Natural Language Processing Unit 2 Evaluation

Date: 17th Sep 2014

Time: 1:30 pm to 6:00 pm IST

Problem # 1: Given audio files of a TV news program in an encoded format, analyze the quality of the program and identify the participants.

(Read the disclaimer in the last section before you begin)

Motivation:

One of the most watched programs of TV news channels such as Times Now, NDTV etc. are the debates. Typically, a given debate involves several participants along with an anchor. A debate may be orderly where the participants and the anchor discuss the issues objectively, allowing one another the necessary time to present their views. Some other debates might be less streamlined and may see several participants speaking at the same time. The style of the anchor and his own biases, viewpoints are also components of a debate.

Our goal is to solve 2 problems that constitute Part 1 and Part 2 of this assignment.

Using the HMM and the other techniques learnt in our NLP course:

1. Compute a quality index for a given audio clip of a debate
2. Identify the speakers in the given audio clip

Problem Statement

You are provided with:

a. Training Datasets: These are text files that are named with a pattern: *_trg_vq.txt. These files represent an audio clip. Each line in a given training file has an integer: 0 <= n <= 7.
b. Test Datasets: These are text files that are named with a pattern: *_test_vq.txt. These files represent an audio clip. Each line in a given test file has an integer: 0 <= n <= 7.
c. Audio clips that have .WAV extension
d. A JSON file named debate_initial.json that has the initial model for the problem we are solving
e. HMM module: myhmm_log.py that implements the HMM algorithms using log scaling

Our goal for the Part 1 of this assignment is to evaluate each observation vector of the audio data (which is sequential in time) in to one of 3 states as below:

(a) Only one person is speaking at a time (State = “single”)
(b) More than one person is speaking at the same time (State = “multi”)
(c) No one is speaking. That is there is silence in that time window. (state = “silent”)

That is, given the observations O = [O_1, $O_{2…,}$ O_k], evaluate the probabilities of every O_i and determine the state for that observation. Note that each observation as above by itself is a sequence, where O_i = [o1, o2…on] where o’s are individual observations of the sequence O_i.

In our training or test files, individual observations o are provided one per line. Using a sequence length, say 10, you need to generate the training (or testing) sequences. For example, if a file has 1000 lines where each line is an individual observation, using a sequence length of 10 you will generate 1000/10 = 100 training vectors, where each of this vector has 10 elements. The goal is to evaluate each of this 100 in to one of the 3 states: (“single”, “multi”, “silent”)

You are provided with an initial model in a JSON file: debate_initial.json. Using this and the training data, you can train an HMM by invoking the forward_backward algorithm. Once you have trained the system, you can predict the output for the test data.

You are required to do the following:

1. Create 3 HMM instances using same initial model file. You can do this by instantiating the class MyHmmLog(initial_model_file_name). Let us call these 3 models as M1, M2, M3
2. The training files that correspond to the 3 states are: single_trg_vq.txt, multi_trg_vq.txt and silent_trg_vq.txt. Train each model M_i with its corresponding training file.
    a. Read the training file in to a variable, remove any trailing white spaces like newline characters
    b. Generate the sequences where each sequence is 10 observations. If there are 1000 lines in your file, you will get 100 sequences, where each sequence is going to be classified in to one of 3 states by our implementation
    c. Train the model M_i with these vectors. You need to invoke: model.forward_backward_multi(List_of_sequences). Note that this is a different implementation of our earlier forward_backward algorithm that took a sequence of observations as input. This implementation takes a “list” of observation sequences as input as against a single observation sequence.
    d. Note that there is only one training file per model, each model should be trained separately with its own training file.

3. After successful execution of steps above, you will now have 3 HMMs each capable of detecting a given output. Now, read each test file, generate sequences as described above, classify each sequence in to one of the 3 states using the trained HMMs. The output for a given sequence is the argmax of m over the models that maximizes the probability of that sequence given the model. To find the probability of a given observation sequence you may use: forward(observation_sequence) as you did in the earlier lab assignment.
4. At this point, on successful completion of prior steps above, you will have for each test file, the sequence vectors and their corresponding state label. For example the output for a given test file may look like: (silent, silent, single, single, silent, multi, single…). Using this output do the following:
    a. The output sequence is ordered in time that correspond to 20 vectors per second where is vector is a sequence of length 10 in our assignment. Using this timeline information, associate each state in the output with a time coordinate. The first element of the output is for the audio at time = 0, the second element of the output corresponds to time = 50 milliseconds and so on. Store these in a variable and write to an output file
    b. For a given audio input file, compute the total number of occurrences of each state – for example {"silent": 10, "single": 326, "multi": 700}
    c. Compute a quality index for the audio clip using the formula: $q = w_1x_1 + w_2x_2 + w_3x_3$ where we can take w's for silent and single to be 10 and that for multi to be -10. Normalize the output by dividing it with the number of vectors in the file.
5. With the outputs produced as above listen to the corresponding audio clip and make your observations on the accuracy of prediction. You can do this for a minimum of 3 test files. Your observations/comments is a deliverable in a file: part1_comments.txt

Part 2:

Repeat the above using the training files: music_trg_vq.txt, modi1_trg_vq.txt, arnab1_trg_vq.txt. The states here are: ("modi", "arnab", "other"). You can leave out the quality index computation, just output the state sequence with time.

Deliverables

1. Source code of your program
2. The outputs produced in part 1 and 2
3. A brief write up on how effective HMM was, where it did well and where the results are not good and what can be done to improve them in files part1_comments.txt and part2_comments.txt
4. Any readme file or documentation that describes your work (optional)

Your submissions should be made to Shruti's mail id ☺ Have fun ☺

Disclaimer:

1. The audio clips for this exercise are obtained from publicly available content and these are used here solely with the purpose of academic purposes. These are not used for any commercial activities.
2. Some or part of audio may contain strong opinions, political messages and reference to people, parties or religions. The purpose of this assignment is solely academic and to learn to process the real life data. The contents of these audio are in no way endorsed or alluded to by the faculty or college or a reflection of their own personal viewpoints.