CS 753: Assignment #2

Preethi Jyothi
pjyothi@cse.iitb.ac.in

IIT Bombay — September 23, 2019

Instructions: This assignment is due on or before 11.55 pm on October 12, 2019. No grace period will be granted for this assignment. The submission portal on Moodle will be closed after 11.55 pm on October 12.

- For this assignment, you can work in groups of two or three. You can also choose to work on it individually. You will be building an ASR system for Swahili using the Kaldi toolkit.
- You will need to install the latest version of Kaldi available here. (Some useful Kaldi resources have been listed at the end of this document.)
- Click here for a detailed structure of your final submission directory. Many parts of this assignment will be auto-graded. Hence, it is **important that you do not deviate from the specified structure.** Compress your submission directory using the command: `tar -cvzf submission.tgz submission` and upload `submission.tgz` to Moodle.

Automatically Recognizing Swahili

Swahili is a widely spoken language in Africa written using the Latin alphabet. During the course of this assignment, you will develop an ASR system for Swahili. You will first set up a baseline system using the starter recipe provided here. If Kaldi is installed in the directory `[kaldi]`, untar within `[kaldi]/egs` to get a directory `assgmt2` and run the following commands:

Command Line

```
$ cd [kaldi]/egs/assgmt2/recipe
$ ./run.sh
```

This command will execute a number of scripts that prepare the datasets, create dictionary/LM FSTs, trains monophone HMMs and decodes a test set to produce a word error rate (WER) of:

Command Line

```
%WER 55.30 [ 605 / 1094, 24 ins, 162 del, 419 sub ] exp/mono/decode_test/wer_10
```

Note: All subsequent evaluations will be done on this test set. On running `run.sh`, you may not get the exact WER specified above but a slightly different number close to 55.3%; this is due to a small amount of non-determinism present in the system.

`run.sh` within assgmt2 is the main wrapper script which runs in under 5 minutes on a machine with four cores. Go through this script carefully to understand the various steps involved. You can set the variable `stage` to determine which stages of `run.sh` will be processed.

Task 1: Building improved monophone HMMs **[4 points]**

The following command within `run.sh` is used to train monophone HMMs for the acoustic model:

Command Line

```
$ steps/train_mono.sh --nj 4 --cmd "$train_cmd" \
        data/train data/lang exp/mono
```

Question 1

Within `steps/train_mono.sh`, look at the variables listed on lines 11–30 (within comments "Begin configuration section" and "End configuration section"). Figure out which variables are important by tuning on your test set and report the best-performing adjustments. Within `submission/REPORT.txt`, write down which variables you tuned (to what value) and what these variables control. Also submit a new `task1/train_mono.sh` with updated parameters that we will use to train monophone HMMs and recover your reported numbers.

Task 2: Lexicon idiosyncrasies **[4 points]**

Examine the file `data/lang/phones.txt` which is a vocabulary file defining all the phones used in our models.

Question 2

(a) Most of the phones contain suffixes of the form "_B", "_I" and so on. What do these indicate? Add your answer to `submission/REPORT.txt`.

(b) Towards the end of the file, you will see a list of phones that start with the special symbol "#". What are they used for and how were they created? Add your answer to `submission/REPORT.txt`.

Task 3: Train triphone HMMs **[4 points]**

Uncomment the following lines to train tied-state triphone HMMs.

Command Line

```
$ steps/align_si.sh --nj "$nj" --cmd "$train_cmd" \
   data/train data/lang exp/mono exp/mono_ali
$ steps/train_deltas.sh --boost-silence 1.25 --cmd "$train_cmd" \
   2000 20000 data/train data/lang exp/mono_ali exp/tri1
```

Question 3

(a) Report the improvements in performance on the test set using triphone models compared to monophone models in `submission/REPORT.txt`. (Note that you will need to update stage 5 in run.sh to use the triphone models during decoding instead of the monophone models.)

(b) `steps/train_deltas.sh` takes two arguments 2000 and 20000. What are these hyperparameters? Tune them and report how this tuning affects performance on the test set within `submission/REPORT.txt`. Submit the final, tuned hyperparameter values within `run.sh` in the call to `steps/train_deltas.sh`. (If you have edited `steps/train_deltas.sh`, please submit a `task3/train_deltas.sh` that we will use to train tied-state triphone HMMs.)

Task 4: Language models [4 points]

For the baseline model, you were given an (unnamed) smoothed trigram language model in `corpus/LM/swahili.small.ar`

Question 4

Build different smoothed Ngram models (of higher order, with interpolation, etc.) with the help of SRILM tools trained on `corpus/LM/train.txt`. (More information about SRILM was provided via Moodle.) Use the test set to identify the best of these language models and submit a file `task4/G.fst` that we can evaluate with the tied-state triphone model in task 3.

Task 5: Rescoring with a larger LM [4 points]

The LM `corpus/LM/swahili.small.arpa` was trained on ≈20K lines of text. Say you are given access to an LM trained on a corpus that is ten times larger consisting of ≈200K lines of text: `corpus/LM/swahili.big.arpa`.

There are two ways in which a larger LM could be used:

(a) Create a new G.fst, followed by a new decoding graph. Run `decode.sh` with this new graph in place.

(b) Create a new G.fst. Use the script `steps/lmrescore.sh` to rescore lattices created with the old LM using the new LM.

Question 5

Implement both the techniques mentioned above and report WERs in `submission/REPORT.txt`. Which of the two is faster? Share the new G.fst within `task5/G.fst`. Additionally, create a new stage in run.sh (numbered 6) that we can run to implement part (a).

Task 6: Analysis of performance on test set [4 points]

Utterance IDs in the files within `data/test` have the format 16k-emission_swahili.*_part001Q where the last character Q stands for the quality of the utterance: "g" ⇒ utterance is good, "m" ⇒ utterance has background music, "n" ⇒ utterance has noise, "l" ⇒ utterance is very noisy.

Instead of providing a single WER for all the utterances in `data/test`, provide WERs for utterances grouped by their quality (Q) tag. Use the ASR system from task 4 for evaluation. Comment on which utterances have the highest and lowest WERs in `submission/REPORT.txt`. Akin to `local/score.sh`, submit a script `task6/score.sh` that takes the same command line arguments. Here is a sample run,

Command Line

```
$ ./task6/score.sh --cmd "run.pl" data/test \
    exp/tri1/graph exp/tri1/decode_test
```

and here is a sample output from running this script:

Command Line

```
(g) %WER 42.30 [ ??? / 994, ? ins, ? del, ? sub ]
(l) %WER 55.81 [ ??? / 50, ? ins, ? del, ? sub ]
(m) %WER 48.30 [ ??? / 25, ? ins, ? del, ? sub ]
(n) %WER 47.55 [ ??? / 25, ? ins, ? del, ? sub ]
```

Any other new script that your `score.sh` calls should be within `task6`.

Task 7: Find the word

For a given word, find all audio instances of the word in the training data. Create a script `task7/findword.sh` which, when run as `./findword.sh "ninapenda"`, will output audio files `word01.wav`, `word02.wav`, etc., containing snippets of audio corresponding to the word `"ninapenda"`. You can use the commandline tool sox to snip an audio file. Any additional files you require should be within `task7`. (You can assume that the given word will be found at least once in the training data.)
(Hint: You will need to force-align the training utterances to its transcripts in order to find timestamps where the utterance can be snipped.)

Task 8: Proof of the pudding

How well does your Swahili ASR system perform on unseen data? Check `corpus/data/truetest/wav.scp` for a set of utterances for which we would like you to predict transcriptions using your best ASR system. This task is hosted on Kaggle at the following link. Before you access the Kaggle page, go to the Kaggle site and create a new login using your roll number/GPO ID. (If you are a team, it is sufficient to create an account using one of your roll numbers.) It is important that you use your roll number (and not any other pseudonym) in order to help us with grading. Upload your prediction file to the Kaggle site in the specified format. Visit here for more details.

The leaderboard will display your roll numbers along with the edit-distance based scores for your predictions. The top N (depending on where there's a clean split in WERs) performers on the leaderboard will gain extra credit points.

Useful Kaldi resources

- Kaldi tutorial for beginners.
- Kaldi lecture slides.
- Kaldi troubleshooting.