The Kaldi Speech Recognition Toolkit

Daniel Povey[1], Arnab Ghoshal[2,3], Gilles Boulianne[4], Lukáš Burget[5,6], Ondřej Glembek[5], Nagendra Goel[6], Mirko Hannemann[5], Petr Motlíček[8], Yanmin Qian[9], Petr Schwarz[5], Jan Silovský[10], Georg Stemmer[11], Karel Veselý[5]

[1] Microsoft Research, USA; [2] University of Edinburgh, UK; [3] Saarland University, Germany; [4] Centre de Recherche Informatique de Montréal, Canada; [5] Brno University of Technology, Czech Republic; [6] SRI International, USA; [7] Go-Vivace Inc., USA; [8] IDIAP Research Institute, Switzerland; [9] Tsinghua University, China; [10] Technical University of Liberec, Czech Republic; [11] University of Erlangen-Nuremberg, Germany

KALDI

http://kaldi.sf.net

Problem Statement

- You are a researcher who wants to try out a new method.
- You want other people to use your idea, if it works.
- You find that older toolkits (HTK, CMUSphinx) are too hard to modify.
- Also their license may not allow you to release your changes (HTK).

The Kaldi project

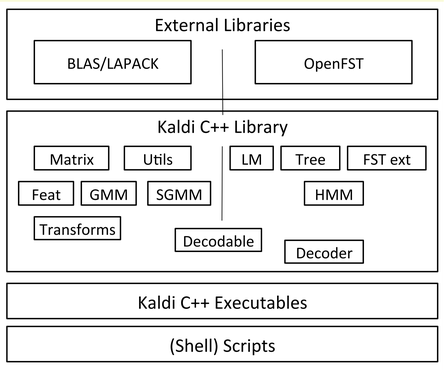
- Open-source speech recognition toolkit, Apache-licensed.
- Began at 2009 Johns Hopkins University CLSP summer workshop.
- Further development included 2 summer workshops in Brno, Czech Republic, and ongoing work by the participants.
- Uses OpenFst code for decoding graph construction.
- Uses the BLAS and LAPACK libraries for linear algebra.
- Our closest competitor is probably the RWTH Aachen toolkit.

Our vision

- Distributed community of users and contributors.
  - ▷ Not a free-for-all; original authors moderate contributions.
  - ▷ The Apache license allows you to fork the project.
- Complete state-of-the-art recipes that run from public data
  - ▷ These already exist for Resource Management and Wall Street Journal
  - ▷ Switchboard recipe exists but is not yet state-of-the-art.
- Code that's well structured and simple to understand.
- Thorough testing and documentation.

The structure of Kaldi

- OpenFst for FST functions
- BLAS/LAPACK for linear algebra
- Core functions in C++
- Many simple command-line utilities
- Example shell scripts



Example of Kaldi code

```
void DiagGmm::LogLikelihoods(const VectorBase<BaseFloat> &data,
                             Vector<BaseFloat> *loglikes) const {
  loglikes->Resize(gconsts_.Dim(), kUndefined);
  loglikes->CopyFromVec(gconsts_);
  if (static_cast<int32>(data.Dim()) != Dim()) {
    KALDI_ERR << "DiagGmm::ComponentLogLikelihood, dimension "
        << "mismatch" << (data.Dim()) << " vs. "<< (Dim());
  }
  Vector<BaseFloat> data_sq(data);
  data_sq.ApplyPow(2.0);

  // loglikes +=  means * inv(vars) * data.
  loglikes->AddMatVec(1.0, means_invvars_, kNoTrans, data, 1.0);
  // loglikes += -0.5 * inv(vars) * data_sq.
  loglikes->AddMatVec(-0.5, inv_vars_, kNoTrans, data_sq, 1.0);
}
```

Acoustic modeling techniques supported in Kaldi

- Acoustic front-end supports MFCC and PLP features, with cepstral mean and variance normalization, LDA, STC/MLLT, HLDA, VTLN, etc.
- HMM/GMM acoustic models; phonetic decision trees.
- Also SGMMs, exponential transform.
- No language modeling code, but support converting ARPA format LMs to FSTSs.
- WFST-based decoders, lattice generation.
- Discriminative training with MMI, boosted MMI (fMPE unfinished).

Example of top-level system building script

```
## Example of WSJ system building.
local/wsj_data_prep.sh /mnt/speech_data/WSJ?/??-{?,??}.?
local/wsj_prepare_dict.sh
local/wsj_format_data.sh
mfccdir=/mnt/my_storage/kaldi_wsj_mfcc
for x in test_eval92 test_eval93 test_dev93 train_si284; do
 steps/make_mfcc.sh data/$x exp/make_mfcc/$x $mfccdir 4
done
# skipped some data-subsetting commands here.
# This setup would use GridEngine.
decode_cmd="queue.pl -q queue_name -l ram_free=1200M,mem_free=1200M"
train_cmd="queue.pl -q queue_name -l ram_free=700M,mem_free=700M"

steps/train_mono.sh --num-jobs 10 --cmd "$train_cmd" \
  data/train_si84_2kshort data/lang exp/mono0a
steps/align_deltas.sh --num-jobs 10 --cmd "$train_cmd" \
   data/train_si84_half data/lang exp/mono0a exp/mono0a_ali
steps/train_deltas.sh --num-jobs 10 --cmd "$train_cmd" \
    2000 10000 data/train_si84_half data/lang exp/mono0a_ali exp/tri1
# ...
```

Segment of triphone training sript

```
for n in `get_splits.pl $nj`; do
  featspart[$n]="ark:apply-cmvn --norm-vars=false \
   --utt2spk=ark:$data/split$nj/$n/utt2spk ark:$alidir/$n.cmvn \
   scp:$data/split$nj/$n/feats.scp ark:- | add-deltas ark:- ark:- |"
done
# tree building, graph compilation omitted.
while [ $x -lt $numiters ]; do
   echo "Iteration $x"
   if echo $realign_iters | grep -w $x >/dev/null; then
     echo "Aligning data"
     for n in `get_splits.pl $nj`; do
       $cmd $dir/log/align.$x.$n.log \
         gmm-align-compiled $scale_opts --beam=10 --retry-beam=40 \
           $dir/$x.mdl "ark:gunzip -c $dir/$n.fsts.gz|" "${featspart[$n]}" \
           "ark:|gzip -c >$dir/$n.ali.gz" || touch $dir/.error &
     done
     wait;
     [ -f $dir/.error ] && echo "Alignment error on iteration $x" && exit 1;
   fi
   for n in `get_splits.pl $nj`; do
     $cmd $dir/log/acc.$x.$n.log \
     gmm-acc-stats-ali  $dir/$x.mdl "${featspart[$n]}" \
      "ark,s,cs:gunzip -c $dir/$n.ali.gz|" $dir/$x.$n.acc || touch $dir/.error &
   done
   wait;
   [ -f $dir/.error ] && echo "Accumulation error on iteration $x" && exit 1;
   $cmd $dir/log/update.$x.log \
     gmm-est --write-occs=$dir/$[$x+1].occs --mix-up=$numgauss $dir/$x.mdl \
       "gmm-sum-accs - $dir/$x.*.acc |" $dir/$[$x+1].mdl || exit 1;
   rm $dir/$x.mdl $dir/$x.*.acc
   rm $dir/$x.occs
   if [[ $x -le $maxiterinc ]]; then
      numgauss=$[$numgauss+$incgauss];
   fi
   x=$[$x+1];
done
```

Comparison with previously published results

	Test set				
	Feb'89	Oct'89	Feb'91	Sep'92	Avg
HTK	2.77	4.02	3.30	6.29	4.10
Kaldi	3.20	4.21	3.50	5.86	4.06

	Test set	
	Nov'92	Nov'93
Bell	11.9	15.4
HTK (+GD)	11.1	14.5
KALDI	11.8	15.0

- These are not our best results: they just show that with similar system setups, we get similar results.
  - ▷ WSJ results in table use bigram LM.
- Current best RM result: 1.78%
  - ▷ System combination: LDA+MLLT+SAT+MMI with LDA+MLLT+SAT+SGMM+fMLLR).
- Current best WSJ result: 4.39% on eval'92 open-vocabulary test set.
  - ▷ Train on SI-284, LDA+MLLT+SAT+SGMM, extended vocabulary, 4-gram LM trained from supplied trainscripts.