CSE-667 Advance Computational Linguistics Mega Lab

Harshit Monish
University at Buffalo
hmonish@buffalo.edu

Abstract

In this project, we assess the importance of each of the different structural and linguistic features on our ability to predict whole-word phonetic duration. We have implemented various models like linear, multi-layer neural network, Continuous Bag of Words, Encoder-Decoder, and Encoder-Decoder with attention for this task using the buckeye dataset.

1 Introduction

We can make the distributional assumption for sound in a word just like we can for words in a sentence and sounds are partly determined by the contexts in which they occur. The smallest units of sound is Phonemes which help in discrimination between words and sounds that are similar and word and sounds that are different. Sounds regularly co-occur with other sounds depending on positions of phonetic representations in articulation of words. Hence the sequence of sounds make words and sequence of words make sentence though there are some differences in the way these units are correlated for example vowel sequencing, nasal and non-nasal vowels,etc and nouns, verbs, and other grammar rules but they do have the similarity of context in them. The word embedding are generated using GLOVe embedding vectors, these vectors are associated with every single word in the whole buckeye corpus. These word embedding is going to encode information like morphology, some aspects of the semantics of the word. Ideally, noun, verb, function words and content words will have systematically different representations from each other. So the word embedding representation may provide some additional signal beyond some representations of the sequence of sounds in a word.
In this project we aim to predict the duration of the pronunciation of a word given the phonemes of the word by implementing various models. Further we compare the results of each models we have implemented.

2 Implementation

Fetching the Data: First we fetch the train and test data from the buckeye corpus dataset. The attributes that are fetched from all the attributes are observed pronunciation of the word (dependent variable) and the duration of the pronunciation of the word (independent variable)

Model Implementations:

1. **Linear Model:** We have implemented a linear model that takes input as number of phonemes for each word and predict the duration of pronunciation. Used Mean Squared Error Loss since the output is real values and used Stochastic Gradient Descent optimizer to minimize the loss function. Next we concatenate the glove word embedding with the number of phones to predict the duration of the pronunciation of the word and reported the results in both the cases.

2. **CBOW Model:** CBOW algorithm is used to predict word pronunciation duration from the context using the concept of window size, i.e given context of phones in window size what will be the target duration. During training we generate the word embedding which is the weight matrix of the model using Vocabulary. We used the pre-trained embedding and vocabulary from previous lab in this model. First we load the vocab object and embedding, next we evaluate the maximum length of phonemes from the entire train data. Then we concatenate the embedding of the phonemes sequentially for each word and add padding according to the max length evaluated. We use Keras pad_sequences API for this which also truncates the vector padding to specified max

length if the sequence length during test time is greater than max length in the train data. We then learn a linear model and a multi layer Neural Network model and have reported the results for each of them.

3. **Encoder-Decoder Model:** In this model we use RNN GRU Encoder, Decoder architecture in which we first add an embedding layer in the encoder followed by a GRU layer, the decoder also have an embedding layer, followed by a GRU layer and then a linear layer with softmax activation function. We first learn the encoder embedding using SGD optimizer for both encoder and decoder and back propagate the loss from decoder to encoder. Next we add a linear model and keeping the encoder weights same we learn the liner model that takes encoder embedded output as input and predicts the duration of the word pronunciation using MSE loss and SGD optimizer.Next we use the CBOW learned embedding in the embedding layer of the encoder2 with encoder1 in the same architecture and report the results. We then further use the Glove word embedding with the encoder1 and CBOW learned embeddings in encoder 2 and concatenate it with encoder3 embedding layer and report the results.

4. **Encoder-Decoder with Attention Model:** This model is same as Encoder-Decoder model except that the decoder uses the attention mechanism. Keeping the experiments same as in the above model with encoder, we use the decoder with attention in this model. In attention decoder we first have an embedding layer followed by a dropout layer, we then have an attention layer as linear layer and attention weights are computed as concatenating the embedding layer output and hidden layer of encoder (which is fed into decoder) and then using soft-max activation function. Then we apply batch matrix multiplication of attention weights and encoder outputs. We then concatenate the embedding layer of decoder with matrix multiplication output and feed this into a linear layer. Then we apply relu activation function on this output and feed it further into a GRU layer. The output of the GRU layer is then fed into a log soft-max function. Again We first learn the encoder embedding using SGD optimizer for both encoder and decoder and back propagate the loss from decoder to encoder. Next we add a linear model and keeping the encoder weights same we learn the liner model that takes encoder embedded output as input and predicts the duration of the word pronunciation using MSE loss and SGD optimizer.Next we use the CBOW learned embedding in the embedding layer the encoder2 with encoder1 in the same architecture and report the results. We then further use the Glove word embedding with the encoder1 and CBOW learned embeddings in encoder 2 and concatenate it with encoder3 embedding layer and report the results..

3 Experimental Results

Dataset : The dataset that was used for this project was buckeye dataset. it contains the attributes like word, dictionary pronunciation, observed pronunciation, segment timestamp, segment duration etc. out of which we focused on observed pronunciation and segment duration attribute. The train and test data was split using 80-20 splits.

Results : The results of all the models are mentioned in Table-1. The implementation results are saved in megalab Ipython Notebook (megalab.ipynb)

1. **Linear Model:** For Linear model the configuration was, learning_rate was 0.001, Number of epochs = 20000

2. **CBOW Model:** For CBOW model with linear layer the configuration was, learning_rate = 0.001, the number of epochs were 4 as we loop over whole train dataset containing 4414 batches and the main loop iterated 4 times. This is the reason why loss is lowest for this model. The input was max training len (56) * embedding dimension(8) and output was single node. For multilayer architecture the input and output layers were same and added a hidden layer with 64 nodes.

3. **Encoder-Decoder Model:** The configuration for this model was, learning_rate = 0.001, number of epochs - 300, embedding dim = 8, hidden layer nodes = 256. The GRU layer had dimension of hidden units. Encoder-Decoder with CBOW embedding configuration was, learning_rate = 0.01, momentum = 0.0001,

number of epochs = 300, hidden size = 8 since CBOW embedding dimension was 8, GRU layer had dimension of hidden units. Encoder-Decoder with CBOW embedding and GLOVe embedding configuration was, learning_rate = 0.01, number of epochs = 500, momentum = 0.0001, hidden units dimension = 8, GRU layer had 2 * dimension of hidden units as glove embeddings were concatenated.

4. **Encoder-Decoder with Attention Model:** The configuration for this model was, learning_rate = 0.001, number of epochs - 300, embedding dim = 8, hidden layer nodes = 256 and decoder has attention implementation. The GRU layer had dimension of hidden units. Encoder-Decoder with CBOW embedding configuration was, learning_rate = 0.01, momentum = 0.0001, number of epochs = 300, hidden size = 8 since CBOW embedding dimension was 8, and decoder has attention implementation, GRU layer had dimension of hidden units. Encoder-Decoder with CBOW embedding and GLOVe embedding configuration was, learning_rate = 0.01, number of epochs = 500, momentum = 0.0001, hidden units dimension = 8, and decoder has attention implementation, GRU layer had 2 * dimension of hidden units as glove embeddings were concatenated.

Model	**Train Loss**	**Test Loss**
1. Linear	0.54	0.76
2. Linear with Glove embedding	0.32	0.29
3. CBOW concatenated linear	0.56	0.186
4. CBOW concatenated with Multi-layer NN	0.54	0.11
5. Encoder-Decoder	0.51	0.68
6. Encoder-Decoder with Hidden and CBOW embeddding	1.29	1.42
7. Encoder-Decoder with CBOW and Glove embeddding	0.33	0.41
8. Encoder-Decoder with Attention	0.58	0.84
9. Encoder-Decoder with Attention with CBOW embeddding	0.61	0.96
10. Encoder-Decoder with Attention with CBOW, GLOVe embeddding	0.41	0.61

Table 1: Results of all the models.