Lab. – YesNo Tutorial

Yuan-Fu Liao

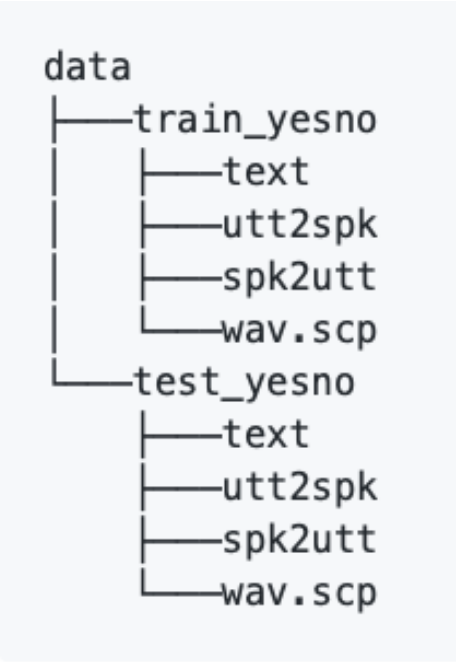
National Taipei University of Technology

Data Description

- 60 .wav files, sampled at 8 kHz
  - All audio files are recorded by an anonymous male contributor of the Kaldi project and included in the project for a test purpose.
- In each file, the individual says 8 words; each word is either "ken" or "lo" ("yes"and "no" in Hebrew)
  - so each file is a random sequence of 8 yes's or no’s.
- The names of files represent the word sequence, with 1 for ken/yes and 0 for lo/no, that is the names will serve as transcript for each sequence.
  - waves_yesno/1_0_1_1_1_0_1_0.wav
  - waves_yesno/0_1_1_0_0_1_1_0.wav
  - ...

Step 1 - Data Preparation

- local/prepare_data.sh waves_yesno
  - text
    - Essentially, transcripts of the audio files.
    - Write an utterance per line, formatted in <utt_id> <transcript>
      - e.g. 0_0_1_1_1_1_0_0 NO NO YES YES YES YES NO NO
  - wav.scp
    - Indexing files to unique ids.
    - <file_id> <path of wave filenames OR command to get wave file>
      - e.g. 0_1_0_0_1_0_1_1 waves_yesno/0_1_0_0_1_0_1_1.wav
  - utt2spk
    - For each utterance, mark which speaker spoke it.
      - Since we have only one speaker in this example, let's use global as speaker_id
    - <utt_id> <speaker_id>
      - e.g. 0_0_1_0_1_0_1_1 global
  - spk2utt
    - Simply inverse-indexed utt2spk (<speaker_id> <all_hier_utterences>)



Step 2 - Dictionary Preparation

- local/prepare_dict.sh
  - data/local/dict
    - lexicon.txt
      - full list of lexeme-phone pairs including *silences*
    - lexicon_words.txt
      - list of word-phone pairs (no silence)
    - silence_phones.txt
      - list of silent phones
    - nonsilence_phones.txt
      - list of non-silent phones
    - optional_silence.txt
      - list of optional silent phones

Step 3 – Language Preparation

- utils/prepare_lang.sh --position-dependent-phones false data/local/dict "<SIL>" data/local/lang data/lang
  - convert our dictionaries into a data structure that Kaldi would accept - weighted finite state transducer (WFST)

Step 4 – Language Model

- local/prepare_lm.sh
  - arpa2fst --disambig-symbol=#0 --read-symbol-table=$test/words.txt input/task.arpabo $test/G.fst

Step 5 - Feature extraction

```
for x in train_yesno test_yesno; do
    steps/make_mfcc.sh --nj 1 data/$x exp/make_mfcc/$x mfcc
    steps/compute_cmvn_stats.sh data/$x exp/make_mfcc/$x mfcc
    utils/fix_data_dir.sh data/$x
done
```

Step 6 – Training

Mono training

steps/train_mono.sh --nj 1 --cmd "$train_cmd" --totgauss 400 data/train_yesno data/lang exp/mono0a

Step 7 – Decoding

```
# Graph compilation
utils/mkgraph.sh data/lang_test_tg exp/mono0a exp/mono0a/graph_tgpr

# Decoding
steps/decode.sh --nj 1 --cmd "$decode_cmd" exp/mono0a/graph_tgpr data/test_yesno exp/mono0a/decode_test_yesno

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