DNN Adaptation Documentation for Kaldi

Anis Chihi

August 25, 2016

Contents

1 Introduction

This document provides a description and a walk-through on how to use the adaptation methods I investigated in my master thesis on Kaldi. It is a short explanation on how to use the implemented functions. I implemented these techniques for two different ASR systems, and I will only describe the folders concerning the state of the art one, since both are quite similar, and for future work, it is more interesting to work on a powerful system. The whole ASR system is under the folder **Kaldi-trunk/egs/libri/speech/s5**. As for the standard ASR systems implemented for Kaldi, all the scripts have to be run from the above mentioned path.

In this paper, we do not describe how to train an ASR system from scratch, but we use an already existing framework.

2 Folders Tree

Under the **Kaldi-trunk/egs/libri/speech/s5** are the whole folders and scripts for running experiments and getting results. the figure below shows the different folders an files that are under **s5**.

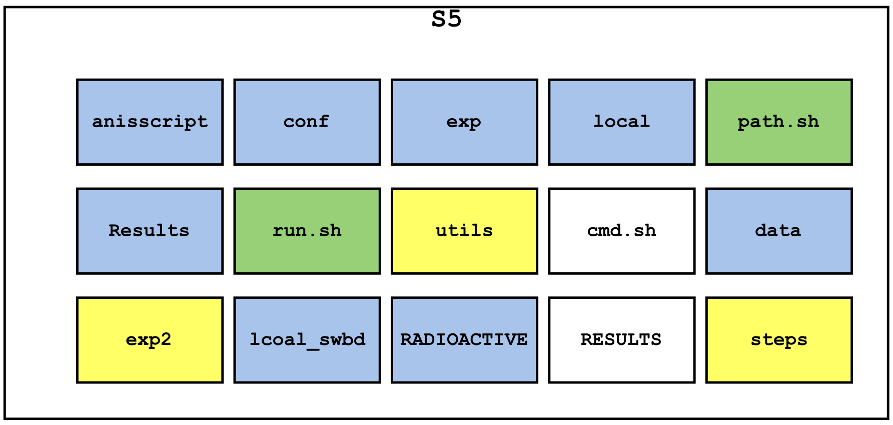

Figure 1: All components under s5 folder. Blue are folders, green are executable bash scripts, yellow are soft links to shared folders and white are text files

Each of the components will be explained in the next subsections. I will treat them in order of importance: from the less important or less used, to the most important.

2.1 conf

This is a folder where some of the configuration files are stored. The one important file is *decode.config* which contains the feature transform to be used for decoding speech. This remains unchanged as long as we use the same baseline DNN. In these adaptation experiments we use (*dnn_fbank_nolda*). The other files inside are not important.

2.2 RESULTS

This is a file where results with different system architectures of Librispeech based ASR systems are cited. This file is present in every egs folder. Not relevant in our case

2.3 cmd.sh

This script contains some global variables that are defined for the good functioning of Kaldi. Normally it is not necessary to take care of this file. It should however be invoked in the beginning of each bash script.

2.4 path.sh

This is a bash script that sets the path to the different Kaldi functions. It is also to be invoked in the beginning of each bash script, to have the possibility to run the functionalities integrated in Kaldi. No need to touch this file, it is preset already. It should be invoked in the beginning of each bash script you might want to implement in the future.

2.5 run.sh

This script is the standard script that should be able to build an ASR system over the Librispeech data from scratch. It is already there when Kaldi was installed. No need to use it in here.

2.6 utils

Those are shared utilities that you can use for every ASR system you want to develop or modify. It contains functions that may be useful for many tasks, like segmenting data, or extracting specific speaker data. It is also a folder that is proper to Kaldi.

2.7 steps

This is a shared folder of functions that are to be used in the context of ASR systems. It involves many functions for training or decoding, that are also called in the context of adaptation. To notice is the subfolder **nnet** that contains many useful functions for dealing with "nnet 1" framework.

2.8 local

This folder contains functions that are proper to building a Librispeech based ASR system. They are proper to kaldi and do not have to be used in context of adaptation

2.9 local_swbd

Just a buffer folder I used, containing local functions relative to switchboard based ASRs. Nothing very important here.

2.10 exp

Here I stored the State of the Art DNN, the framework of the former GMM-HMM system on which this DNN is based and some other stuff. We discuss its content in the following subsections.

2.10.1 tri4b

This is a soft link to the former GMM-HMM used to build the DNN we used. This folder has to stay as it is!! It is not to be changed in any way, because Wilhelm uses it. We need basically the HCLG graphs that are stored in there for decoding speech. That's the only reason we need it.

2.10.2 dnn_fbank_nolda

This is the framework of the DNN we use for our adaptation stuff. To get a better idea of the frame of this folder, check Kaldi ASR website, under nnet 1. This folder also contains recognition results of some speakers evaluation sets we used for adaptation. Those are basically the baseline values that the DNN achieves over our set of adaptation speakers.

2.10.3 adaptation_ali,adaptation_ali_2,adaptation_lat

These folders contain the alignments and the lattices of the adaptation data used respectively in supervised and unsupervised adaptation. We will talk about how these folders are produced and how the adaptation data correspondent to each speaker is extracted in the coming sections.

2.11 data

This folder contains the data (speech data, transcriptions, features...) we use. it basically contains the adaptation data, which is split corresponding to the number of utterances contained in it. It contains some other data from the Librispeech corpus, from which we extracted the adaptation data. An important folder called **lang** is in there. This is a soft link to one of the folders used by wilhelm. This folder contains basically the lexicon and the dictionnary that map phonemes to words. It is not to be changed, and it has to be used when aligning the adaptation data. through these alignments we get the adaptation data in a form we can use for supervised adaptation. We will come to this matter later.

2.12 exp2

This folder contains basically all the experiments results obtained through adaptation. This is a softlink to a folder based in *spielwiese 2* hardrive. Only two of these folders are relevant:

2.12.1 dnn_adaptation_libri

This folder contains all the results (results, adapted dnns, decoding outputs...) following the adaptation experiments we conducted on the state of the art ASR. A Better explanation to what is in here, and how it is to be interpreted comes later in this document.

2.12.2 dnn_adaptation

This folder contains all the results obtained through adapting a former system ("ASR System 1"). It has the same architecture as dnn_adaptation_libri.

2.13 Results

This folder contains the results of adaptation. It does not contain the frameworks or the decoding outputs. It just contains the WERs extracted after adaptation. If you need only the numerical results, than this folder should be enough. Otherwise you have to check *dnn_adaptation_libri*.

2.14 anisscript

This folder, like the name shows contains all the bash scripts and the configuration files that I implemented for the sake of adaptation. We will look at it in the next chapter

2.15 RADIOACTIVE

This is a buffer folder. It contains files that are generated automatically during adaptation. No needs to do changes here. It is basically a buffer for summing probabilities in the case of KLD regularized adaptation.

3 anisscript

3.1 dnn_proto

This folder contains the prototypes of the DNNs we would like to adapt. Basically the prototype of the baseline DNN or the prototype of the DNN where only the input layer can be retrained. It can also contain the DNN prototype where only the output layer can be retrained. Those prototypes are then invoked in the **config** folders so the training algorithm knows which DNN to retrain.

3.2 config folders

These folders contain configuration information about the way we retrain the DNNs in the context of adaptation. Information like the feature transform to be used, number of epochs, learning rate, prototype of the DNN to be trained...

config

This folder contains the configuration files for different number of epochs used for adaptation and some other options concerning the framework

config_1,config_cpu,config_dev_2

Not useful, just some other configuration files used for some circumstances.

config_dev_1_0,config_dev_1_1,config_dev_1_2,config_dev_1_3,

Those are configuration folders used when adapting a development set. Each folder contains files with the number of adaptation epochs and other stuff. Each one of the folder is different from the others by the learning rate chosen for adaptation.

config_input,config_output

Those folders contain also the configuration parameters of adaptation. They contain also the dnn prototypes to be used when retraining only the first and the last layers of the dnns. In Kladi there is no possibility to explicitly choose the layers you want to retrain. You basically need to manually change the prototype of the dnn (*final.nnet*) by setting the learning coefficients to zero, there where you don't want the layers to be retrained. This can be done like the following:

1. Copy the final.nnet file of the needed DNN using the following script: nnet-copy –binary=false final.nnet output-file
2. Open output-file. The structure of the neural network will appear in the form of a text file.
3. Set the <LearnCoef> to zero for the layers you don't want to update
4. Save the new DNN prototype.

3.3 extras,old

Those are files that I used to use for the adaptation of the old ASR system. No need for them.

3.4 run_dnn_baseline.sh,run_gmm_baseline.sh

Those are files I used for generating ASR System 1. It was basically the first ASR System I generated and used for adaptation. No need for these files, only if a new comer wants to see how things work. You need to run the gmm generation first and then the DNN. Maybe some variables need to be changed.

3.5 spk_list

Just a buffer file to extract the adaptation data depending on the speaker we want to adapt.

3.6 decode_eval2000.sh

Just a script for decoding a test set used for evaluation in state of the art systems. Some variables might need some changes there. This script can be used in order to get an idea about how good your system is in comparison to other systems present in the literature.

3.7 run_dnn_adaptation.sh

This is the script that is used to adapt and the decode the results of the adaptation for a certain speaker. An example of how to use this script is to be found under **s5/anisscript/utilities/scriptor/scriptor.sh**. This script works only for entire adaptation of the network or only input and output. KLD regularization is taken care by another script.

3.8 run_dnn_adaptation_kld.sh

This script is used in the framework of KLD regularized adaptation. With this script, you can choose the ρ values you want to use for adaptation. An example script can be found under **s5/anisscript/utilities/kld/kld_05.sh**, where we use a value of 0.5 as KLD regularization coefficient.

3.9 run_dnn_adaptation_unsup.sh

This script should normally be used for unsupervised adaptation in the case of adapting a whole network or just input or output. Though, I didn't update this script. So to do the unsupervised adaptation, you could use the script we describe later (**run_dnn_adaptation_kld_unsup**). The idea is to set the KLD coef to 0 like shown in the example:
s5/anisscript/utilities/scriptor/scriptor_unsup.sh.

3.10 run_dnn_adaptation_kld_unsup.sh

This script is used for the unsupervised adaptation in the case of KLD regularization. For an example script, you can check the script that runs a KLD regularized adaptation with $\rho = 0.75$ in **s5/anisscript/utilities/kld/kld_unsup_075.sh**

3.11 utilities

This folder contains useful scripts, used for gathering adaptation data, splitting it, extracting alignments...

3.11.1 baseline_decode_adapt.sh

This script decodes the adaptation sets of the speakers used for adaptation. It is useful to get these results in the case of unsupervised adaptation in a test only setup.

3.11.2 baseline_decode_eval

This script decodes the evaluation data of each speaker from the adaptation set. It gives the best WER for each of the evaluation sets when using the baseline system. Those values are used as the baseline performance for each of the speakers evaluation sets.

3.11.3 best_wer_collector.sh

This script collects all the results over the speakers after adaptation in one file. It list the speaker id, number of adaptation utterances, epochs of adaptation, WER... Example: If you need the results of adaptation (without regularization and over all the network) of speaker **2002-139469**, then you go to **s5/exp2/dnn_adaptation_libri/lr_1_all/2002-139469/adapt_results.txt**, And you will find all the results in a table form. This script is used always after finishing an adaptation experiment to collect all the needed data in one file. An example of how it is used can be found under the adaptation script **s5/anisscript/utilities/scriptor/scriptor.sh**.

3.11.4 get_adapt_ali.sh

This one is very important. This script gets the ground truth alignments of the adaptation data, which will be then used as labels for retraining (adapting) the network with the adaptation data. The alignments will be then stored in a folder that will be used later in the adaptation scripts (Supervised case) described in sections 3.7 and 3.8 .

3.11.5 get_adapt_lat.sh

This script is also very important, since it gets the alignments for the unsupervised adaptation task. It decodes the data used for adaptation with the baseline system, stores the lattices we get for each of the decoding results. From these lattices, we can get pdf-posteriors we use as labels for the unsupervised adaptation in the script written for that purpose, basically the scripts described in sections 3.9 and 3.10.

3.11.6 get_data.sh

This script extracts the data from a certain corpus speaker-wise. you have just to put the speaker id, and the corpus from where you want to extract the data.

3.11.7 split_data.sh

This script splits the data into adaptation data with different data sizes, and to an evaluation data for each of the speakers.

3.11.8 kld,scriptors

These folders contain some example scripts of how to run adaptation.

4 Running an example script

Now let's see how these scripts are run.

4.1 Supervised KLD regularized adaptation

An example script can be found under **s5/anisscript/utilities/scriptor_kld.sh**. first you have to name the variable *adaptation_method* that determines the name of the export folder where the results will be stored. It will be under **s5/exp2/dnn_adaptation_libri**. You have then to declare the *speakers* you want to adapt to. These speakers should have been already selected in beforehand by using **get_data.sh** and split into adaptation and evaluation sets using **split_data.sh** and their alignments should have been extracted using **get_adapt_ali.sh**. In this file is then run the script **run_dnn_adaptation_kld.sh** over each speaker. This script takes the following arguments:

1. Speaker-id: Id of the speaker you want to run the adaptation over it.
2. Config file: this file is contained in the config folder described in 3.2. This file contains several configuration parameter on how to retrain the DNN for adaptation, like for example the network you want to retrain (baseline DNN), the number of epochs, the learning rate...
3. The adaptation method: This is the variable that defines the name of the folder where The experiment results are stored.
4. The KLD coefficient of the SI DNN (ρ). If set to 0, no regularization is done, if set to 1, the DNN doesn't learn anything through adaptation
5. The coefficient of the adaptation data ($\rho - 1$).

After all the adaptation experiments are run for the chosen speakers, comes the script **best_wer_collector.sh**. This script like described in the section 3.11.3, collects all the results like described above.

4.2 Unupervised KLD regularized adaptation

An example script can be found under **s5/anisscript/utilities/scriptor_unsup_kld.sh**. This script is very similar to the script of supervised KLD regularized adaptation. Instead of getting the alignments of the speakers adaptation data in beforehand, you need to first decode the adaptation data and get the corresponding lattices, to generate the unsupervised labels. To do that use the script **get_adapt_lat.sh** described earlier in this document, over each of the speakers you might want to adapt.