A REPROT FOR DEEP LEARNING LAB EXERCISE 7

Ziyi Guo, MSc Data Science

zg2u21@sonton.ac.uk

1 SEQUENCE-TO-SEQUENCE MODELLING

In the first part of the experiment, a basic sequence-to-sequence model is implemented as tested for a translation task. The forward pass method in the Encoder class for passing the input through the embedding to LSTM and return the hidden and the cell state is as below.

```python
class Encoder(nn.Module):
    def __init__(self, input_dim, emb_dim, hid_dim):
        super().__init__()
        self.hid_dim = hid_dim
        self.embedding = nn.Embedding(input_dim, emb_dim)
        self.rnn = nn.LSTM(emb_dim, hid_dim)
    def forward(self, src):
        embedded = self.embedding(src)
        _, (hidden, cell) = self.rnn(embedded)
        return (hidden, cell)
```

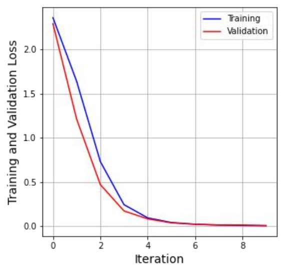

Fig 1. Training and Validation Loss of Sequence Model

It can be observed that with the decrease of the translation loss in training, the model has an approximately decreasing validation loss, indicating that the proposed forward method in the sequence-to-sequence model is working well.

Based on the trained model, experiments are carried out to translate sequences with the decoder function as below.

```python
def decode(code: str):
    out = ' '
    for chunk in code.split(' '):
        num = ds.encode_morse('^ ' + chunk + ' $').unsqueeze(1)
        pred = model(num.cuda(), maxlen=2)
        pred = pred[1:].view(-1, pred_dim).argmax(-1)
        out += ds.decode_alpha(pred.cpu())[::-1]
    return out
```

With input sequences of " .- -. ... .-- . .-. / - .... . / ..-. --- .-.. .-.. --- .-- .. -. --. ", " • .-- .... -.-- / .. ... / - .... . / --- .-. -.. . .-. / --- ..-. / - .... . / --- ..- - .--. ..- - / .-. . ...- . .-. ... . -.. ", " • .-- .... .- - / .. ... / - .... . / .--. --- .. -. - / --- ..-. / - . .- -.-. .... . .-. / ..-. --- .-. -.-. .. -. --. ", the sequence-to-sequence model outputs the sentences " ***answer the following*** ", " • ***why is the order of the output reversed*** "," • ***what is the point of teacher forcing*** ". Reversed output is due to the reversed reading of the input sentences, which illustrates short term dependencies in LSTM and makes optimisation easier. Teacher forcing is to directly use the previous term corresponding to the ground truth of training data as the input of the following state instead of the output of the previous state, which could help with convergence speed and model performance.

Further exploring the influence of sequence lengths, experiments are carried out on the sequences in various lengths based on the corresponding International Morse code of English letters. The code and the results are as below.

```python
print("Target:  a", " ","Output:",decode(".-"))
print("Target:  b", " ","Output:",decode("-..."))
print("Target:  c", " ","Output:",decode("-.-."))
print("Target:  h", " ","Output:",decode("...."))
print("Target:  q", " ","Output:",decode("--.-"))
print("Target:  y", " ","Output:",decode("-.--"))
print("Target:  z", " ","Output:",decode("--.."))
```

Target	a	b	c	h	**q**	y	z
Input	.-	-...	-.-.	....	--.-	-.--	--..
Model Output	a	b	c	h	**y**	y	z

Table 1. Translations of Morse Code

It can be observed that the predictions of the model is likely to be influenced by sequence length in a basic letter degree as it outputs the incorrect translation of the letter ***q***, but the phenomenon is not significant. However, when tuning the sequences of the sentence " ***answer the following*** " and deleting the blank between the two short chunks only, the output of the model becomes " ***pswer the following*** "; when deleting all the blank between the chunks and making it a long chunk, the output becomes " ***f*** ". Obviously, the model outputs unexpected translations due to the long sequence lengths. It can be inferred that the model hardly learned long sequences in the training data and performs poorly in such translations.