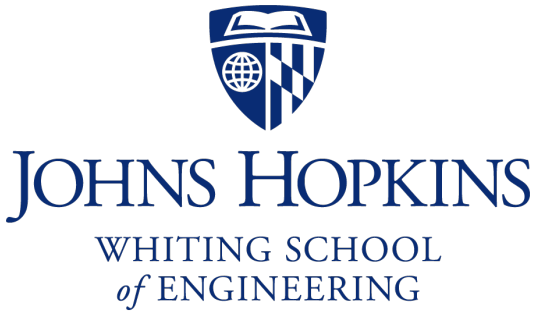
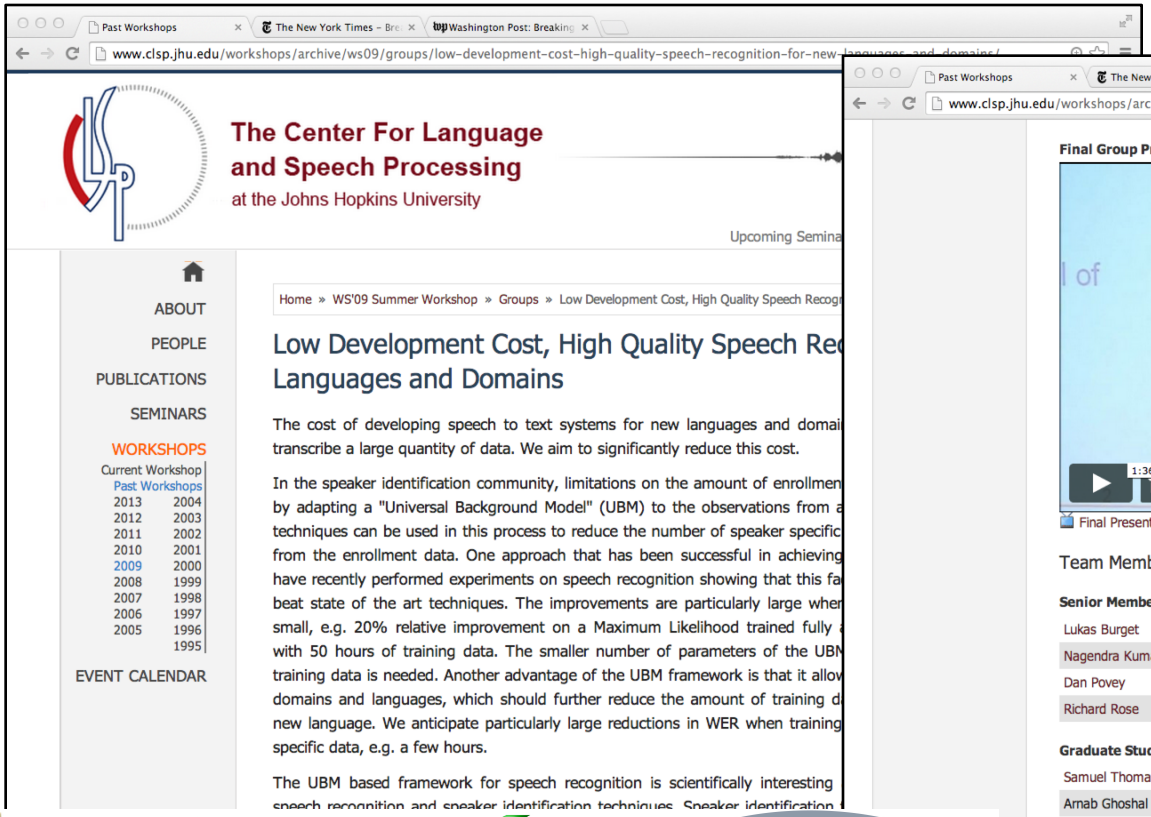
Low-Resource Speech-to-Text with Kaldi

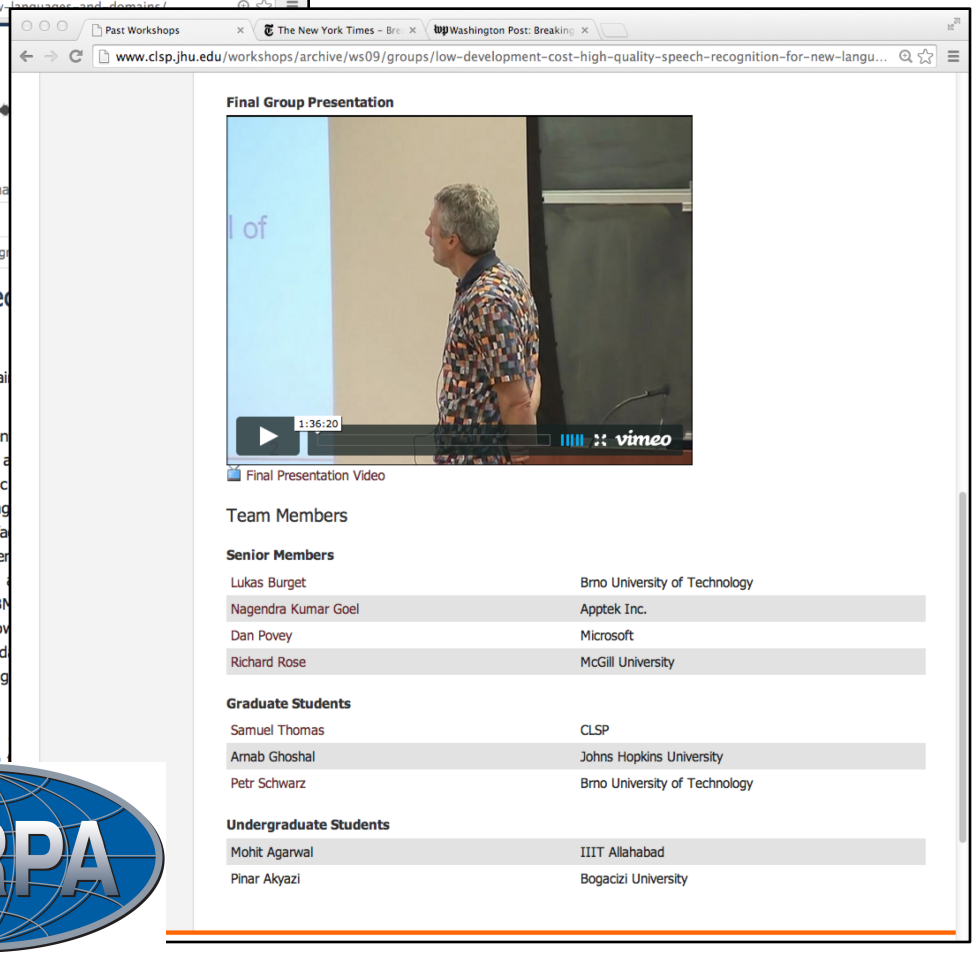
Sanjeev Khudanpur, Jan Trmal, Vijay Peddinti
Johns Hopkins University
Center for Language and Speech Processing
May 9, 2016



In the beginning, there was nothing

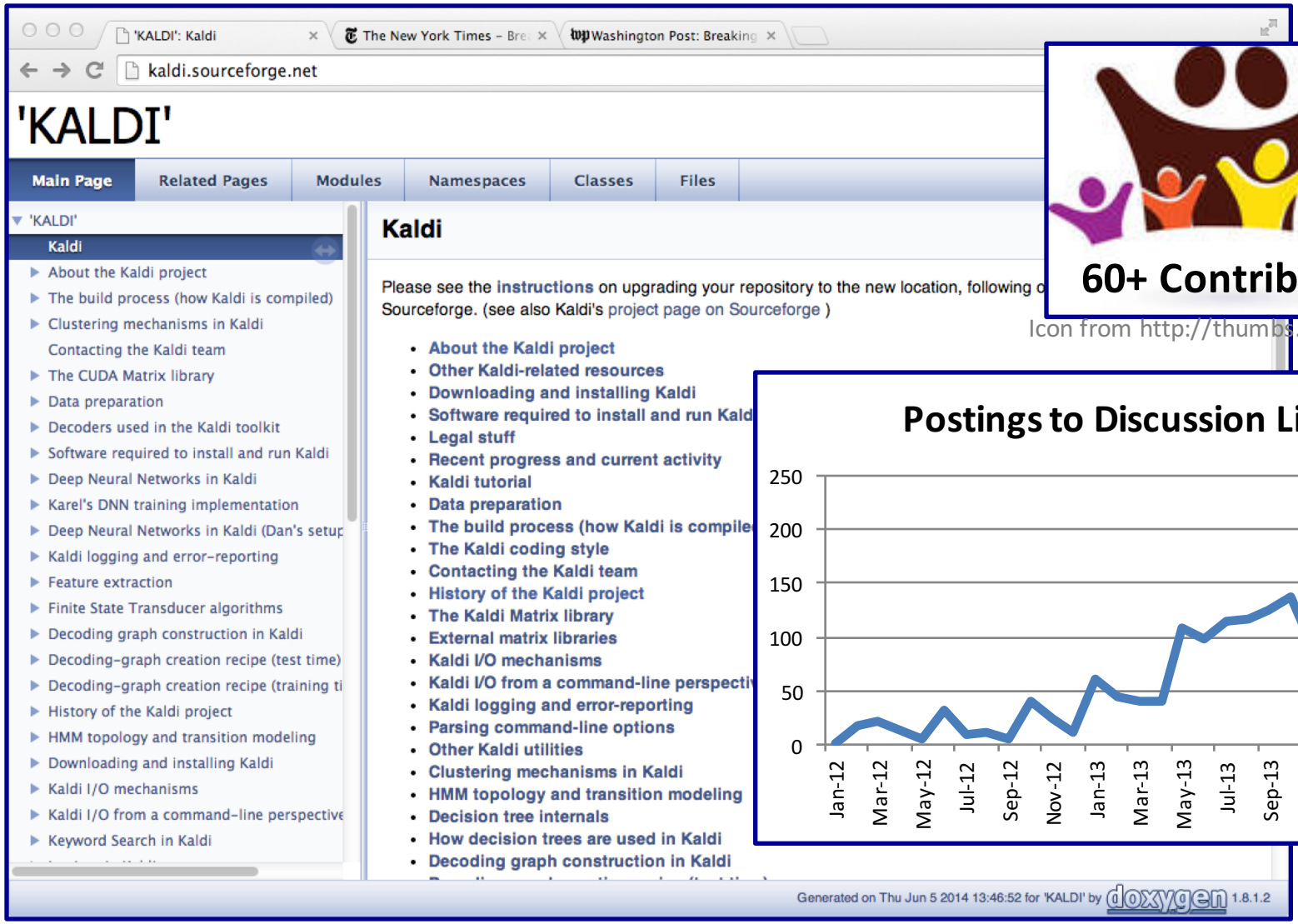
- Then Kaldi was born in Baltimore, MD, in 2009.





NSF Google DARPA

Kaldi then grew up & became …



Icon from http://thumbs.gograph.com

Meanwhile, Speech Search went from "Solved" to "Unsolved" … Again

- NIST TREC SDR (1998)
  - Spoken "document" retrieval from STT output as good as retrieval from reference transcripts
  - Speech search was declared a solved problem!
- NIST STD Pilot (2006)
  - STT was found to be inadequate for spoken "term" detection in conversational telephone speech
- Limited language diversity in CTS corpora
  - English Switchboard, Call Home and Fisher
  - Arabic and Mandarin Chinese Call Home

In 2012, IARPA launched BABEL

One month after Dan Povey returned to Kaldi's birthplace

- Automatic transcription of conversational telephone speech was still the core challenge.
- But with a few subtle, crucial changes
  - Focused attention on low-resource conditions
  - Required concurrent progress in multiple languages
    - **PY1**: Cantonese, Tagalog, Pashto, Turkish and Vietnamese
    - **PY2**: Assamese, Bengali, Haitian Creole, Lao, Zulu and Tamil
  - Reduced system development time from year to year
  - Used keyword search metrics to measure progress

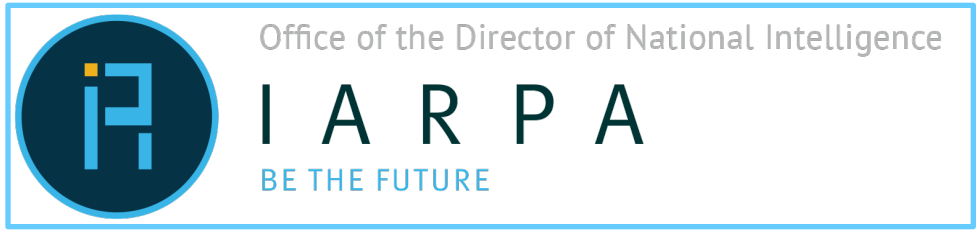
Kaldi Today

A community of Researchers Cooperatively Advancing STT

- C++ library, command-line tools, STT "recipes"
  - Freely available via GitHub (Apache 2.0 license)
- 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 (based on Google scholar data)
  - 400+ citations in 2015 (based on Google scholar data)
  - Used by several US and non-US companies
- Main "trunk" maintained by Johns Hopkins
  - Forks contain specializations by JHU and others

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**



Building an STT System with Kaldi

- Data preparation
  - Acoustic model training data
  - Pronunciation lexicon
  - Language model training data
- Basic GMM system building
  - Acoustic model training
  - Language model training
- Basic Decoding
  - Creating a static decoding graph
  - Lattice rescoring
- Basic DNN system building
- Going beyond the basics

Setting up Paths, Queue Commands, …

Terminal

```
export KALDI_ROOT=`pwd`/../../..
[ -f $KALDI_ROOT/tools/env.sh ] && . $KALDI_ROOT/tools/env.sh
export PATH=$PWD/utils/:$KALDI_ROOT/tools/openfst/bin:$PWD:$PATH
[ ! -f $KALDI_ROOT/tools/config/common_path.sh ] && echo >&2 "The standard file\
 $KALDI_ROOT/tools/config/common_path.sh is not present -> Exit!" && exit 1
. $KALDI_ROOT/tools/config/common_path.sh
export LC_ALL=C
```

```
nj=16
dev_nj=6

if [ $stage -le 1 ]; then
  e
  l
  l
  u
  l
fi
```

Terminal

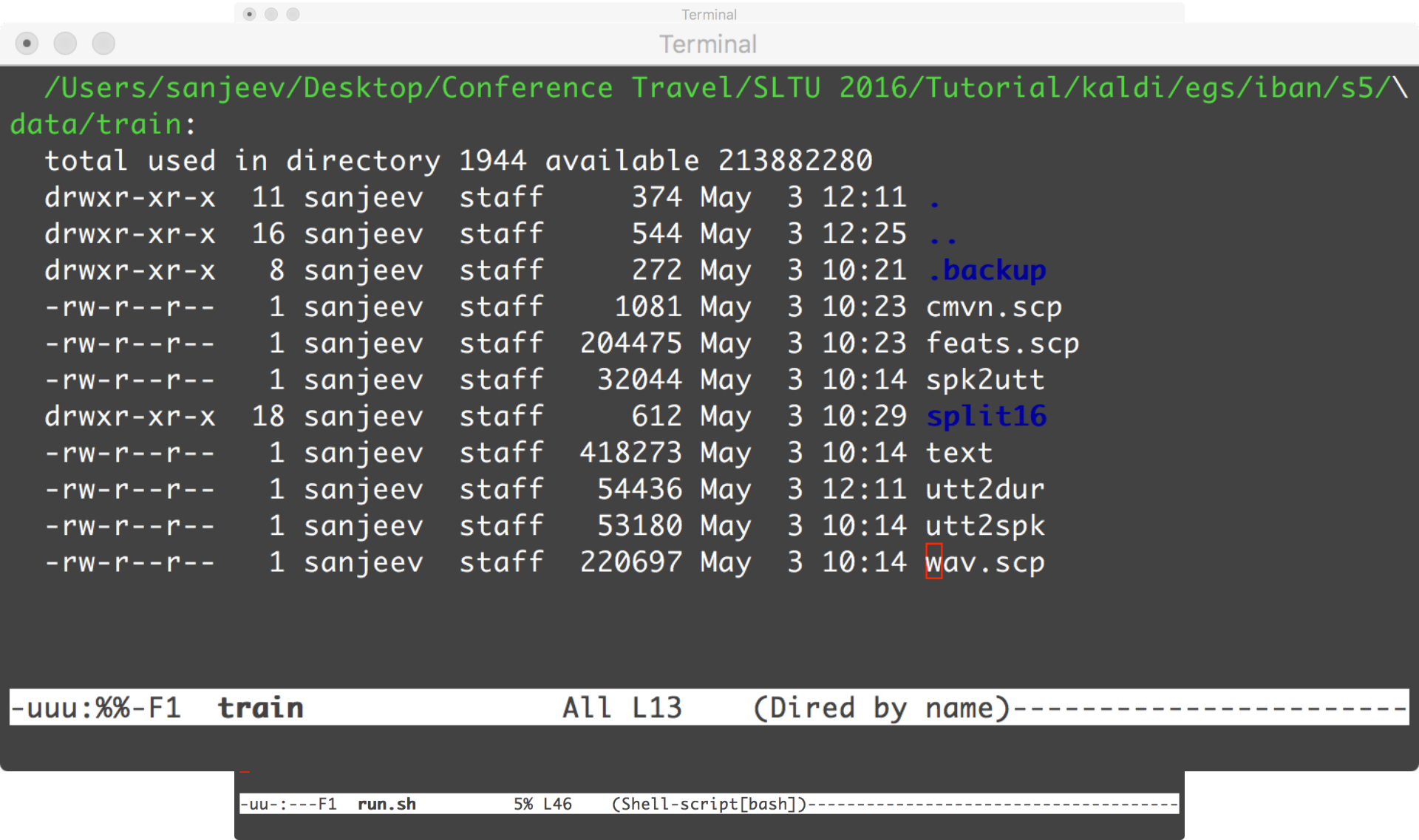
```
export train_cmd="run.pl --max-jobs-run 32"
export decode_cmd="run.pl --max-jobs-run 32"

if
  #
  f
#export train_cmd="queue.pl"
#export decode_cmd="queue.pl --mem 4G"

  d
fi
```

```
-uu-:---F1  run.sh          5% L46     (Shell-script[bash])-----------------------------------
```

Building an STT System with Kaldi

- Data preparation
  - **Acoustic model training data**
  - Pronunciation lexicon
  - Language model training data
- Basic GMM system building
  - Acoustic model training
  - Language model training
- Basic Decoding
  - Creating a static decoding graph
  - Lattice rescoring
- Basic DNN system building
- Going beyond the basics

Preparing Acoustic Training Data

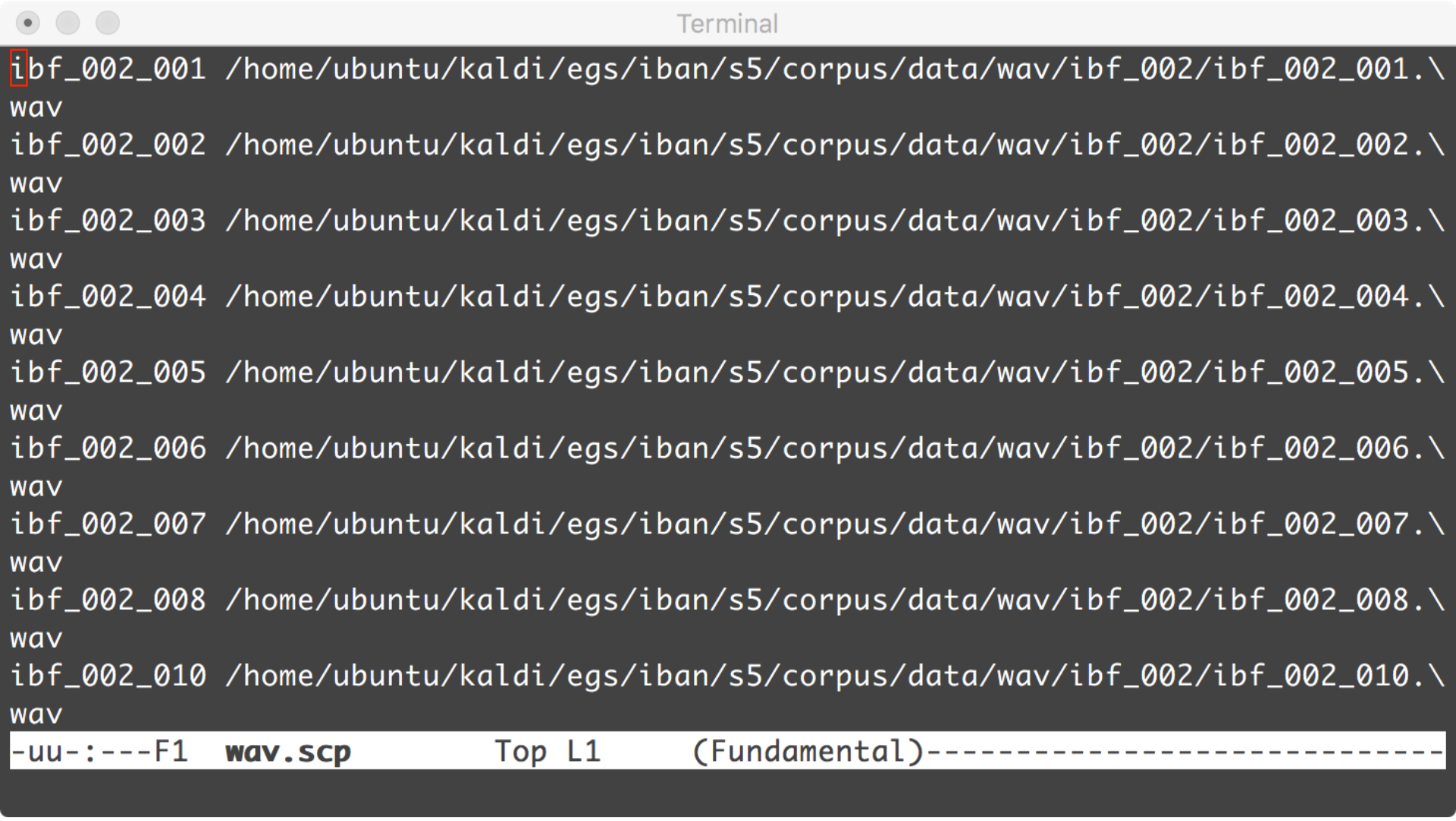


data/train/text

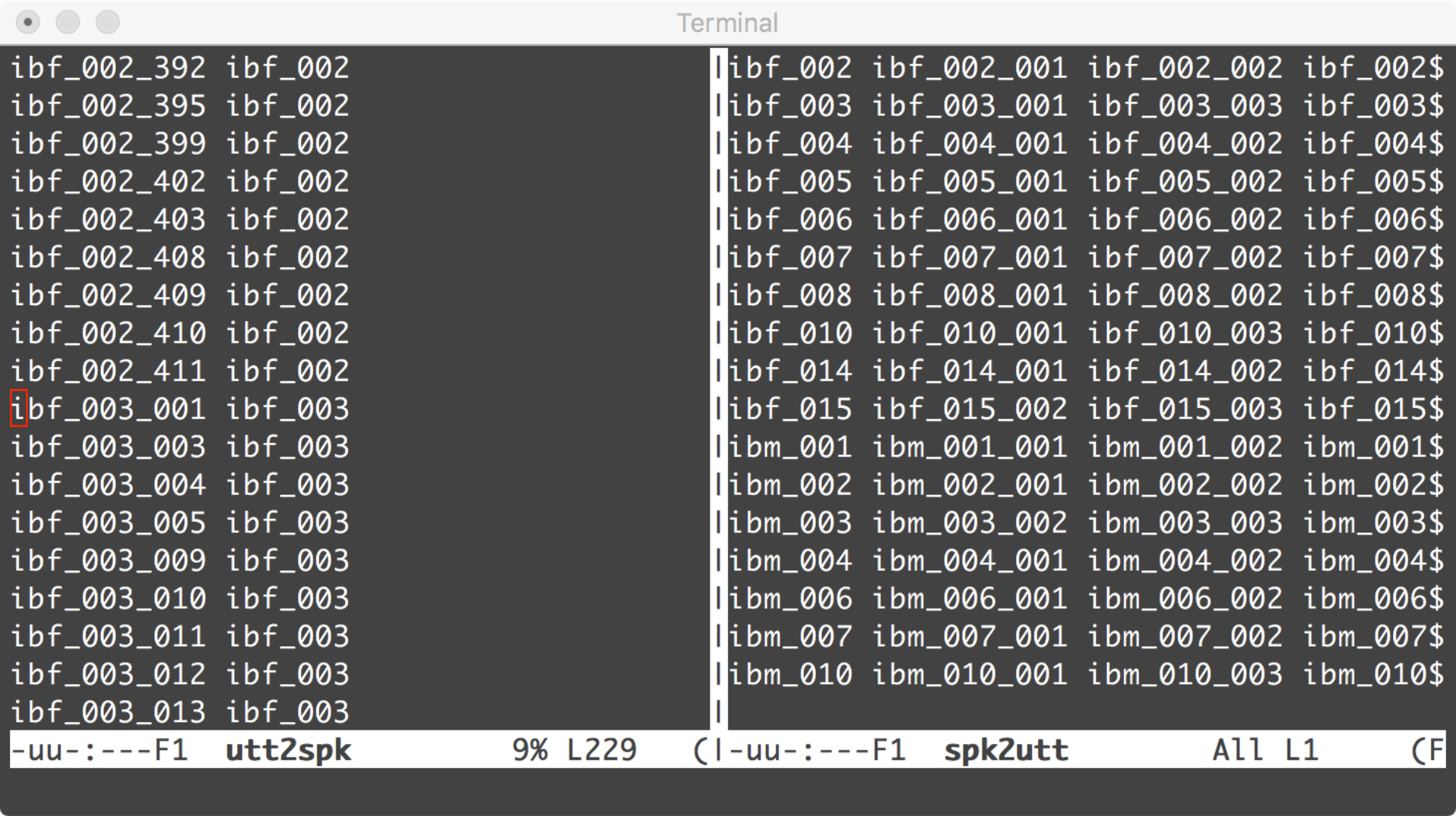
Terminal

```
ibf_002_165  bagiiban miri fm bisi nyediaka tempat ke empat bengkah program gaw\
ai rambau musim pengerami gawai ke taun tu
ibf_002_166  program gawai tu nyengkaum jaku pesan ari penulung menteri perengk\
a ke diguna mensia mayuh datuk sylvester entri muran enggau
ibf_002_167  program anak mit begawai ba studio ke dikeluarka mary bajang
ibf_002_169  kepala bagi iban rtm miri encik simon suti madahka bala nembiak ke\
 masuk program setengah jam nya datai ari sk lambir village sk beluru central e\
nggau sk kelapa sawit numor dua
ibf_002_171  program nya deka ditabur ba serata menua nengah waifm ba ari ti ke\
terubah gawai pukul sepuluh pagi
ibf_002_172  sebengkah agi program gawai nya berami gawai miri fm dua ribu dua \
belas pengelama empat puluh lima minit
ibf_002_174  program nya deka dikeluar ba ari ti kedua gawai pukul dua ngalih h\
ari
ibf_002_175  nya naka kami naburka berita ari waifm berita nya tadi ditusun bev\
erly kaur sereta disalin shirley limban salam satu malaysia
ibf_002_177  pukul tujuh pagi
ibf_002_179  diatu kami naburka berita ari waifm
-uu-:---F1  text              4% L89    (Fundamental)----------------------------
Undo!
```

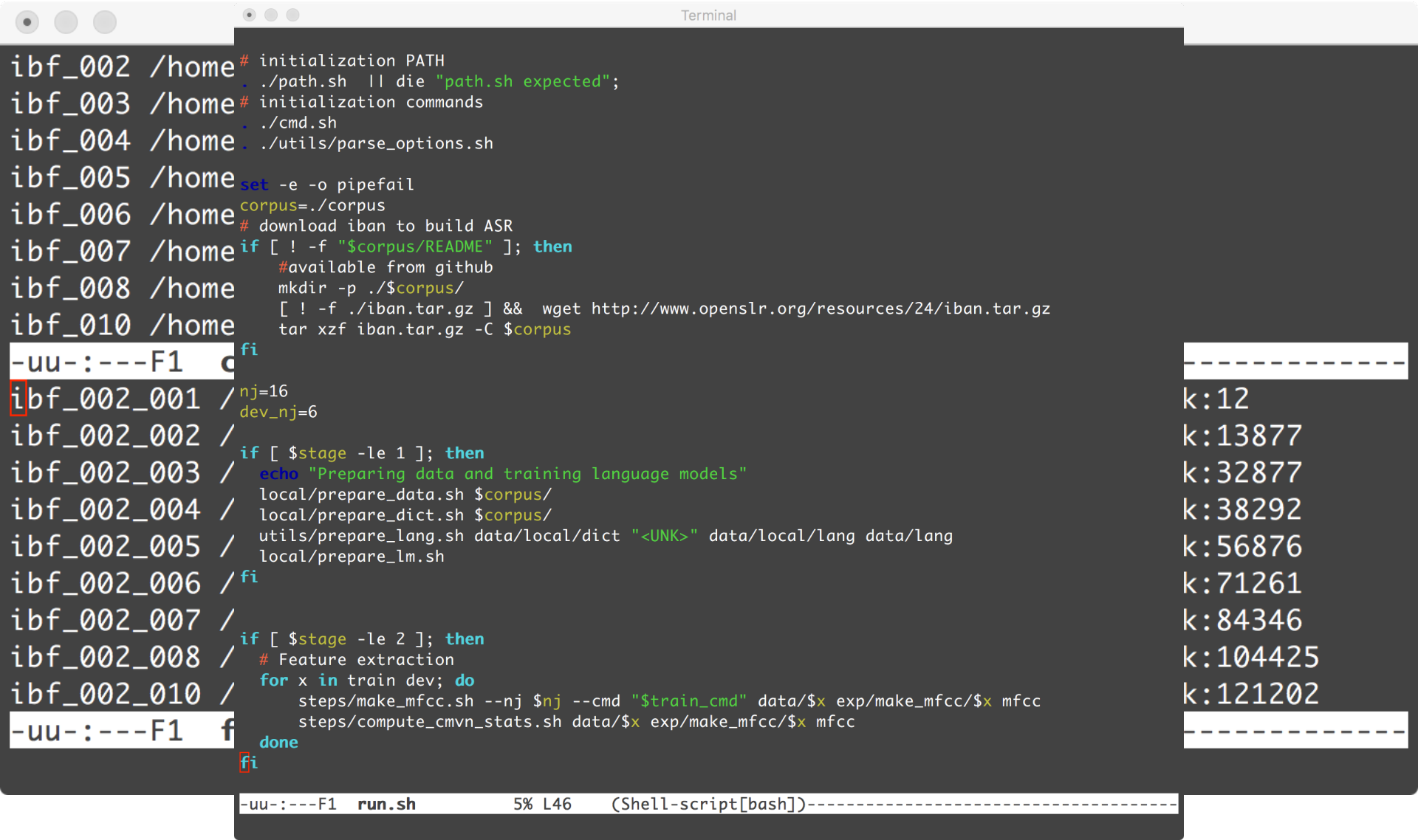
data/train/wav.scp



data/train/(utt2spk|spk2utt)



data/train/(cmvn.scp|feats.scp)



Building an STT System with Kaldi

- Data preparation
  - Acoustic model training data
  - **Pronunciation lexicon**
  - Language model training data
- Basic GMM system building
  - Acoustic model training
  - Language model training
- Basic Decoding
  - Creating a static decoding graph
  - Lattice rescoring
- Basic DNN system building
- Going beyond the basics

Preparing the Pronunciation Lexicon

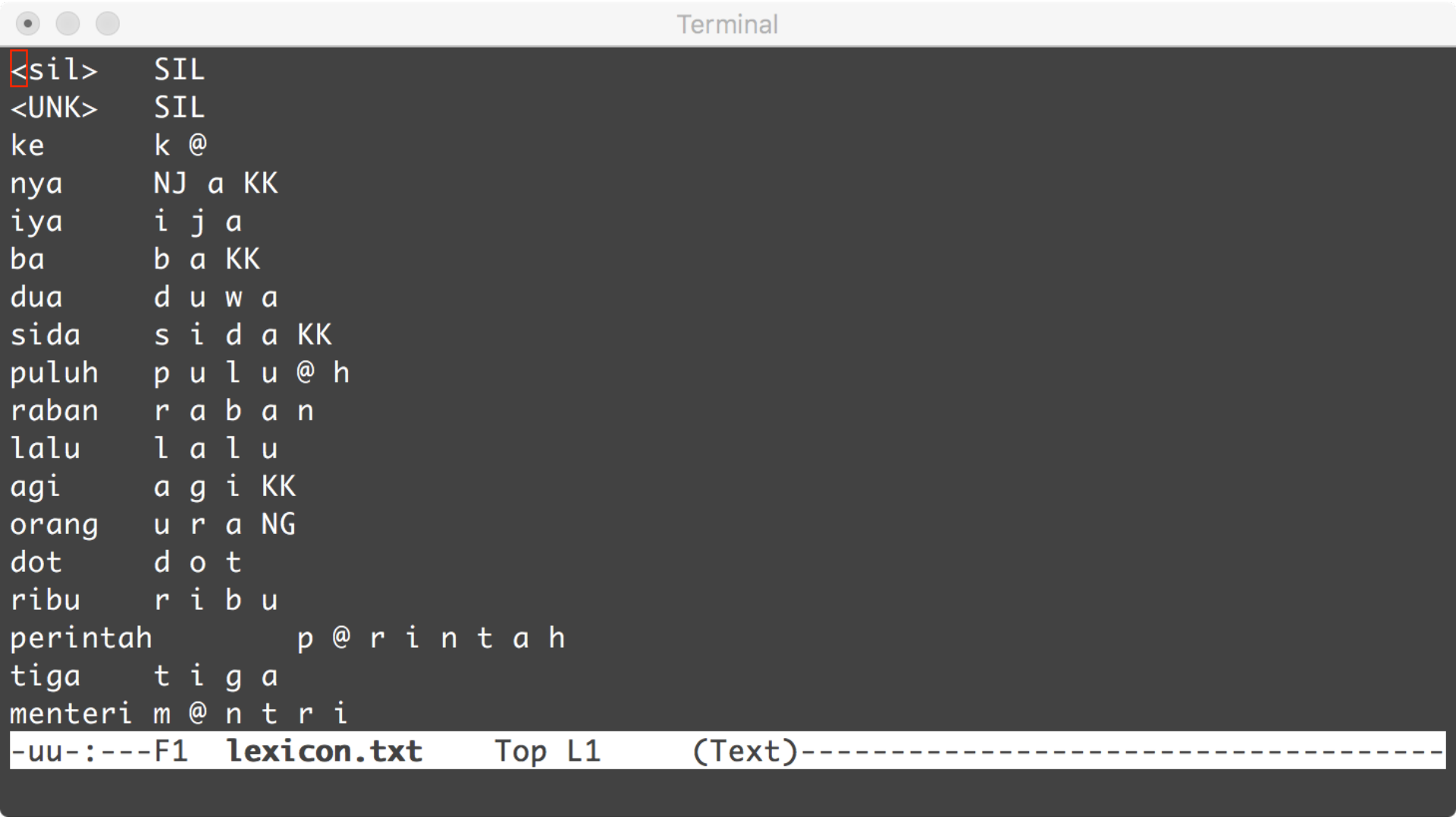
Terminal

```
  /Users/sanjeev/Desktop/Conference Travel/SLTU 2016/Tutorial/kaldi/egs/iban/s5\
/data/local/dict:
  total used in directory 3408 available 213871436
  drwxr-xr-x  9 sanjeev  staff     306 May  2 20:07 .
  drwxr-xr-x  5 sanjeev  staff     170 May  2 20:15 ..
  -rw-r--r--  1 sanjeev  staff       0 May  3 10:14 extra_questions.txt
  -rw-r--r--  1 sanjeev  staff  788889 May  3 10:14 lexicon.txt
  -rw-r--r--  1 sanjeev  staff  934289 May  2 20:07 lexiconp.txt
  -rw-r--r--  1 sanjeev  staff      78 May  3 10:14 nonsilence_phones.txt
  -rw-r--r--  1 sanjeev  staff       6 May  3 10:14 oov.txt
  -rw-r--r--  1 sanjeev  staff       4 May  3 10:14 optional_silence.txt
  -rw-r--r--  1 sanjeev  staff       4 May  3 10:14 silence_phones.txt
```

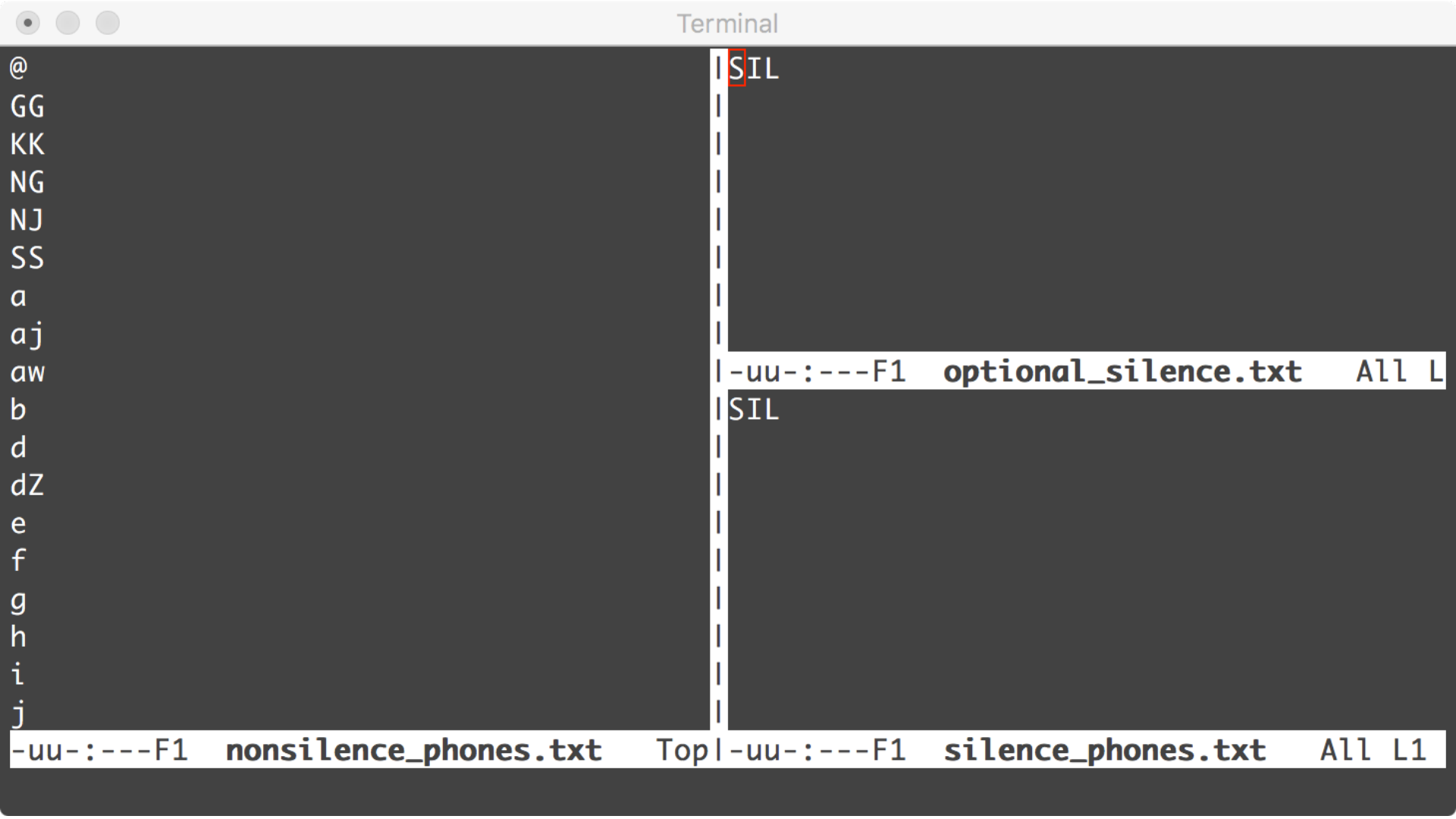
```
-uuu:%%-F1  dict                    All L5     (Dired by name)----------------------
```

```
-uu-:---F1  run.sh          5% L46    (Shell-script[bash])-----------------------------------
```

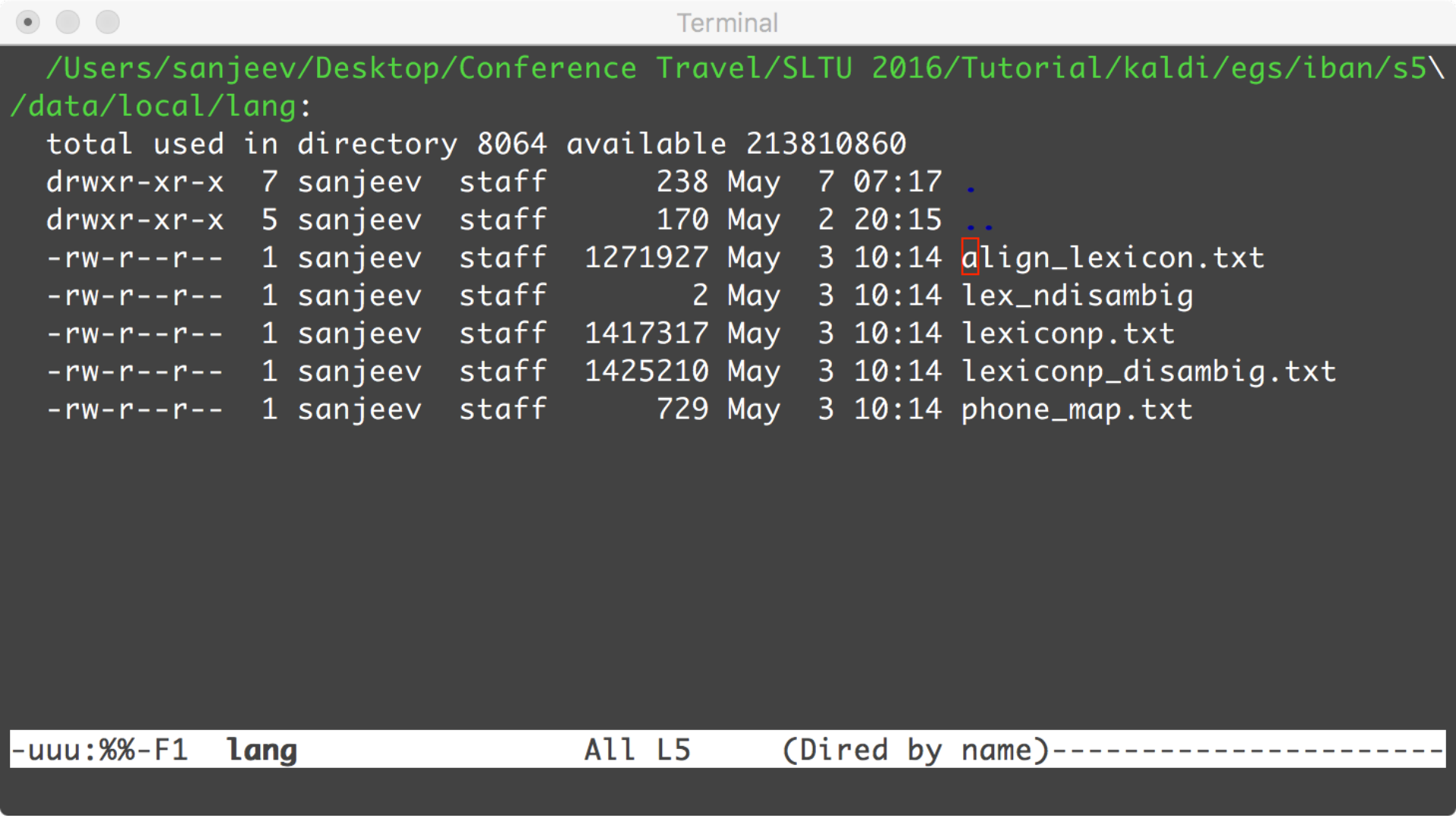
data/local/dict/lexicon.txt



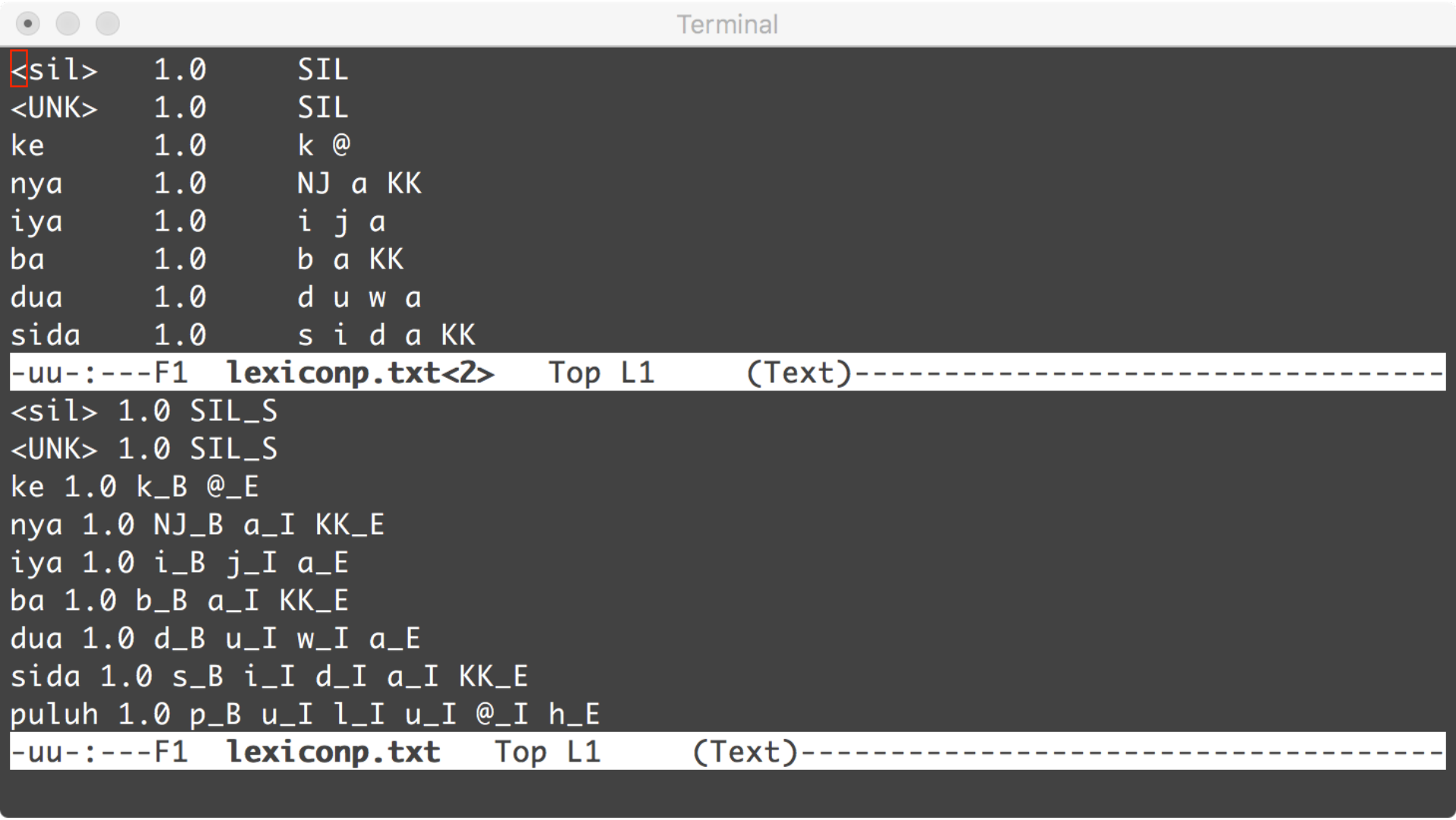
data/local/dict/*silence*.txt



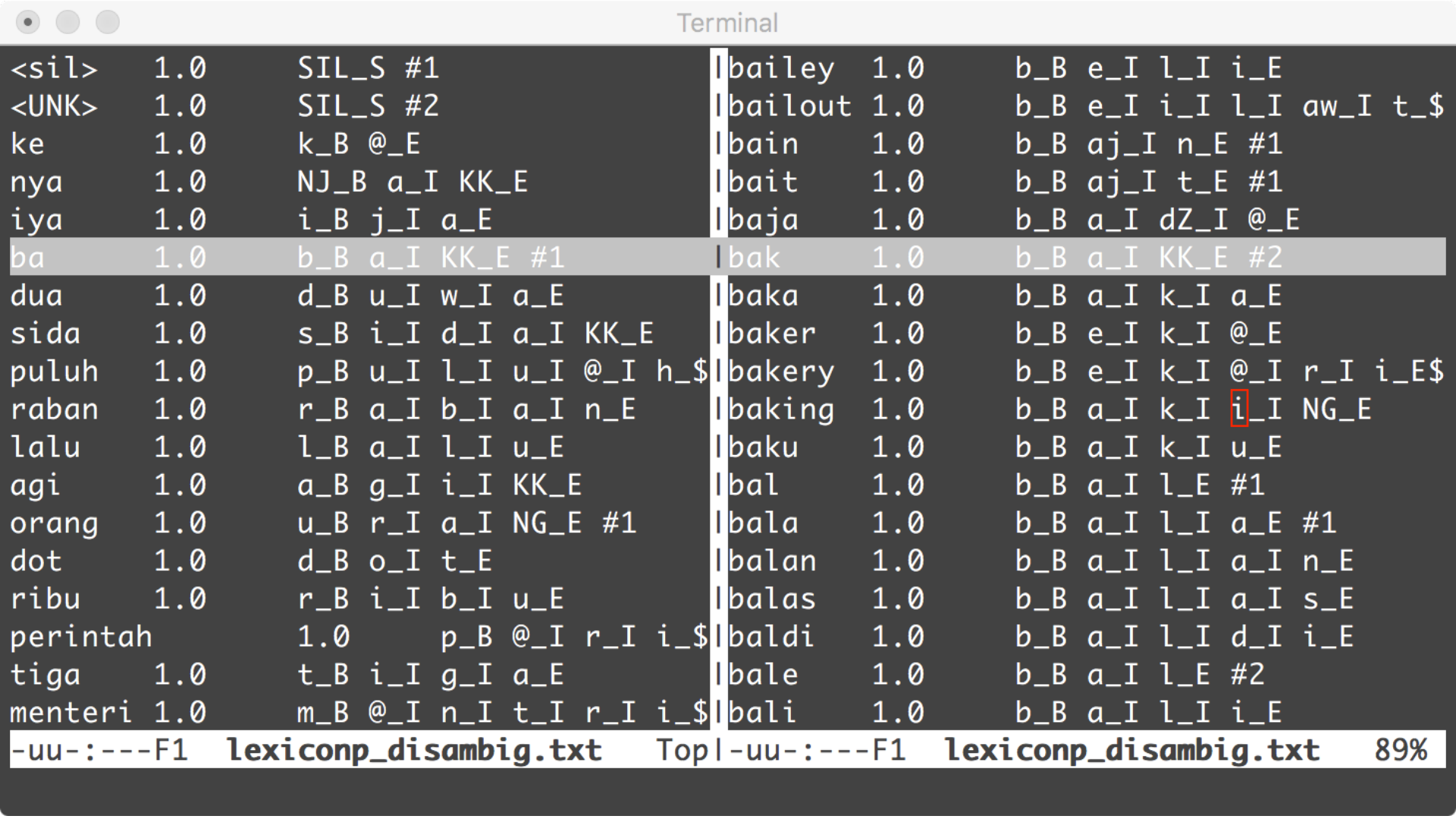
data/local/lang



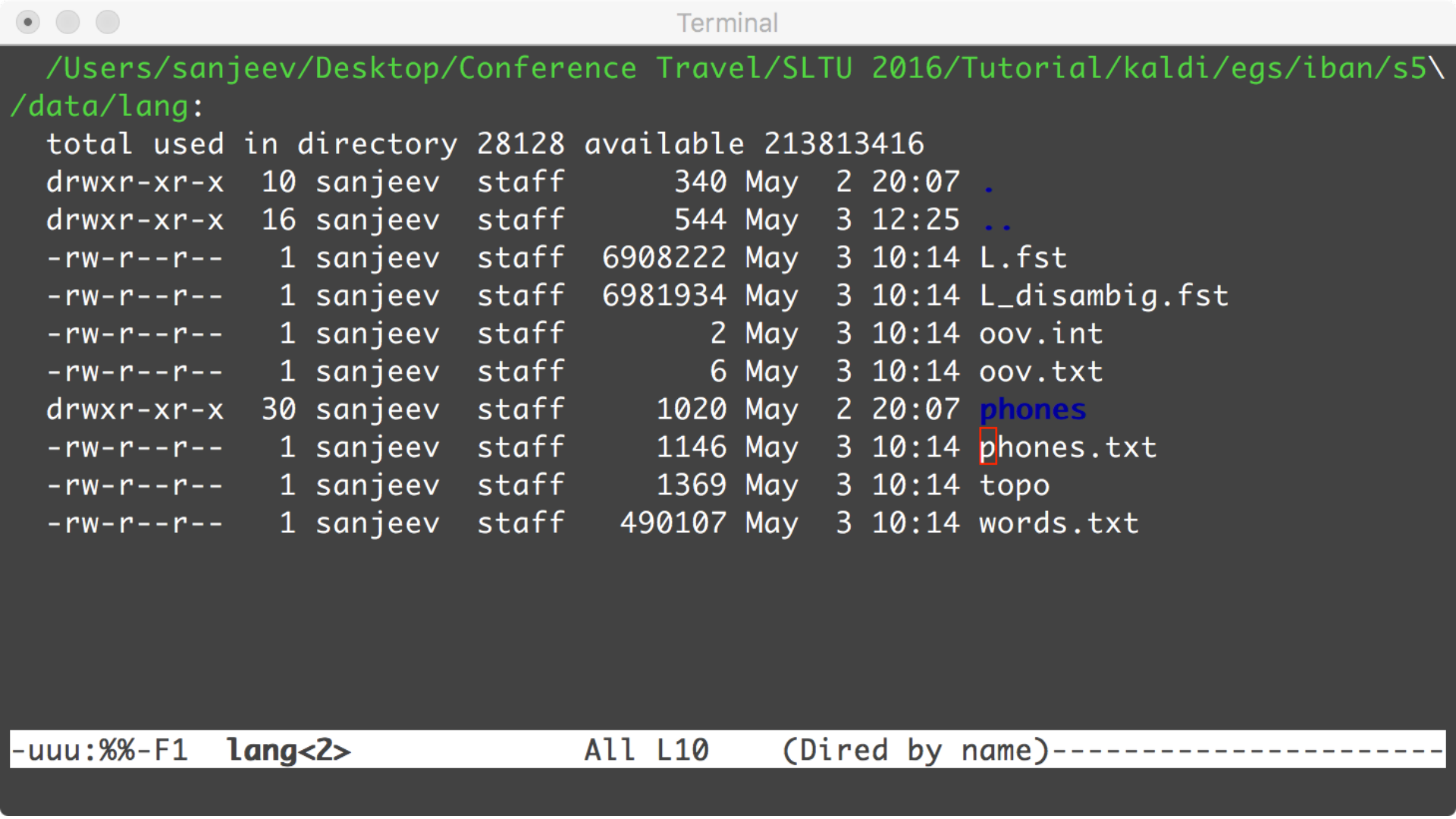
Word Boundary Tags



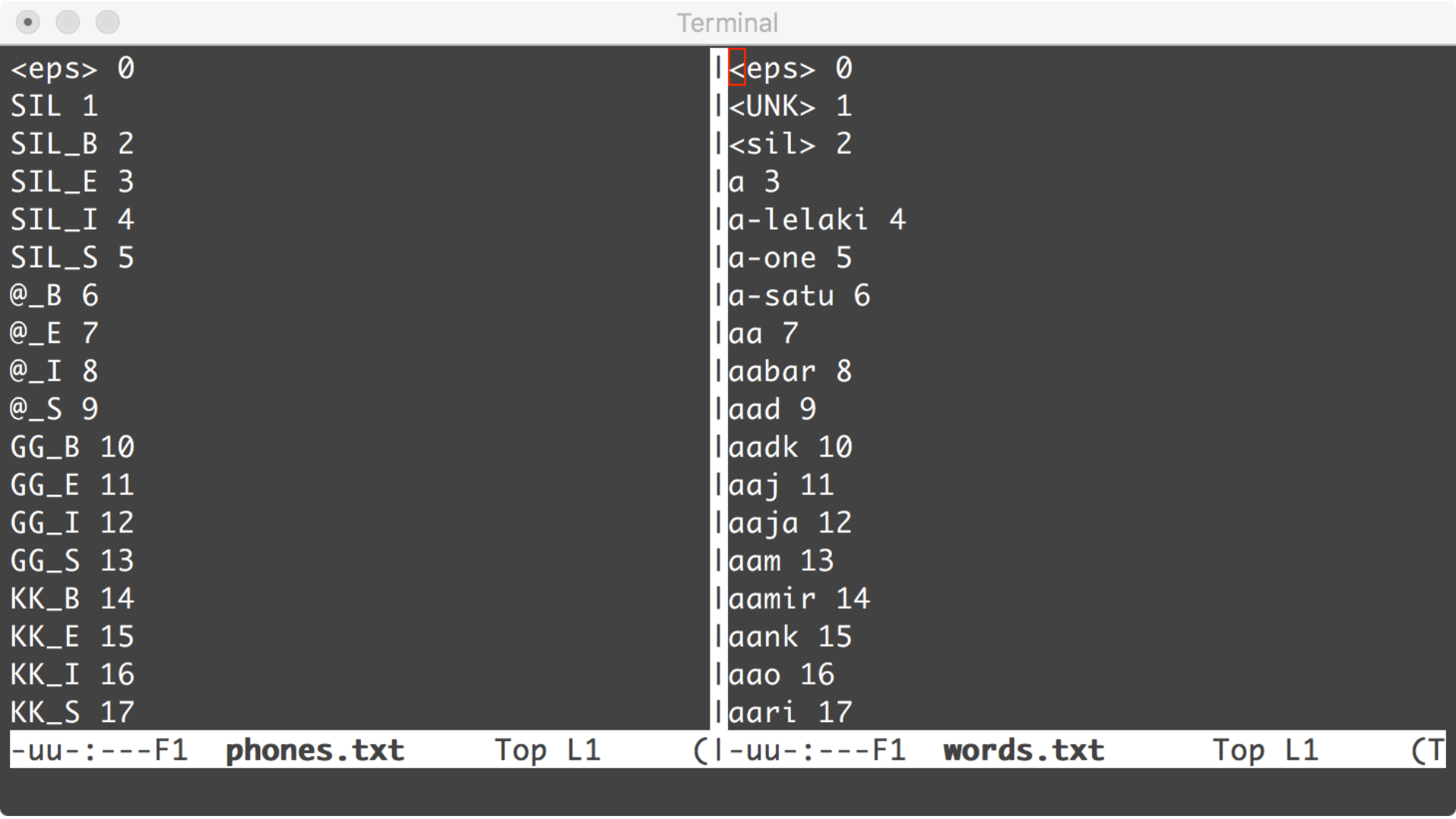
Disambiguation Symbols



data/lang


Terminal
/Users/sanjeev/Desktop/Conference Travel/SLTU 2016/Tutorial/kaldi/egs/iban/s5\
/data/lang:
total used in directory 28128 available 213813416
drwxr-xr-x 10 sanjeev staff 340 May 2 20:07 .
drwxr-xr-x 16 sanjeev staff 544 May 3 12:25 ..
-rw-r--r-- 1 sanjeev staff 6908222 May 3 10:14 L.fst
-rw-r--r-- 1 sanjeev staff 6981934 May 3 10:14 L_disambig.fst
-rw-r--r-- 1 sanjeev staff 2 May 3 10:14 oov.int
-rw-r--r-- 1 sanjeev staff 6 May 3 10:14 oov.txt
drwxr-xr-x 30 sanjeev staff 1020 May 2 20:07 phones
-rw-r--r-- 1 sanjeev staff 1146 May 3 10:14 phones.txt
-rw-r--r-- 1 sanjeev staff 1369 May 3 10:14 topo
-rw-r--r-- 1 sanjeev staff 490107 May 3 10:14 words.txt
-uuu:%%-F1 lang<2> All L10 (Dired by name)----------

data/lang/(phones|words).txt

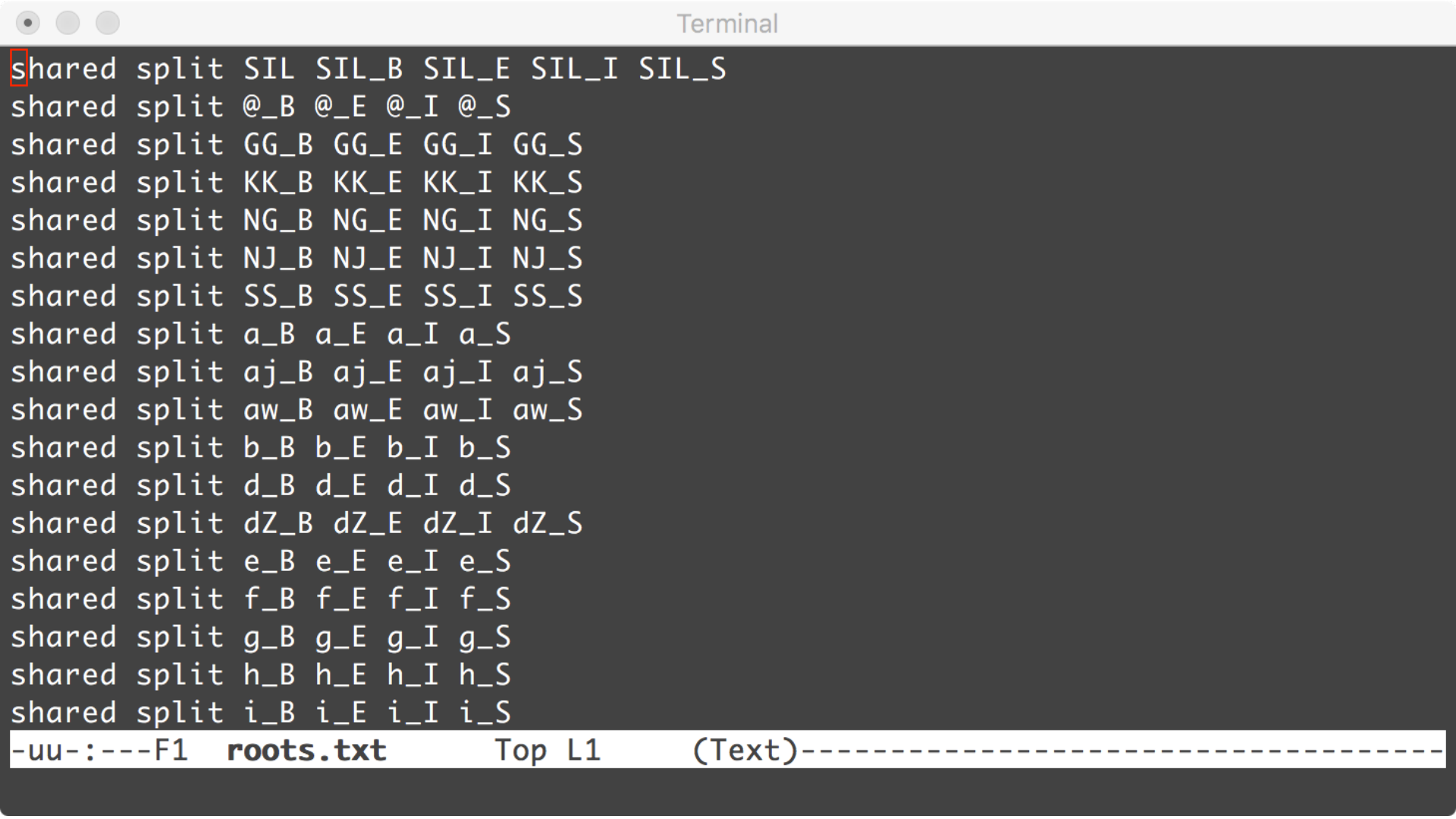


data/lang/topo

Terminal

```
<ForPhones>
1 2 3 4 5
</ForPhones>
<State> 0 <PdfClass> 0 <Transition> 0 0.25 <Transition> 1 0.25 <Transition> 2 0\
.25 <Transition> 3 0.25 </State>
<State> 1 <PdfClass> 1 <Transition> 1 0.25 <Transition> 2 0.25 <Transition> 3 0\
.25 <Transition> 4 0.25 </State>
<State> 2 <PdfClass> 2 <Transition> 1 0.25 <Transition> 2 0.25 <Transition> 3 0\
.25 <Transition> 4 0.25 </State>
<State> 3 <PdfClass> 3 <Transition> 1 0.25 <Transition> 2 0.25 <Transition> 3 0\
.25 <Transition> 4 0.25 </State>
<State> 4 <PdfClass> 4 <Transition> 4 0.75 <Transition> 5 0.25 </State>
<State> 5 </State>
</TopologyEntry>
</Topology>


-uu-:---F1  topo          Bot L12    (Fundamental)------------------------------
```

data/lang/phones/roots.txt



data/lang/phones/extra_questions.txt

Terminal

```
@_B GG_B KK_B NG_B NJ_B SS_B a_B aj_B aw_B b_B d_B dZ_B e_B f_B g_B h_B i_B j_B\
 k_B l_B m_B n_B o_B oj_B p_B r_B s_B t_B tS_B u_B v_B w_B x_B z_B
@_E GG_E KK_E NG_E NJ_E SS_E a_E aj_E aw_E b_E d_E dZ_E e_E f_E g_E h_E i_E j_E\
 k_E l_E m_E n_E o_E oj_E p_E r_E s_E t_E tS_E u_E v_E w_E x_E z_E
@_I GG_I KK_I NG_I NJ_I SS_I a_I aj_I aw_I b_I d_I dZ_I e_I f_I g_I h_I i_I j_I\
 k_I l_I m_I n_I o_I oj_I p_I r_I s_I t_I tS_I u_I v_I w_I x_I z_I
@_S GG_S KK_S NG_S NJ_S SS_S a_S aj_S aw_S b_S d_S dZ_S e_S f_S g_S h_S i_S j_S\
 k_S l_S m_S n_S o_S oj_S p_S r_S s_S t_S tS_S u_S v_S w_S x_S z_S
SIL
SIL_B
SIL_E
SIL_I
SIL_S
```

-uu-:---F1 **extra_questions.txt**   All L1   (Text)----------

Building an STT System with Kaldi

- Data preparation
  - Acoustic model training data
  - Pronunciation lexicon
  - **Language model training data**
- Basic GMM system building
  - Acoustic model training
  - Language model training
- Basic Decoding
  - Creating a static decoding graph
  - Lattice rescoring
- Basic DNN system building
- Going beyond the basics

Preparing the Language Model

Terminal

```
local/train_lms_srilm.sh --train-text data/train/text data/ data/srilm

nl -nrz -w10  corpus/LM/iban-bp-2012.txt | sort -R > data/local/external_text
local/train_lms_srilm.sh --train-text data/local/external_text data/ data/srilm\
_external

# let's do ngram interpolation of the previous two LMs
# the lm.gz is always symlink to the model with the best perplexity, so we use \
that

mkdir -p data/srilm_interp
for w in 0.9 0.8 0.7 0.6 0.5; do
    ngram -lm data/srilm/lm.gz  -mix-lm data/srilm_external/lm.gz \
          -lambda $w -write-lm data/srilm_interp/lm.${w}.gz
    echo -n "data/srilm_interp/lm.${w}.gz "
    ngram -lm data/srilm_interp/lm.${w}.gz -ppl data/srilm/dev.txt | paste -s
done | sort  -k15,15g  > data/srilm_interp/perplexities.txt
-uu-:---F1  prepare_lm.sh   14% L19    (Shell-script[bash])--------------------
```

```
-uu-:---F1  run.sh          5% L46    (Shell-script[bash])-----------------------------------
```

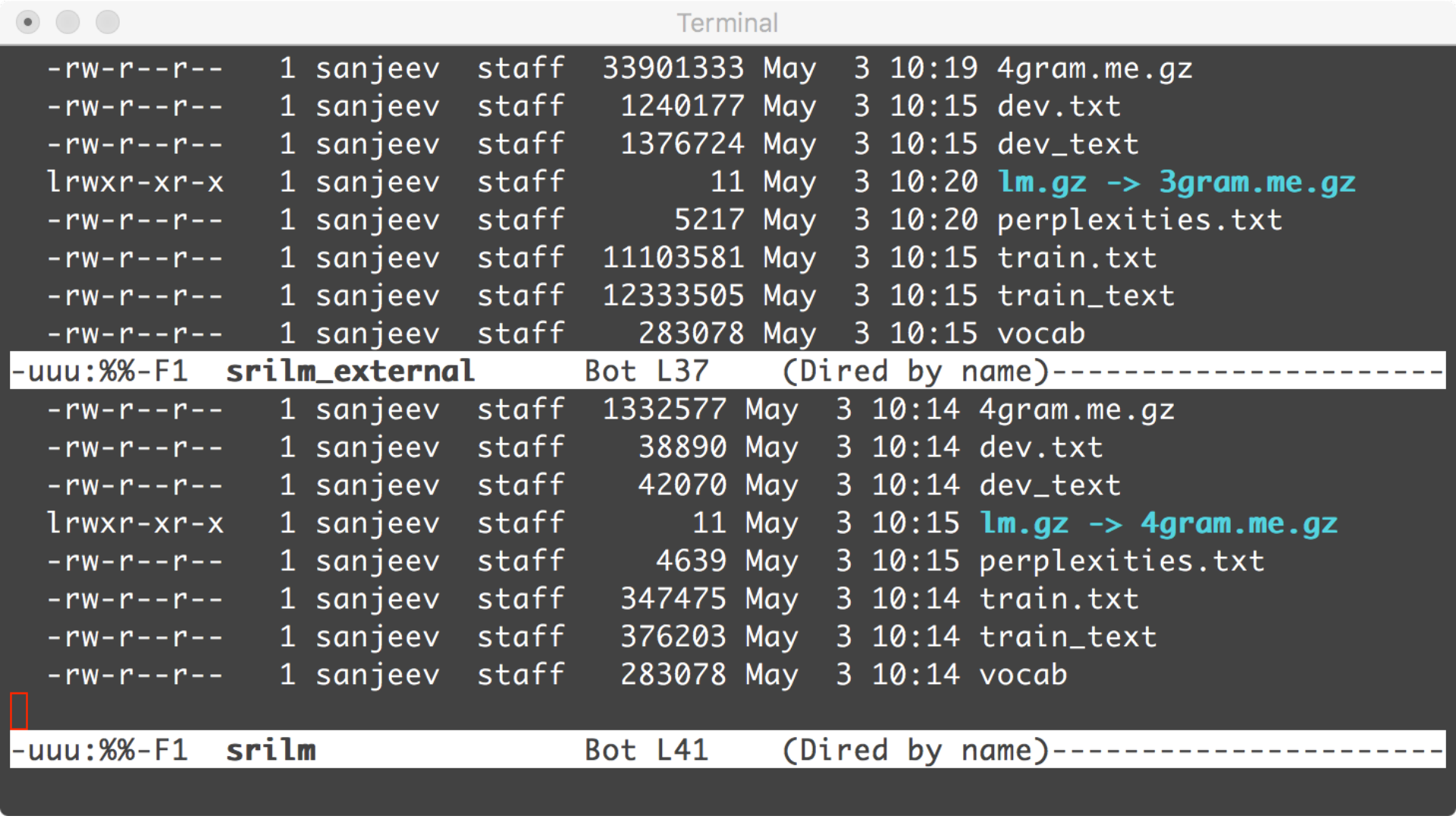
local/train_lms_srilm.sh

Terminal

```
echo "-------------------"
echo "Kneser-Ney 3grams"
echo "-------------------"
ngram-count -lm $tgtdir/3gram.kn011.gz -kndiscount1 -gt1min 0 -kndiscount2 -gt2\
min 1 -kndiscount3 -gt3min 1 -order 3 -text $tgtdir/train.txt -vocab $tgtdir/vo\
cab -unk -sort -map-unk "$oov_symbol"
ngram-count -lm $tgtdir/3gram.kn012.gz -kndiscount1 -gt1min 0 -kndiscount2 -gt2\
min 1 -kndiscount3 -gt3min 2 -order 3 -text $tgtdir/train.txt -vocab $tgtdir/vo\
cab -unk -sort -map-unk "$oov_symbol"
ngram-count -lm $tgtdir/3gram.kn022.gz -kndiscount1 -gt1min 0 -kndiscount2 -gt2\
min 2 -kndiscount3 -gt3min 2 -order 3 -text $tgtdir/train.txt -vocab $tgtdir/vo\
cab -unk -sort -map-unk "$oov_symbol"
ngram-count -lm $tgtdir/3gram.kn023.gz -kndiscount1 -gt1min 0 -kndiscount2 -gt2\
min 2 -kndiscount3 -gt3min 3 -order 3 -text $tgtdir/train.txt -vocab $tgtdir/vo\
cab -unk -sort -map-unk "$oov_symbol"
```

-uu-:---F1 **train_lms_srilm.sh** 48% L152 (Shell-script[bash])----------

local/train_lms_srilm.sh (cont'd)



Interpolated Language Models

Terminal

```
# let's do ngram interpolation of the previous two LMs
# the lm.gz is always symlink to the model with the best perplexity, so we use \
that

mkdir -p data/srilm_interp
for w in 0.9 0.8 0.7 0.6 0.5; do
    ngram -lm data/srilm/lm.gz  -mix-lm data/srilm_external/lm.gz \
          -lambda $w -write-lm data/srilm_interp/lm.${w}.gz
    echo -n "data/srilm_interp/lm.${w}.gz "
    ngram -lm data/srilm_interp/lm.${w}.gz -ppl data/srilm/dev.txt | paste -s
done | sort  -k15,15g  > data/srilm_interp/perplexities.txt

# for basic decoding, let's use only a trigram LM
[ -d data/lang_test/ ] && rm -rf data/lang_test
cp -R data/lang data/lang_test
lm=$(cat data/srilm/perplexities.txt | grep 3gram | head -n1 | awk '{print $1}'\
)
local/arpa2G.sh $lm data/lang_test data/lang_test
-uu-:---F1  prepare_lm.sh   29% L24     (Shell-script[bash])--------------------
```

local/arpa2G.sh

Terminal

```
$decompress | \
  grep -v '<s> <s>' | grep -v '</s> <s>' |  grep -v '</s> </s>' | \
  arpa2fst - | \
  fstprint | \
  utils/eps2disambig.pl | \
  utils/s2eps.pl | \
  fstcompile --isymbols=$langdir/words.txt \
  --osymbols=$langdir/words.txt  --keep_isymbols=false --keep_osymbols=false | \
\
 fstrmepsilon | fstarcsort --sort_type=olabel > $destdir/G.fst || exit 1
fstisstochastic $destdir/G.fst || true;

if $cleanup; then
  rm $destdir/lm_tmp.gz  2>/dev/null || true;
fi

exit 0

-uu-:---F1  arpa2G.sh       Bot L108    (Shell-script[bash])----------------------
```

Building an STT System with Kaldi

- Data preparation
  - Acoustic model training data
  - Pronunciation lexicon
  - Language model training data
- Basic GMM system building
  - **Acoustic model training**
  - Language model training
- Basic Decoding
  - Creating a static decoding graph
  - Lattice rescoring
- Basic DNN system building
- Going beyond the basics

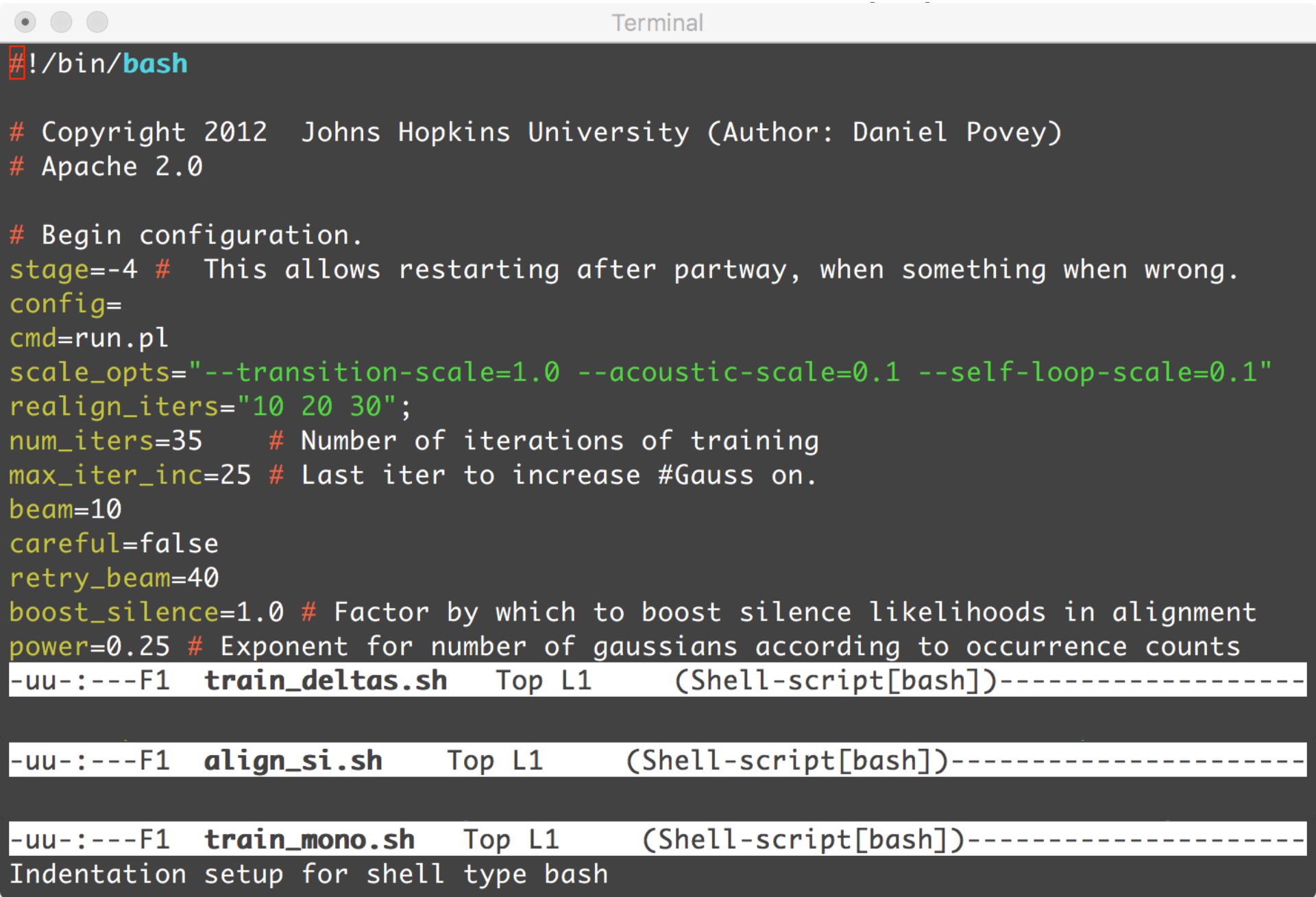
GMM Training (1)

Terminal

Terminal

```
if [ $stage -le 3 ]; then
  ### Monophone
  echo "Starting monophone training."
  utils/subset_data_dir.sh data/train 1000 data/train.1k
  steps/train_mono.sh --nj $nj --cmd "$train_cmd" data/train.1k data/lang exp/m\
ono
  echo "Mono training done."

-uu-:---F1   run.sh          19% L52     (Shell-script[bash])----------------------
if [ $stage -le 4 ]; then
  ### Triphone
  echo "Starting triphone training."
  steps/align_si.sh --nj $nj --cmd "$train_cmd" \
      data/train data/lang exp/mono exp/mono_ali
  steps/train_deltas.sh --boost-silence 1.25  --cmd "$train_cmd"  \
      3200 30000 data/train data/lang exp/mono_ali exp/tri1
  echo "Triphone training done."

-uu-:---F1   run.sh          28% L67     (Shell-script[bash])----------------------
```

-uu-:---F1 run.sh 19% L64 (Shell-script[bash])----------


Terminal
#!/bin/bash
# Copyright 2012 Johns Hopkins University (Author: Daniel Povey)
# Apache 2.0
# Begin configuration.
stage=-4 # This allows restarting after partway, when something when wrong.
config=
cmd=run.pl
scale_opts="--transition-scale=1.0 --acoustic-scale=0.1 --self-loop-scale=0.1"
realign_iters="10 20 30";
num_iters=35   # Number of iterations of training
max_iter_inc=25 # Last iter to increase #Gauss on.
beam=10
careful=false
retry_beam=40
boost_silence=1.0 # Factor by which to boost silence likelihoods in alignment
power=0.25 # Exponent for number of gaussians according to occurrence counts
-uu-:---F1 train_deltas.sh   Top L1   (Shell-script[bash])----------
-uu-:---F1 align_si.sh   Top L1   (Shell-script[bash])----------
-uu-:---F1 train_mono.sh   Top L1   (Shell-script[bash])----------
Indentation setup for shell type bash

cluster-phones, compile-questions, build-tree

Terminal

```
if [ $stage -le -2 ]; then
  echo "$0: getting questions for tree-building, via clustering"
  # preparing questions, roots file...
  cluster-phones $context_opts $dir/treeacc $lang/phones/sets.int \
    $dir/questions.int 2> $dir/log/questions.log || exit 1;
  cat $lang/phones/extra_questions.int >> $dir/questions.int
  compile-questions $context_opts $lang/topo $dir/questions.int \
    $dir/questions.qst 2>$dir/log/compile_questions.log || exit 1;

  echo "$0: building the tree"
  $cmd $dir/log/build_tree.log \
    build-tree $context_opts --verbose=1 --max-leaves=$numleaves \
    --cluster-thresh=$cluster_thresh $dir/treeacc $lang/phones/roots.int \
    $dir/questions.qst $lang/topo $dir/tree || exit 1;

  $cmd $dir/log/init_model.log \
    gmm-init-model  --write-occs=$dir/1.occs  \
      $dir/tree $dir/treeacc $lang/topo $dir/1.mdl || exit 1;
-uu-:---F1  train_deltas.sh   48% L94    (Shell-script[bash])--------------------
```

GMM Training (4)

Terminal

```
  ### Triphone + LDA and MLLT
  # Training
  echo "Starting LDA+MLLT training."
  steps/align_si.sh  --nj $nj --cmd "$train_cmd"  \
      data/train data/lang exp/tri2a exp/tri2a_ali

  steps/train_lda_mllt.sh  --cmd "$train_cmd"  \
    --splice-opts "--left-context=3 --right-context=3" \
-uu-:---F1  run.sh          54% L116   (Shell-script[bash])----------------------
  ### Triphone + LDA and MLLT + SAT and FMLLR
  # Training
  echo "Starting SAT+FMLLR training."
  steps/align_si.sh  --nj $nj --cmd "$train_cmd" \
      --use-graphs true data/train data/lang exp/tri2b exp/tri2b_ali
  steps/train_sat.sh --cmd "$train_cmd" 4200 40000 \
      data/train data/lang exp/tri2b_ali exp/tri3b
  echo "SAT+FMLLR training done."

-uu-:---F1  run.sh          67% L138   (Shell-script[bash])----------------------
```

GMM Training (5)

Terminal

```
fi


if [ $stage -le 8 ]; then
  echo "Starting SGMM training."
  steps/align_fmllr.sh --nj $nj --cmd "$train_cmd" \
      data/train data/lang exp/tri3b exp/tri3b_ali

  steps/train_ubm.sh  --cmd "$train_cmd"  \
      600 data/train data/lang exp/tri3b_ali exp/ubm5b2

  steps/train_sgmm2.sh  --cmd "$train_cmd"  \
       5200 12000 data/train data/lang exp/tri3b_ali exp/ubm5b2/final.ubm exp/s\
gmm2_5b2
  echo "SGMM training done."

  (
  echo "Decoding the dev set using SGMM models"
-uu-:---F1  run.sh          80% L167   (Shell-script[bash])----------------------
```

Building an STT System with Kaldi

- Data preparation
  - Acoustic model training data
  - Pronunciation lexicon
  - Language model training data
- Basic GMM system building
  - Acoustic model training
  - Language model training
- Basic Decoding
  - **Creating a static decoding graph**
  - Lattice rescoring
- Basic DNN system building
- Going beyond the basics

Building HCLG (1)

Terminal

```
#!/bin/bash
# Copyright 2010-2012 Microsoft Corporation
#           2012-2013 Johns Hopkins University (Author: Daniel Povey)
# Apache 2.0

# This script creates a fully expanded decoding graph (HCLG) that represents
# all the language-model, pronunciation dictionary (lexicon), context-dependenc\
y,
-uu-:---F1  mkgraph.sh      Top L1     (Shell-script[bash])----------------------
if [[ ! -s $lang/tmp/LG.fst || $lang/tmp/LG.fst -ot $lang/G.fst || \
      $lang/tmp/LG.fst -ot $lang/L_disambig.fst ]]; then
  fsttablecompose $lang/L_disambig.fst $lang/G.fst | fstdeterminizestar --use-l\
og=true | \
    fstminimizeencoded | fstpushspecial | \
    fstarcsort --sort_type=ilabel > $lang/tmp/LG.fst || exit 1;
  fstisstochastic $lang/tmp/LG.fst || echo "[info]: LG not stochastic."
fi

-uu-:---F1  mkgraph.sh      46% L74    (Shell-script[bash])----------------------
```

Building HCLG (2)

Terminal

```
clg=$lang/tmp/CLG_${N}_${P}.fst

if [[ ! -s $clg || $clg -ot $lang/tmp/LG.fst ]]; then
  fstcomposecontext --context-size=$N --central-position=$P \
   --read-disambig-syms=$lang/phones/disambig.int \
   --write-disambig-syms=$lang/tmp/disambig_ilabels_${N}_${P}.int \
    $lang/tmp/ilabels_${N}_${P} < $lang/tmp/LG.fst |\
    fstarcsort --sort_type=ilabel > $clg
  fstisstochastic $clg  || echo "[info]: CLG not stochastic."
fi

if [[ ! -s $dir/Ha.fst || $dir/Ha.fst -ot $model  \
    || $dir/Ha.fst -ot $lang/tmp/ilabels_${N}_${P} ]]; then
  if $reverse; then
    make-h-transducer --reverse=true --push_weights=true \
      --disambig-syms-out=$dir/disambig_tid.int \
-uu-:---F1  mkgraph.sh      53% L87     (Shell-script[bash])----------------------
```

Building HCLG (3)

Terminal

```
if [[ ! -s $dir/HCLGa.fst || $dir/HCLGa.fst -ot $dir/Ha.fst || \
      $dir/HCLGa.fst -ot $clg ]]; then
  if $remove_oov; then
    [ ! -f $lang/oov.int ] && \
      echo "$0: --remove-oov option: no file $lang/oov.int" && exit 1;
    clg="fstrmsymbols --remove-arcs=true --apply-to-output=true $lang/oov.int $\
clg|"
  fi
  fsttablecompose $dir/Ha.fst "$clg" | fstdeterminizestar --use-log=true \
    | fstrmsymbols $dir/disambig_tid.int | fstrmepslocal | \
     fstminimizeencoded > $dir/HCLGa.fst || exit 1;
  fstisstochastic $dir/HCLGa.fst || echo "HCLGa is not stochastic"
fi

if [[ ! -s $dir/HCLG.fst || $dir/HCLG.fst -ot $dir/HCLGa.fst ]]; then
  add-self-loops --self-loop-scale=$loopscale --reorder=true \
    $model < $dir/HCLGa.fst > $dir/HCLG.fst || exit 1;
```

-uu-:---F1 **mkgraph.sh** 70% L112 (Shell-script[bash])----------

Building HCLG (4)

Terminal

```
fi

if [[ ! -s $dir/HCLG.fst || $dir/HCLG.fst -ot $dir/HCLGa.fst ]]; then
  add-self-loops --self-loop-scale=$loopscale --reorder=true \
    $model < $dir/HCLGa.fst > $dir/HCLG.fst || exit 1;

  if [ $tscale == 1.0 -a $loopscale == 1.0 ]; then
    # No point doing this test if transition-scale not 1, as it is bound to fai\
l.
    fstisstochastic $dir/HCLG.fst || echo "[info]: final HCLG is not stochastic\
."
  fi
fi

# keep a copy of the lexicon and a list of silence phones with HCLG...
# this means we can decode without reference to the $lang directory.


-uu-:---F1  mkgraph.sh      80% L125    (Shell-script[bash])----------------------
```

Decoding and Lattice Rescoring

Terminal

Terminal

```
  # Graph compilation
  utils/mkgraph.sh data/lang_test exp/sgmm2_5b2 exp/sgmm2_5b2/graph
  utils/mkgraph.sh data/lang_big/ exp/sgmm2_5b2 exp/sgmm2_5b2/graph_big

  steps/decode_sgmm2.sh --nj $dev_nj --cmd "$decode_cmd" \
      --transform-dir exp/tri3b/decode_dev \
      exp/sgmm2_5b2/graph data/dev exp/sgmm2_5b2/decode_dev

  steps/lmrescore_const_arpa.sh  --cmd "$decode_cmd" \
      data/lang_test/ data/lang_big/ data/dev \
      exp/sgmm2_5b2/decode_dev exp/sgmm2_5b2/decode_dev.rescored

  steps/decode_sgmm2.sh --nj $dev_nj --cmd "$decode_cmd" \
      --transform-dir exp/tri3b/decode_dev \
      exp/sgmm2_5b2/graph_big data/dev exp/sgmm2_5b2/decode_dev.big
  echo "SGMM decoding done."
  ) &
fi
-uu-:---F1   run.sh          88% L184    (Shell-script[bash])----------------------
```

steps/decode_sgmm2.sh

Terminal

```
#!/bin/bash

# Copyright 2012  Johns Hopkins University (Author: Daniel Povey).  Apache 2.0.

# This script does decoding with an SGMM system, with speaker vectors.
# If the SGMM system was
# built on top of fMLLR transforms from a conventional system, you should
# provide the --transform-dir option.

# Begin configuration section.
stage=1
transform_dir=    # dir to find fMLLR transforms.
nj=4 # number of decoding jobs.
acwt=0.1  # Just a default value, used for adaptation and beam-pruning..
cmd=run.pl
beam=13.0
gselect=15  # Number of Gaussian-selection indices for SGMMs.  [Note:
            # the first_pass_gselect variable is used for the 1st pass of
-uu-:---F1  decode_sgmm2.sh    Top L1     (Shell-script[bash])------------------
```

Building an STT System with Kaldi

- Data preparation
  - Acoustic model training data
  - Pronunciation lexicon
  - Language model training data
- Basic GMM system building
  - Acoustic model training
  - Language model training
- Basic Decoding
  - Creating a static decoding graph
  - **Lattice rescoring**
- Basic DNN system building
- Going beyond the basics

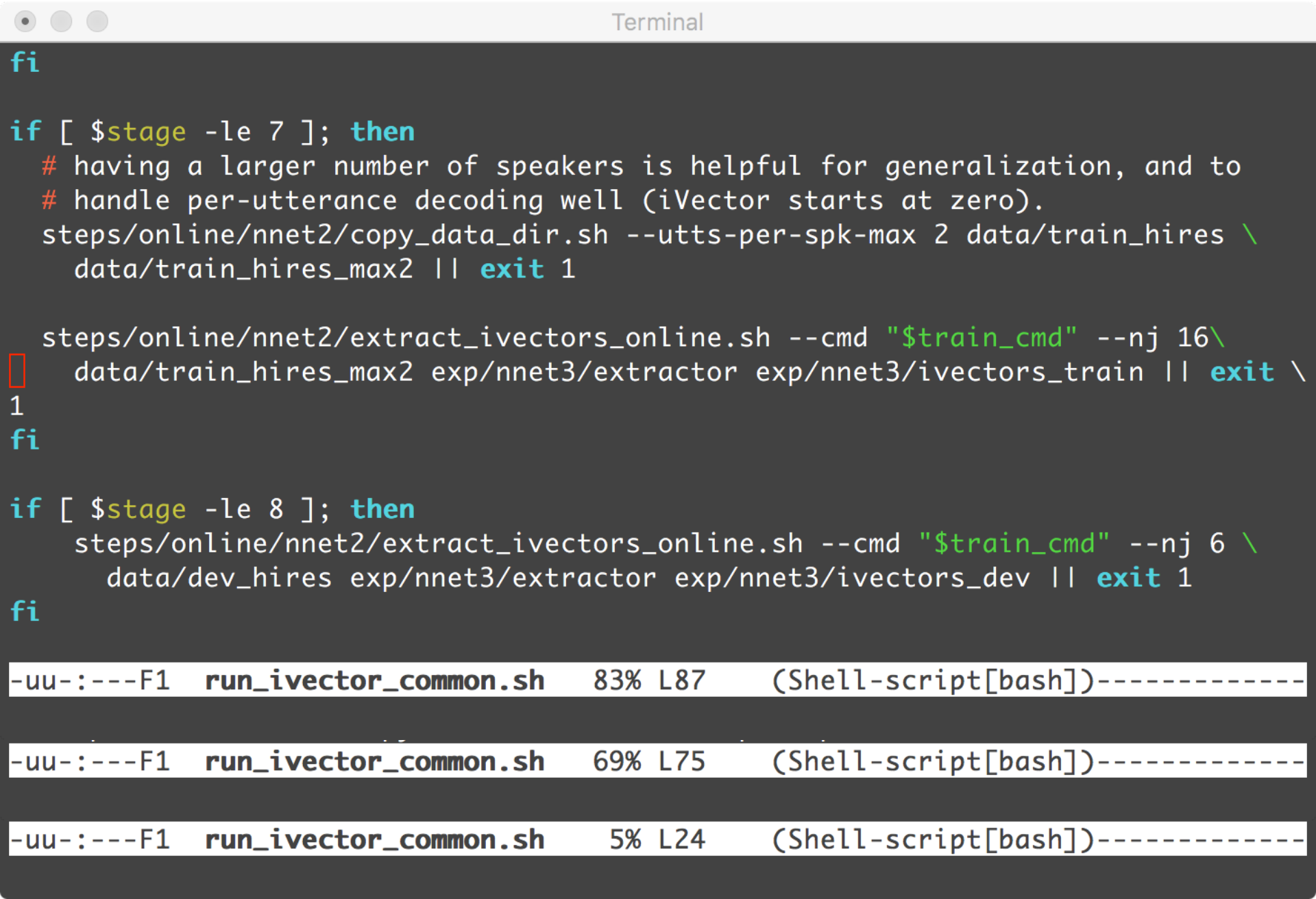
steps/lmrescore_const_arpa.sh

Terminal

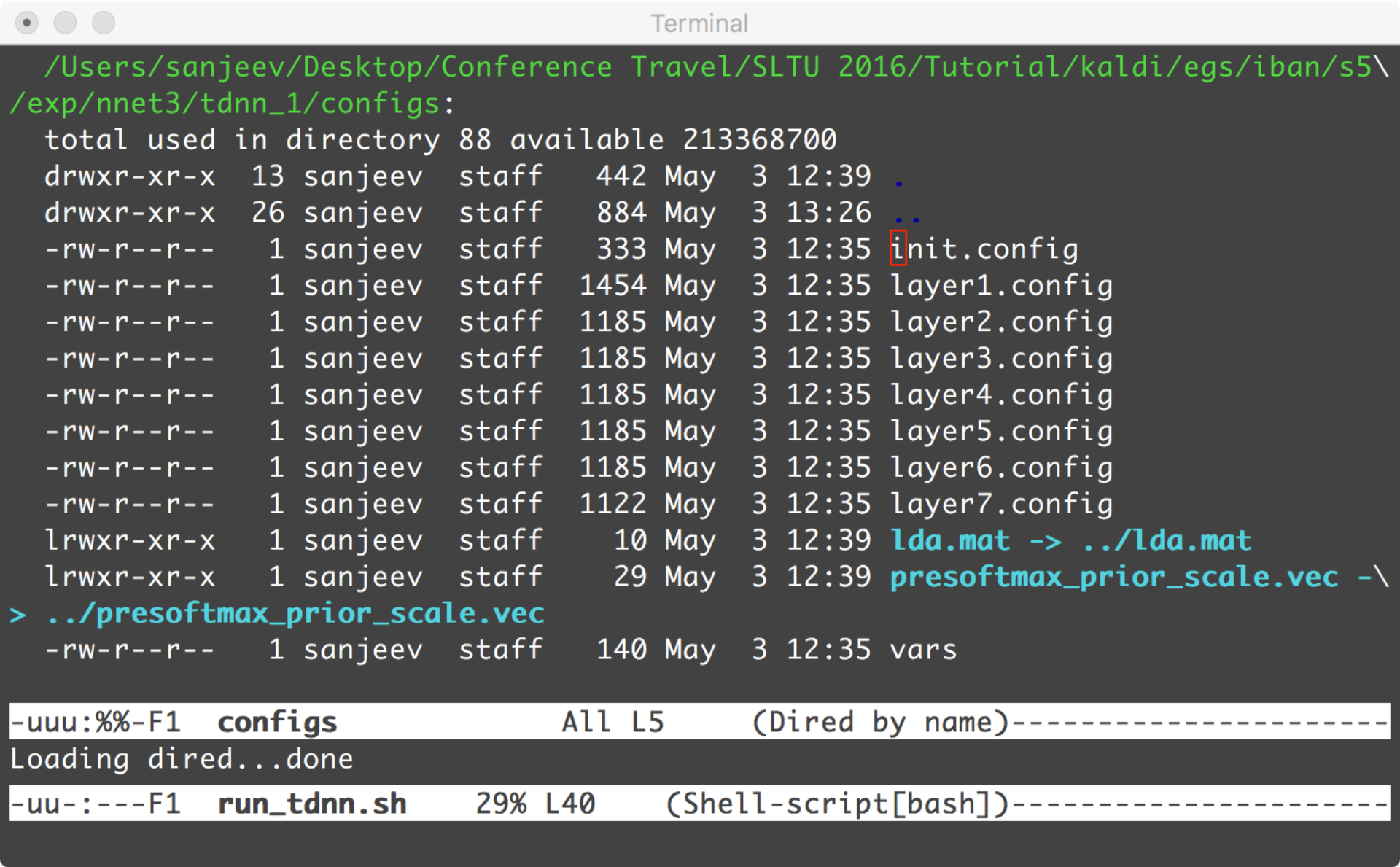
```
if ! cmp -s $oldlang/words.txt $newlang/words.txt; then
  echo "$0: $oldlang/words.txt and $newlang/words.txt differ: make sure you kno\
w what you are doing.";
fi

oldlmcommand="fstproject --project_output=true $oldlm |"

-uu-:---F1  lmrescore_const_arpa.sh    55% L46     (Shell-script[bash])----------

if [ $stage -le 1 ]; then
  $cmd JOB=1:$nj $outdir/log/rescorelm.JOB.log \
    lattice-lmrescore --lm-scale=-1.0 \
    "ark:gunzip -c $indir/lat.JOB.gz|" "$oldlmcommand" ark:-  \| \
    lattice-lmrescore-const-arpa --lm-scale=1.0 \
    ark:- "$newlm" "ark,t:|gzip -c>$outdir/lat.JOB.gz" || exit 1;
fi

-uu-:---F1  lmrescore_const_arpa.sh    70% L57     (Shell-script[bash])----------
```

Building an STT System with Kaldi

- Data preparation
  - Acoustic model training data
  - Pronunciation lexicon
  - Language model training data
- Basic GMM system building
  - Acoustic model training
  - Language model training
- Basic Decoding
  - Creating a static decoding graph
  - Lattice rescoring
- **Basic DNN system building**
- Going beyond the basics


Terminal
fi
if [$stage -le 7]; then
# having a larger number of speakers is helpful for generalization, and to
# handle per-utterance decoding well (iVector starts at zero).
steps/online/nnet2/copy_data_dir.sh --utts-per-spk-max 2 data/train_hires \
data/train_hires_max2 || exit 1
steps/online/nnet2/extract_ivectors_online.sh --cmd "$train_cmd" --nj 16\
data/train_hires_max2 exp/nnet3/extractor exp/nnet3/ivectors_train || exit \
1
fi
if [$stage -le 8]; then
steps/online/nnet2/extract_ivectors_online.sh --cmd "$train_cmd" --nj 6 \
data/dev_hires exp/nnet3/extractor exp/nnet3/ivectors_dev || exit 1
fi
-uu-:---F1 run_ivector_common.sh 83% L87 (Shell-script[bash])----------
-uu-:---F1 run_ivector_common.sh 69% L75 (Shell-script[bash])----------
-uu-:---F1 run_ivector_common.sh 5% L24 (Shell-script[bash])----------

steps/nnet3/tdnn/make_configs.py



steps/nnet3/train_dnn.py

Terminal

```
steps/nnet3/train_dnn.py --stage $train_stage \
  --cmd="$decode_cmd" \
  --trainer.optimization.num-jobs-initial 2 \
  --trainer.optimization.num-jobs-final 4 \
  --trainer.num-epochs 4 \
  --trainer.add-layers-period 1 \
  --feat.online-ivector-dir exp/nnet3/ivectors_train\
  --feat.cmvn-opts "--norm-means=false --norm-vars=false" \
  --trainer.num-epochs 2 \
  --trainer.optimization.initial-effective-lrate 0.005 \
  --trainer.optimization.final-effective-lrate 0.0005 \
  --trainer.samples-per-iter 120000 \
  --cleanup.preserve-model-interval 10 \
  --feat-dir data/train_hires \
  --ali-dir exp/nnet3/tri3b_ali_sp \
  --lang data/lang \
  --dir=$dir  || exit 1;
-uu-:---F1  run_tdnn.sh     51% L60     (Shell-script[bash])----------------------
```

Building an STT System with Kaldi

- Data preparation
  - Acoustic model training data
  - Pronunciation lexicon
  - Language model training data
- Basic GMM system building
  - Acoustic model training
  - Language model training
- Basic Decoding
  - Creating a static decoding graph
  - Lattice rescoring
- Basic DNN system building
- **Going beyond the basics**

Advanced Methods:

Staying Ahead in the STT Game

- STT technology is advancing very rapidly
  - Amazon, Apple, Baidu, Facebook, Google, Microsoft
- Kaldi leads and keeps up with major innovations
  - From SGMMs to DNN (2012)
  - From “English” to low-resource languages (2013)
  - From CPUs to GPUs (2014)
  - From close-talking to far-field microphones (2015)
  - From well-curated to “wild type” corpora (2016)
- A preview of some upcoming developments

Advanced Methods:

Staying Ahead in the STT Game

- STT technology is advancing very rapidly
  - Amazon, Apple, Baidu, Facebook, Google, Microsoft
- Kaldi leads and keeps up with major innovations
  - **From SGMMs to DNN (2012)**
  - From "English" to low-resource languages (2013)
  - From CPUs to GPUs (2014)
  - From close-talking to far-field microphones (2015)
  - From well-curated to "wild type" corpora (2016)
- A preview of some upcoming developments

Deep Neural Networks for STT

INTERSPEECH 2011

Conversational Speech Transcription Using Context-Dependent Deep Neural Networks

Frank Seide[1], *Gang Li*,[1] *and Dong Yu*[2]

[1]Microsoft Research Asia, Beijing, P.R.C.
[2]Microsoft Research, Redmond, USA
{fseide,ganl,dongyu}@microsoft.com

Abstract

We apply the recently proposed Context-Dependent Deep-Neural-Network HMMs, or CD-DNN-HMMs, to speech-to-text transcription. For single-pass speaker-independent recognition on the RT03S Fisher portion of phone-call transcription benchmark (Switchboard), the word-error rate is reduced from 27.4%, obtained by discriminatively trained Gaussian-mixture HMMs, to 18.5%—a 33% relative improvement.

CD-DNN-HMMs combine classic artificial-neural-network HMMs with traditional tied-state triphones and deep-belief-network pre-training. They had previously been shown to reduce errors by 16% relatively when trained on tens of hours of data using hundreds of tied states. This paper takes CD-DNN-HMMs further and applies them to transcription using over 300 hours of training data, over 9000 tied states, and up to 9 hidden layers, and demonstrates how sparseness can be exploited.

ers), and task (from voice queries to speech-to-text transcription). This is demonstrated on a publicly available benchmark, the Switchboard phone-call transcription task (2000 NIST Hub5 and RT03S sets). We should note here that ANNs have been trained on up to 2000 hours of speech before [7], but with much fewer output units (monophones) and fewer hidden layers.

Second, we advance the CD-DNN-HMMs by introducing weight sparseness and the related learning strategy and demonstrate that this can reduce recognition error or model size.

Third, we present the statistical view of the multi-layer perceptron (MLP) and DBN and provide empirical evidence for understanding which factors contribute most to the accuracy improvements achieved by the CD-DNN-HMMs.

2. The Context-Dependent Deep Neural Network HMM

DNN Acoustic Models for the Masses

- Nontrivial to get the DNN models to work well
  - Design decisions: # layers, # nodes, # outputs, type of nonlinearity, training criterion
  - Training art: learning rates, regularization, update stability (max change), data randomization, # epochs
  - Computational art: matrix libraries, memory mgmt
- Kaldi recipes provide a robust starting point

Corpus	Training Speech	SGMM WER	DNN WER
BABEL Pashto	10 hours	69.2%	67.6%
BABEL Pashto	80 hours	50.2%	42.3%
Fisher English	2000 hours	15.4%	10.3%

Advanced Methods:

Staying Ahead in the STT Game

- STT technology is advancing very rapidly
  - Amazon, Apple, Baidu, Facebook, Google, Microsoft
- Kaldi leads and keeps up with major innovations
  - From SGMMs to DNN (2012)
  - **From "English" to low-resource languages (2013)**
  - From CPUs to GPUs (2014)
  - From close-talking to far-field microphones (2015)
  - From well-curated to "wild type" corpora (2016)
- A preview of some upcoming developments

Low-Resource STT for the Masses

- Kaldi provides language-independent recipes
  - Typical BABEL Full LP condition
    - 80 hours of transcribed speech, 800K words of LM text, 20K word pronunciation lexicon
  - Typical BABEL Limited LP condition
    - 10 hours of transcribed speech, 100K words of LM text, 6K word pronunciation lexicon

Language	Cantonese		Tagalog		Pashto		Turkish	
Speech	80h	10h	80h	10h	80h	10h	80h	10h
CER/WER	48.5%	61.2%	46.3%	61.9%	50.7%	63.0%	51.3%	65.3%
ATWV	0.47	0.26	0.56	0.28	0.46	0.25	0.52	0.25

Advanced Methods:

Staying Ahead in the STT Game

- STT technology is advancing very rapidly
  - Amazon, Apple, Baidu, Facebook, Google, Microsoft
- Kaldi leads and keeps up with major innovations
  - From SGMMs to DNN (2012)
  - From "English" to low-resource languages (2013)
  - **From CPUs to GPUs (2014)**
  - From close-talking to far-field microphones (2015)
  - From well-curated to "wild type" corpora (2016)
- A preview of some upcoming developments

Parallel (GPU-based) Training

- Original neural network training algorithms were inherently sequential (e.g. SGD)
- Scaling up to “big data” becomes a challenge
- Several solutions have emerged recently
  - 2009: Delayed SGD (Yahoo!)
  - 2011: Lock-free SGD (Hogwild! U Wisconsin)
  - 2012: Gradient averaging (DistBelief, Google)
  - 2014: Model averaging (NG-SGD, Kaldi)

PARALLEL TRAINING OF DNNS WITH NATURAL GRADIENT AND PARAMETER AVERAGING

Daniel Povey, Xiaohui Zhang & Sanjeev Khudanpur
Center for Language and Speech Processing & Human Language Technology Center of Excellence,
The Johns Hopkins University, Baltimore, MD 21218, USA
{dpovey@gmail.com}, {xiaohui,khudanpur@jhu.edu}

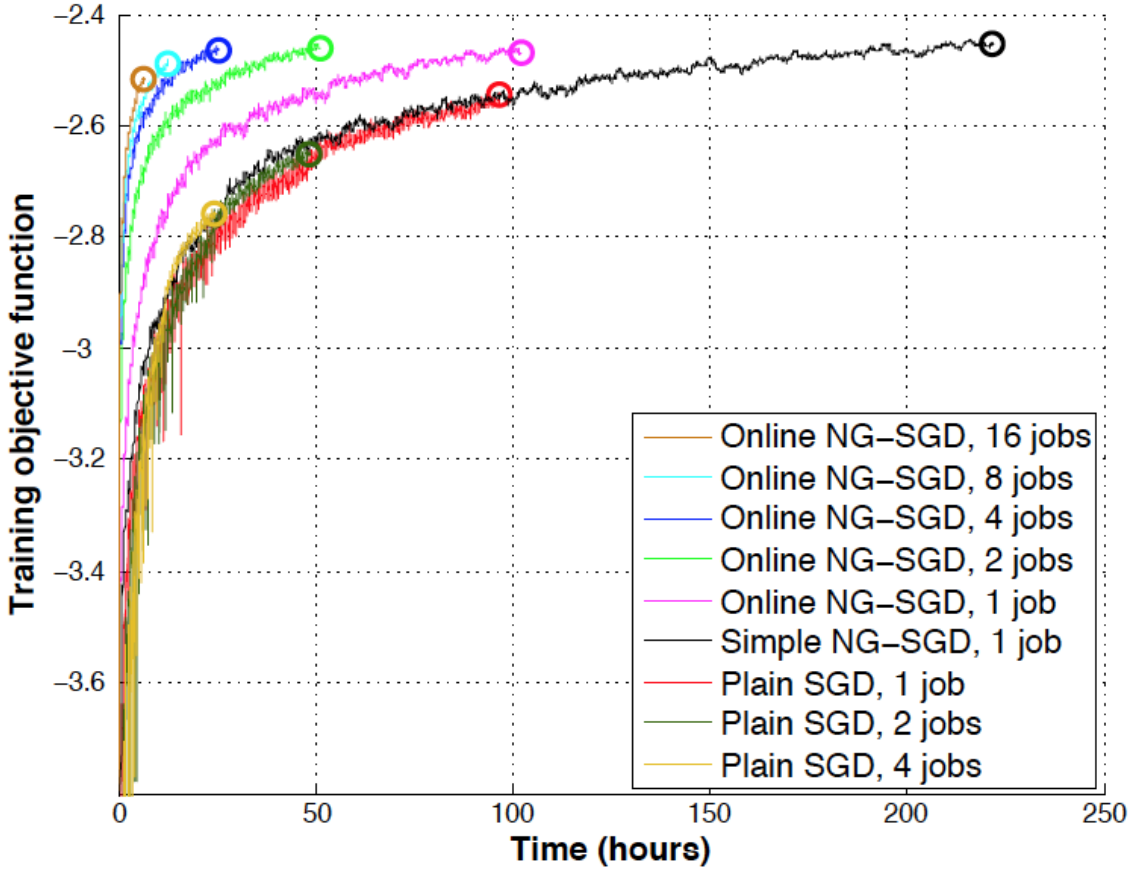
ABSTRACT

We describe the neural-network training framework used in the Kaldi speech recognition toolkit, which is geared towards training DNNs with large amounts of training data using multiple GPU-equipped or multi-core machines. In order to be as hardware-agnostic as possible, we needed a way to use multiple machines without generating excessive network traffic. Our method is to average the neural network parameters periodically (typically every minute or two), and redistribute the averaged parameters to the machines for further training. Each machine sees different data. By itself, this method does not work very well. However, we have another method, an approximate and efficient implementation of Natural Gradient for Stochastic Gradient Descent (NG-SGD), which seems to allow our periodic-averaging method to work well, as well as substantially improving the convergence of SGD on a single machine.

Model Averaging with NG-SGD

- Train DNNs with large amount of data
  - Utilize a cluster of CPUs or GPUs
  - Minimize network traffic (esp. for CPUs)
- Solution: exploit data parallelization
  - Update model in parallel over many mini-batches
  - Infrequently average models (parameters)
- Use “Natural-Gradient” SGD for model updating
  - Approximates conditioning via inverse Fisher matrix
  - Improves convergence even without parallelization

Parallelization Matters!



- Typically, a GPU is 10x faster than a 16 core CPU
- Linear speed-up till ca 4 GPUs, then diminishing

Advanced Methods:

Staying Ahead in the STT Game

- STT technology is advancing very rapidly
  - Amazon, Apple, Baidu, Facebook, Google, Microsoft
- Kaldi leads and keeps up with major innovations
  - From SGMMs to DNN (2012)
  - From "English" to low-resource languages (2013)
  - From CPUs to GPUs (2014)
  - **From close-talking to far-field microphones (2015)**
  - From well-curated to "wild type" corpora (2016)
- A preview of some upcoming developments

IARPA's Open Challenge

- Automatic speech recognition software that works in a variety of acoustic environments and recording scenarios is a holy grail of the speech research community.

- IARPA's Automatic Speech recognition In Reverberant Environments (ASpIRE) Challenge is seeking that grail.

Rules of the ASpIRE Challenge

- 15 hours of speech data were posted on the IARPA website
  - Multi-microphone recordings of conversational English
  - 5h development set (**dev**), 10h development-test set (**dev-test**)
  - Transcriptions provided for dev, only scoring for dev-test output
  - For training data selection, system development and tuning
- 12 hours of new speech data during the evaluation period
  - Far-field speech (**eval**) from noisy, reverberant rooms
  - Single-microphone or multi-microphone conditions
- Word error rate is the measure of performance
  - Single-microphone submissions were due on 02/18/2015
  - Results were officially announced on 09/10/2015

Examples of ASpIRE Audio

- Typical sample
  - Suggested by Dr. Mary Harper
- Almost manageable
  - Easy for humans, 26% errors for ASR
- Somewhat hard
  - Easy for humans, 41% errors for ASR
- Much harder
  - Not easy for humans, 60% errors for ASR
- *#@! !#% #%^^
  - Very hard for humans, no ASR output

Kaldi ASR Improvements for ASpIRE

- **Time delay neural networks** (TDNN)
  - A way to deal with long acoustic-phonetic context
  - A structured alternative to deep/recurrent neural nets
- **Data augmentation** with simulated reverberations
  - A way to mitigate channel distortions not seen in training
  - A form of multi-condition training of ASR models
- **i-vector based** speaker & environment **adaptation**
  - A way to deal with speaker & channel variability
  - Adapted [with a twist] from Speaker ID systems

Kaldi ASR Improvements, ASpIRE++

- **Pronunciation** and inter-word silence **modeling**
  - Inspired by pronunciation-prosody interactions
  - A simple context-dependent model of inter-word silence
- Recurrent neural network language models (RNNLM)
  - A (known) way to model long-range word dependencies
  - Incorporated post-submission into JHU ASpIRE system
- Ongoing Kaldi investigations that hold promise
  - Semi-supervised discriminative training of (T)DNNs
  - Long short-term memory (LSTM) acoustic models
  - Connectionist temporal classification (CTC) models

Time Delay Neural Networks

(See our paper at INTERSPEECH 2015 for details)

A time delay neural network architecture for efficient modeling of long temporal contexts

Vijayaditya Peddinti[1], *Daniel Povey*[1,2], *Sanjeev Khudanpur*[1,2]

[1]Center for Language and Speech Processing &
[2]Human Language Technology Center of Excellence
Johns Hopkins University, Baltimore, MD 21218, USA

vijay.p,khudanpur@jhu.edu, dpovey@gmail.com

Abstract

Recurrent neural network architectures have been shown to efficiently model long term temporal dependencies between acoustic events. However the training time of recurrent networks is higher than feedforward networks due to the sequential nature of the learning algorithm. In this paper we propose a time delay neural network architecture which models long term temporal dependencies with training times comparable to standard feed-forward DNNs. The network uses sub-sampling to reduce computation during training. On the Switchboard task we show a relative improvement of 6% over the baseline DNN model. We present results on several LVCSR tasks with training data ranging from 3 to 1800 hours to show the effectiveness of the TDNN architecture in learning wider temporal dependencies in both small and large data scenarios.

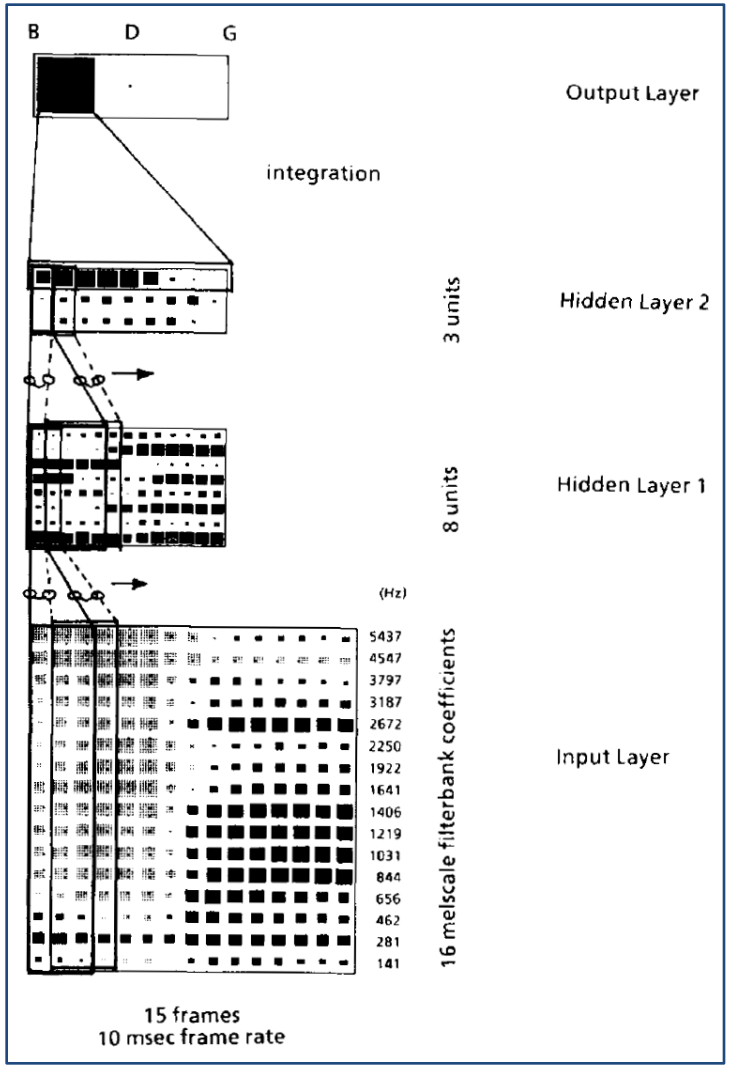
Index Terms: time delay neural networks, acoustic modeling, recurrent neural networks

be reduced, while ensuring that information from all time steps in the input context is processed by the network.

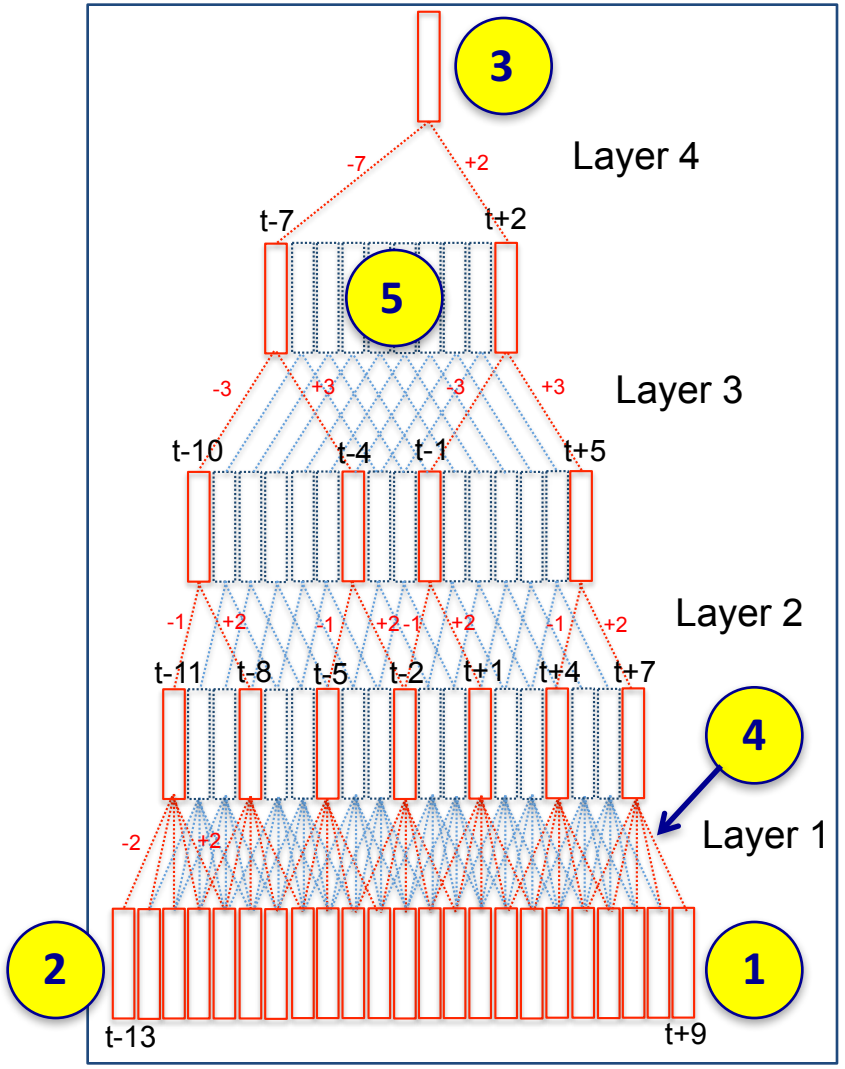
Neural network architectures have been shown to benefit from speaker adaptation. However, speaker adaptation techniques like fMLLR [4] require two passes of decoding. The 2-pass decoding strategy is difficult to use in online speech recognition applications. iVectors which capture both speaker and environment specific information have been shown to be useful for instantaneous and discriminative adaptation of the neural network [5, 6]. In this paper we use iVector based neural network adaptation.

The paper is organized as follows. Section 2 mentions relevant work, Section 3 describes the neural network architecture and training recipe in greater detail. Section 4 describes the experimental setup. Section 5 presents and analyzes the results primarily on the Switchboard [7] task. It also presents results on other LVCSR tasks which have 3-1800 hours of training data. Section 6 presents the conclusions and the future work.

A 28 Year Old Idea, Resurrected



Alex Waibel, Kevin Lang, et al (1987)



Our TDNN Architecture (2015)

Improved ASR on Several Data Sets

Standard ASR Test Sets	Size	DNN	TDNN	Rel. Δ
Wall Street Journal	80 hrs	6.6%	6.2%	5%
TED-LIUM	118 hrs	19.3%	17.9%	7%
Switchboard	300 hrs	15.5%	14.0%	**10%**
Libri Speech	960 hrs	5.2%	4.8%	7%
Fisher English	1800 hrs	22.2%	21.0%	**5%**

- Consistent 5-10% reduction in word error rate (**WER**) over DNNs on most datasets, including conversational speech.
- TDNN training speeds are on par with DNN, and nearly an order of magnitude faster than RNN

ASpIRE (Fisher Training)	1800 hrs	47.7%	47.6%	

Data Augmentation for ASR Training

(See our paper at INTERSPEECH 2015 for details)

Audio Augmentation for Speech Recognition

Tom Ko[1], *Vijayaditya Peddinti*[2], *Daniel Povey*[2,3], *Sanjeev Khudanpur*[2,3]

[1]Huawei Noah's Ark Research Lab, Hong Kong, China
[2]Center for Language and Speech Processing &
[3]Human Language Technology Center of Excellence,
Johns Hopkins University, Baltimore, MD, 21218, USA

{tomkocse,dpovey}@gmail.com, {vijay.p,khudanpur}@jhu.edu

Abstract

Data augmentation is a common strategy adopted to increase the quantity of training data, avoid overfitting and improve robustness of the models. In this paper, we investigate audio-level speech augmentation methods which directly process the raw signal. The method we particularly recommend is to change the speed of the audio signal, producing 3 versions of the original signal with speed factors of 0.9, 1.0 and 1.1. The proposed technique has a low implementation cost, making it easy to adopt. We present results on 4 different LVCSR tasks with training data ranging from 100 hours to 1000 hours, to examine the effectiveness of audio augmentation in a variety of data scenarios. An average relative improvement of 4.3% was observed across the 4 tasks.

Index Terms: speech recognition, data augmentation, deep neural network

2. Audio perturbation

In this section we describe a speed-perturbation technique for data augmentation and compare it with the existing augmentation technique VTLP [3]. Speed perturbation produces a warped time signal. Given an audio signal $x(t)$, time warping by a factor α gives the signal $x(\alpha t)$. It can be seen from the Fourier transform of $x(\alpha t)$, $\alpha^{-1}\hat{x}(\alpha^{-1}\omega)$, that the warping factor produces shifts in the frequency components of the $\hat{x}(\omega)$ by an amount proportional to frequency ω. In [8] it was shown that this corresponds approximately to a shift of the spectrum in the mel spectrogram, since the mel scale is approximately logarithmic. It can be seen that these changes in the mel spectrogram are similar to those produced using VTLP. However, unlike VTLP, speed perturbation results in a change in the duration of the signal which also affects the number of frames in the utterance.

Simulating Reverberant Speech for Multi-condition (T)DNN Training

- Simulate ca 5500 hours of reverberant, noisy data from 1800 hours of the Fisher English CTS corpus
  - Replicate each of the ca 21,000 conversation sides 3 times
  - Randomly change the sampling rate [up to ±10%]
  - Convolve each conversation side with one of 320 real-life room impulse responses (RIR) chosen at random
  - Add noise to the signal (when available with the RIR)
- Generate (T)DNN training labels from clean speech
  - Align "pre-reverb" speech to ca 7500 CD-HMM states
- Train DNN and TDNN acoustic models
  - Cross-entropy training followed by sequence training

Result of Data Augmentation

Acoustic Model	Data Augmentation	Dev WER
TDNN A (230 ms)	None (1800h, clean speech)	47.6%
TDNN A (230 ms)	+ 3 x (reverberation + noise)	31.7%
TDNN B (290 ms)	+ 3 x (reverberation + noise)	**30.8%**
TDNN A (230 ms)	+ sampling rate perturbation	31.0%
TDNN B (290 ms)	+ sampling rate perturbation	31.1%

- Data augmentation with simulated reverberation is beneficial
  - Likely to be a very important reason for relatively good performance
- Sampling rate perturbation didn't help much on ASpIRE data
- Sequence training helped reduce WER on the dev set
  - Required modifying the sMBR training criterion to realize gains
  - But the **gains did not carry over** to dev-test set

i-vectors for Speaker Compensation

(See our paper at INTERSPEECH 2015 for details)

Reverberation robust acoustic modeling using i-vectors with time delay neural networks

Vijayaditya Peddinti[1], *Guoguo Chen*[1], *Daniel Povey*[1,2], *Sanjeev Khudanpur*[1,2]

[1]Center for language and speech processing &
[2]Human Language Technology Center of Excellence
The Johns Hopkins University, Baltimore, MD 21218, USA
{vijay.p,guoguo,khudanpur}@jhu.edu, dpovey@gmail.com

Abstract

In reverberant environments there are long term interactions between speech and corrupting sources. In this paper a time delay neural network (TDNN) architecture, capable of learning long term temporal relationships and translation invariant representations, is used for reverberation robust acoustic modeling. Further, iVectors are used as an input to the neural network to perform instantaneous speaker and environment adaptation, providing 10% relative improvement in word error rate. By subsampling the outputs at TDNN layers across time steps, training time is reduced. Using a parallel training algorithm we show that the TDNN can be trained on $\sim$ 5500 hours of speech data in 3 days using up to 32 GPUs. The TDNN is shown to provide results competitive with state of the art systems in the IARPA ASpIRE challenge, with 27.7% WER on the *dev_test* set.

Index Terms: far field speech recognition, time delay neural networks, reverberation

1. Introduction

such a wide temporal context enables the network to deal with late reverberations.

iVectors which capture both speaker and environment specific information have been shown to be useful for rapid adaptation of the neural network [6, 7, 8]. iVector based adaptation has also been shown to be effective in reverberant environments [9]. In this paper we use this adaptation technique.

We show experimental results on the ASpIRE far-field speech recognition challenge held by IARPA [10]. This challenge uses the English portion of the Fisher database [11] for acoustic and language model training. We show that in this large data scenario the proposed network architecture, combined with a parallel training technique [12], can train on multi-condition training data of $\sim$ 5500 hours, using up to 32 GPUs, in 3 days.

Using the TDNN architecture helps us to achieve results close to those of the best combined system submitted to the ASpIRE challenge, while using only a single system. Our system was able to achieve 27.7% WER on the *dev-test* set, while the best system achieved 27.2% WER.

Using i-vectors Instead of fMLLR

and using unnormalized MFCCs to compute i-vectors

- 100-dim i-vectors are appended to MFCC inputs of the TDNN
  - i-vectors are computed from raw MFCCs (i.e. no mean subtraction etc)
  - UBM posteriors however use MFCCs normalized over a 6 sec window
- i-vectors are computed for each training utterance
  - Increases speaker- and channel variability seen in training data
  - May model transient distortions? e.g. moving speakers, passing cars
- i-vectors are calculated for every ca 60 sec of test audio
  - UBM prior is weighted 10:1 to prevent overcompensation
  - Weight of test statistics is capped at 75:1 relative to UBM statistics

Speaker Compensation Method	Dev WER
TDNN without i-vectors	34.8%
+ i-vectors (from all frames)	33.8%
+ i-vectors (from reliable speech frames)	**30.8%**

Pronunciation and Silence Probabilities

(See our paper at INTERSPEECH 2015 for details)

PRONUNCIATION AND SILENCE PROBABILITY MODELING FOR ASR

Guoguo Chen[1], *Hainan Xu*[1], *Minhua Wu*[1], *Daniel Povey*[1,2], *Sanjeev Khudanpur*[1,2]

[1]Center for Language and Speech Processing
[2]Human Language Technology Center of Excellence
The Johns Hopkins University, Baltimore, MD 21218, USA

guoguo@jhu.edu, hxu31@jhu.edu, mwu56@jhu.edu, dpovey@gmail.edu, khudanpur@jhu.edu

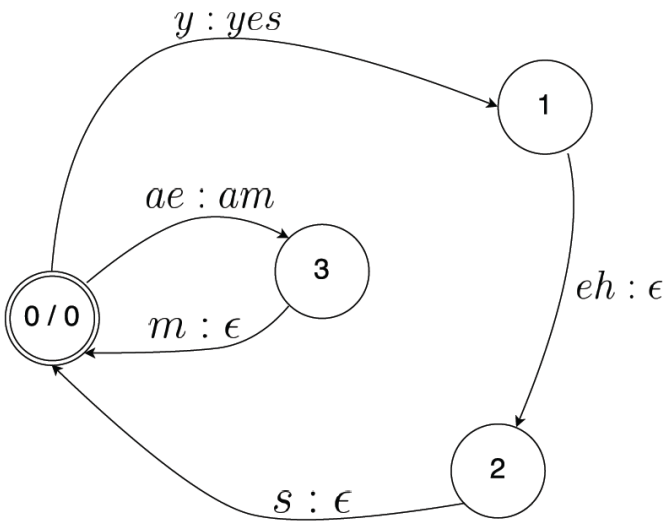
Abstract

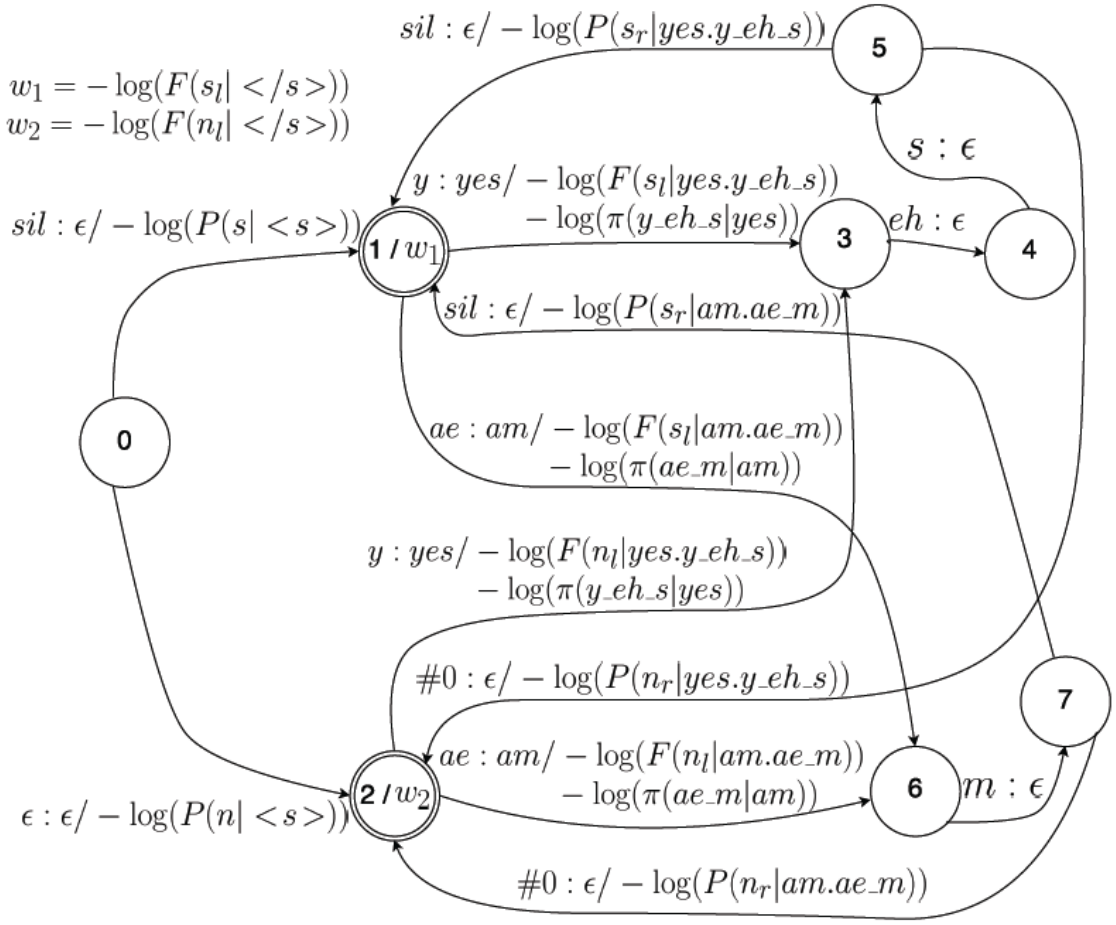
In this paper we evaluate the WER improvement from modeling pronunciation probabilities and word-specific silence probabilities in speech recognition. We do this in the context of Finite State Transducer (FST)-based decoding, where pronunciation and silence probabilities are encoded in the lexicon (L) transducer. We describe a novel way to model word-dependent silence probabilities, where in addition to modeling the probability of silence following each individual word, we also model the probability of each word appearing after silence. All of these probabilities are estimated from aligned training data, with suitable smoothing. We conduct our experiments on four commonly used automatic speech recognition datasets, namely Wall Street Journal, Switchboard, TED-LIUM, and Librispeech. The

Implicit pronunciation modeling relies on the underlying acoustic-phonetic models to account for pronunciation variations, and therefore removes the necessity to explicitly determine and represent them in the lexicon. In some methods, acoustic model parameters of a phoneme (e.g., Gaussian densities) are tied with those of its alternative realizations, thus capturing alternative pronunciations [3, 12, 13]. Others view pronunciations as a bundle of features, and pronunciation variation is viewed as feature-change or asynchrony [14, 15].

While variability in the pronunciation of individual words has been studied extensively, relatively little has been studied about inter-word silence and its dependence on the prosodic and syntactic structure of the utterance. In [3, 10], for instance, three types of silence are permitted following each pronunciation in the lexicon: a zero-silence, a short pause and a long silence. It is

Trigram-like Inter-word Silence Model





$$P\left(s\middle|a\_b\right)=P\left(s\middle|a\_\right)F\left(s\middle|\_b\right)$$

$$P\left(s\middle|a\_\right)=\frac{c\left(as\right)+\lambda_2 P\left(s\right)}{c\left(a\right)+\lambda_2}$$

$$F\left(s\middle|\_b\right)=\frac{c\left(sb\right)+\lambda_3}{\sum_{a'}c\left(a'*b\right)P\left(s\middle|a'\_\right)+\lambda_3}$$

Is "Prosody" Finally Helping STT?

Task	Test Set	Baseline	+ Sil/Pron
WSJ	Eval 92	4.1	3.9
Switchboard	Eval 2000	20.5	20.0
TED-LIUM	Test	18.1	17.9
Libri Speech	Test Clean	6.6	6.6
	Test Other	22.9	22.5

- Modeling pronunciation and silence probabilities yields modest but consistent improvement on many large vocabulary ASR tasks

Pronunciation/Silence Probabilities	Dev WER
No probabilities in the lexicon	32.1%
+ pronunciation probabilities	31.6%
+ inter-word silence probabilities	**30.8%**

Recurrent Neural Network based Language Models

(See our paper at INTERSPEECH 2010 for the first “convincing” results)

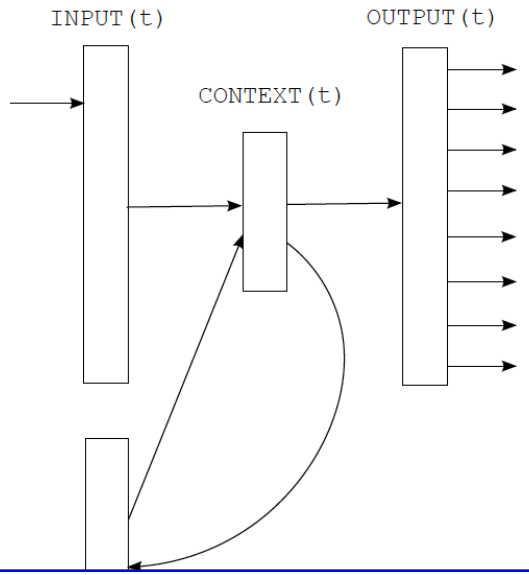
Recurrent neural network based language model

Tomáš Mikolov[1,2], *Martin Karafiát*[1], *Lukáš Burget*[1], *Jan “Honza” Černocký*[1], *Sanjeev Khudanpur*[2]

[1]Speech@FIT, Brno University of Technology, Czech Republic
[2] Department of Electrical and Computer Engineering, Johns Hopkins University, USA
{imikolov,karafiat,burget,cernocky}@fit.vutbr.cz, khudanpur@jhu.edu

Abstract

A new recurrent neural network based language model (RNN LM) with applications to speech recognition is presented. Results indicate that it is possible to obtain around 50% reduction of perplexity by using mixture of several RNN LMs, compared to a state of the art backoff language model. Speech recognition experiments show around 18% reduction of word error rate on the Wall Street Journal task when comparing models trained on the same amount of data, and around 5% on the much harder NIST RT05 task, even when the backoff model is trained on much more data than the RNN LM. We provide ample empirical evidence to suggest that connectionist language models are superior to standard n-gram techniques, except their high computational (training) complexity.

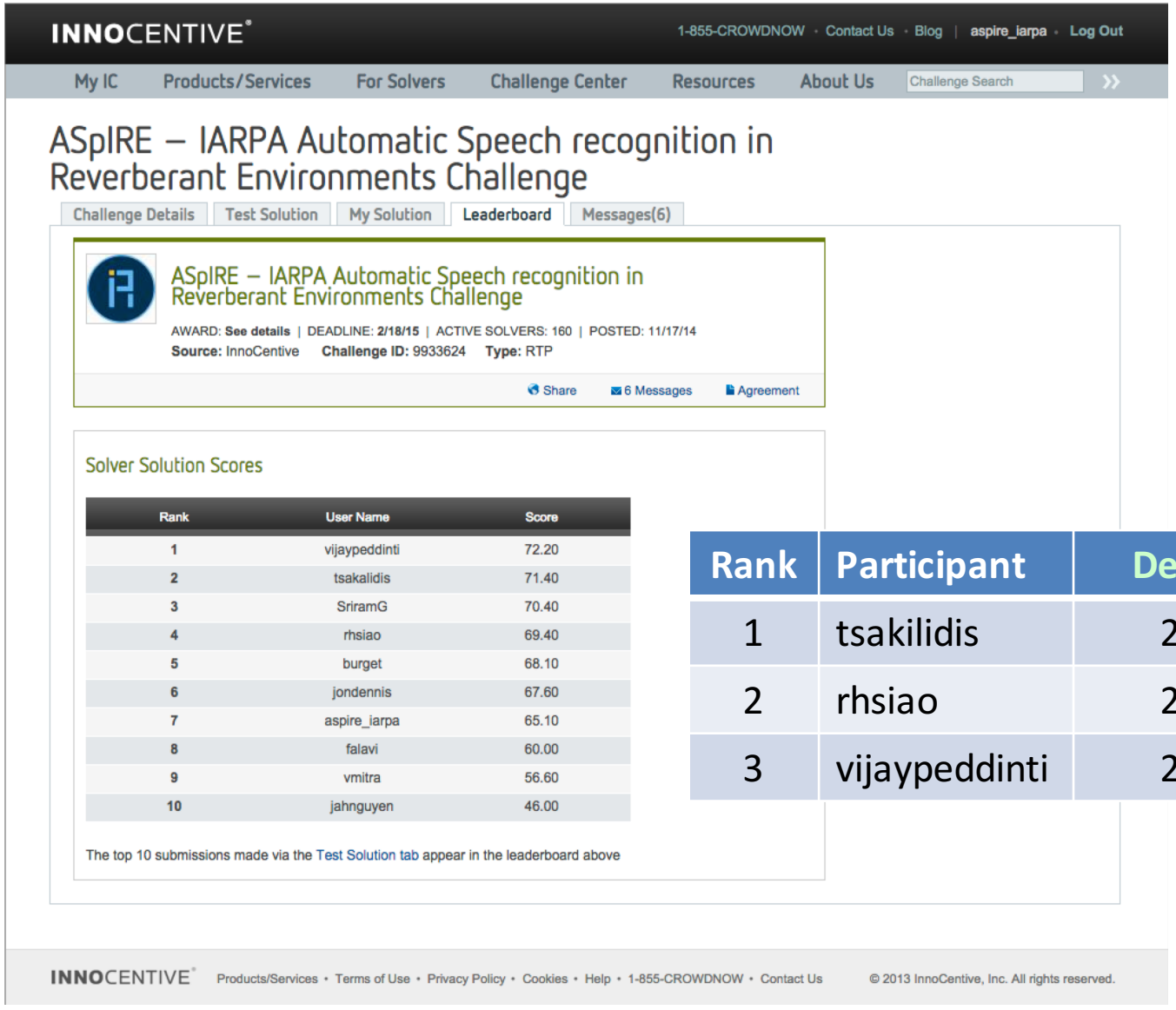


RNN LM on ASpIRE Data

Language Model and Rescoring Method	Dev WER
4-gram LM and lattice rescoring	30.8%
RNN-LM and 100-best rescoring	30.2%
RNN-LM and 1000-best rescoring	29.9%
RNN-LM (4-gram approximation) lattice rescoring	29.9%
RNN-LM (6-gram approximation) lattice rescoring	**29.8**%

- An RNN LM consistently outperforms the N-gram LM
- The Kaldi lattice rescoring appears to cause no loss in performance
  - Approximation entails not "expanding" the lattice to represent each unique history separately
  - When two paths merge in an N-gram lattice, only one s(t) is chosen at random and propagated forward

The IARPA ASpIRE Leader Board



Rank	Participant	Dev WER	System Type
1	tsakilidis	27.2%	Combination
2	rhsiao	27.5%	Combination
3	vijaypeddinti	27.7%	Single System

http://www.dni.gov/index.php/newsroom/press-releases/210-press-releases-2015/1252-iarpa-announces-winners-of-its-aspire-challenge

OFFICE OF THE DIRECTOR OF NATIONAL INTELLIGENCE

LEADING INTELLIGENCE INTEGRATION

IARPA Announces Winners of its ASpIRE Challenge

NEWS RELEASE

FOR IMMEDIATE RELEASE
ODNI News Release No. 14-15
September 10, 2015

IARPA Announces Winners of its ASpIRE Challenge

WASHINGTON – The Intelligence Advanced Research Projects Activity (IARPA), within the Office of the Director of National Intelligence (ODNI), today announced the winners of its speech recognition challenge, *Automatic Speech recognition in Reverberant Environments (*ASpIRE*)*. The winning teams from the Johns Hopkins University, Raytheon BBN Technologies, the Institute for Infocomm Research, and Brno University of Technology will share $110,000 in prizes.

http://www.dni.gov/index.php/newsroom/press-releases/210-press-releases-2015/1252-iarpa-announces-winners-of-its-aspire-challenge

IARPA Announces Winners of its ASpIRE Challenge

2. The **Multiple Microphone Condition** tested accuracy of speech recognition on recordings from six different microphones recording at once.

All of the ASpIRE challenge winners delivered systems with more than a 50% reduction in word error rate (WER) compared to the IARPA baseline system. WER is the standard measure of accuracy for speech recognition systems; lower WER scores indicate more accurate systems.

The winners in the Single Microphone category are:

- the team from the Center for Language and Speech Processing, Johns Hopkins University (Vijayaditya Peddinti, Guoguo Chen, Dr. Daniel Povey, Dr. Sanjeev Khudanpur);

Performance on Evaluation Data

Participant	Test WER	System Type
Kaldi	44.3%	**Single System**
BBN (and others)	44.3%	Combination
I^2R (Singapore)	44.8%	Combination

Acoustic Model	Language Model	Dev WER	Test WER	Eval WER
TDNN B (CE training)	4-gram	30.8%	27.7%	**44.3**%
TDNN B (sMBR training)	4-gram	29.1%	28.9%	**43.9**%
TDNN B (CE training)	RNN	29.8%	26.5%	43.4%
TDNN B (sMBR training)	RNN	28.3%	28.2%	43.4%

Keys to Good Performance on ASpIRE

- Time delay neural networks (TDNN)
  - Deal well with long reverberation times
- i-vector based adaptation compensation
  - Deals with speaker & channel variability
- Data augmentation with simulated reverberations
  - Deals with channel distortions not seen in training
- Pronunciation and inter-word silence probabilities
  - Helpful in adverse acoustic conditions

The JHU ASpIRE System

(See our ASRU 2015 paper for details)

JHU ASPIRE SYSTEM : ROBUST LVCSR WITH TDNNS, I-VECTOR ADAPTATION AND RNN-LMS

Vijayaditya Peddinti[1], *Guoguo Chen*[1], *Vimal Manohar* [1], *Tom Ko* [3] *Daniel Povey*[1,2], *Sanjeev Khudanpur*[1,2]

[1]Center for language and speech processing &
[2]Human Language Technology Center of Excellence
The Johns Hopkins University, Baltimore, MD 21218, USA
[3]Huawei Noah's Ark Research Lab, Hong Kong, China

{vijay.p,guoguo,khudanpur}@jhu.edu, {vimal.manohar91, tomkocse, dpovey}@gmail.com

ABSTRACT

Multi-style training, using data which emulates a variety of possible test scenarios, is a popular approach towards robust acoustic modeling. However acoustic models capable of exploiting large amounts of training data in a comparatively short amount of training time are essential. In this paper we tackle the problem of reverberant speech recognition using 5500 hours of simulated reverberant data. We use time-delay neural network (TDNN) architecture, which is capable of tackling long-term interactions between speech and corrupting sources in reverberant environments. By sub-sampling the outputs at TDNN layers across time steps, training time is substantially reduced. Combining this with distributed-optimization we show that the TDNN can be trained in 3 days using up to 32 GPUs. Further, iVectors are used as an input to the neural network to perform in-

iVectors which capture both speaker and environment specific information have been shown to be useful for rapid adaptation of the neural network [4, 5, 6]. iVector based adaptation has also been shown to be effective in reverberant environments [7]. In this paper we use this adaptation technique.

We show experimental results on the ASpIRE far-field speech recognition challenge held by IARPA [8]. This challenge uses the English portion of the Fisher database [9] for acoustic and language model training. We show that in this large data scenario the proposed network architecture, combined with a distributed optimization technique [10], can train on multi-condition training data of $\sim$ 5500 hours, using up to 32 GPUs, in 3 days.

Using the TDNN architecture helps us to achieve results close to those of the best combined system submitted to the ASpIRE chal-

Semi-supervised MMI Training

(See our paper at INTERSPEECH 2015 for details)

Semi-supervised Maximum Mutual Information Training of Deep Neural Network Acoustic Models

Vimal Manohar, Daniel Povey*†, Sanjeev Khudanpur*†*

*Center for Language and Speech Processing
† Human Language Technology Center of Excellence
The Johns Hopkins University, Baltimore, MD 21218, USA
vmanoha1@jhu.edu, danielpovey@gmail.com, khudanpur@jhu.edu

Abstract

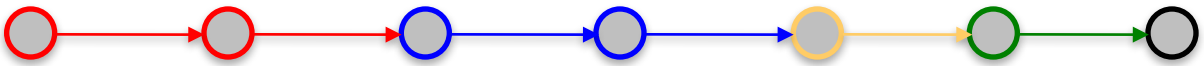
Maximum Mutual Information (MMI) is a popular discriminative criterion that has been used in supervised training of acoustic models for automatic speech recognition. However, standard discriminative training is very sensitive to the accuracy of the transcription and hence its implementation in a semi-supervised setting requires extensive filtering of data. We will show that if the supervision transcripts are not known, the natural analogue of MMI is to minimize the conditional entropy of the lattice of possible transcripts of the data. This is equivalent to the weighted average of MMI criterion over different reference transcripts, taking those reference transcripts and their weighting from the lattice itself. In this paper we describe experiments where we applied this method to the semi-supervised training of Deep Neural Network acoustic models. In our experimental setup, the proposed method gives up to 0.5% absolute

untranscribed data for cross-entropy training using a slightly different method which we will described here. We avoid the untranscribed data "polluting" the last layer of the network by giving it a separate final layer, using ideas inspired by multilingual DNN training [15, 16].

Discriminative training is very sensitive to the accuracy of the transcripts [17, 18, 19]. Therefore sequence-discriminative self-training methods do not work well without some form of confidence-based filtering, as used in [20, 21, 17, 22]. However, we show in this paper that by using an alternative objective function, Negative Conditional Entropy (NCE) on the untranscribed portion of the data, we can obtain improvements from untranscribed data without filtering. Entropy minimization has previously been used as an objective for semi-supervised learning in a facial recognition problem [23] and for sequence-discriminative training of GMM acoustic models for speech recognitinon [24]. In this paper, we apply the idea to semi-

Discriminative (MMI) Training:

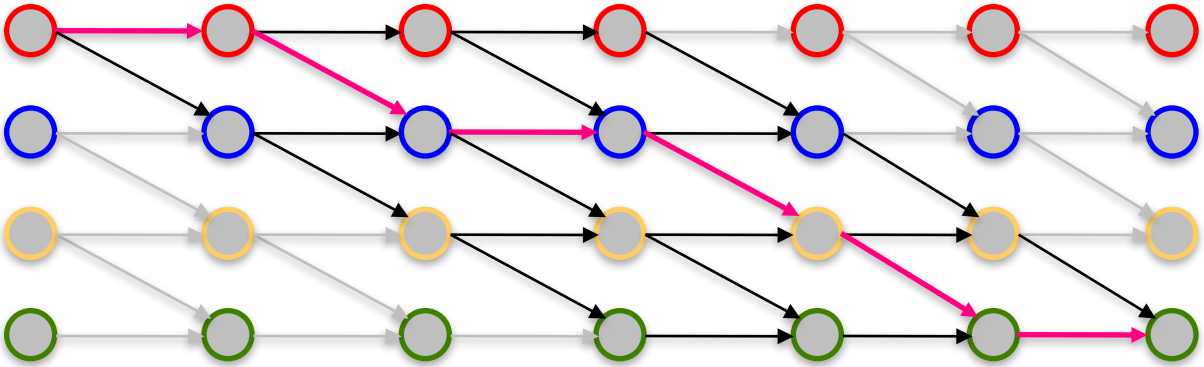
a hand-waving, mostly correct introduction



$$\hat{\theta}_{ML} = \arg\max_{\theta} \sum_{t=1}^{T} \log P\left(O_t | W_t; \theta\right)$$

$$KL\left(\hat{P} \| P_\theta\right)$$

Cross-entropy training



$$\hat{\theta}_{MMI} = \arg\max_{\theta} \sum_{t=1}^{T} \log \left\{ \frac{P\left(O_t | W_t; \theta\right)}{\sum_{W_t'} P\left(O_t | W_t'; \theta\right) P\left(W_t'\right)} \right\}$$

$$I\left(W \wedge O; \theta\right)$$

Sequence training

Semi-Supervised Sequence Training

- Sequence training improves substantially over basic cross-entropy training of DNN acoustic models
- Semi-supervised cross-entropy training – by adding unlabeled data – also improves substantially over basic cross-entropy training on labeled data
- But semi-supervised sequence training is "tricky"
  - Sensitivity to incorrect transcription seems greater
  - Confidence-based filtering or weighting must be applied
  - Empirical results are not very satisfactory

Semi-supervised Sequence Training:
without committing to a single transcription

- View MMI training as minimizing a conditional entropy

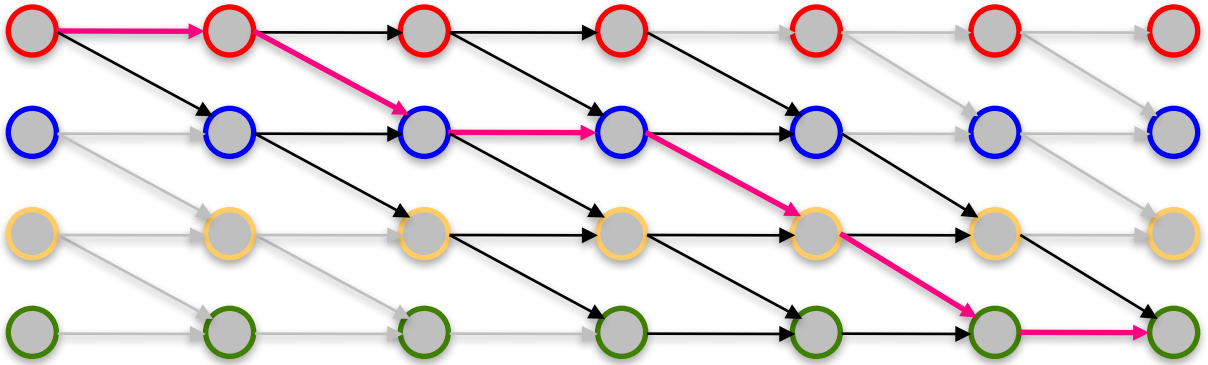
$$I(W \wedge O;\theta) = \frac{1}{T}\sum_{t=1}^{T}\log\frac{P(O_t|W_t;\theta)}{P(O_t;\theta)} = \frac{1}{T}\sum_{t=1}^{T}\log\frac{P(O_t|W_t;\theta)}{\sum_{W'}P(O_t|W';\theta)P(W')}$$

$$I(W \wedge O;\theta) = H(W) - H(W|O;\theta) = H(W) - \frac{1}{T}\sum_{t=1}^{T}H(W|O_t;\theta)$$

- The latter does not require committing to a single W_t
  - Well suited for unlabeled speech
  - Entails computing a sum over all *W*'s in the lattice

Computing Lattice Entropy Using Expectation Semi-rings

- How to efficiently compute $-H\left(W\middle|O_t;\theta\right)=\sum_{\pi\in L}P\left(\pi\right)\log P\left(\pi\right)$



- Take inspiration from the computation of $Z\left(O_t;\theta\right)=\sum_{\pi\in L}P\left(\pi\right)$
- Replace arc-probabilities p_i with the pair (p_i , $p_i log\{p_i\}$)

Semi-ring Element & Operators	(p , p×log{p})
(p_1 , $p_1 log\{p_1\}$) + (p_2 , $p_2 log\{p_2\}$)	(p_1+p_2 , $p_1 log\{p_1\}+p_2 log\{p_2\}$)
(p_1 , $p_1 log\{p_1\}$) × (p_2 , $p_2 log\{p_2\}$)	($p_1 p_2$, $p_1 p_2 log\{p_2\}+p_2 p_1 log\{p_1\}$)

Semi-supervised Sequence Training:

Key Details Needed to Make it Work

- View training criterion as MCE instead of MMI
  - i.e. *arg min H(W|O;ϑ)* instead of *arg max I(W⋀O;ϑ)*
  - Efficiently compute *H(W|O;ϑ)* for the lattice, and its gradient, via Baum Welch with special semi-rings
- Use separate output (soft-max) layers in the DNN for labeled and unlabeled data
  - Inspired by multilingual DNN training methods
- Use a slightly different “prior” for converting DNN posterior probabilities to acoustic likelihoods

Results for Semi-Supervised MMI on Fisher English CTS

Known use of unlabeled data

DNN Training Method (hours of speech)	Dev WER	Test WER
Cross-Entropy Training (100h labeled)	32.0	31.2
CE (100h labeled + 250h self-labeled)	30.6	29.8
CE (100h labeled + 250h weighted)	30.5	29.8

Better use of unlabeled data

Sequence Training (100h labeled)	29.6	28.5
Seq Training (100h labeled +250h weighted)	29.9	28.8
Seq Training (100h labeled + 250h MCE)	29.4	28.1
Sequence Training (350h labeled)	28.5	27.5

- Recovers about 40% of the supervised training gain
  - Investigation underway for 2000h of unlabeled speech
- Repeatable results on BABEL datasets with 10h supervised training + 50-70h unsupervised

Advanced Methods:

Staying Ahead in the STT Game

- STT technology is advancing very rapidly
  - Amazon, Apple, Baidu, Facebook, Google, Microsoft
- Kaldi leads and keeps up with major innovations
  - From SGMMs to DNN (2012)
  - From "English" to low-resource languages (2013)
  - From CPUs to GPUs (2014)
  - From close-talking to far-field microphones (2015)
  - **From well-curated to "wild type" corpora (2016)**
- A preview of some upcoming developments

Heterogeneous Training Corpora

- Transcribed speech from different collections are not easy to merge for STT training
  - Genre and speaking style differences
  - Different channel conditions
  - Slightly different transcription conventions
- Typical result: the corpus matched to test data gives best STT results; others don't help, sometimes hurt!
- SCALE 2015 case study with Pashto CTS
  - Collected in country, and transcribed, by same vendor
  - Roughly 80 hours each in the
    - Appen LILA corpus and IARPA BABEL corpus
  - Pronunciation lexicon to cover transcripts; same phone set

A Study in Pashto

(See our manuscript submitted to INTERSPEECH 2016)

USING OF HETEROGENEOUS CORPORA FOR TRAINING OF AN ASR SYSTEM

Jan Trmal[1,2], *Gaurav Kumar*[1,2], *Vimal Manohar*[1], *Sanjeev Khudanpur*[1,2], *Matt Post*[2], *Paul McNamee*[2]

[1]Center for Language and Speech Processing
[2]Human Language Technology Center of Excellence
The Johns Hopkins University, Baltimore, MD 21218, USA

ABSTRACT

The paper summarizes the development of the LVCSR system built as a part of the Pashto speech-translation system at the SCALE (Summer Camp for Applied Language Exploration) 2015 workshop on "Speech-to-text-translation for low-resource languages". The Pashto language was chosen as a good "proxy" low-resource language, exhibiting multiple phenomena which make the speech-recognition and and speech-to-text-translation systems development hard.

Even when the amount of data is seemingly sufficient, given the fact that the data originates from multiple sources, the preliminary experiments reveal that there is little to no benefit in merging (concatenating) the corpora and more elaborate ways of making use of all of the data must be worked out.

This paper concentrates only on the LVCSR part and presents a range of different techniques that were found to be useful in order to benefit from multiple different corpora

ی ی

Fig. 1. ARABIC LETTER FARSI YEH and ARABIC LETTER ALEF MAKSURA

layout although the Arabic and Urdu layouts are used as well. The alternative layouts have a majority of the Pashto characters (and people freely substitute those which are missing with visually similar characters). Also, different fonts can have small deficiencies in rendering of glyphs, especially during kerning or joining of characters and the users often try to fix this by substituting a different character that looks better (i.e. closer to the expected shape) in the given context.

A direct impact of this is that a word as a sequence of glyphs (visual representations of characters) can be represented as multiple sequences of unicode codepoints (numerical codes

A Study in Pashto

- Transcriptions require extensive cross-corpus normalization
- Even after that, language models don't benefit much from corpus pooling
- Simple corpus pooling doesn't improve acoustic modeling either
- DNNs with shared "inner" layers and corpus-specific input and output layers work best

Training Data	Single Model	Interpolation Weights LM A	LM B	LM T	Interpolated Model
Text A	99.2	0.8	0.2	0.0	92.9
Text B	141.9	0.1	0.8	0.1	140.0
Text T	86.7	0.0	0.0	1.0	86.7

DNN Training Data	STT Word Error Rates Test A	Test B	Test T
Single corpus (matched)	55.4%	**46.8%**	**24.8%**
Two corpora (Pashto A + B)	**51.9%**	48.2%	52.6%

Multi-corpus (A+B) Training Strategy	STT Word Error Rates Test A	Test B	Test T
Shared DNN layers (except 1)	53.2%	47.4%	27.0%
Shared DNN layers (except 2)	51.2%	45.0%	25.4%
+ Optimized Language Model	50.8%	44.8%	25.4%
+ Duration Modeling	**50.4%**	**44.3%**	**24.8%**

Advanced Methods:

Staying Ahead in the STT Game

- STT technology is advancing very rapidly
  - Amazon, Apple, Baidu, Facebook, Google, Microsoft
- Kaldi leads and keeps up with major innovations
  - From SGMMs to DNN (2012)
  - From "English" to low-resource languages (2013)
  - From CPUs to GPUs (2014)
  - From close-talking to far-field microphones (2015)
  - From well-curated to "wild type" corpora (2016)
- **A preview of some upcoming developments**

Other Additions and Innovations

- Semi-supervised (MMI) training
  - Using unlabeled speech to augment a limited transcribed speech corpus
- Multilingual acoustic model training
  - Using other-language speech to augment a limited transcribed speech corpus
- Removing reliance on pronunciation lexicons
  - Grapheme based models and acoustically aided G2P
- **Chain models**
  - 10% more accurate STT, plus
  - 3x faster decoding, and 5x-10x faster training

The Genesis of Chain Models

- Connectionist Temporal Classification
  - The latest shiny toy in neural network-based acoustic modeling for STT (ICASSP and InterSpeech 2015)
  - Nice STT improvements shown on Google datasets
  - We haven't seen STT gains on our datasets
- Chain Models
  - Inspired by (but quite different from) CTC
  - Sequence training of NNs without CE pre-training
  - Nice STT improvements over previous best systems
  - 3x decoding time speed-up; 5x-10x training speed-up

2006: A New Kid on the NNet Block

Connectionist Temporal Classification: Labelling Unsegmented Sequence Data with Recurrent Neural Networks

Alex Graves[1] ALEX@IDSIA.CH
Santiago Fernández[1] SANTIAGO@IDSIA.CH
Faustino Gomez[1] TINO@IDSIA.CH
Jürgen Schmidhuber[1,2] JUERGEN@IDSIA.CH

[1] Istituto Dalle Molle di Studi sull'Intelligenza Artificiale (IDSIA), Galleria 2, 6928 Manno-Lugano, Switzerland
[2] Technische Universität München (TUM), Boltzmannstr. 3, 85748 Garching, Munich, Germany

Abstract

Many real-world sequence learning tasks require the prediction of sequences of labels from noisy, unsegmented input data. In speech recognition, for example, an acoustic signal is transcribed into words or sub-word units. Recurrent neural networks (RNNs) are powerful sequence learners that would seem well suited to such tasks. However, because they require pre-segmented training data, and post-processing to transform their outputs into label sequences, their applicability has so far been limited. This paper presents a novel method for training RNNs to label unsegmented sequences directly, thereby solving both problems. An experiment on the TIMIT speech corpus demonstrates its advantages over both a baseline HMM and a hybrid HMM-RNN.

belling. While these approaches have proved successful for many problems, they have several drawbacks: (1) they usually require a significant amount of task specific knowledge, e.g. to design the state models for HMMs, or choose the input features for CRFs; (2) they require explicit (and often questionable) dependency assumptions to make inference tractable, e.g. the assumption that observations are independent for HMMs; (3) for standard HMMs, training is generative, even though sequence labelling is discriminative.

Recurrent neural networks (RNNs), on the other hand, require no prior knowledge of the data, beyond the choice of input and output representation. They can be trained discriminatively, and their internal state provides a powerful, general mechanism for modelling time series. In addition, they tend to be robust to temporal and spatial noise.

2015: The New Kid Comes of Age

LEARNING ACOUSTIC FRAME LABELING FOR SPEECH RECOGNITION WITH RECURRENT NEURAL NETWORKS

Haşim Sak, Andrew Senior, Kanishka Rao, Ozan İrsoy, Alex Graves, Françoise Beaufays, Johan Schalkwyk

Google

{hasim, andrewsenior, kanishkarao, gravesa, fsb, johans}@google.com

ABSTRACT

We explore alternative acoustic modeling techniques for large vocabulary speech recognition using Long Short-Term Memory recurrent neural networks. For an acoustic frame labeling task, we compare the conventional approach of cross-entropy (CE) training using fixed forced-alignments of frames and labels, with the Connectionist Temporal Classification (CTC) method proposed for labeling unsegmented sequence data. We demonstrate that the latter can be implemented with finite state transducers. We experiment with phones and context dependent HMM states as acoustic modeling units. We also investigate the effect of context in acoustic input by training unidirectional and bidirectional LSTM RNN models. We show that a bidirectional LSTM RNN CTC model using phone units can perform as well as an LSTM RNN model trained with CE using HMM state alignments. Finally, we also show the effect of sequence discriminative training on these models and show the first results for sMBR training of CTC models.

implemented in finite-state transducer (FST) framework and explain how CTC models can be used in decoding (Section 2.2). We also describe the use of sequence discriminative training with our sequence models (Section 2.3). In Section 4, we describe experiments with two acoustic modeling units – phones and HMM states. We also investigate the effect of acoustic context for LSTM RNN acoustic models by training unidirectional and bidirectional models.

2. ACOUSTIC MODELING WITH LSTM RNN

There are a number of alternative approaches for acoustic modeling with neural networks for automatic speech recognition (ASR). Fundamentally the unit to be modeled by the network must be chosen (e.g. phone, HMM state, context dependent HMM state, diphone, word etc.). Training may use a hard (Viterbi) alignment with a single class label per frame, or a soft (Baum-Welch) alignment giving a probability distribution. Further, a variety of objective functions

2015: The New Kid Comes of Age

ACOUSTIC MODELLING WITH CD-CTC-SMBR LSTM RNNS

Andrew Senior, Haşim Sak, Félix de Chaumont Quitry, Tara Sainath, Kanishka Rao

Google

{hasim,andrewsenior,fcq,tsainath,kanishkarao}@google.com

ABSTRACT

This paper describes a series of experiments to extend the application of Context-Dependent (CD) long short-term memory (LSTM) recurrent neural networks (RNNs) trained with Connectionist Temporal Classification (CTC) and sMBR loss. Our experiments, on a noisy, reverberant voice search task, include training with alternative pronunciations and the application to child speech recognition; combination of multiple models, and convolutional input layers. We also investigate the latency of CTC models and show that constraining forward-backward alignment in training can reduce the delay for a real-time streaming speech recognition system. Finally we investigate transferring knowledge from one network to another through alignments.

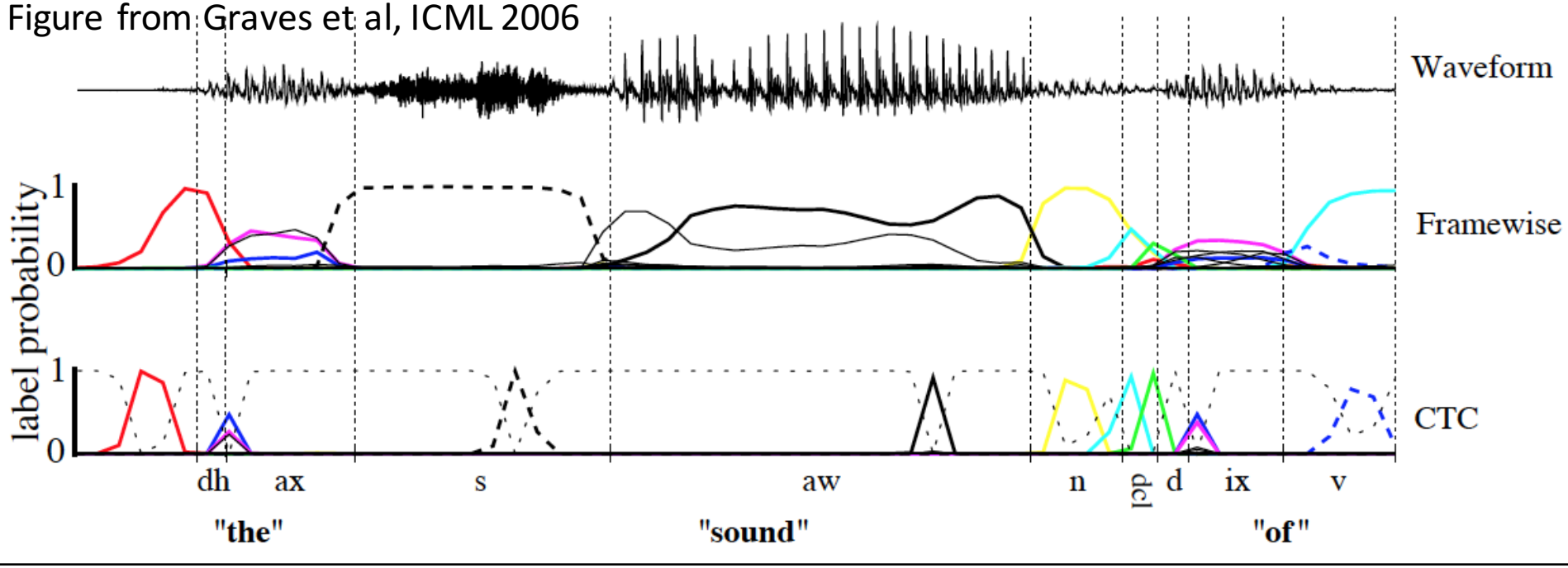
Index Terms: Long Short Term Memory, Recurrent Neural Networks, Connectionist Temporal Classification, sequence discriminative training, knowledge transfer.

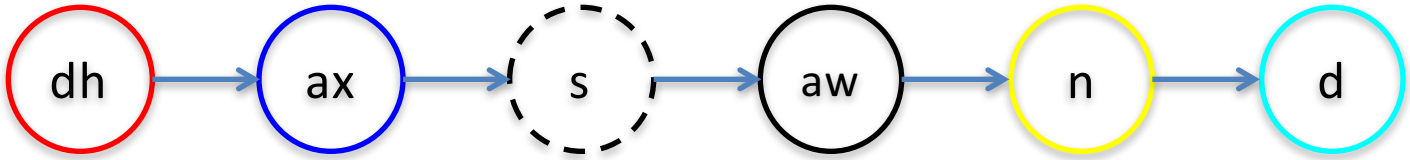
the labels indicate the segmentation of the sequence with repeated labels indicating longer durations, with CTC an output may only be high for a single frame to indicate the presence of the symbol, with other frames labelled "blank," and duration information is discarded. During training CTC constantly aligns every sequence and trains to maximize the total probability of all valid label sequences. Because of the memory of the LSTM model this means that the outputs no longer need to occur at the same time as the input features to which they correspond.

In our previous work [2] we have shown that models with a blank symbol that are initialized with CTC can be improved upon with sMBR sequence-discriminative training. We then showed [3] that such models, using long-duration features (95ms of speech represented as 8 stacked overlapping log-mel filterbank features, generated with a 25ms window FFT every 10ms), downsampled and processed every 30ms, can outperform conventionally-trained LSTM models when using context dependent phone targets [5]. We use the term CD-CTC-sMBR LSTM RNN for these models.

CTC, Explained … in Pictures

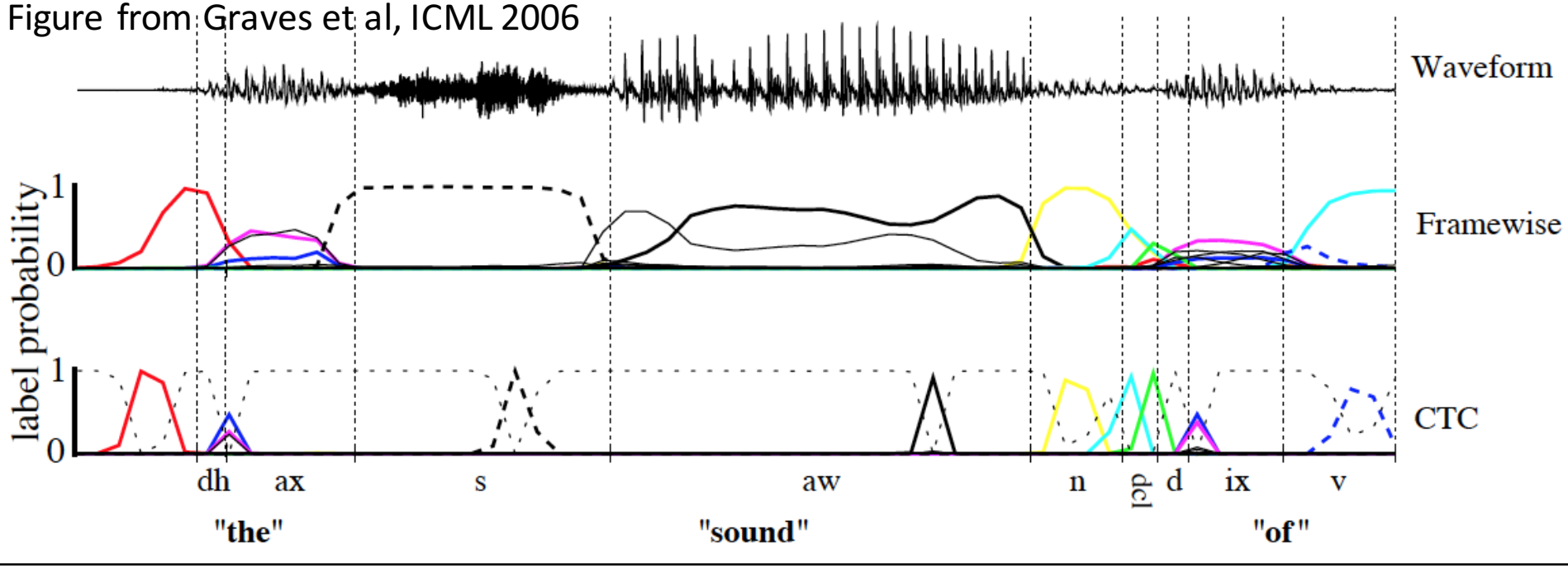
Figure from Graves et al, ICML 2006

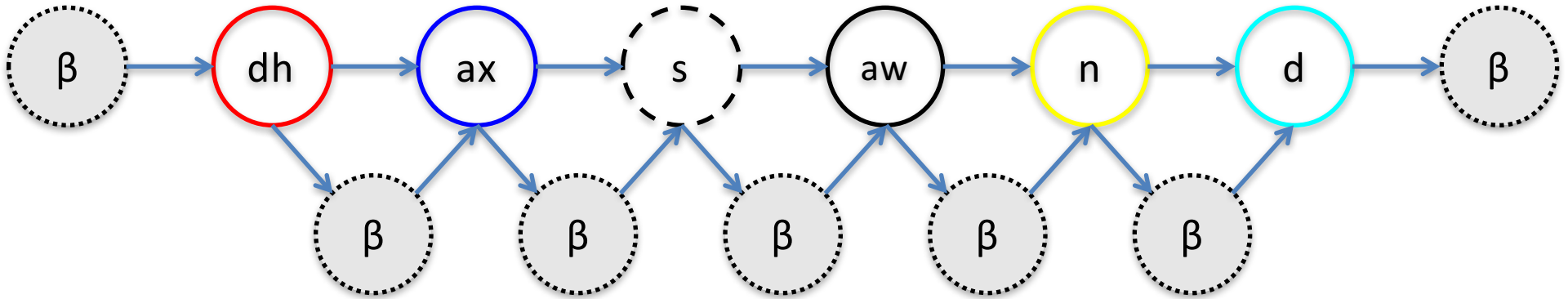




CTC, Explained … in Pictures

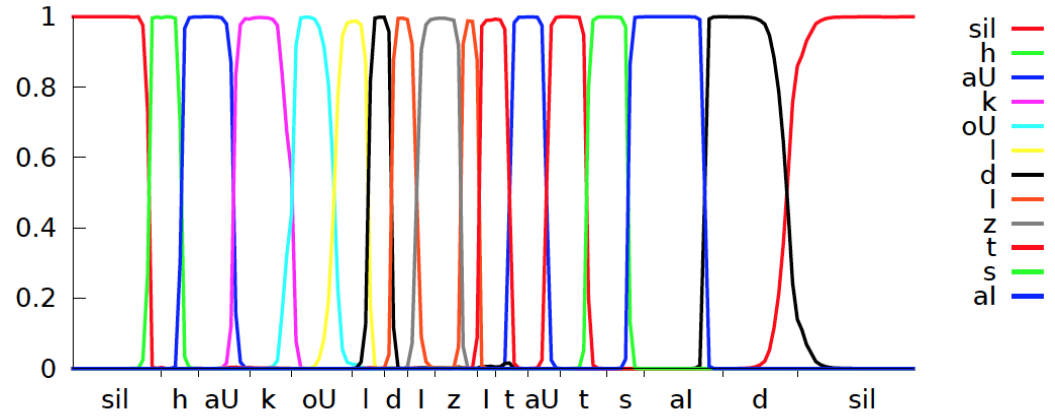
Figure from Graves et al, ICML 2006





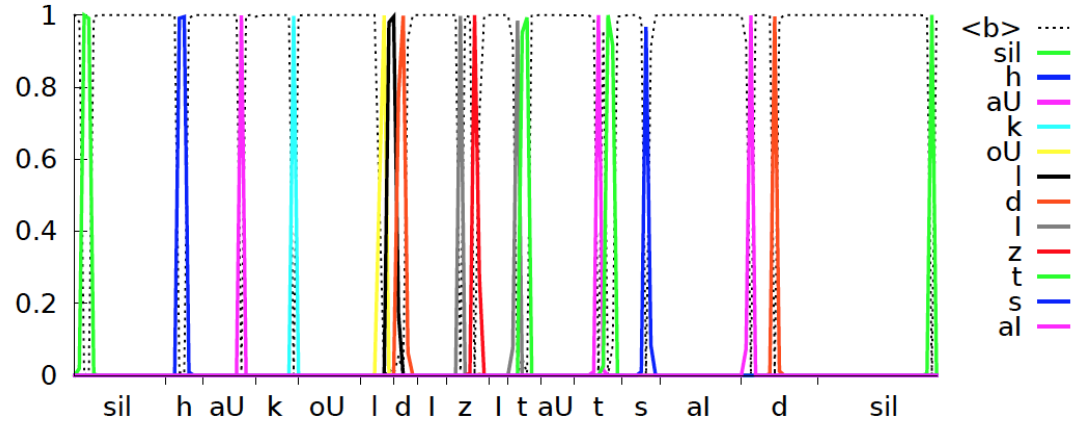
DNN versus CTC: STT Performance

Figures and Tables from Sak et al, ICASSP 2015



(b) CE bidirectional LSTM with phone labels

DNN	Target	CE	sMBR
LSTM	Senone	10.0%	8.9%
BLSTM	Senone	9.7%	9.1%



(i) CTC bidirectional LSTM with phone labels (10% PER)

CTC	Target	CE	sMBR
LSTM	Phone	10.5%	9.4%
BLSTM	Phone	9.5%	8.5%

First, the Bad News …

- We haven't been able to get CTC models to give us any noticeable improvement over our best (TDNN or LSTM-RNN) models on our data
  - It appears to be easier to get them to work when one has several 1000 hours of labeled speech
  - But we care about lower-resource scenarios

… and then the Good News

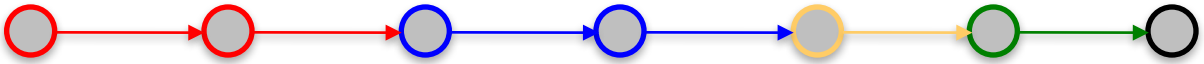
- We are able to get similar improvements using a different model, which is inspired by ideas from the CTC papers
  - Use simple “1-state” HMMs for each CD phone
  - Reduce frame rate from 100 Hz to 33 Hz
  - Permit slack in the frame-to-state alignment

Chain Models and LF-MMI Training

- A new class of acoustic models for hybrid STT
  - "1-state" HMM for each context-dependent phone
  - LSTM/TDNNs compute state posterior probabilities
- MFCCs are down-sampled from 100Hz to 33Hz
  - Inspired by CTC
- A new lattice-free MMI training method
  - Improved parallelization, sequence training on GPUs
    - Larger mini-batches, smaller I/O bandwidth
  - Does not require CE training before MMI training
  - Uses "flexible label alignment" inspired by CTC

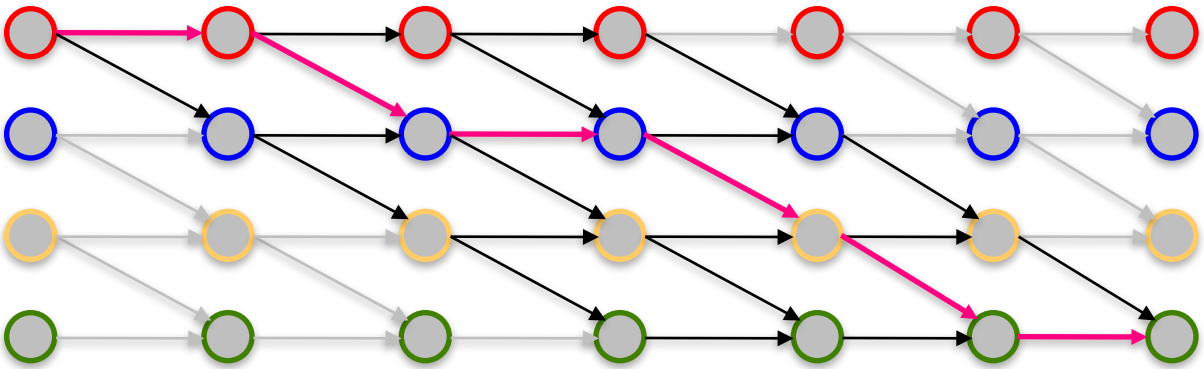
Discriminative (MMI) Training:

a hand-waving, mostly correct introduction



$$\hat{\theta}_{ML} = \arg\max_{\theta} \sum_{t=1}^{T} \log P\left(O_t | W_t; \theta\right)$$

$$KL\left(\hat{P} \| P_\theta\right)$$



$$\hat{\theta}_{MMI} = \arg\max_{\theta} \sum_{t=1}^{T} \log \left\{ \frac{P\left(O_t | W_t; \theta\right)}{\sum_{W_t'} P\left(O_t | W_t'; \theta\right) P\left(W_t'\right)} \right\}$$

$$I\left(W \wedge O; \theta\right)$$

Lattice-Free MMI Training

- Denominator (phone) graph creation
  - Use a phone 4-gram language model, L
  - Compose H, C and L to obtain denominator graph
    - This FSA is the same for all utterances; suits GPU training
    - Use (heuristic) sentence-specific initial probabilities
- Numerator graph creation
  - Generate a phone graph using transcripts
    - This FSA encodes frame-by-frame alignment of HMM states
  - Permit some alignment “slack” for each frame/label
  - Intersect slackened FSA with the denominator FSA

Lattice-free MMI Training (cont'd)

- LSTM-RNNs trained with this MMI training procedure are highly susceptible to over-fitting
- Essential to **regularize** the NN training process
  - A second output layer for CE training
  - Output L_2 regularization
  - Use a leaky HMM

Regularization			Hub-5 '00 Word Error Rate	
Cross Entropy	L_2 Norm	Leaky HMM	Total	SWBD
N	N	N	16.8%	11.1%
Y	N	N	15.9%	10.5%
N	Y	N	15.9%	10.4%
N	N	Y	16.4%	10.9%
Y	Y	N	15.7%	10.3%
Y	N	Y	15.7%	10.3%
N	Y	Y	15.8%	10.4%
Y	Y	Y	**15.6%**	10.4%

STT Results for Chain Models

300 hours of SWBD Training Speech; Hub-5 '00 Evaluation Set

Training Objective	Model (Size)	Total WER	SWBD WER
Cross-Entropy	TDNN A (16.6M)	**18.2%**	12.5%
CE + sMBR	TDNN A (16.6M)	**16.9%**	11.4%
Lattice-free MMI	TDNN A (9.8M)	16.1%	10.7%
	TDNN B (9.9M)	15.6%	10.4%
	TDNN C (11.2M)	**15.5%**	10.2%
LF-MMI + sMBR	TDNN C (11.2M)	**15.1%**	10.0%

- LF-MMI reduces WER by ca 10%-15% *relative*
- LF-MMI is better than standard CE + sMBR training (ca 8%)
- LF-MMI improves very slightly with additional sMBR training

Chain Models and LF-MMI Training

STT Performance on a Variety of Corpora

Corpus and Audio Type	Training Speech	CE + sMBR Error Rate	LF-MMI Error Rate
AMI IHM	80 hours	23.8%	22.4%
AMI SDM	80 hours	48.9%	46.1%
TED-LIUM	118 hours	11.3%	12.8%
Switchboard	300 hours	16.9%	15.5%
Fisher + SWBD	2100 hours	15.0%	13.3%

- Chain models with LF-MMI reduce WER by 6%-11% (*relative*)
- LF-MMI improves a bit further with additional sMBR training
- FL-MMI is 5x-10x faster to train, 3x faster to decode

A Recap of Chain Models

- A new class of acoustic models for hybrid STT
  - "1-state" HMM for context-dependent phones
  - LSTM-RNN acoustic models (TDNN also compatible)
- A new lattice-free MMI training method
  - Better suited to using GPUs for parallelization
  - Does not require CE training before MMI training
- Improved speed and STT performance
  - 6%-8% relative WER reduction over previous best
  - 5-10x improvement in training time; 3x decoding time

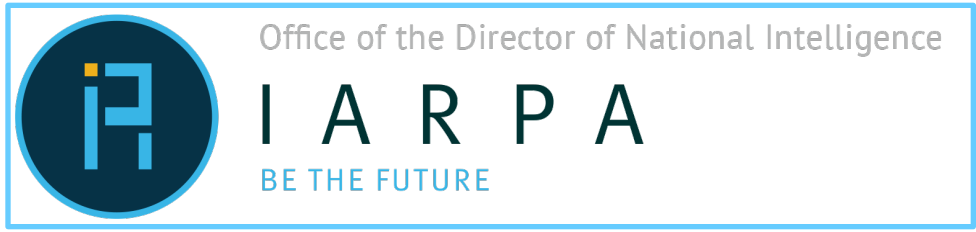
Summary of Advanced Methods:

Staying Ahead in the STT Game

- STT technology is advancing very rapidly
  - Amazon, Apple, Baidu, Facebook, Google, Microsoft
- Kaldi leads and keeps up with major innovations
  - From SGMMs to DNN (2012)
  - From “English” to low-resource languages (2013)
  - From CPUs to GPUs (2014)
  - From close-talking to far-field microphones (2015)
  - From well-curated to “wild type” corpora (2016)
  - Chain models for better STT, faster decoding (2017)
- **and the list goes on ...**

Team Kaldi @ Johns Hopkins

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



Kaldi Points-of-Contact

- **Kaldi mailing list**
  - kaldi-help@googlegroups.com
- Daniel Povey
  - povey@gmail.com
- Jan “Yenda” Trmal
  - trmal@jhu.edu
- Sanjeev Khudanpur
  - khudanpur@jhu.edu
  - 410-516-7024