CS 753 Spring 2023: Assignment #1

Instructor: Preethi Jyothi Email: pjyothi@cse.iitb.ac.in Total: 42 points

Jan 30, 2023

Instructions: This assignment is due on or before 11.59 pm on Feb 17, 2023. The submission portal on Moodle will be open until Feb 19th with a 5% penalty for each additional day after the 17th.

- This is a group assignment. You can form groups with 2-3 members.
- There are two parts to this assignment. Part 1 comprises theory questions. For part 1, you should submit a pdf with scans of your answers. Please make sure your answer FSTs are very legibly drawn. If you're introducing any new notation (beyond what we used in class) when drawing FSTs, it should be clearly outlined in your answer pdf. In Part 2, you will build an ASR system for isolated commands.
- Click here for a detailed structure of your final submission directory. Parts of this assignment will be auto-graded. Hence, it is **very important that you do not deviate from the specified structure.** Deviations from the specified structure will be penalized. All the files that need to be submitted are highlighed in red below; all the submitted files will be within the parent directory `submission/`. Compress your submission directory using the command: `tar -cvzf submission.tgz submission` and upload `submission.tgz` to Moodle.

Part 1A: Syllable-based ASR (10 points)

a) Syllables are sub-word units consisting of multiple phones. In this problem, a syllable is defined as a sequence of phones that appear in the order VC, CV or CVC (where V stands for a vowel and C stands for a consonant). We shall assume that our lexicon consists only of words that have at least one way of being written as a sequence of such syllables (e.g., the lexicon will have no words with 3 consecutive consonants). E.g., the words "up", "shoe" and "tall" are composed of single syllables of the form VC, CV and CVC, respectively, and the word "codas" could be broken up either as "cod-as" (CVC, VC) or "co-das" (CV, CVC). Further, we allow a syllable to straddle word boundaries: e.g., the sequence "map it" could be broken up either as "map-it" (CVC, VC) or "ma-pit" (CV, CVC).

Recall that a WFST-based ASR decoder uses the composed machine, $H \circ C \circ L \circ G$. We want to modify the components H, C, L and G in order to work with syllables instead of individual phones. The HMMs that constitute H now correspond to syllables rather than phones (i.e., the output alphabet of H is the set of all syllables). Suppose we do not wish to change the pronunciation lexicon L, so that it remains phone-based (i.e. each word is represented as a sequence of phones).

Then, how should C be defined? Draw a state diagram of C assuming there are only two phones c and v, a consonant and a vowel, and three syllables $\alpha = vc$, $\beta = cv$ and $\gamma = cvc$. **[3 pts]**

b) We define a sequence of "contextualized syllables" to be a sequence of triplets of the form (P_1, S, P_2), where P_1 and P_2 are two phones preceding and succeeding a syllable S (either of them could be empty, if S is the first or last syllable in an utterance). For instance, the syllable sequence "re-map-it" would correspond to the contextualized syllable sequence "(ϵ, re, m), (e, map, i), (p, it, ϵ)."

Suppose that we are given an H that outputs contextualized syllables, and L, G remain as before. Then, how should C be defined? Draw a state diagram of C assuming the same phone and syllable alphabet as in part a). **[7 pts]**

What to submit: Submit a pdf `part1A/answers.pdf` that contains the FSTs drawn for both parts of this question.

Part 1B: Handling disfluencies in ASR (10 points)

Recall that a pronunciation lexicon L maps a sequence of phones to a sequence of words. In this problem, we shall modify L in order to handle some limited forms of interruptions in speech (a.k.a. disfluencies). We will consider a dictionary of two words: W_1 with the phone sequence "a b c" and W_2 with the phone sequence "x y z".

a) Draw the state diagram of the finite-state machine L. **[2 pts]**

b) We want to modify L such that it accounts for "breaks" when the speaker stops in the middle of a word and says the word all over again. For instance, the word W_1 may be pronounced as "a b ⟨`break`⟩ a b c," where ⟨`break`⟩ is a special token produced by the acoustic model. In a *valid* phone sequence, breaks are allowed to appear only within a word, and not at the end or beginning of a word. Further, two consecutive ⟨`break`⟩ tokens are not allowed. But a word can be pronounced with an arbitrary number of breaks. E.g. W_1 can be pronounced also as "a b ⟨`break`⟩ a ⟨`break`⟩ a b ⟨`break`⟩ a b c". Let L_1 be an FST (obtained by modifying L from the previous part) that accepts all valid phone sequences with breaks, and outputs a corresponding sequence of words. Draw the state diagram of L_1. **[3 pts]**

c) Next, we want to modify L_1 such that it can account for both "breaks" and "pauses." A pause corresponds to when the speaker briefly stops in the middle of a word and continues. For instance, the word W_1 may be pronounced as "a b ⟨`pause`⟩ c", "a ⟨`break`⟩ a ⟨`pause`⟩ b ⟨`break`⟩ a b c," etc. where ⟨`pause`⟩ is another special token produced by the acoustic model. In a valid phone sequence, these special tokens are allowed to appear only within a word, and two consecutive special tokens are not allowed. Let L_2 be an FST (obtained by modifying L_1 from the previous part) that accepts all valid phone sequences with breaks and pauses, and outputs a corresponding sequence of words. Draw the state diagram of L_2. **[5 pts]**

What to submit: Submit a pdf `part1B/answers.pdf` that contains the FSTs drawn for all three parts of this question.

Part 2: Speech Command Recognition (22 points)

Speech Commands is an audio dataset consisting of isolated-word commands spoken by different speakers. (All the audio files are about 1 sec long.)

Task 0: Set up a baseline system **[2 points]**

Follow the PyTorch tutorial on audio command recognition that takes you through the entire recognition pipeline: Format the dataset, create train/test splits, build a convolutional neural network to train the model and finally evaluate it on test examples.

This tutorial can be run on Google's Colab. These runs will significantly benefit from access to GPUs. The second paragraph in the tutorial describes how to enable GPU access in your Colab account.

In the first cell under "Importing the Dataset", you will find two lists named `train_set` and `test_set`. Confirm that the length of `train_set` and `test_set` are 84843 and 11005, respectively. Next, change the code so as to select elements from these lists based on the indices in the following two files to construct modified training and test data sets, respectively: https://www.cse.iitb.ac.in/~pjyothi/cs753/train_list.txt and https://www.cse.iitb.ac.in/~pjyothi/cs753/test_list.txt. Make sure that the resulting `train_set` and `test_set` lengths are 12000 and 4000, respectively. (The audio from the first training file will be someone saying "backward" and the audio from the last training file will be someone saying "zero".) Set the number of training epochs `n_epoch` to 20. The entire training process should take less than a minute and the final test accuracies will be in the range $66\% - 71\%$.

What to submit: Submit a text file `part2/task0/accuracy.txt` that contains the test accuracy on `test_set`. Submit a file `part2/task0/notebook.txt` with a link to your Colab notebook that we can run end-to-end to reproduce your results. `part2/task0/accuracy.txt` and `part2/task0/notebook.txt` should each contain only a single line with the test accuracy (up to two decimal points, no $\%$ symbol) and a link to your notebook, respectively. All subsequent tasks that require such `.txt` files should follow the same structure. The very last line in your notebook should print the final test accuracy.

Task 1: Larger model **[4 points]**

Create a new class for and train the M11 network listed in this reference. What is the new accuracy with this much larger model?

What to submit: Submit a file `part2/task1/notebook.txt` with a link to your Colab notebook that contains code for the new class for M11 and to train/test it. Submit a text file `part2/task1/accuracy.txt` that contains the new test accuracy on `test_set`.

Task 2: Confusion Matrix **[4 points]**

Write a new function `find_confusion_matrix(model)` that takes a trained model as its input and computes the distribution across all training classes predicted for the samples in `test_set` using the trained model. You can choose to print out the confusion matrix any way you prefer.

What to submit: Submit a file `part2/task2/notebook.txt` with a link to your Colab notebook that contains the new function `find_confusion_matrix(model)`.

Task 3: LSTM-based ASR **[10 points]**

Question 1

Instead of the convolutional M5 network you used in Task 0 that directly processes raw audio data, design an LSTM-based recurrent neural network that unrolls across the length of the input audio and uses **Mel Frequency Cepstral Coefficients** (MFCC) features as inputs at each timeframe. Here, as in Task 0, you have a categorical classification problem and the loss will be a negative log-likelihood loss.

1. You should use the same `train_set` and `test_set` from Task 0.

2. You cannot use more than three layers in your LSTM network and each layer cannot have more than 400 hidden units. Use the Adam optimizer with a learning rate and a learning rate schedule of your own choosing.

3. You may refer to the following notebook to set up an LSTM network for speech command recognition. As mentioned in this notebook, use 12 MFCC features for your input.

Play around with the network architecture and various hyperparameters to derive the best possible performance on `test_set`.

What to submit: Submit a text file `part2/task3/accuracy.txt` with the best test accuracy using your tuned hyperparameters and design choices. Submit a link to your notebook in `part2/task3/notebook.txt` that we can run on Colab to reproduce your results. As with Task 0, this should run end-to-end to produce the final test results. (All the hyperparameters should be set to the best values.) The very last line in your notebook should print the final test accuracy.

Task 4: Performance on a blind test set **[2 points]**

Question 2

How well do your systems perform on unseen utterances? Within `part2/task4/choice.txt`, choose one of the following labels for the type of trained system you would like to use to evaluate on unseen utterances: {M5,M11,LSTM}. Depending on your choice, we will use your corresponding notebook to evaluate a blind test set consisting of completely new utterances. A leaderboard with the top-scoring N groups and the corresponding scores will be posted on Moodle. You will receive full points for this question if your blind test accuracy is higher than what we get with the baseline recipe in Task 0. The N top-scoring performers on the leaderboard will gain extra credit points. (N will be chosen later after examining all the blind test accuracies.)