KALDI tutorial

KALDI is an open source speech recognition toolkit implementing state-of-the-art algorithms for feature extraction, acoustic modeling and decoding. Its codebase is written entirely in C++ and the corresponding executables are invoked from bash scripts called "recipes" which depend on the dataset that they process.

Installation[2]

In order to install KALDI you need to clone the repository where it resides:
"git clone https://github.com/kaldi-asr/kaldi.git kaldi-trunk –origin golden"
The INSTALL file contains all the required information about the installation process.

Getting started

Source code and executables are organized according to the purpose they serve (decoding, feature extraction, neural network configuration etc.) and are placed in kaldi-trunk/src. Invoking a script without arguments will print instructions on how to run it and what arguments it actually needs.

Example recipes for well-known datasets are inside kaldi-trunk/egs. The latest recipe for each dataset is inside the s5 folder. For example, the latest recipe for the Wall Street Journal dataset is kaldi-trunk/egs/wsj/s5/run.sh and it is executed by running run.sh while being inside that directory. Do not be intimidated by the length of the script. The basic part is the data preparation at the beginning; the rest are the training and evaluation of different recognition systems.

We will focus on the WSJ recipe throughout the tutorial. Apart from the run.sh script, inside kaldi-trunk/egs/wsj/s5 are folders (local, steps, utils) containing dataset-dependent scripts.

Going though the WSJ recipe

- Data preparation [1]
  - The script begins by setting the variables which determine the dataset location (*wsj0,wsj1*). For this lab, you only have wsj0 data.
  - Invoke dataset dependent script (*wsj_data_prep.sh*) to make sure all data needed are available and to build auxiliary scripts for the training steps (e.g. spk2utt, utt2spk scripts). This script should be adapted to the data at our disposal. It also extracts the language model which (unless we want to use a different one) is distributed with the WSJ dataset.
  - Proceed to create the dictionary (*wsj_prepare_dict.sh*) and the language directory, which contains extra information regarding the language model (*prepare_lang.sh*).
  - Reorganize the data directory to facilitate the next steps (*wsj_format_data.sh*).

utterance-id	[frame1 features
	frame2 features
	. . .]

Table 1: KALDI table format

- Feature extraction
  - Extract MFCC features (*steps/make_mfcc.sh*). Make sure that the data are where the script expects them to be and in the correct form (case-sensitive to file names).
- Split the dataset into smaller chunks which will be used for training various systems (*utils/subset_data_dir.sh*).
- Training
  - The usual procedure to evaluate an ASR system starts with system training (e.g. *train_mono.sh*), proceeds to build the decoding graph HCLG (*utils/mkgraph.sh*) and ends with system evaluation (*steps/decode.sh*) [5]. The final result of the evaluation is inside exp/decode.../scoring_kaldi/best_wer.
  - The system we just created can be used to extract the observations-phones alignment, which we could use to initiate the training of e.g. a triphone model [3],[4].

Useful scripts

KALDI provides various executables that can prove useful not only for ASR system training but also for debugging/testing purpose. In particular, inside kaldi-trunk/src/bin:

- *ali-to-phones* : converts alignments from KALDI raw format ("transition-ids") to phones. To view the alignments use *show-alignments*. (For extra insight into KALDI HMM transition modeling see "HMM topology and transition modeling" in [1], however it is not really necessary for a basic use of the toolkit.)
- *copy-matrix* (or *featbin/copy-feats*): copies a KALDI matrix. During copying we can save the matrix in text format. The read and write specifiers that are referred to in the scripts are "ark" ,"txt" and "scp" for binary ,text and scp (readable by text editors) scripts.

  In general, KALDI processes scp scripts (ending in .scp) and binary data. KALDI matrices have the form seen in the table and are saved into binary format (.ark files).

References

[1] http://kaldi-asr.org/.

[2] http://www.danielpovey.com/files/lecture1.pdf.

[3] http://www.danielpovey.com/files/lecture2.pdf.

[4] http://www.danielpovey.com/files/lecture3.pdf.

[5] http://www.danielpovey.com/files/lecture4.pdf.