NATIONAL TECHNICAL UNIVERSITY OF ATHENS
SCHOOL OF ELECTRICAL & COMPUTER ENGINEERING

Speech & Language Processing

Summer Semester

2nd Lab: Speech Recognition with Kaldi toolkit

1 Description

The purpose of this exercise is to implement a speech processing and recognition system with the Kaldi toolkit, which is widely used in research and industry for the training of state-of-the-art speech recognition systems.

More specifically, the system you will develop concerns Phone Recognition using USC-TIMIT recordings. You will be given audio data from 4 different speakers, with the corresponding transcriptions, to train and evaluate your system.

The system design process can be divided into 4 parts.

The first part aims to extract appropriate acoustic features from the voice data (Mel-Frequency Cepstral Coefficients). These features are essentially a number of cepstrum coefficients that are extracted after analyzing the signals with a specially designed filter array (Mel filterbank). This array is inspired by the non-linear way in which the human ear perceives sound and is inspired by psychoacoustic studies.

The second part concerns the creation of language models from the transcripts of the data set, which will give an a-priori bias to the final system.

The third part concerns the training of acoustic models using the acoustic features that have been exported.

Finally, by combining the above units, the final speech recognition system can be constructed, which, given a voice signal, it produces the audio features and uses them to decode the signal into a sequence of sounds or words or phonemes.

2 Background

During the preparation you should get acquainted with specific concepts that will be used during the lab. Specifically, we would like you to know about the following concepts:

1. Mel-frequency Cepstral Coefficients (MFCCs)
2. Language Models
3. Acoustic Models

In your final report, you are tasked to briefly develop the above concepts and comment on their performance. Do not stay in the steps of the basic algorithms but try to suggest / evaluate how the voice recognition system you have developed could be improved.

In preparation for the workshop you can refer to the following material:

- Chapter 14 of the textbook **[R & S] Theory and Applications of Digital Speech Processing** of Lawrence R. Rabiner and Ronald W. Schafer (Pearson, 2011), on Automatic Speech Recognition (ASR) systems.
- Mel-frequency Cepstral Coefficients (MFCCs)
- GMM-HMM for acoustic modeling
- Kaldi tutorial 1
- Kaldi tutorial 2
- Kaldi tutorial 3

3 Preparatory Steps

1. Install Kaldi on your computer according to the instructions given in https://helios.ntua.gr/.

2. Familiarize yourself with the Kaldi tool. The language in which it was developed is C++, but the main functions we are interested in are called from bash scripts. There exist implemented recipes for speech recognition models for common datasets inside the Kaldi *egs* folder. For the dataset we will provide you will have to mix existing scripts from the base recipe (*wsj*) with your own to create a new recipe.

3. Download the data from the following link:

   `https://drive.google.com/file/d/1_mIoioHMeC2HZtIbGs1LcL4kkIF696nB/view?usp=sharing`

   The data includes recordings from 4 speakers with names: m1, m3 (male) and f1, f5 (female). Each speaker has produced 460 utterances (Note: speaker m1 lacks utterances 231 to 235 due to a recording error).

   The audio files are located in the *wav* folder. Filenames are prefixed by speaker name and followed by the utterance id. In the file *transcription.txt* you will find the text spoken by the speakers in each utterance (1st line → 1st utterance, 2nd line → 2nd utterance etc.) and in the folder *filesets* you will find which utterances correspond to the training set, the validation set and the testing set (training, validation, testing).

4. Recipe construction:

   - Inside the *egs* folder, create a *usc* subfolder, which will be your working directory.
   - Create the folder *data* and the subfolders *data/train*, *data/dev*, *data/test* inside *usc*, in which you will create index files that will describe the training, validation and testing data respectively.
   - In each of these 3 folders you must create the following files:
     - *uttids*: contains in each line a unique symbolic name for each utterance in the corresponding training/validation/test subset (i.e. the contents of the files in the folder *filesets*) which from now on will be referred to as utterance_ids
     - *utt2spk*: contains on each line the speaker that corresponds to each utterance and is of the form:

       utterance_id_1 <*space*>*speaker_idutterance_id*_2 <space> speaker_id etc.

       where as speaker_id we select m1, m3, f1, f5 respectively
     - *wav.scp*: contains the location of the audio file that corresponds to each sentence and is of the format:

       utterance_id_1 <space> /path/to/wav1
       utterance_id_2 <space> /path/to/wav2
       etc.

     - *text*: contains the text that corresponds to each utterance and is of the form:

       utterance_id_1 <space> <utterance 1 text>
       utterance_id_2 <space> <utterance 2 text>
       etc.

   - Finally, for each *text* file you create, you must replace the words in the sentences with the corresponding phoneme sequences. For this reason you are given along with the rest of the data the dictionary (lexicon.txt), which maps each word of the English language to the corresponding sequence of phonemes. Be careful in this step to first convert all characters to lower case, as well as remove special characters (e.g. periods, dashes, etc.) except for single quotes ('). Also, at the beginning and at the end you have to add the silence phoneme (sil). The following example is given for the 1st utterance of speaker f1 (f1_001):

     This was easy for us.

     Will be converted to:

     sil dh ih s w ao z iy z iy f r er ah s sil

4 Main lab steps

4.1 Preparing the USC-TIMIT speech recognition system

1. From the Wall Street Journal (*wsj*) recipe under `egs/wsj/s5` you must copy the files `path.sh` and `cmd.sh`.
2. Inside the file path.sh you must set the `KALDI_ROOT` variable to point to your Kaldi installation directory. You will source this file (`source path.sh`) in the beginning of every bash script you write so that the Kaldi commands are available to your environment.
3. Inside `cmd.sh` you must change the values of the variables `train_cmd`, `decode_cmd` and `cuda_cmd` to `run.pl`.
4. Copy the folders `egs/wsj/s5/steps` and `egs/wsj/s5/utils` inside your recipe.
5. Create the folder `local` and create a soft link to the file `steps/score_kaldi.sh` inside `local`.
6. Create the folder *conf* and copy inside it the file mfcc.conf provided to you in the helper code.
7. Finally, create the following folders: `data/lang`, `data/local/dict`, `data/local/lm_tmp`, `data/local/nist_lm`.

4.2 Building the language model

1. Inside the folder `data/local/dict` you will put the files needed for constructing the language model.
   - Create `silence_phones.txt` and `optional_silence.txt` containing only the silence phoneme (sil).
   - Create `nonsilence_phones.txt`, which contains all other phonemes (one per line, sorted).
   - Create `lexicon.txt` containing the vocabulary. Since we perform phoneme recognition (and not word recognition), `lexicon.txt` contains a 1-1 mapping of each phoneme to itself. In each line of `lexicon.txt` you must put a phoneme then a <space> and then the same phoneme again. Do not forget to include the silence phoneme (sil).
   - Create `lm_train.text` based on the file `text` you created during the lab preparation, by adding the special tokens <s> and </s> in the beginning and the end of each line respectively. Similarly create `lm_dev.text` and `lm_test.text`.
   - Finally, create an empty file `extra_questions.txt`.
2. Inside `data/local/lm_tmp` you will store the intermediate files needed during the language model training process. Use the command `build-lm.sh`, provided by the IRSTLM package that you installed along with Kaldi (it should be available after you run `source path.sh`.

   build-lm.sh -i <path to lm_train.text> -n <order of language model> -o <path_to_output_file.ilm.gz>

   Note: Train a unigram and a bigram language model.
3. Inside `data/local/nist_lm` you will save the compiled language model in ARPA format. Use **compile-lm**.

   compile-lm <path_to_intermediate_lm.ilm.gz> -t=yes /dev/stdout | grep -v unk | gzip -c > <path_to_output_file.arpa.gz>

   Name the unigram model `lm_phone_ug.arpa.gz` and the bigram model `lm_phone_bg.arpa.gz`
4. Inside `data/lang` you will create the lexicon FST (L.fst) using the Kaldi command **prepare_lang.sh**.
5. Use the `sort` command to sort the files `wav.scp`, `text` και `utt2spk` inside `data/train`, `data/dev` και `data/test`.
6. Use the script `utils/utt2spk_to_spk2utt.pl` to create a new file `spk2utt` inside `data/train`, `data/dev` και `data/test`.
7. Finally create the grammar FST (G.fst). You can base the procedure on the `egs/timit` recipe, provided by Kaldi, using the `local/timit_format_data.sh`. You may need to modify the file.

Q1: For the language models you created, calculate the perplexity on the validation and the test sets. Present your scores and explain what is the perplexity of a language model?

4.3 Acoustic feature extraction

Extract the MFCC features for all 3 sets, using the Kaldi scripts (make_mfcc.sh, compute_cmvn_stats.sh).

Q2: The second command performs the so-called Cepstral Mean and Variance Normalization. What purpose does this step serve? (Bonus: Use the math formulas of CMVN to justify/explain your answer.)

Q3: How many acoustic frames were extracted for each of the first 5 sentecnes of the training set? What is the dimension of the extracted features?

4.4 Training of acoustic models and decoding

1. Train a monophone GMM-HMM acoustic model using the training data.
   (Hint: `steps/train_mono.sh`)

2. Create the HCLG graph, using the grammar (G.fst) you created in the previous step. Try both unigram and bigram grammars. (Hint: `utils/mkgraph.sh`)

3. Decode the sentences inside the validation and test sets using Viterbi algorithm.
   (Hint: `steps/decode.sh`)

4. Use the Phone Error Rate (PER) metric to assess your models:

$$PER = 100\frac{insertions + substitutions + deletions}{\#phonemes}$$

   where $\#phonemes$ is the total number of phonemes in the transcription.

   Discuss what the 2 hyperparameters for the scoring process are in this step and what they represent. What values were chosen for your best model? (Hint: Use the `local/score.sh` script. Results are stored in `exp_mono_bg/decode_test/scoring_kaldi/best_wer`)

5. Perform phoneme alignment using the monophone model you just trained. Then using these alignments, train a triphone model. Create the new HCLG graph, and use it for decoding and evaluation. Present and compare your results. (Hint: `steps/align_si.sh`, `steps/train_deltas.sh`)

Q4: Explain the structure of a GMM-HMM acoustic model. What is the purpose of Markov models in this case and what is the purpose of the Gaussian mixtures? How is this model trained? Describe in detail the training process of a monophonic model and the algorithms used.

Q5: Explain how we calculate the a-posteriori probability according to the Bayes rule for speech recognition problem. Specifically, how do we determine the most likely word (or phoneme in our case) given a sequence of acoustic features?

Q6: Describe the HCLG graph.

4.5 Bonus: DNN-HMM model using PyTorch

For this step you will use the provided helper code. Install the kaldi-io-for-python library (`https://github.com/vesis84/kaldi-io-for-python`). You can use the following command:

```
pip install git+https://github.com/vesis84/kaldi-io-for-python
```

Then download the helper code from `https://github.com/slp-ntua/slp-labs` inside your recipe.

1. Extract the triphone alignments for the train, validation and test sets.

2. Extract the cmvn statistics as in `run_dnn.sh`

3. Use the provided class TorchSpeechDataset to read the train, validation and test data. This class contains the properties: `feats`, which is an array that stores the MFCCs, `labels` where the phonemes are stored as numeric labels starting from 0, `uttids` where the utterance_ids are stored and *end_indices* where we save the locations (indices) of the `feats` table where the features for each sentence end and the next sentence begins.

4. Create a Deep Neural Network (DNN) that classifies the phonemes, given an input frame. You need to train your network on the train set, and use the validation set for hyperparameter selection. The network must consist of feedforward layers, followed by dropout and relu activations (except the last output layer). 1D Batch Normalization can be added to the input of each Layer.

5. Using your best model, predict the phonemes of the test set and save the a posteriori probabilities for each acoustic frame in a format recognizable by Kaldi scripts (example will be given).

6. Finally, use the `decode_dnn.sh` script to perform decoding on the test set using the a-posteriori probabilities, as in `run_dnn.sh`. Discuss your results.

Q7: Explain how the DNN-HMM model differs from a GMM-HMM model and how a DNN is combined with an HMM. What might be the benefit of adding DNNs over GMMs? Would it be possible to train a DNN-HMM from the beginning?

Q8: Explain Batch Normalization.

Deliverables:

1. A report, where you discuss the concepts covered in this lab (MFCCs, language and acoustic models). Do not limit yourselves to the steps of the basic algorithm but try to suggest how the speech recognition system you have developed could be improved. Also include a detailed description of each step, answers to the questions and a presentation of your results. Try to keep your report to the point.

2. Code with comments.