Training process for mastering in speaker recognition

Version1 – Bar Madar, 19.11.19

Part 1 - Theoretical knowledge

Read the following articles and tutorials above. Make sure you understand every component, algorithm and methodological approach that used as a part of the speaker recognition system and process. this is a necessary part before the practical training.

- ### Task 1.1 - Basic knowledge of speaker recognition
  1. Speaker recognition by machine and human - https://www.researchgate.net/publication/282940395_Speaker_Recognition_by_Machines_and_Humans_A_tutorial_review
  2. Nave Thesis – sending you by mail

- ### Task 1.2 - Frontend steps
  1. Features extraction (MFCC) - http://practicalcryptography.com/miscellaneous/machine-learning/guide-mel-frequency-cepstral-coefficients-mfccs/
  2. Voice Activity Detection (VAD) - http://practicalcryptography.com/miscellaneous/machine-learning/voice-activity-detection-vad-tutorial/
  3. Data augmentation techniques – RIR and MUSAN corporates - https://danielpovey.com/files/2017_icassp_reverberation.pdf https://arxiv.org/pdf/1510.08484.pdf
  4. X-VECTOR and TDNN architecture- https://www.danielpovey.com/files/2015_interspeech_multisplice.pdf https://www.danielpovey.com/files/2018_icassp_xvectors.pdf https://www.youtube.com/watch?v=8nZjiXEdMH0

- ### Task 1.3 - Backend steps
  1. Linear Discriminant Analysis(LDA) - http://www.music.mcgill.ca/~ich/classes/mumt611_07/classifiers/lda_theory.pdf
  2. Probabilistic Linear Discriminant Analysis(PLDA) - http://people.irisa.fr/Guillaume.Gravier/ADM/articles2018/Probabilistic_Linear_Discriminant_Analysis.pdf
  3. PLDA adaptation for mismatch domain - http://danielpovey.com/files/2014_slt_dnn.pdf https://arxiv.org/pdf/1812.10260.pdf
  4. Scoring Normalization - https://www.sciencedirect.com/science/article/pii/S1051200499903603
  5. Measuring performances with DET curve - https://pdfs.semanticscholar.org/ae28/6f0d7fe2555ed16f405c59ac81eb94a9aec2.pdf

- **<u>Task 1.4 – Presentation of what you learned so far</u>**
  1. Create a presentation that include the following subjects-
     a. Basically explanation of the speaker recognition task
     b. Block diagram of the automatic speaker recognition system based on x-vector speaker representation and PLDA classifier.
     c. Explanations of each component on the system
     d. Ways to measuring the performances of the system
  2. Schedule a time to represent it to the research group

Part 2 - Install support systems

There are 2 ways working with kaldi software - install it on the private PC or building a Docker environment that support kaldi and running it on the server. Both ways have their advantages and disadvantages. We need to success working on kaldi in both ways to make our work more efficient.

- **Task 2.1 - Install kaldi on PC**

1. Ensure Python3 is install on your PC
2. Open a github profile – you will need it to clone and build the toolkit and keep it updated.
3. Go to the master branch of the Kaldi project on github - https://github.com/kaldi-asr/kaldi, clone the project to your PC and follow the installation instructions on the main page (you have to work with LINUX operating system).
4. After you finished all the installation steps, running the “yesno” recipe to check if the KALDI project is successfully installed on your PC and succeed to build the code -
   a. Open the terminal and go to the main brunch of the KALDI project.
   b. Type “cd egs/yesno” to get into the “yesno” recipe.
   c. Type “./run.sh” to run the recipe. The code should run, and return 0% error (indication on the terminal).
   d. Congratulation you have KALDI on your PC!

- **Task 2.2 - Install SOX on PC**

SoX is a cross-platform (Windows, Linux, MacOS X, etc.) command line utility that can convert various formats of computer audio files in to other formats. It can also apply various effects to these sound files. With sox, we will implement the data augmentation, changing the format of our audio files, down-sample or up-sample the data, and cutting audio files with specific time steps.

1. Go to SOX home page - http://sox.sourceforge.net/
2. Follow the installation instructions – you can download the installation files or clone it using git.
3. Open the SoX documentation file - http://sox.sourceforge.net/sox.pdf
4. Download a “.wav” file, save it on a folder inside the sox path and do the following actions using the sox (due to the documentation file):
   a. Change to “.sph” format.
   b. Check the sample rate of the audio file.
   c. down-sampling to 8 Khz.
   d. Up-sampling t 16 Khz.
   e. Choose a time interval and generate a new audio file between this interval with a different format and sampling.
   f. Congratulation you have SoX on your PC and know how to use it!

- **<u>Task 2.3 – Building a Docker that support KALDI and working on the server</u>**
  1. Create a user on the group servers, been able to access to it via SSH.
  2. Open your own folder on the main path at the server.
  3. Read the following tutorial to learn more about docker - https://docker-curriculum.com/
  4. Using the following links and command to build your own docker that support the kaldi project (the image has also Python3 and SoX) -
     https://itml.miraheze.org/wiki/Docker
     https://github.com/kaldi-asr/kaldi/tree/master/docker
     Sudo docker run –it –mount type=bind, source=/storage/,target=/common_space_docker/ -p 7777:1111 --name DOCKER_NAME --runtime=nvidia kaldiasr/kaldi:gpu-latest
  5. Follow the steps on the following link to activate the ssh-server and how to using the ssh to run the docker -
     https://itml.miraheze.org/wiki/Docker
  6. Run the docker using the ssh and make an alias for the python3 using -
     $sudo alias python="python3"
  7. Copy the KALDI folder from the main path of the docker to the mounted folder (common_space_docker).
  8. Run the docker, go to the "yesno" recipe path and type "./run.sh". ensure that everything run like expected with 0% error.
  9. Congratulation you have KALDI docker running on the server!

- **<u>Task 2.4 – Get access to the docker using pycharm on your PC</u>**

  A simply way to run the kaldi project on the server is to get access to the docker with the pycharm on your pc (lab's PC or laptop using VPN) – add deployment server. You can change the code and get access to the files in the docker with the pycharm and run the code with the ssh access to the server.
  1. Open the Pycharm on your PC.
  2. Preferences >> build, execution, deployment >> deployment
  3. Add new deployment.
  4. Choose name and type SFTP.
  5. SFTP host: <server ip>
  6. Port: 7777 (compatible with the initial run of the docker)
  7. Root path: /
  8. Username: root (compatible with activate ssh-server)
  9. Password: XXXXXX (compatible with activate ssh-server)
  10. Mapping: map your local path to the deployment path on server – the local path is where files from the server will download to.
  11. Push test connection and verify that it is working.
  12. Congratulation you can access your docker from the pycharm!

Part 3 - Practical experience

On this step, we will experience on using KALDI and it's methodology. First we will run a recipe and check it performance with pretrained models of the x-vector extractor and PLDA, after we will train those models. At the end, we will experience on running a speaker diarization recipe.

- **Task 3.1 – the basic of KALDI**
  1. On the the link below, follow the kaldi for dummies tutorial - http://kaldi-asr.org/doc/kaldi_for_dummies.html
  2. On the link below, follow the data preparation tutorial - http://kaldi-asr.org/doc/data_prep.html
  3. If you have any question about KALDI, you can search for it in KALDI documentation - http://kaldi-asr.org/doc/index.html
  4. Join to the "kaldi-help" group, where you can ask questions about kaldi and see other problems and answers - http://kaldi-asr.org/forums.html

- **Task 3.2 – get all the neccessary data bases to run the sre recipes**
  1. SRE 04, 05, 06, 10, 16, 18, 19 SRE10 – sharing on the google drive.
  2. MIXER6 – sharing on the google drive.
  3. SWBD – sharing on google drive.
  4. MUSAN – download from https://www.openslr.org/17/
  5. RIR – download directly from the run.sh script on each recipe.
  6. VOXCELEB – download from http://www.robots.ox.ac.uk/~vgg/data/voxceleb/
  7. Make sure you have all the data on you PC or server. Take a look on the format and sampling of each data set. Also, get familiar with each data set structure (divide to sections, test/train, trial keys, etc.).

- **Task 3.3 – download the pretrained models (xvector and PLDA)**
  1. In order to run the sre16 recipe without training the PLDA and xvector extractor, we have to download the pretrained models, and copy it to the recipe path as described on - https://davidryansnyder.github.io/2017/10/04/model_sre16_v2.html

- **Task 3.4 – run the sre16 recipe without train the xvector and PLDA**
  1. On kaldi path go to egs/sre16/v2.
  2. Go over all the steps in run.sh except those who train the x-vector extractor and the PLDA models.
  3. Look carefully on each script you run as a part of the recipe, and follow its input and output. Make sure you understand each step of the recipe.
  4. Check if your system performances are matched to those who written in comment on the run.sh recipe.
  5. Write a Python code that get the scoring of the system as an input and its output is a DET curve with EER written to a blank file.

- **Task 3.5 – run the full sre16 recipe**
  1. run the same sre16 recipe. Now the recipe includes training of the x-vector extractor and PLDA model.
  2. Make sure you run the recipe on the docker so you can use the GPU's for the training process.

- **Task 3.6 – run the SRE19 recipe**
  1. Ask from Bar the SRE19 recipe and run it. It includes mix of SRE and Voxceleb training set, augmentation for the Voxceleb and in-domain data, and plda adaptation.
  2. Test the model on SRE18 test set, using NIST scoring software.
     a. Normalize the scoring to get act_C close to min_C as much as you can.
  3. Generate a DET curve of the system performances.

- **Task 3.7 – compare the SRE19 and SRE16 recipe**
  1. Test the 2 recipes both with sre18 test set.
  2. Get the scoring results of each recipe using the sre18 scoring software.
  3. Compare the performances of the systems using a single DET graph with 2 curves.

- **Task 3.8 – Speaker diarization**
  1. Read the article below and implement the speaker diarization system regard to it, using the call_home_v2 recipe - https://towardsdatascience.com/speaker-diarization-with-kaldi-e30301b05cc8
  2. Go to Nave's Thesis and read the section of speaker diarization.
  3. Run the BGU SRE18 speaker diarization recipe - https://github.com/navealg/SRE18_BGU
  4. check the performance and compare to the result on BGU SRE18 system description.