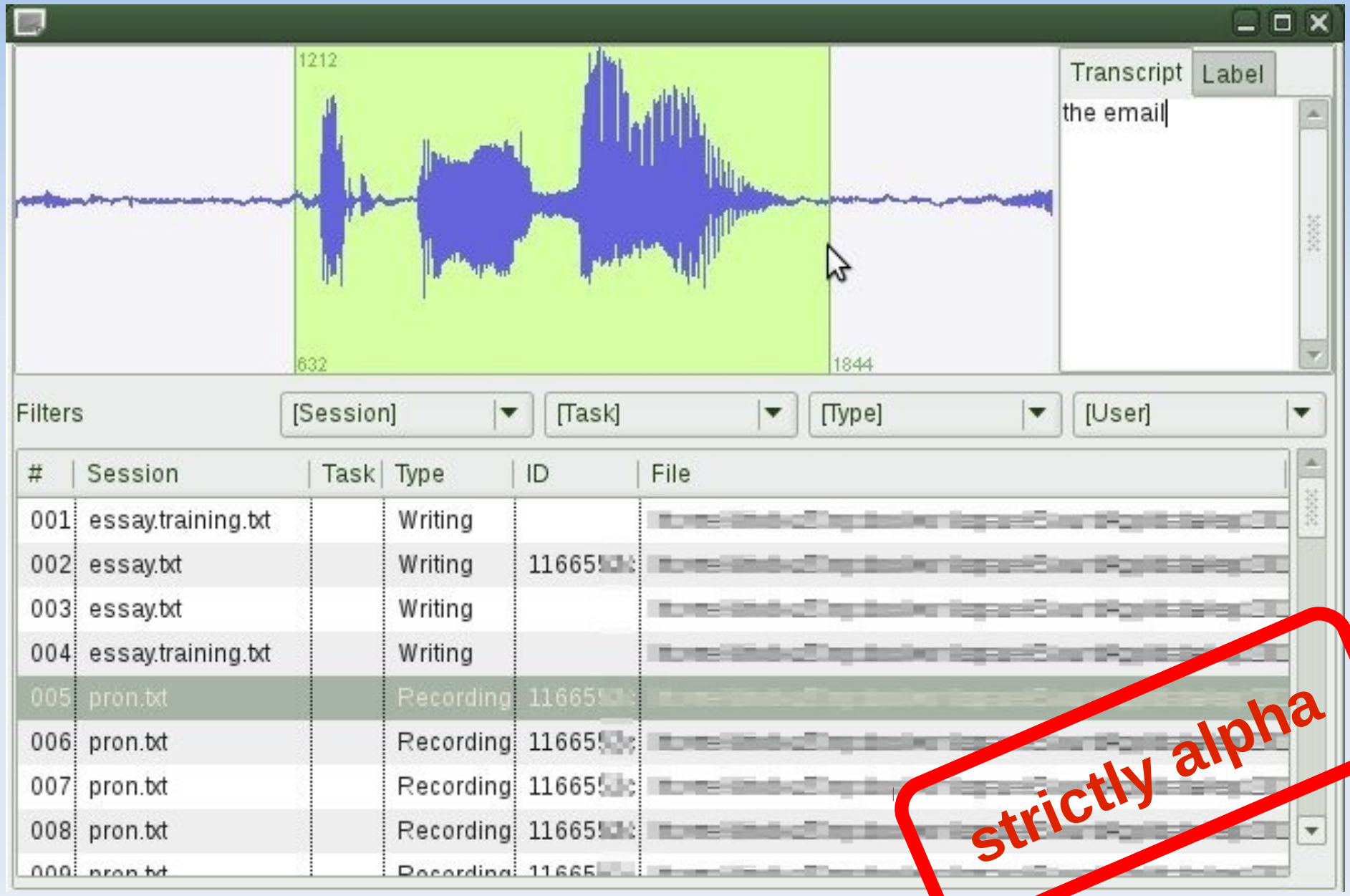
Task Player

- sound recording over the web

Customizable features

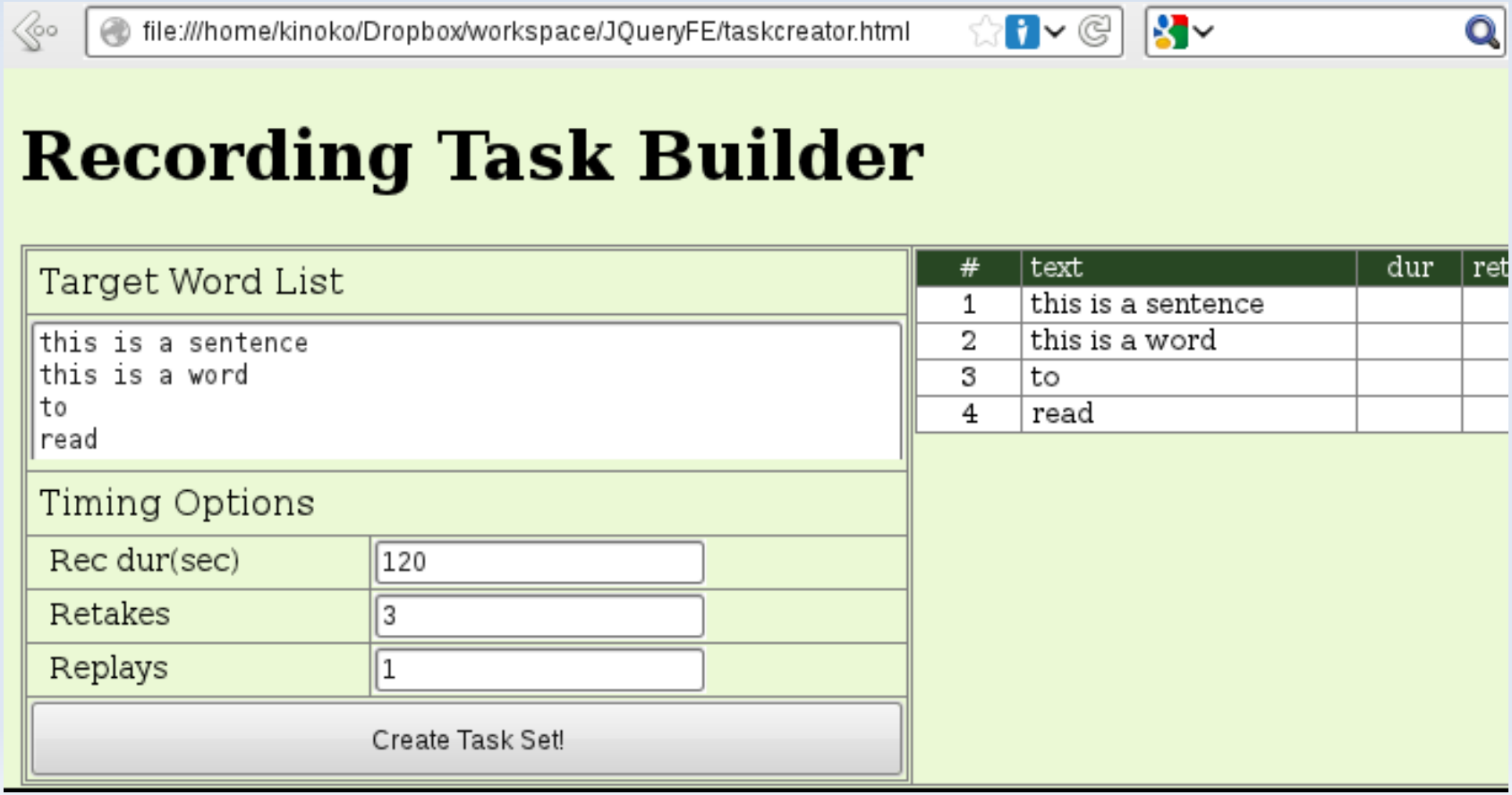
- pictures / sounds
- timed start / stop
- max duration
- number of replays / recordings
- simultaneous play & rec (for shadowing)

Task Evaluator



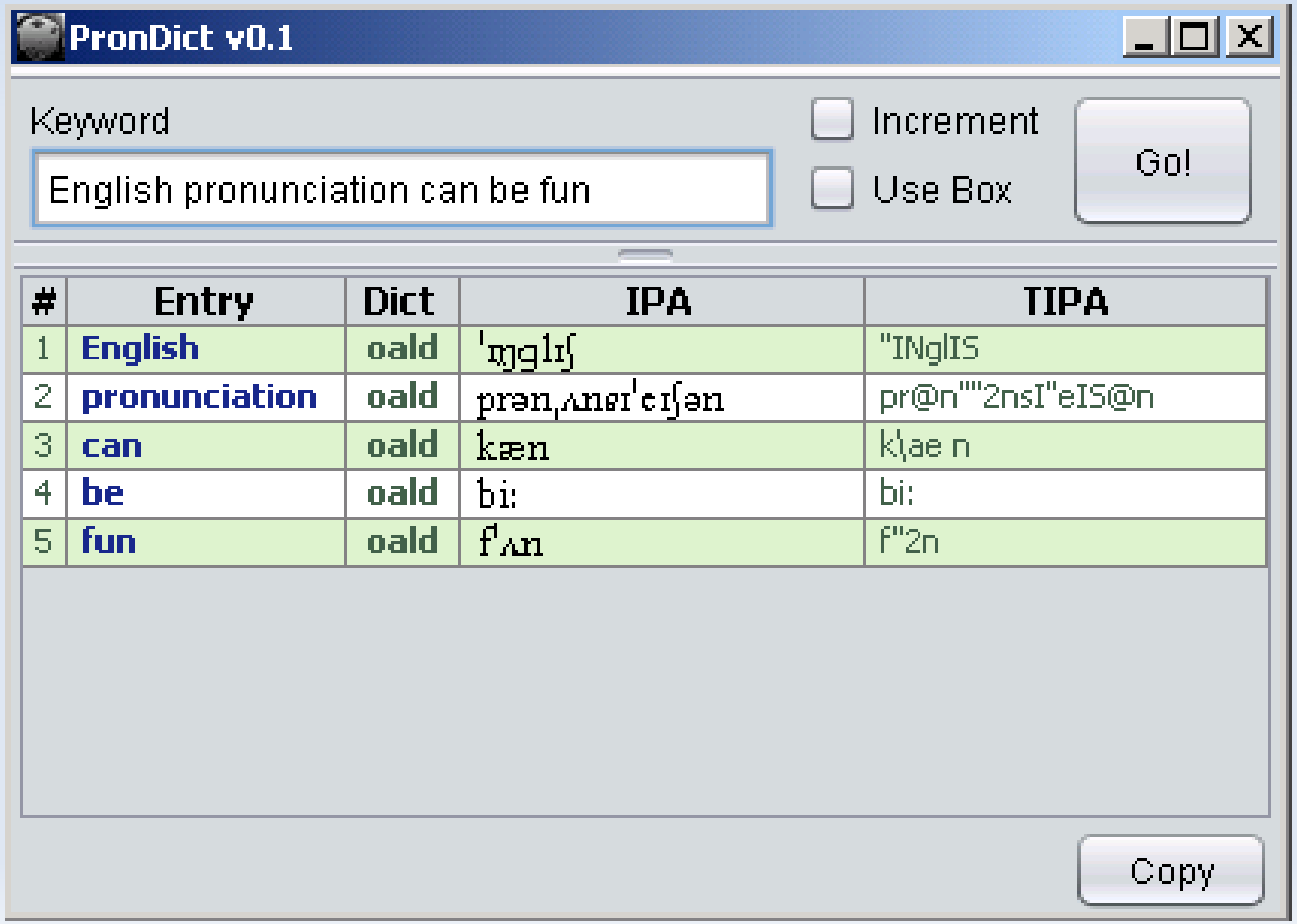
Task Builder

- via Java source code (present practice)
  - but not recommended
- via web browser (HTML + JavaScript)
  - under development



Helping Tools

- pronunciation lookup
- to help creating course material
- dictionary: OALD (+CMU in latest versions)



In Action

- use cases (2011-2012)
  - university conversation classes
  - pre/post tests for US study tour
  - phonology experiment: local + overseas (Korea)

Learners' Speech Corpus

- over 120 students
- Japanese learners of English
- over two semester
- in CALL rooms
- 100 hours of data collected
  - 30 hrs direct speech
  - 70 hrs free speech
- transcription: in progress

Analysis

- Sphinx 4 with
  WSJ (Wall Street Journal) acoustic model
- freely available (several versions)
  - frequencies

sampling	16,000 Hz
min	130 Hz
max	6800 Hz

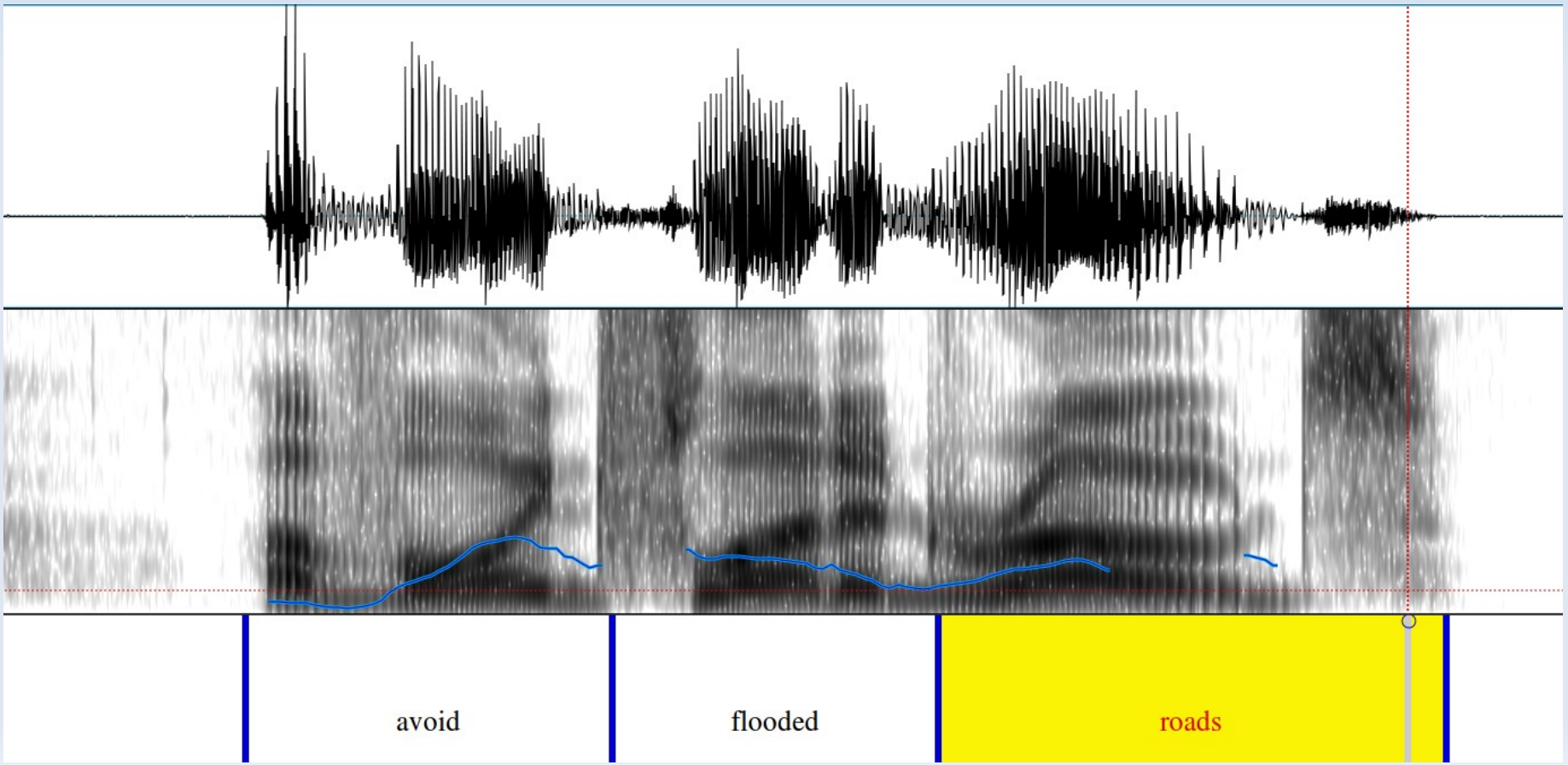
  - vector length: 39
  - Gaussians: 8

- *Speech Corpus: DARPA Spoken Language Program, 1991, read texts from Wall Street Journal news*
  → frequently used for evaluation ($1,500)

Native Data

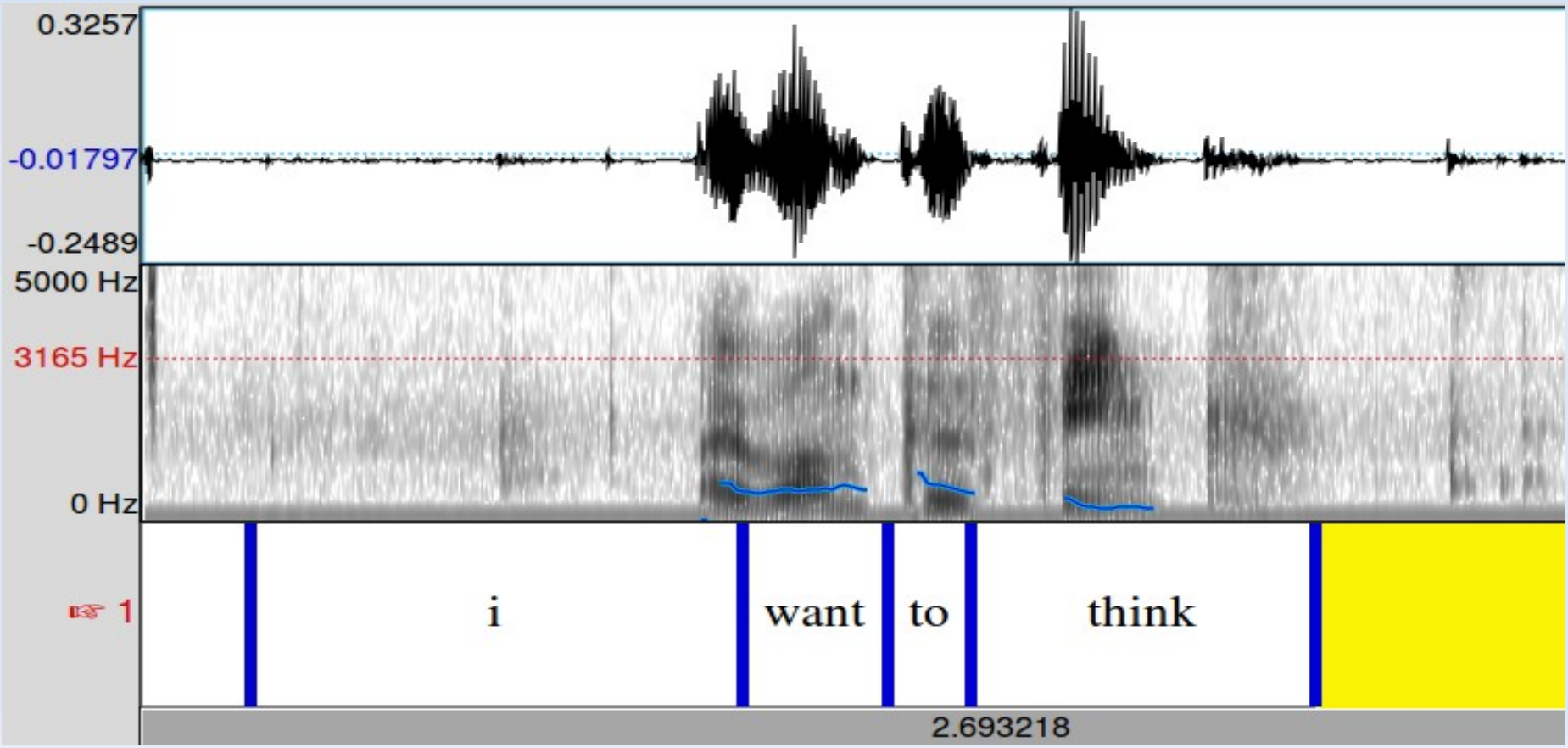
- textbook audio
- native data
- forced alignment

- high accuracy
- small misalignments
  → frame-size effect



Learners' Data

- learners' data
- CALL environment
- low voice + high noise
- apparent misalignments
- frame-size effect



Encountered Problems

- recording speech (handling devices, buffers)
- GUI design + threads
- graphics for sound waves
- embedding fonts
- data caching: memory / local / remote
- deployment (Java Web Start)
- Cross-Site Request Forgery (django's csrf token)
- Sphinx 4 versions
  → use the latest builds!

Project Schedule

- Phase 1 : **2011 – 2012**
  - technological backbone : sound exchange
  - client-server communication
- Phase 2 : **2012 – 2013**
  - increase user experience
  - gain user base
- Phase 3 : **2013 – 2014**
  - adding automatization, training learners' AM
  - releasing source code and API for the public

Resources

- Slides & other presentations

  `www.pinlab.info/talks`

- Blog: technical problems related to the project
  → with solutions

  `www.pinlab.info/blog`

- CMU Sphinx 4
  - ASR related lectures, tutorials, resources

  `www.cmusphinx.sourceforge.net`

Work in progress