Kaldi

Open source software

- Kaldi: complete toolkit in C++ with multiple recipes (bash scripts)
- RWTH ASR - The RWTH Aachen University Speech Recognition System (Commercial restrictions)
- Theano: deep learning research in python
- Torch 7: a computing framework for LuaJIT (Lua scripting language)
- Caffe: Deep learning framework
- Minerva: a fast and flexible tool for deep learning on multi-GPU (python numpy)

INSTALL

```
$ git clone https://github.com/kaldi-asr/kaldi.git
$ cd kaldi/tools
$ make –j 4
$ cd ../src
$ ./configure
$ make depend –j 4
$ make –j 4
```

EXAMPLE 1: yesno (run.sh)

- cd egs/yesno/s5
- ./run.sh
- %WER 0.00 [0 / 240, 0 ins, 0 del, 0 sub] exp/mono0a/decode_test_yesno/wer_10

EXAMPLE 1: yesno (data)

head -2 data/test_yesno/*

==> data/test_yesno/cmvn.scp <==

global /veu4/usuaris31/xtrans/docencia/rtthss2015/kaldi/egs/yesno/s5/mfcc/cmvn_test_yesno.ark:7

==> data/test_yesno/feats.scp <==

0_1_1_1_1_1_1_1 /veu4/usuaris31/xtrans/docencia/rtthss2015/kaldi/egs/yesno/s5/mfcc/raw_mfcc_test_yesno.1.ark:16

1_0_0_0_0_0_0_0 /veu4/usuaris31/xtrans/docencia/rtthss2015/kaldi/egs/yesno/s5/mfcc/raw_mfcc_test_yesno.1.ark:8165

==> data/test_yesno/spk2utt <==

global 0_1_1_1_1_1_1_1 1_0_0_0_0_0_0_0 1_0_0_0_0_0_0_1 1_0_0_0_0_0_1_1 1_0_0_0_1_0_0_1 1_0_0_1_0_1_1_1
1_0_1_0_1_0_0_1 1_0_1_1_0_1_1_1 1_0_1_1_1_0_1_0 1_0_1_1_1_1_0_1 1_1_0_0_0_0_0_1 1_1_0_0_0_1_1_1
1_1_0_0_1_0_1_0 1_1_0_0_1_0_1_1 1_1_0_0_1_1_1_0 1_1_0_1_0_1_0_0 1_1_0_1_0_1_1_0 1_1_0_1_1_0_0_1
1_1_0_1_1_0_1_1 1_1_0_1_1_1_1_0 1_1_1_0_0_0_0_1 1_1_1_0_0_1_0_1 1_1_1_0_0_1_1_1 1_1_1_0_1_0_1_0
1_1_1_0_1_0_1_1 1_1_1_1_0_0_1_0 1_1_1_1_0_1_0_0 1_1_1_1_1_0_0_0 1_1_1_1_1_1_0_0 1_1_1_1_1_1_1_1

==> data/test_yesno/text <==

0_1_1_1_1_1_1_1 NO YES YES YES YES YES YES YES

1_0_0_0_0_0_0_0 YES NO NO NO NO NO NO NO

==> data/test_yesno/utt2spk <==

0_1_1_1_1_1_1_1 global

1_0_0_0_0_0_0_0 global

==> data/test_yesno/wav.scp <==

0_1_1_1_1_1_1_1 waves_yesno/0_1_1_1_1_1_1_1.wav

1_0_0_0_0_0_0_0 waves_yesno/1_0_0_0_0_0_0_0.wav

EXAMPLE 2: voxforge (download)

- Set *DATA_ROOT* variable in *./path.sh* to point to a directory residing (about 25GB).
- Use ./getdata.sh to download VoxForge's 16KHz versions of the compressed data archives to
  ${DATA_ROOT}/tgz
  and extract them to
  ${DATA_ROOT}/extracted.

you can add a "--deltgz true" parameter to remove tgz: /getdata.sh --deltgz true

EXAMPLE 2: voxforge (directories)

- **local/** - hosts scripts that are specific to each recipe. Using new data with Kaldi involves writing and modifying local scripts.
- **conf/** - small configuration files, specifying things like e.g. feature extraction parameters and decoding beams.
- **steps/** - scripts implementing various acoustic model training methods, calling Kaldi's binary tools and the scripts in "utils/" (shared)
- **utils/** - scripts performing small low-level tasks. (shared)
- **data/** - This is where various metadata, produced at the recipe run time is stored, e.g. lexicon,
- **exp/** - The most important output of the recipe goes there. This includes acoustic models and recognition results.

EXAMPLE 2: voxforge (data preparation)

```
dialects="((American)|(British)|(Australia)|(Zealand))”
selected=${DATA_ROOT}/selected
# Select a subset of the data to use
local/voxforge_select.sh --dialect $dialects \
    ${DATA_ROOT}/extracted ${selected}

# Mapping the anonymous speakers to unique IDs
local/voxforge_map_anonymous.sh ${selected}

# Initial normalization of the data
local/voxforge_data_prep.sh --nspk_test ${nspk_test} ${selected}
```

EXAMPLE 2: voxforge (language model)

```
locdata=data/local
loctmp=$locdata/tmp
order=3
cut -f2- -d' ' < $locdata/train_trans.txt |\
   sed -e 's:[ ]\+: :g' |\
   sort -u > $loctmp/corpus.txt
ngram-count -order $order -write-vocab $locdata/vocab-full.txt \
  -wbdiscount -text $loctmp/corpus.txt -lm $locdata/lm.arpa
```

(SRILM The SRI Language Modeling Toolkit)

EXAMPLE 2: voxforge (phonetic dictionary)

local/voxforge_prepare_dict.sh.

- Downloads the CMU's pronunciation dictionary
- Creates a list of the words that are found in the train set, but not in cmudict
- Pronunciations for these words are automatically generated using Sequitur G2P, which is installed under tools/g2p.
- Because the training of Sequitur models takes a lot of time this script is downloading and using a pre-built model trained on cmudict instead.

Prepare data/lang and data/local/lang directories (L.fst)

Prepare the grammar transducer (G.fst) for testing

EXAMPLE 2: voxforge (feature extraction)

```
mfccdir=${DATA_ROOT}/mfcc
for x in train test; do
   steps/make_mfcc.sh --cmd "$train_cmd" --nj $njobs \
      data/$x exp/make_mfcc/$x $mfccdir || exit 1;
   steps/compute_cmvn_stats.sh \
     data/$x exp/make_mfcc/$x $mfccdir || exit 1;
done
```

EXAMPLE 2: voxforge (train models)

```
# Train monophone models on a subset of the data
utils/subset_data_dir.sh data/train 1000 data/train.1k  || exit 1;
steps/train_mono.sh --nj $njobs --cmd "$train_cmd" data/train.1k
data/lang exp/mono  || exit 1;
# Get alignments from monophone system.
steps/align_si.sh --nj $njobs --cmd "$train_cmd" \
  data/train data/lang exp/mono exp/mono_ali || exit 1;
# train tri1 [first triphone pass]
steps/train_deltas.sh --cmd "$train_cmd" \
  2000 11000 data/train data/lang exp/mono_ali exp/tri1 || exit 1;
```

EXAMPLE 2: voxforge (decode mono)

```
# Monophone decoding
utils/mkgraph.sh --mono data/lang_test exp/mono \
  exp/mono/graph || exit 1
# exp/mono/decode/
steps/decode.sh --config conf/decode.config --nj $njobs \
 --cmd "$decode_cmd" \
 exp/mono/graph data/test exp/mono/decode
```

EXAMPLE 2: voxforge (decode tri1)

```
# decode tri1
utils/mkgraph.sh data/lang_test exp/tri1 exp/tri1/graph || exit 1;
steps/decode.sh --config conf/decode.config --nj $njobs \
   --cmd "$decode_cmd" \
  exp/tri1/graph data/test exp/tri1/decode

draw-tree data/lang/phones.txt exp/tri1/tree \
  | dot -Tps -Gsize=8,10.5 | ps2pdf - tree.pdf
```

Evaluation

- for x in `find exp -name "decode*"` ; do [-d $x] && grep WER $x/wer_* | utils/best_wer.sh; done

WFST

- Mehryar Mohri, Fernando C. N. Pereira, and Michael Riley. "Speech recognition with weighted finite-state transducers". Handbook on Speech Processing and Speech Communication 2008 (a.k.a. The Holly Book of WFSTs)
- Kaldi decoding-graph creation recipe: http://kaldi.sourceforge.net/graph_recipe_test.html
- Decoding graph construction in Kaldi: A visual walkthrough
  http://vpanayotov.blogspot.com.es/2012/06/kaldi-decoding-graph-construction.html