COMP541/441: Deep Learning Assignment - 3 Report

Gamze Keçibaş[*]
Department of Mechanical Engineering
Koc University
Istanbul ,34450
gkecibas16@ku.edu.tr

Abstract

In the assignment, training process of recurrent neural networks (RNNs) for different tasks is studied in the task. There are two main steps in the assignment as RNN for classifcation and implementing a language model considering instructions. The assignment is developed using Apple M2. Epoch time in results represents the model training without data preprocessing. Total time shows taking time from raw input to get prediction.

1 RNNs for Classification

In the part, binary classification of the sequences have constant length to predict whether the given sequence is followed by a consonant or a vowel. There is a *RNNModel* class to construct the classification model structure in 'models.py'. The *RNNModel* class also has a *forward method*, which defines the forward pass of the network. It takes in an input tensor input and returns an output tensor. Here's how the *forward* pass works:

1. The input tensor is passed through the 'embedding layer' to get the embeddings for each word.

2. The embeddings are reshaped to add a singleton dimension at the second position.

3. A tuple of two zero tensors is created and used as the initial hidden state for the LSTM layer.

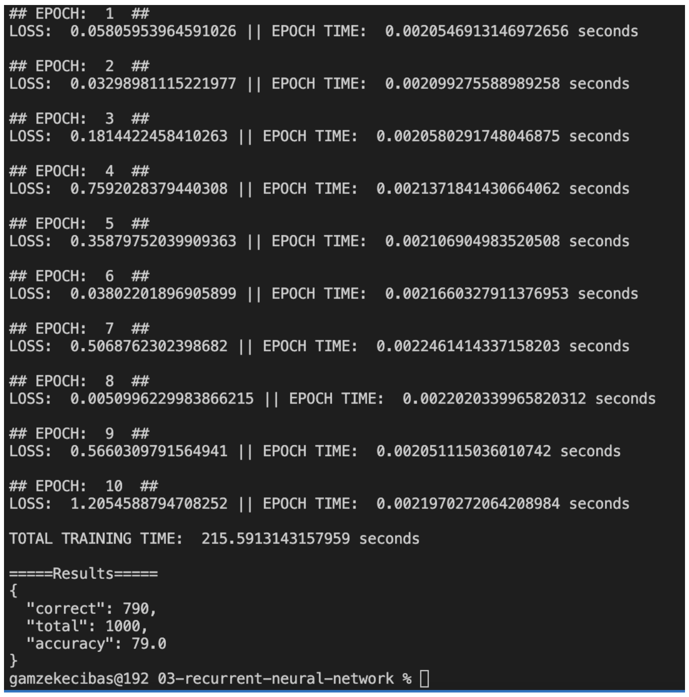
4. The LSTM layer is applied to the input embeddings and the initial hidden state, and it returns the final hidden state and the output.

5. The final hidden state is passed through the 'linear layer' to get the final output.

6. The output tensor is squeezed at the second position (i.e., the singleton dimension is removed) and returned.

When the model is trained, the accuracy rate is equal to 74%-79% after 10 epochs and 215.6 seconds. The accuracy rate may change because of shuffling. Each epochs take around 0.002 seconds. All results are presented below:

[*]Use footnote for providing further information about author (webpage, alternative address)—*not* for acknowledging funding agencies.



```
## EPOCH:  1  ##
LOSS:  0.05805953964591026 || EPOCH TIME:  0.0020546913146972656 seconds

## EPOCH:  2  ##
LOSS:  0.03298981115221977 || EPOCH TIME:  0.002099275588989258 seconds

## EPOCH:  3  ##
LOSS:  0.1814422458410263 || EPOCH TIME:  0.0020580291748046875 seconds

## EPOCH:  4  ##
LOSS:  0.7592028379440308 || EPOCH TIME:  0.0021371841430664062 seconds

## EPOCH:  5  ##
LOSS:  0.35879752039909363 || EPOCH TIME:  0.002106904983520508 seconds

## EPOCH:  6  ##
LOSS:  0.03802201896905899 || EPOCH TIME:  0.0021660327911376953 seconds

## EPOCH:  7  ##
LOSS:  0.5068762302398682 || EPOCH TIME:  0.0022461414337158203 seconds

## EPOCH:  8  ##
LOSS:  0.0050996229983866215 || EPOCH TIME:  0.0022020339965820312 seconds

## EPOCH:  9  ##
LOSS:  0.5660309791564941 || EPOCH TIME:  0.002051115036010742 seconds

## EPOCH:  10  ##
LOSS:  1.2054588794708252 || EPOCH TIME:  0.0021970272064208984 seconds

TOTAL TRAINING TIME:  215.5913143157959 seconds

=====Results=====
{
  "correct": 790,
  "total": 1000,
  "accuracy": 79.0
}
gamzekecibas@192 03-recurrent-neural-network % 
```

RNNClassifier also has a *predict method*, which takes in a string context and returns a prediction (0 or 1). Here's how the predict method works:

1. The input string is converted into a list of integers, with each integer corresponding to the index of a character in the vocabulary.

2. The list of integers is converted into a PyTorch tensor.

3. The tensor is passed through the model object to get the output.

4. The output is thresholded at 0 to get a prediction (0 if the output is negative, 1 if the output is positive or 0). The prediction is returned.

2 Implementing a Language Model

The class *LMRNNModel* is a PyTorch module that defines a language model based on a long short-term memory (LSTM) recurrent neural network (RNN). It takes a dictionary size (the size of the vocabulary) as input and defines the following layers:

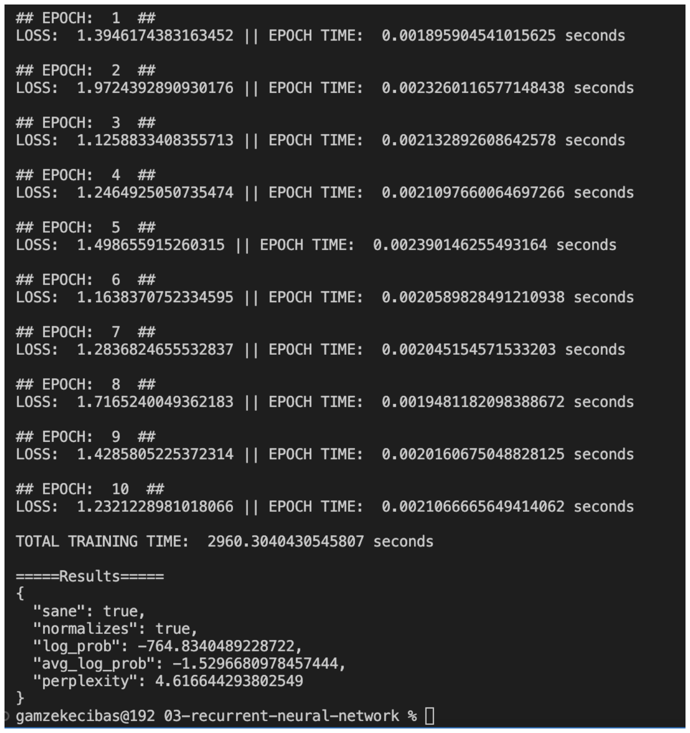
1. An embedding layer ('embed_dim'): This layer converts the input indices (integers representing words in the vocabulary) into dense vectors of a fixed size.

2. An LSTM layer: This layer takes the embedded inputs and produces a hidden state for each time step. The LSTM layer has 'hidden_num' hidden units.

3. A linear layer: This layer takes the hidden state produced by the LSTM layer and produces an output of size 'dict_size', which represents the logits for the next word in the sequence.

The forward method of the *LMRNNModel* class defines how the input is processed through these layers. It first converts the input indices into dense embeddings using the embedding layer, then passes these embeddings through the LSTM layer to produce the hidden state. Finally, it applies the linear layer to the hidden state to produce the logits.

The 'RNNLanguageModel' class is a subclass of 'LanguageModel' that wraps an *LMRNNModel* object and provides additional methods for generating predictions based on the model. The 'get_next_char_log_probs' method takes a context (a string of characters) as input and returns the log probabilities of the next character in the sequence given the context. The 'get_log_prob_sequence' method takes a sequence of characters and their context as input and returns the log probability of the entire sequence given the context.



```
## EPOCH:  1  ##
LOSS:  1.3946174383163452 || EPOCH TIME:  0.001895904541015625 seconds

## EPOCH:  2  ##
LOSS:  1.9724392890930176 || EPOCH TIME:  0.0023260116577148438 seconds

## EPOCH:  3  ##
LOSS:  1.1258833408355713 || EPOCH TIME:  0.002132892608642578 seconds

## EPOCH:  4  ##
LOSS:  1.2464925050735474 || EPOCH TIME:  0.0021097660064697266 seconds

## EPOCH:  5  ##
LOSS:  1.498655915260315 || EPOCH TIME:  0.002390146255493164 seconds

## EPOCH:  6  ##
LOSS:  1.1638370752334595 || EPOCH TIME:  0.0020589828491210938 seconds

## EPOCH:  7  ##
LOSS:  1.2836824655532837 || EPOCH TIME:  0.002045154571533203 seconds

## EPOCH:  8  ##
LOSS:  1.7165240049362183 || EPOCH TIME:  0.0019481182098388672 seconds

## EPOCH:  9  ##
LOSS:  1.4285805225372314 || EPOCH TIME:  0.0020160675048828125 seconds

## EPOCH:  10  ##
LOSS:  1.2321228981018066 || EPOCH TIME:  0.0021066665649414062 seconds

TOTAL TRAINING TIME:  2960.3040430545807 seconds

=====Results=====
{
  "sane": true,
  "normalizes": true,
  "log_prob": -764.8340489228722,
  "avg_log_prob": -1.5296680978457444,
  "perplexity": 4.616644293802549
}
gamzekecibas@192 03-recurrent-neural-network % 
```

When the language model is trained during 10 epochs, perplexity value is around 4 end of the 2960 seconds.