Introduction to the Kaldi Toolkit

Compiled from:

- Dan Povey, Speech recognition with Kaldi lectures, http://www.danielpovey.com/kaldi-lectures.html
- 高予真, An Introduction to the Kaldi Speech Recognition Toolkit, http://berlin.csie.ntnu.edu.tw/Courses/Speech%20Recognition/Lectures2013/SP2013F_Lecture14-Introduction%20to%20the%20Kaldi%20toolkit.pdf
- 篠崎隆宏, Kaldiツールキットを用いた音声認識システムの構築 - 東京工業大学, http://www.ts.ip.titech.ac.jp/demos/csjkaldisp2016oct.pdf

and many other sources.

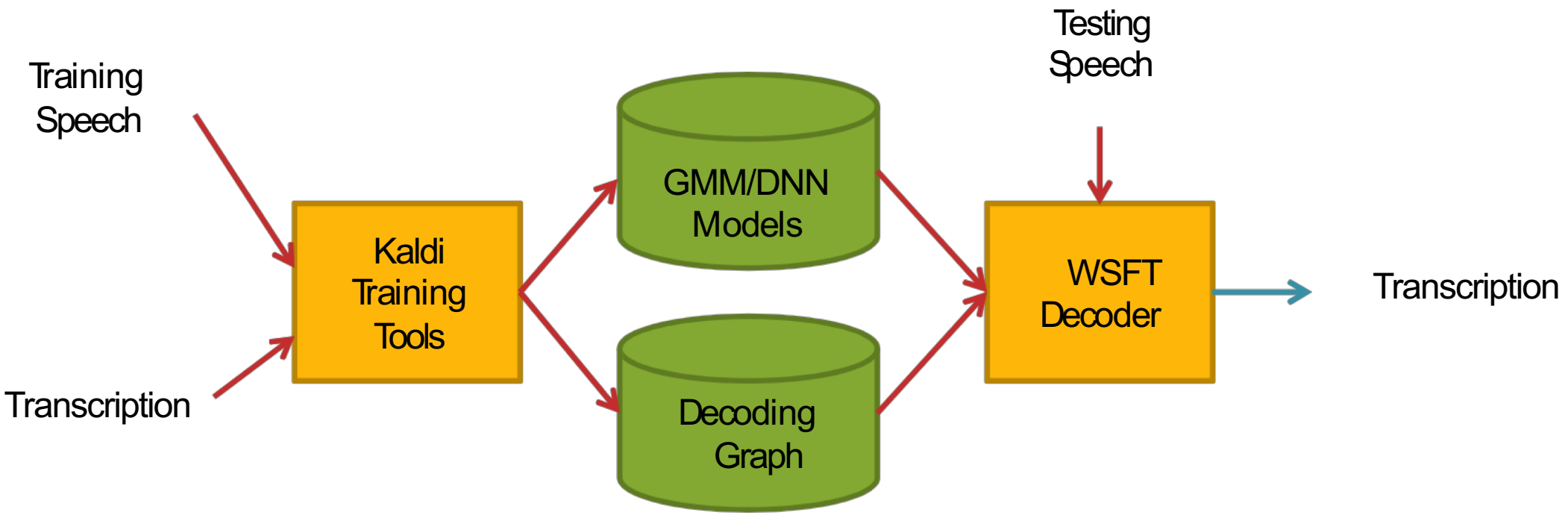
Yuan-Fu Liao
National Taipei University of Technology
yfliao@ntut.edu.tw

Outline

- Some Background
- Our Systems & Applications Implemented using Kaldi
- Overview of Kaldi Features

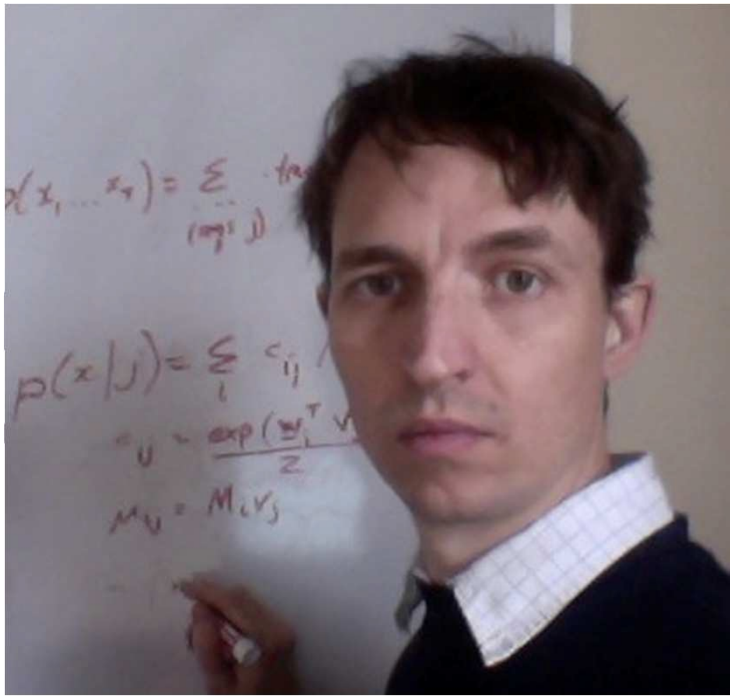
The Kaldi Toolkit

- Kaldi is specifically designed for speech recognition research application
  - very gracious and mature.



The Kaldi Toolkit

- Kaldi: an Ethiopians discovered the coffe



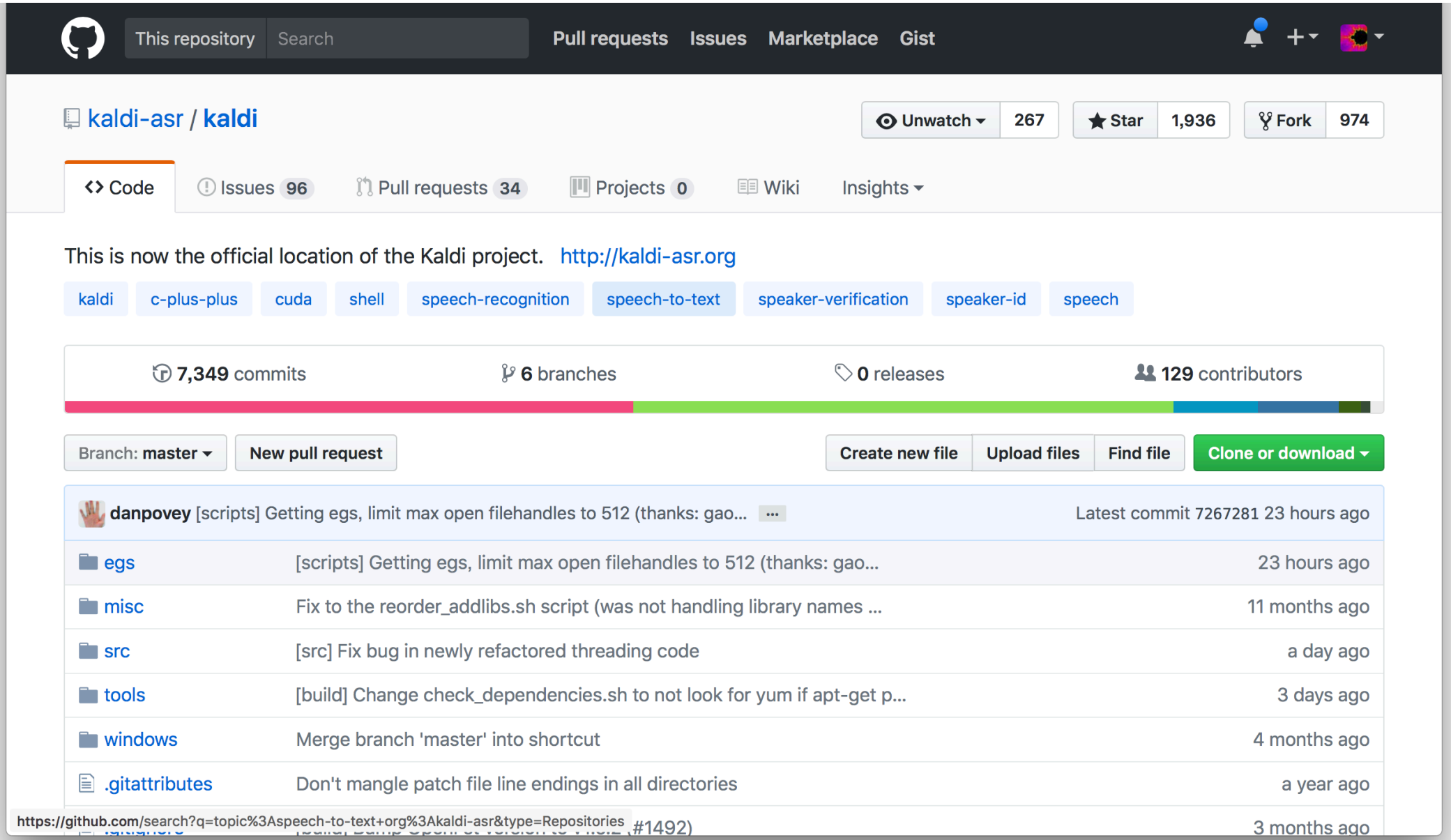
- A WFST-based speech recognition toolkit written mainly by Daniel Povey
- Initially born in a speech workshop in JHU in 2009, with some guys from Brno University of Technology

Kaldi Today

A community of Researchers Cooperatively Advancing STT

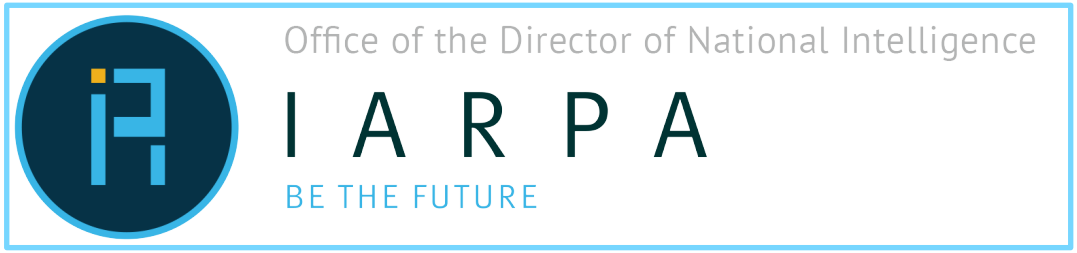
- C++ library, command-line tools, STT "recipes"
  - Freely available via GitHub (Apache 2.0 license)
  - Documentation and example scripts
- Top STT performance in open benchmark tests
  - E.g. NIST OpenKWS (2014) and IARPA ASpIRE (2015)
- Widely adopted in academia and *industry*
  - 300+ citations in 2014
  - 400+ citations in 2015
  - Used by several US and non-US companies
- Main "trunk" maintained by Johns Hopkins
  - Forks contain specializations by JHU and others

https://github.com/kaldi-asr/kaldi



Co-PI's, PhD Students and Sponsors

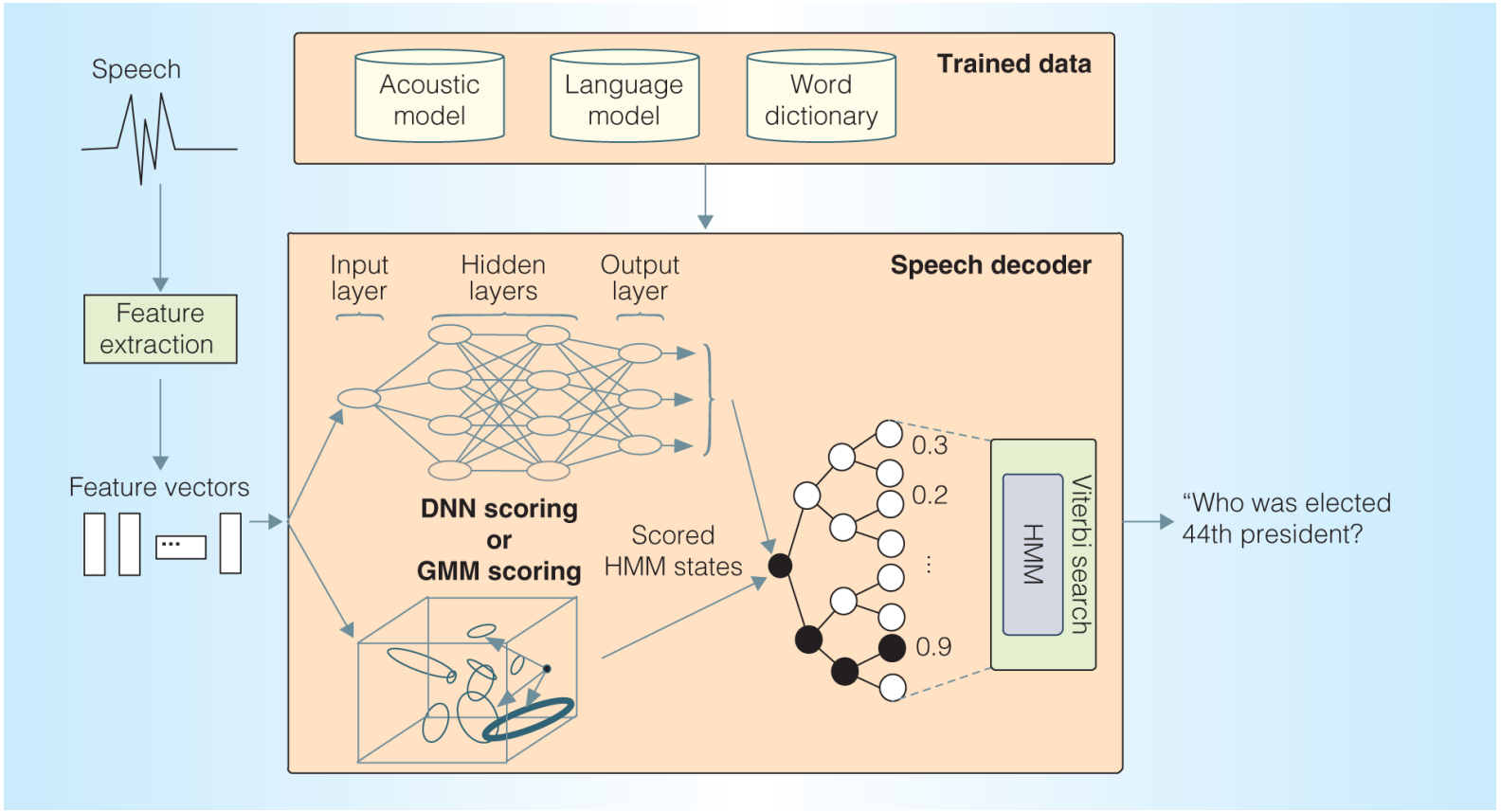
- Sanjeev Khudanpur
- Daniel Povey
- Jan Trmal
- Guoguo Chen
- Pegah Ghahremani
- Vimal Manohar
- Vijayaditya Peddinti
- Hainan Xu
- Xiaohui Zhang
- and **several others**

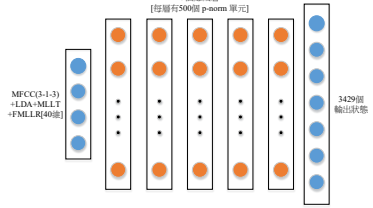


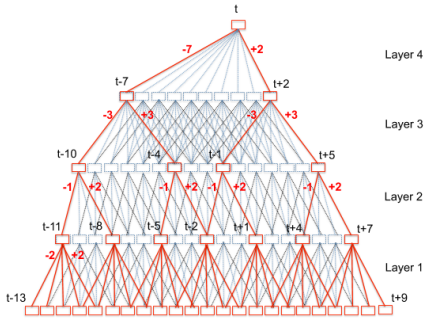
Outline

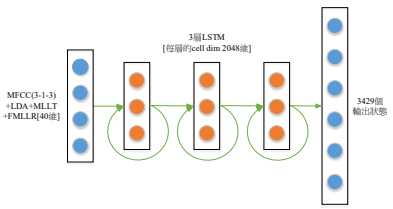
- Some Background
- Our Systems & Applications Implemented using Kaldi
- Overview of Kaldi Features

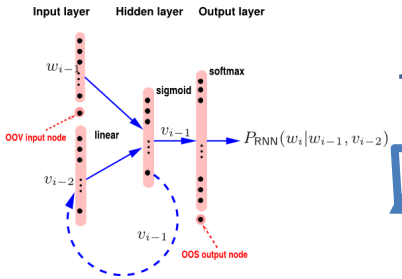
Evolutions of Our Multilingual ASR using the Kaldi Toolkit

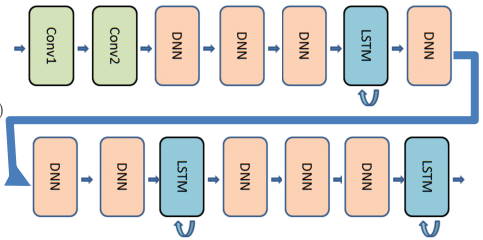








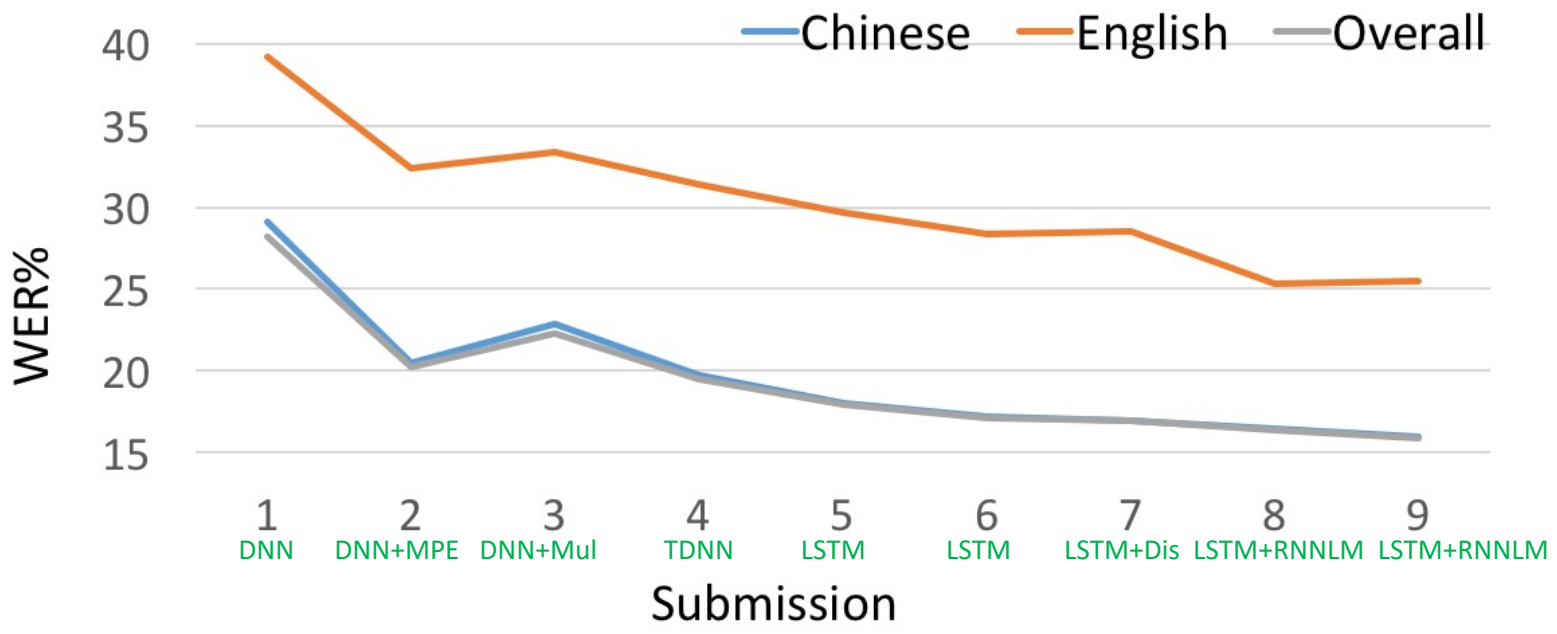




DNN TDNN LSTM RNNLM CNN+DNN+LSTM

Performance of Our Systems on OC16 MixASR-CHEN Challenge Extended Submission

- Multilingual (Chinese+English) ASR

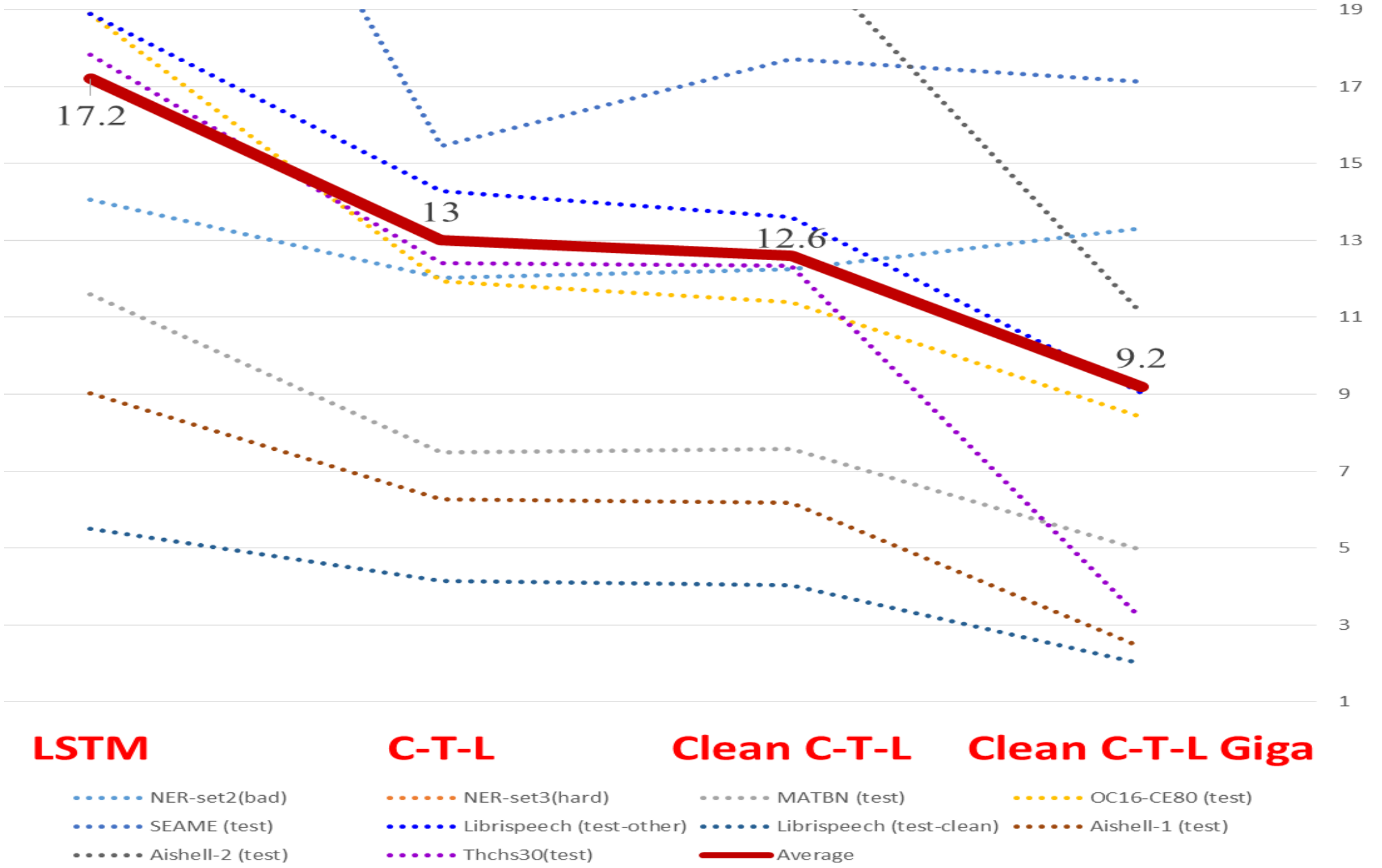


Prof. Yuanfu Liao from National Taipei University of Technology*!*

Overall performance Rank2

English performance Rank1

Performance of Our Systems on various Databases (CER in %)

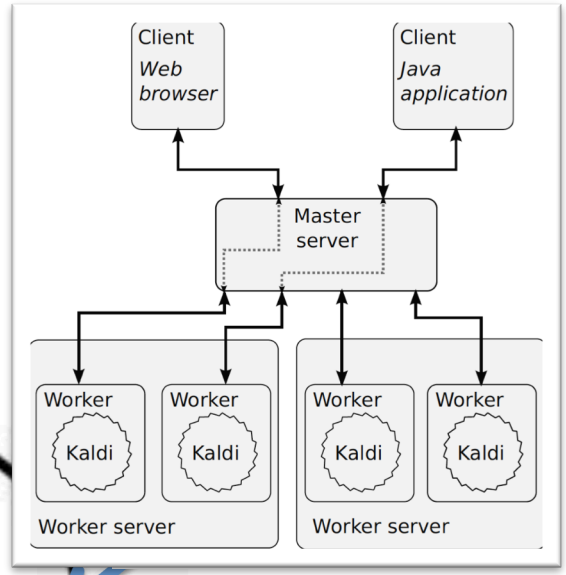


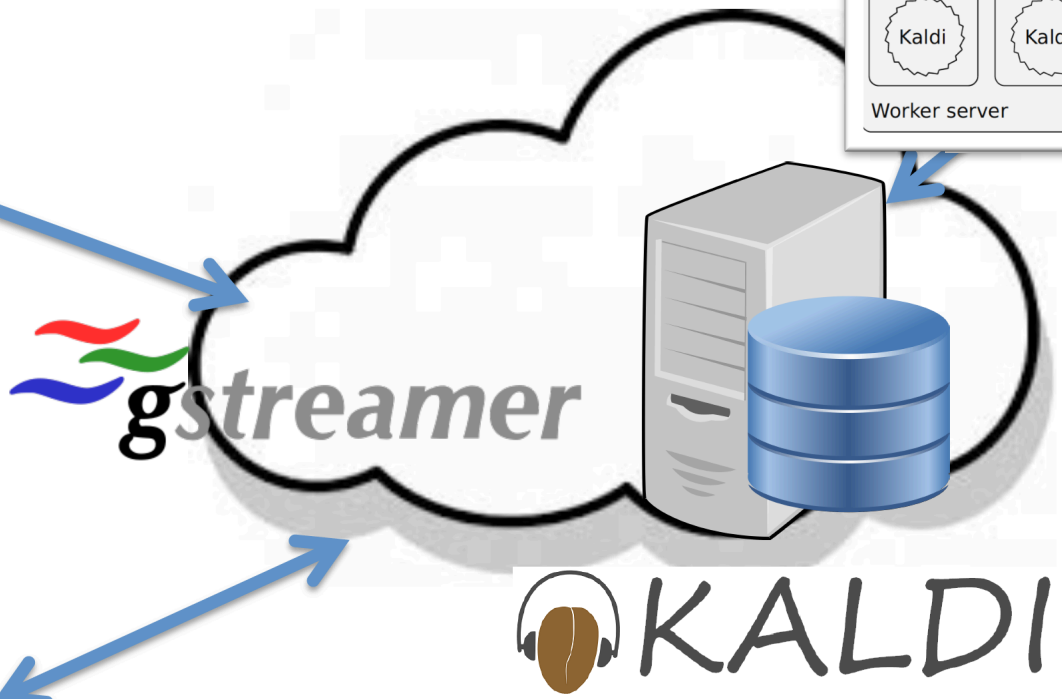
Applications Framework

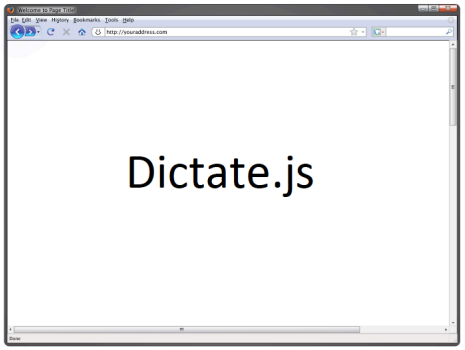
- Kaldi/GStreamer Server

Kõnele app

https://github.com/Kaljurand/K6nele



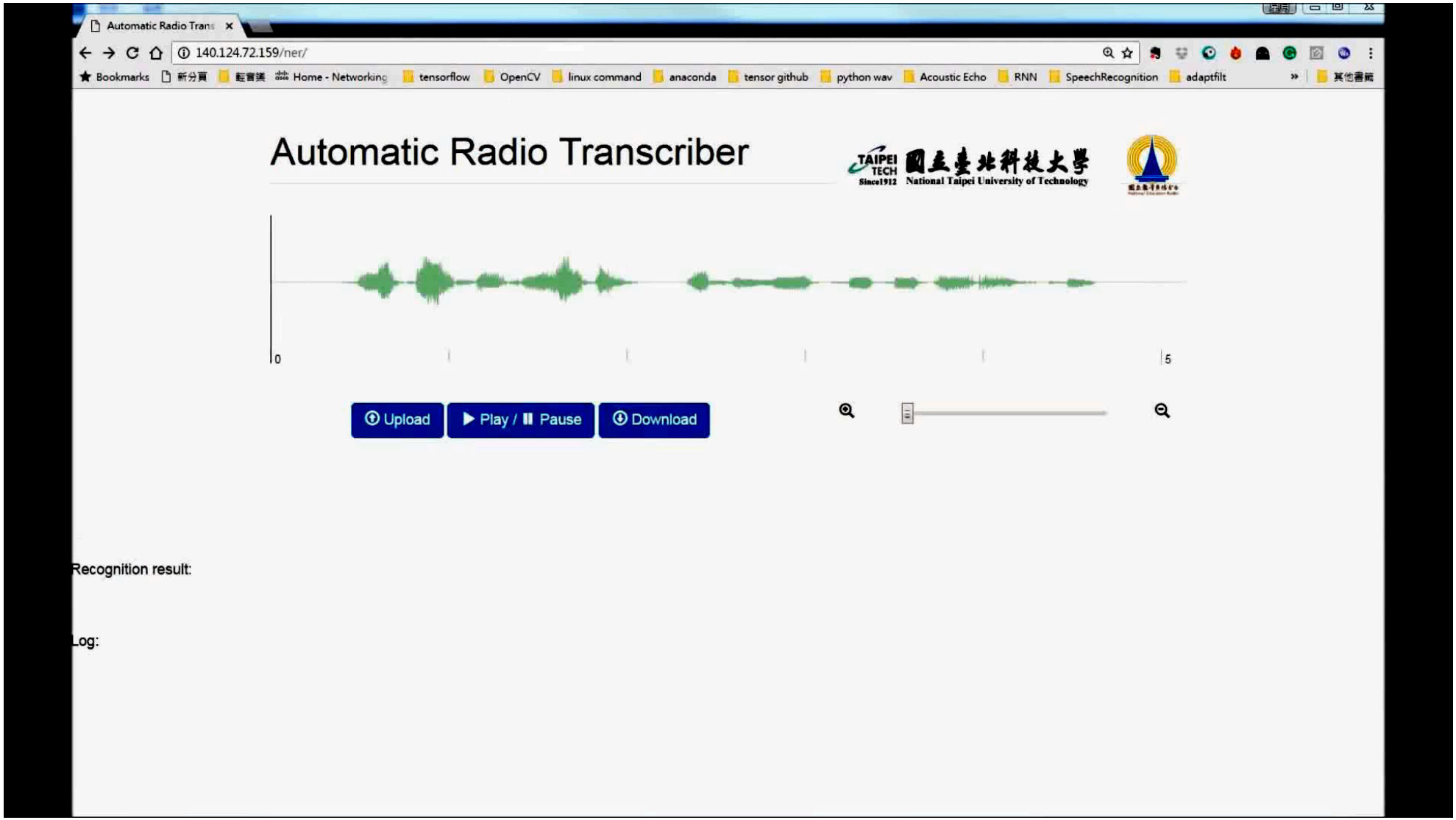




https://github.com/alumae/kaldi-gstreamer-server

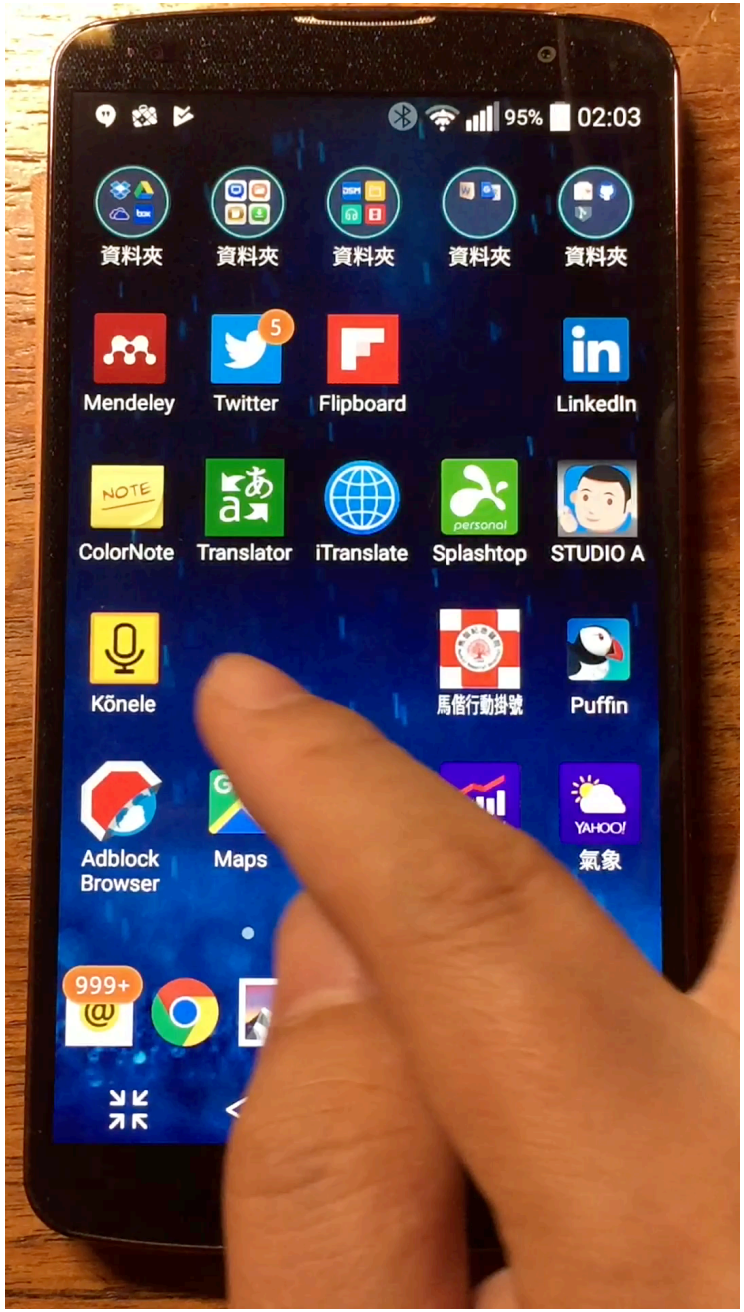
Demo – Automatic Radio Transcriber

- http://140.124.72.159/ner



Demo – Web Search Query

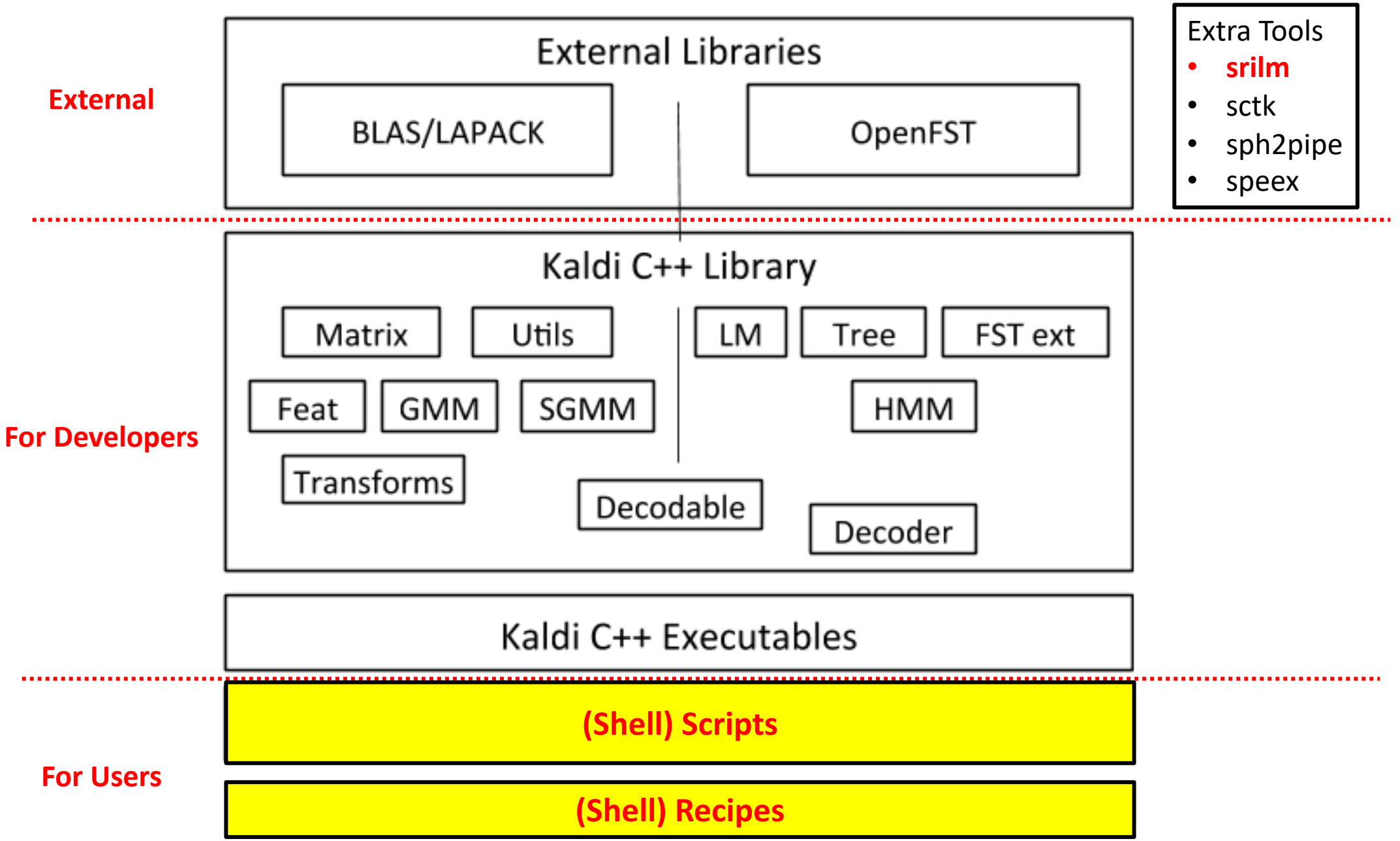
- WebSocket
  - ws://140.124.72.159:8804/client/ws/speech4
- Http
  - http://140.124.72.159:8804/client/dynamic/recognize4



Outline

- Some Background
- Our Systems & Applications Implemented using Kaldi
- Overview of Kaldi Features

Kaldi Dependency Structure (approx)



Kaldi I/O

- Pipelines are heavily used
  - extended filenames: “-”, “gunzip –c foo.gz|”, “/offset/into/file:12345”
- Save various files in Kaldi proprietary ark format
  - generic mechanism to index objects by strings
  - typically map from “utterance id” to “something”
- Uniform handling of input and output by Rspecifier / Wspecifier

Reading data: Rspecifier

rspecifier	meaning
ark:foo.ark	Read archive file foo.ark
scp:foo.scp	Read the script file foo.scp
ark:-	Enter from standard input
ark:gunzip –c foo.ark.gz|	Load compressed foo.ark.gz

Writing data: Wspecifier

wspecifier	meaning
ark:foo.ark	Write the archive to foo.ark
scp:foo.scp	Write script to foo.scp
ark:-	Output in standard output (in binary format)
ark,t:-	Output to standard output in text format
ark,t:|gzip –c > foo.gz	Compress text format output and write it to foo.gz
ark,scp:foo.ark,foo.scp	Write in both formats of archive script at the same time

General Properties of Kaldi

- A C++ library of various speech tools
- The command-line tools are just thin wrappers of the underlying library

```
gmm-decode-faster --verbose=2 \
    --config=conf/file \
    --print-args=true \
    --acoustic-scale=0.09 \
    model.mdl \
    ark:decoding_graph.input \
    scp:feature.input \
    ark:text.output
```

Standard Arguments

Application-specific Arguments

Input/Output Files

Example Scripts

- Many scripts and Scripts quite complex
- Much of the configurability of Kaldi takes place at shell-script level
- This helps keep the C++ code simple
- Note use of pipes: features, alignments etc. are passed through pipes.

Scripts (example fragment)

```
#!/bin/bash
…
while [ $x -lt $numiters ]; do
 if echo $mllt_iters | grep -w $x >/dev/null; then # Do MLLT update.
   ( ali-to-post ark:$dir/cur.ali ark:- | \
     weight-silence-post 0.0 $silphonelist $dir/$x.mdl ark:- ark:- | \
     gmm-acc-mllt --binary=false $dir/$x.mdl "$featsub" ark:- $dir/$x.macc ) \
       2> $dir/macc.$x.log  || exit 1;

   est-mllt $dir/$x.mat.new $dir/$x.macc 2> $dir/mupdate.$x.log || exit 1;
   gmm-transform-means --binary=false $dir/$x.mat.new $dir/$x.mdl $dir/$[$x+1].mdl \
   2> $dir/transform_means.$x.log || exit 1;
   compose-transforms --print-args=false $dir/$x.mat.new $cur_lda $dir/$x.mat || exit 1;
   cur_lda=$dir/$x.mat

   feats="ark:splice-feats scp:data/train.scp ark:- | transform-feats $cur_lda ark:- ark:-|"
   # Subset of features used to train MLLT transforms.
   featsub="ark:scripts/subset_scp.pl 800 data/train.scp | splice-feats scp:- ark:- |
         transform-feats $cur_lda ark:- ark:-|"
else
 ….
```

Recipes

- Many Many Recipes
- Most of the time, we only have to add our own corpus

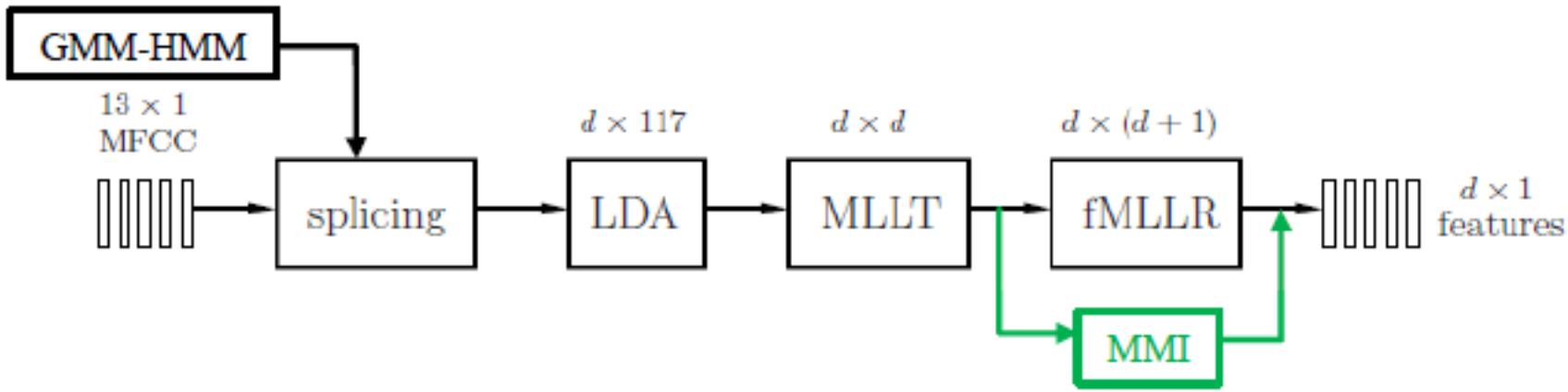
- Automatic Speech Recognition (ASR)
  - wsj
  - Swbd
  - librispeech
  - tedlium
  - timit
- Speaker/Language Identification
  - sre08
  - sre10
  - lre07
- Distant AS/noise/microphone array
  - aurora4
  - chime3
  - chime4
- Mandarin
  - thchs30
  - hkust
- Audio Event Detection
  - bn_music_speech

Pros and Cons of using Kaldi

- Pros
  - Modular source, open license (even commercial use is OK)
  - Plenty of example scripts
  - Optimized for LVCSR tasks
  - Using pipes to significantly reduce disk I/O
- Cons
  - Almost impossible to use without some knowledge on shell scripting
  - Commands and defaults change frequently
  - A little hard to work with on Windows

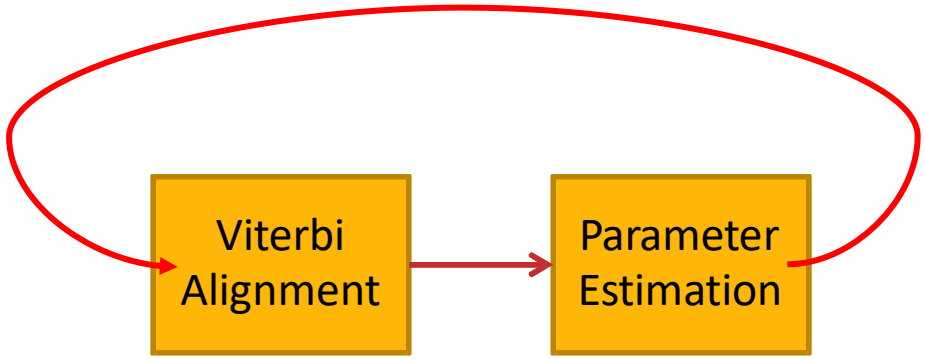
Feature Processing

- Basic MFCCs and PLPs
- Conventional Delta operations
- An unified framework for feature transformation
  - The “Transform” part does not know what the transform is at all
  - The “Estimate” part supports LDA, HLDA, fMLLR, MLLT, VTLN, etc.



Acoustic Modeling

- Standard maximum likelihood and MPE training of GMMs and subspace GMMs
- They don't like the idea of "embedded training", instead…



- "We don't believe it's better than Viterbi; and Viterbi makes it convenient to write alignments to disk." – Daniel Povey

Acoustic Modeling

- Context-dependent acoustic modeling, with crazily wide context support (for example hepta-phone)

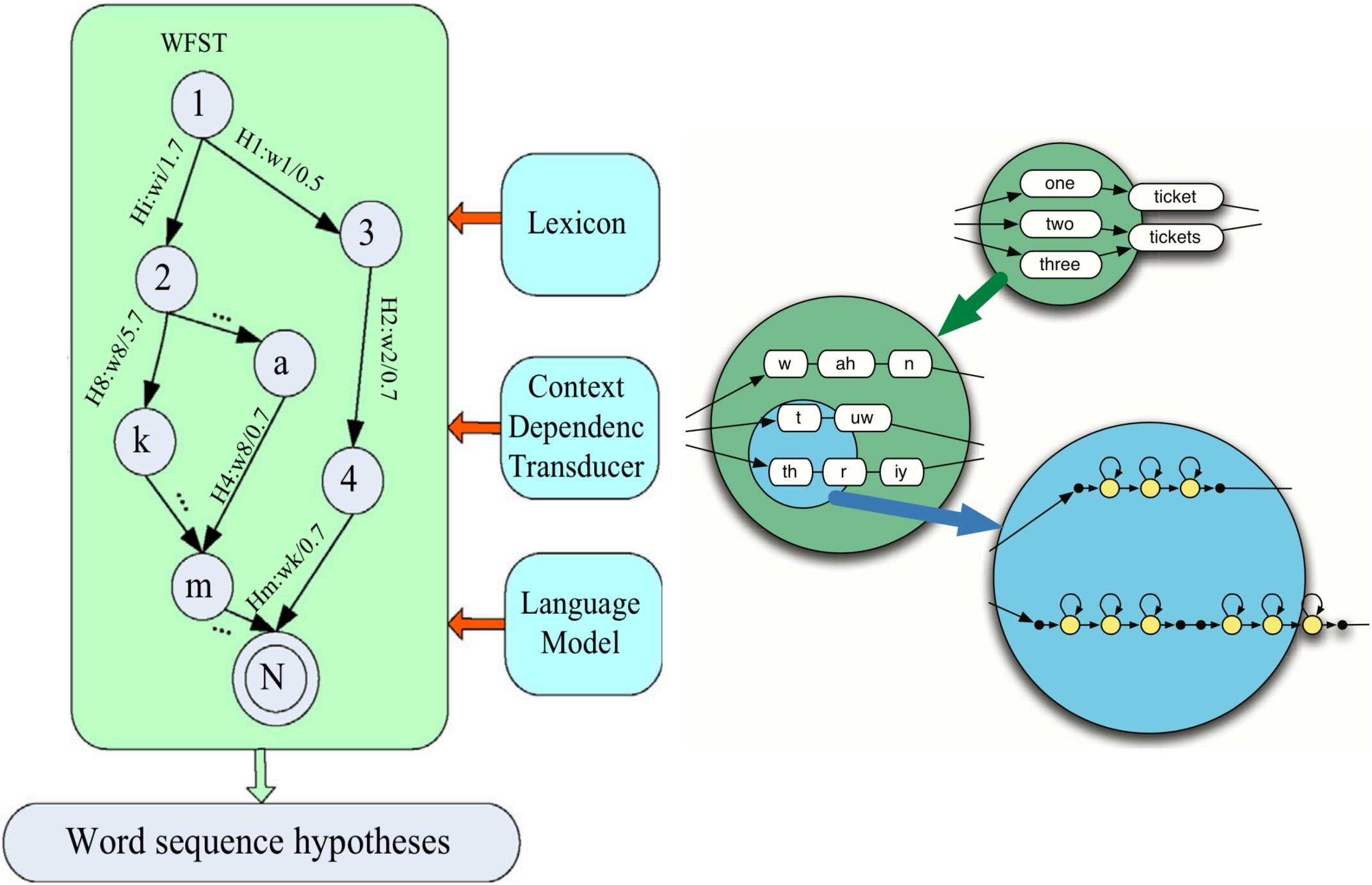
- Tree clustering according to:
  - Pre-defined questions
  - Data-driven clustering
  - Phone position and stress

Decoding

- Represents HMM as a series of GMM/DNN and a transition graph, which is encoded in the decoding graph

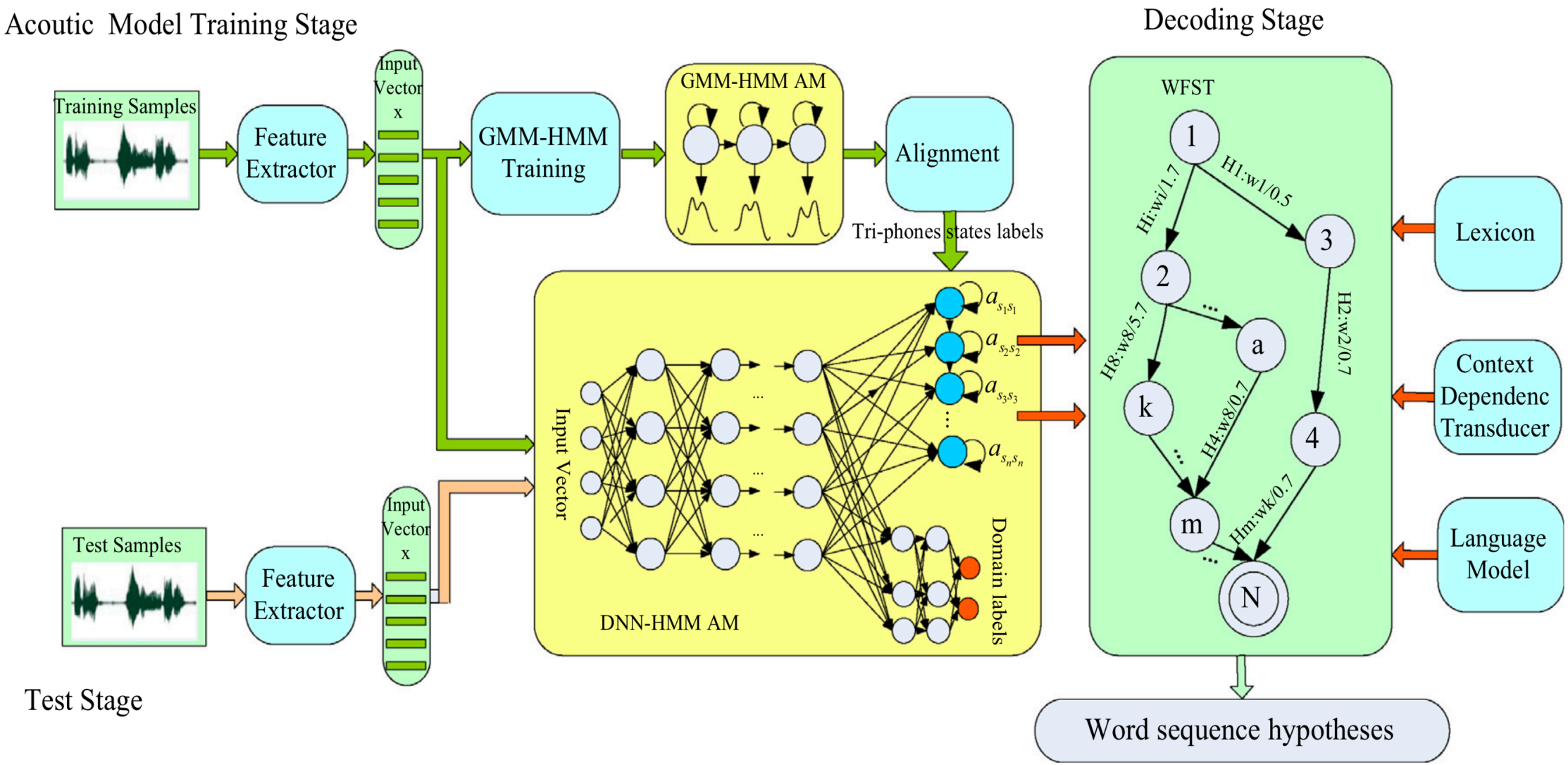
  - The decoding graph: $\min(\det(H \circ C \circ L \circ G))$
    - H: mapping from PDFs to context labels
    - C: mapping from context labels to phones
    - L: mapping from phones to words
    - G: grammar or language model

- Decoding is done by just finding the Viterbi path in the decoding graph


WFST
1
Hi:wi/1.7
H1:w1/0.5
3
2
H8:w8/5.7
a
H2:w2/0.7
k
H4:w8/0.7
4
m
Hm:wk/0.7
N
Lexicon
Context
Dependenc
Transducer
Language
Model
one
two
three
ticket
tickets
w
ah
n
t
uw
th
r
iy
Word sequence hypotheses

Decoding

- Decoders do not “know about” the HMM topology or GMMs, only a “decodable” interface, which has function that says “give me score for this”
- This make it easier to incorporate Kaldi with other kind of acoustic models


Acoutic Model Training Stage
Training Samples
Feature Extractor
Input Vector x
GMM-HMM Training
GMM-HMM AM
Alignment
Tri-phones states labels
Input Vector
DNN-HMM AM
Domain labels
Test Samples
Feature Extractor
Input Vector x
Test Stage
Decoding Stage
WFST
Hi:wi/1.7
H1:w1/0.5
H2:w2/0.7
H8:w8/5.7
H4:w8/0.7
Hm:wk/0.7
Lexicon
Context Dependenc Transducer
Language Model
Word sequence hypotheses