Contents

Installation of Kaldi

A step-by-step installation script

The step-by-step installation script discussed here is based on the `setup.sh` file given at https://raw.githubusercontent.com/AssemblyAI/kaldi-asr-tutorial/master/setup.sh.

This file should be saved in the **parent** directory of `kaldi`. For example, if we want to install `kaldi` in `/home/xxx/projects/kaldi`, then we need to put it at `/home/xxx/projects`.

Note that we have added section lines in the `setup.sh` file below to separate the installation into different sections. Also, we use a new name `install_kaldi_sbs.sh` to providing instance meaning and reduce the risk to have name collision.

Checking the CUDA toolkit

Kaldi can use GPU to speed up the NN training. If we have GPU and want to use it, we need to have CUDA toolkit installed. To check if the toolkit is installed, we run

```
nvcc --version
```

If it displays that `nvcc` cannot be found, we just need to install it using `sudo apt install nvidia-cuda-toolkit`

The installation script

`install_kaldi_sbs.sh`:

```
#!/bin/sh

#-------------------------------------------------------
# Section 1. Set up the number of CPU cores for make
#
if test -n "$1"; then
    cores=$1
```

```
else
    cores=1
fi
echo "Using $cores core(s) for \`make\` and \`make depend\`"

#-------------------------------------------------------
# Section 2. update and upgrade package using apt
#
echo "Upgrading packages"
yes | sudo apt update
yes | sudo apt upgrade

#-------------------------------------------------------
# Section 3. Install the required packages for Kaldi
#
echo "Installing required packages"
yes | sudo apt install unzip git-all
pkgs="wget g++ make automake autoconf sox gfortran libtool subversion python2.7 python3.8 zlib
yes | sudo apt-get install $pkgs

#-------------------------------------------------------
# Section 4. Install tools for Kaldi
#
echo "Installing Kaldi"
echo "Installing Kaldi - Tools install"
git clone https://github.com/kaldi-asr/kaldi.git kaldi --origin upstream
cd ./kaldi/tools
extras/install_mkl.sh
extras/check_dependencies.sh

#-------------------------------------------------------
# Section 5. Check the dependencies needed for the installation of Kaldi
#
echo "Did you receive the \`all OK.\` message? (y/n)"
read response
if [ $response == "n" ]; then
    echo "Please install all required dependencies and then continue the installation with ./k
    exit
fi

#-------------------------------------------------------
# Section 6. Build tools for Kaldi
#
if [ 1 == "$cores" ]; then
    yes | make CXX=g++
else
    yes | make CXX=g++ -j $cores
fi
```

```
#-------------------------------------------------------
# Section 7. Install Kaldi from source
#
echo "Installing Kaldi - Src install"
cd ../src
./configure --shared
if [ 1 == "$cores" ]; then
    yes | make CXX=g++
    yes | make depend CXX=g++
else
    yes | make CXX=g++ -j $cores
    yes | make depend CXX=g++ -j $cores
fi

# The if block can be changed to, if running directly:
# sudo make -j $cores clean depend; sudo make -j $cores

#-------------------------------------------------------
# Section 8. Install a demo
#
echo "Kaldi Install Complete"
echo "Cloning Project Repository"
cd ../egs
git clone https://github.com/AssemblyAI/kaldi-asr-tutorial.git
cd kaldi-asr-tutorial/s5
```

Manual installation of dependencies

We have issues running the above script. Specifically, when we check the dependencies at the end of Section 4, we see that some modules are not installed. These modules are listed in the terminal, and we can copy the list and do a sudo installation.

Here is the display:

```
extras/check_dependencies.sh: Some prerequisites are missing; install them using the command:
  sudo apt-get install zlib1g-dev automake autoconf sox gfortran libtool subversion python2.7
Did you receive the `all OK.` message? (y/n)
```

We just need to install using the suggested command. Here `apt-get` can be replaced by `apt`. Note that these packages should have been installed in Section 3; we are not sure what has prevented their automatic installation.

After the above installation, we run `extras/check_dependencies.sh` again, and everything is fine. We then move on to use the manual installations steps in Sections 5 and 6.

Manual installation of IRSTLM

At the end of Section 6, there is a warning:

```
Warning: IRSTLM is not installed by default anymore. If you need IRSTLM
```

```
Warning: use the script extras/install_irstlm.sh
```

For convenience, we just go ahead to install using the following:

```
sudo ./extras/install_irstlm.sh
```

At the end of installation, we are reminded

```
Please source the tools/env.sh in your path.sh to enable it
```

Note that this is automatically done in each `path.sh` file under the `s5` folder. The corresponding line is

```
[ -f $KALDI_ROOT/tools/env.sh ] && . $KALDI_ROOT/tools/env.sh
```

Manual installation of Kaldi from source

After we install the dependencies, we just move forward to install Kaldi from source as indicated in Section 7. Note that we need to use sudo for each step.

Testing the installation

We can follow Step 8 to install the demo project. Then, we can use the steps at Kaldi Speech Recognition for Beginners - A Simple Tutorial (assemblyai.com) to test the installation.

To run more examples, we can download the LibriSpeech test dataset, named `test-clean`.

The speech signals are saved in the flac format. We can use

```
ffmpeg -i flac_file.flac wav_file.wav
```

Issues

Python version

We use conda to handle the versions of Python. We can start the installation in any env. It seems that some required packages are built using Python2.7, and some other Python3.8. We have Python3.10 installed already, and it is used to replace Python3.8. It seems that there is no need to install Python 3.8 specifically.

Python-not-found issue

On Ubuntu, when we run Kaldi, we may see the following problem:

```
# /usr/bin/env: python: No such file or directory
```

The solution to this Python-not-found issue is to run `sudo apt install python-is-python3`.

GPU out of memory

This happened when we train thchs32. According to Dan, https://groups.google.com/g/kaldi-help/c/ExgtEg_vnxc, we can change the number of jobs; we need to change to no more than the number of GPUs we have (e.g. 1). So, we need to change at the following snippet in `run_tdnn-f_common_skip.sh`:

```
 steps/nnet3/chain/train.py --stage $train_stage \
    --cmd "$cuda_cmd" \
    --feat.cmvn-opts "--norm-means=false --norm-vars=false" \
    --chain.xent-regularize $xent_regularize \
    --chain.leaky-hmm-coefficient 0.1 \
    --chain.l2-regularize 0.0 \
    --chain.apply-deriv-weights false \
    --chain.lm-opts="--num-extra-lm-states=2000" \
    --egs.dir "$common_egs_dir" \
    --egs.stage $get_egs_stage \
    --egs.opts "--frames-overlap-per-eg 0" \
    --egs.chunk-width $frames_per_eg \
    --trainer.num-chunk-per-minibatch 128 \
    --trainer.frames-per-iter 1500000 \
    --trainer.num-epochs 6 \
    --use-gpu='wait' \
    --trainer.optimization.num-jobs-initial 2 \
    --trainer.optimization.num-jobs-final 2 \
```

We need change the last two `2`s to `1`s.

Suggested installation steps

These are just thoughts, which have not been done yet.

According to the above steps, we can have the following installation steps in a few steps:

Step 1. Check dependencies. Run this script in the directory outside `kaldi`.

`install_kaldi_deps.sh`:

```
#!/bin/sh

#-------------------------------------------------------
# Update and upgrade packages using apt
#
echo "Upgrading packages"
yes | sudo apt update
yes | sudo apt upgrade

#-------------------------------------------------------
# Install the required packages for getting Kaldi
#
echo "Installing required packages for getting Kaldi"
yes | sudo apt install unzip git-all

#-------------------------------------------------------
# Install tools for Kaldi and check dependicies
#
echo "Installing Kaldi"
echo "Installing Kaldi - Tools install"
git clone https://github.com/kaldi-asr/kaldi.git kaldi --origin upstream
cd ./kaldi/tools
```

```
extras/install_mkl.sh
extras/check_dependencies.sh
```

Step 2. Install dependencies according to the suggested command in the last step. Run `sudo ./extras/check_dependencies.sh` again to check the dependencies. If we receive the "all OK." message, then move on to the next step.

Step 3. Make sure we are in the `kaldi/tools` directory. Run the following script. Note that we can specify the number of threads for the building of the tools. This is somehow related to the number of CPU cores. If not specified, we use 4.

`install_kaldi.sh`:

```
#!/bin/bash

if test -n "$1"; then
    cores=$1
else
    cores=4
fi
echo "Using $cores core(s) for \`make\` and \`make depend\`"

#-------------------------------------------------------
# Build tools for Kaldi
#
if [ 1 == "$cores" ]; then
    yes | make CXX=g++
else
    yes | make CXX=g++ -j $cores
fi

#-------------------------------------------------------
# Install Kaldi from source
#
echo "Installing Kaldi - Src install"
cd ../src
./configure --shared
if [ 1 == "$cores" ]; then
    yes | make CXX=g++
    yes | make depend CXX=g++
else
    yes | make CXX=g++ -j $cores
    yes | make depend CXX=g++ -j $cores
fi
```