Word Boundary Prediction

Assignment 3 – ELL888

IIT Delhi

Anish Mahishi – 2014MT10585
Adarsh Onkar – 2014MT10582
Ankit – 2014TT10866
Date – 10/05/2018

Problem Statement

Given a speech audio file in .wav for format, and the list of words spoken in it, predict the word boundaries of corresponding words spoken in the audio.

Training Data

Training data consists of approximately 13600 samples where each sample consists of

1. A wave (.wav) file.
2. A .word file containing the set of words spoken along with their boundaries.

Architecture

Pre-processing

1. **Reading the .wav file:**
   For reading each audio file, we use *SoundFile* is an audio library based on *libsndfile*, *CFFI* and *NumPy*
   Input: A .wav file.
   Output: A 1-D array of numbers representing the audio file.
2. **Representing words:**
   The English words are represented using GloVe vectors. We use the GloVe embedded representation of the words as inputs to our model. For all words which are not in the vocabulary, we randomly sample from the GloVe vectors to get a representation of the word.
3. **Scaling Word Boundaries:**
   The word boundaries in the training data are in the order of 10^3. To reduce the mean-squared error while training, we downscale the word boundary values by 1000 (i.e. we divide the values by 1000).
4. **Training**
   Since the input sequence is of variable length, we had to train the model with a batch size of one, this took a lot of time during training, and thus we divided the training dataset into several parts, with each part containing the same number of words in a sentence. For example, all the data having six words in the sentence were trained together, and so on. This increased the speed of training.

Model

We have tried 3 different approaches using Encoder-Decoder (1st), Attention (2nd). The 3rd approach is actually a tweak over the 2nd model. In addition to the standard Encoder-Decoder model, we have also attached a regressor to the outputs of the decoder. The output of the regressor at the tth time step is the start time or/and the end time of the word given as input to the decoder at the tth time step.

- **Input to Encoder:** The input file array which we get as output of after passing the .wav file to *SoundFile* API.
- **Input to Decoder:** The input to tth time step is a GloVe vector representing the tth word.
- **Output of the Decoder:** The output of the decoder is a 100x50 vector.
- **Input to the Regressor:** At each time step**,** the output of the decoder is given as input to the regressor.
- **Output of the Regressor:** We have tried two different models for this purpose:
  1. The first model outputs both the start time and end time of the tth word.
  2. In the second model, start time and time of the tth word are predicted separately using 2 different models.

  The second model is in general found to work much better than the first in prediction, so we proceed to work with the second model only.

Approach 1: Encoder Decoder without Attention

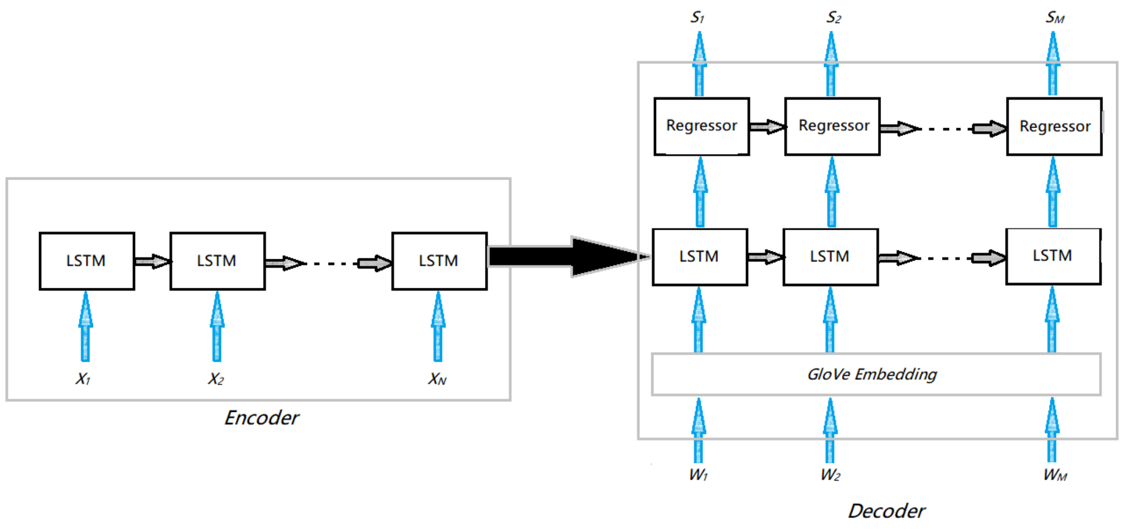

Figure 1: Basic Encoder Decoder model used for Approach 1.

- **N =** Length of the input vector (which represents the speech .wav file)
- **X_i =** ith value in the N-dimensional input vector.
- **M =** No. of words spoken in the speech file.
- **S_i =** Starting time of the ith word.

For the model, used for predicting the end of the words, we just replace $\mathbf{S_i}$ with $\mathbf{E_i}$**,** where,

- $\mathbf{E_i}$ **=** Ending time of the ith word.

- **Metrics:**
  Correct Detection Rate = 16%
  False Detection Rate = 2%
  Missed Detection Rate = 82%

- **Possible reason for poor performance:** Sometimes, the boundary points predicted tend to be a little off from the correct boundary. This is classified as missed detection. Hence, the poor performance.

Approach 2: Encoder Decoder with Attention Mechanism

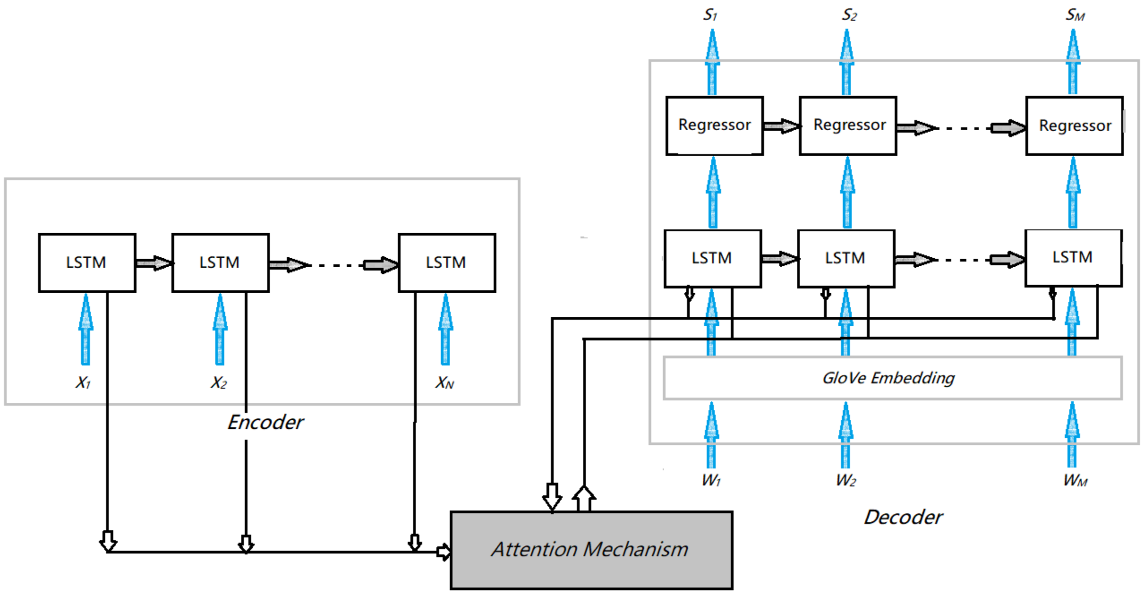

Figure 2: Encoder-Decoder with Attention mechanism used for Approach 2.

- **Attention Mechanism:** The scoring function is dot product of the hidden state of the decoder and that of the encoder outputs. Then we do a softmax over all the scoring functions over each element of the input sequence and then we pass the weighted sum into the decoder. Although this approach did not increase the correct detection rate.

- **Metrics:**
  Correct Detection Rate = 17%
  False Detection Rate = 2%
  Missed Detection Rate = 81%

- **Possible reason for poor performance:** Sometimes, the boundary points predicted tend to be a little off from the correct boundary. This is classified as missed detection. Hence, the poor performance. To tackle this we modified the true word boundaries, which you'd be seeing in Approach 3.

Approach 3: A tweak over approach 2

In this approach we added noise to the output word boundaries while training, for example, if the actual word boundary is [4000,6000] then we trained for [4500,5500]. This increased the performance (correct detection rate by around 12%).

- **Metrics:**

We computed the variance between the actual boundary and the detected boundary. We've computed the variance in each sample and then we averaged the computed variances over each sample in the entire training set. Also, the variance of the training data over each sample.

Correct Detection Rate = 28%
False Detection Rate = 6%
Missed Detection Rate = 66%
Variance over variances = 6.599
Mean over variances = 2.46

Other Approaches

- We could use a simple 1 – 0 over audio samples without using any word information.
- Another approach is that we could use Connectionist Temporal Classification (CTC) over a simple Encoder-Decoder Model.