ADL HW1

R09521603

March 21, 2022

1 Q1: Data Preprocessing

1.1

a

$$\text{WordSet} = \text{SortByFrequency}(\text{Words})$$

The Words is the set of all the words in the data, we sort the it by the appearance frequency.

$$\text{word\_id} = \text{WordSet}[\text{word}]$$

We receive the id according to their position in sorting list

b

We use the pretrained GloVe embedding.

1.2

For the preprocess step, I use the sample code. In both intent classification and slot tagging, we embed the data words using the same process. First we collect all the words and use Counter object to count the frequency of each word. Then we only convert the most common **vocab_size** words into embedding vector. To convert the Vocab contains two steps, First assign each word in most common word a unique id (as for padding and unknown we use 0 and 1 respectively), Second we find each word's embedding vector using pretrained GloVe. As it is impossible to convert all the word into pretrained glove embedding, all the other words is not in the pretrained Glove we assign a random vector with same dimension as GloVe. We than create a tensor R^2 with size **vocab_size**x300. for the ground truth label. For intent classification we use Set object to convert all the intent into numerate label. For slot tagging we also use Set object to convert the slot label.

2 Q2: Describe your intent classification model

2.1 Describe

a

For each data set first we convert pad the input into **max_len** and convert each word into its id format, so each input will become an array **x** with size **max_len**.
Than we process data as

$$\mathbf{x} = \text{Embedding}(\mathbf{x}), \quad \text{size}(\mathbf{x}) = \textbf{max\_len} \text{ x } 300$$

$$\mathbf{out_t}, \mathbf{h_t} = \text{GRU}(w_t, h_t - 1), \quad \text{With layer number} = 2, \text{ and bidirectional=True}$$

$$\mathbf{out} = mean(\mathbf{out}_t), \quad \text{for } \mathbf{t}\text{=1,2,....}\mathbf{max\_len}$$

$$\mathbf{out} = \text{FCN}(\mathbf{out})$$

We then receive a **out** tensor with length equal to number of class.

b

public score on kaggle is 0.90666

c

i use the crossentropy loss

$$-\sum_{C=1}^{M} \mathbf{1}_{y_{o,c}} log(p_{o,c})$$

where
M = number of classes,
$\mathbf{1}_{y_{o,c}}$ = binary indicator if observation's label is c,
$p_{o,c}$ = probability of observation belong to c

d

Optimizer = AdamW
Learning rate = 1e-3
Batch Size = 128

3 Q3: Describe your slot tagging model

3.1 Describe

a

For each input we do the same process as intent classification, but smaller **max_len**.

$$\mathbf{x} = \text{Embedding}(\mathbf{x})$$

$$\mathbf{out_t}, \mathbf{h_t} = \text{GRU}(w_t, h_t - 1), \quad \text{With layer number = 2, and bidirectional=True}$$

$$\mathbf{out_t} = \text{FCN}(\mathbf{out_t})$$

All $\mathbf{out_t}$ has same dimension as tag number. And when calculating the loss, we ignore the error from padded character.

b

Public score on kaggle is 0.77533

c

I use the crossentropy loss as Q2

d

Optimizer = AdamW
Learning rate = 1e-3
Batch Size = 128

4 Q4: Sequence Tagging Evaluation

	precision	recall	f1-score	support
date	0.75	0.73	0.74	206
first$_n ame$	0.95	0.92	0.94	102
last$_n ame$	0.82	0.83	0.83	78
people	0.78	0.76	0.77	238
time	0.90	0.88	0.89	218
micro avg	0.83	0.81	0.82	842
macro avg	0.84	0.82	0.83	842
weighted avg	0.83	0.81	0.82	842

Token Accuracy calculate the correct number of all the tokens in the data.
Joint Accuracy only calculate the correct number of data if all the token in data is correct.
seqeval calculate three types of metric and average across five type's of tag.

$$precision = \frac{TP}{TP + FP}$$

$$recall = \frac{TP}{TP + FN}$$

TP means True Positive which is the positive prediction that is truely positive
FP means False Positive which is the positive prediction that is truely negative
FN means False Positive which is the negative prediction that is truely positive

$$f1score = \frac{2 * precision * recall}{precision + recall}$$

support is the number to calculate each tag class.

5 Q5: Compare with different configurations

I tried different backbone such as Vanilla-RNN, LSTM and GRU with other configurations remain same.

Intent Backbone	Evaluation Accuracy
Vanilla-RNN	0.8736514449119568
LSTM	0.8837890625
GRU	0.894112765789032

The following graph is the evaluation accuracy for three different backbone

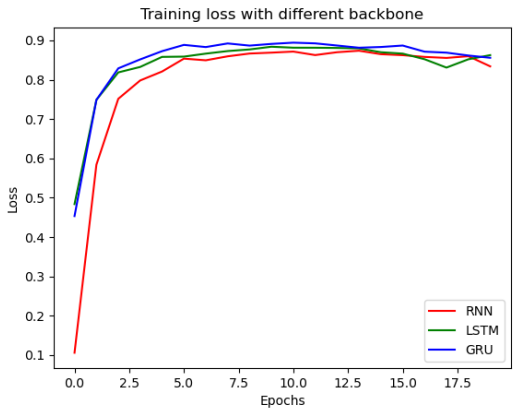

Figure 1: intent classification

As the best evaluation accuracy show that GRU perform the best accuracy, i choose GRU model as my submission. As the graph point out, GRU and the LSTM has the similar convergence curve which is faster than Vanilla-RNN. But as i pad the whole input into 128, we can find out that the speed of LSTM is the lowest one.

Intent Backbone	Training Time
Vanilla-RNN	01:11
LSTM	02:14
GRU	01:55

The following comparison is for slot tagging

Intent Backbone	Evaluation Accuracy
Vanilla-RNN	0.7895132211538461
LSTM	0.8184344951923077
GRU	0.8228665865384616

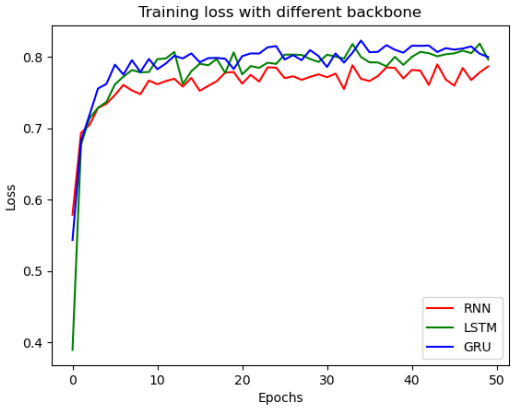

Figure 2: intent classification

As the table also shown that the accuracy of GRU is also the highest in this task, so i also choose GRU as my backbone. But in this task we can see that sometimes LSTM might gain accurate perform in some training epochs compare to GRU.

References