Applied Deep Learning
Homework 1 Report

r11922139 許洸誠

1. Data Process

1.1 Input Data Preprocessing

The $vocab\_size$ most frequent words were selected from the input sentence data to build the **vocab** object.

$$vocab\_size = 3000$$

The input data are in json files, which are transformed into a list of dictionary (each datum is stored in a dictionary). The dictionaries are then used in pytorch dataset and dataloader.

We also define our own collate_fn for dataloader that returns a dictionary **batch**, where

batch['text' | 'tokens'] = 2d word_id matrix of size $(batch\_size,\ max\_len)$

batch['intent' | 'tags'] = labels of each datum

batch['seq_len'] = list of lengths of each sentence

batch['id'] = list of id of each datum

Note that each sentence may have different length, so each sentence is padded to maximum length of all sentences.

1.2 Embedding Preprocessing

The words in vocab are used to build the embedding matrix using pretrained GloVe. The resulting embedding matrix is of size $(vocab\_size * embed\_dim)$, where

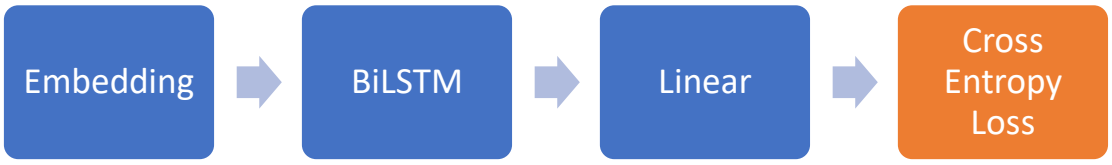
$$vocab\_size = 3000$$

$$embed\_dim = 300$$

If some word is not covered in GloVe, then it is represented by a randomly generated row vector in the embedding matrix.

The cover rates of 3000 words in GloVe are 0.908 and 0.759 for intent classification data and slot tagging data respectively.

2. Intent Classification Model



■ Input → Embeddings

The input x of size $(batch\_size, max\_len)$ is fed into the embedding layer to obtain the embedding tensor emb of size $(batch\_size, max\_len, embed\_dim)$.

$$emb[i][j] = embeddings[x[i][j]]$$

where

$emb[i][j]$ is the embedding vectors of j-th word in i-th sentences

$x[i][j]$ = word id of j-th word in i-th sentence

$$batch\_size = 128$$

■ Embeddings → BiLSTM

The input emb from embedding layer is sent into BiLSTM layer which gives

- out: output tensor, the output feature of last layer at every time step.
- h: hidden states of all layers at last time step.
- c: cell states of all layers at last time step.

$$out, (h, c) = BiLSTM(emb)$$

tensor	out	h	c
size	$(batch\_size,$ $max\_len,$ $2 * hidden\_size)$	$(2 * num\_layer,$ $batch\_size,$ $hidden\_size)$	$(2 * num\_layer,$ $batch\_size,$ $hidden\_size)$

where

$$hidden\_size = 512$$

$$num\_layer = 2$$

■ BiLSTM → Linear

In intene classification problem, hidden states of last layer from both directions are concatenated as feature vectors for each sentence.

$$h = concatenate(h[-1], h[-2])$$

where h is of size $(batch\_size, 2 * hidden\_size)$.

The features are then fed into a linear layer with dropout rate 0.2 to obtain the classification score of each sentence for each intent.

$$score = Linear(h)$$

where score is of size $(batch\_size, num\_intent)$

■ Cross Entropy Loss

The loss is calculated from $score$ by using cross entropy loss function.

For each datum, cross entropy loss is equal to taking softmax of score vector and then calculate the negative log likelihood (NLL).

$$loss = NLL\big(SM(h)\big)$$

Finally, we take the mean of all as our final loss value for this batch.

Performance

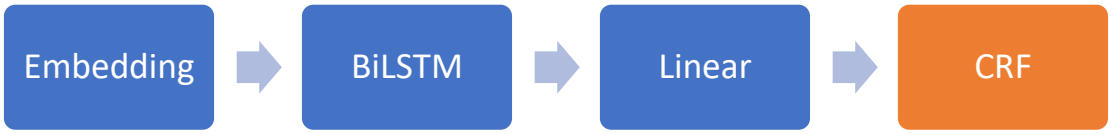
Data	DEV	Kaggle Pulic	Kaggle Private
Accuracy	0.926	0.915	0.90933

Optimizer

Adam optimizer is used for updating model paramters.

$$batch\_size = 128$$
$$learning\_rate = 10^{-3}$$
$$num\_epoch = 50$$

3. Slot Tagging Model



Most of the model in slot tagging problem is the same the one in intent classification, but there are a few difference.

■ BiLSTM

In intent classification, we use h from BiLSTM to predict the score. In slot tagging, we are dealing with sequence labeling, so we take out tensor from BiLSTM.

$out[i][j]$ = feature vector of j-th word in i-th sentence

■ CRF

In sequence labeling problem, sometimes it is not good to simply label each word to the tag with max score of all classes, which ignores the context among labelings.

To take advantage of the context of labeling, we use Conditional Random Field (CRF) as our final layer.

In simple words, CRF score each tagging sequence.

There are 2 types of score in CRF

1. Emission Score (from BiLSTM layer)
2. Transition Score (parameters to be trained)

Score = Transition Score + Emission Score

We predict the tagging of sentence to be the one with max score from CRF layer using Viterbi algorithm.

The loss is calculated as follows:

Label tagging sequences from 1 to N.
Let s_i be the score of tagging sequence i.
Let s_{true} be the score of true labeling.
Define loss to be

$$loss = -\log\left(\frac{SM(s_{true})}{\sum_{i=1}^{N} SM(s_i)}\right)$$

(Take softmax and then negative log likelihood)

The equation can be further simplified to

$$loss = \log\left(\sum_{i=1}^{N} e^{s_i}\right) - s_{true}$$

Performance
See 4.

Optimizer
Adam optimizer is used to update model parameters.

$$batch\_size = 128$$
$$learning\_rate = 5 * 10^{-4}$$
$$num\_epoch = 50$$

4. Sequence Tagging Evaluation

	precision	recall	f1-score	support
date	0.76	0.72	0.74	206
first_name	0.97	0.97	0.97	102
last_name	0.94	0.81	0.87	78
people	0.77	0.75	0.76	238
time	0.86	0.84	0.85	218
micro avg	0.83	0.80	0.81	842
macro avg	0.86	0.82	0.84	842
weighted avg	0.83	0.80	0.81	842

Validation acc and prec: (0.9681916106957293, 0.8235294117647058)

$$Joint\_Accuracy = \frac{number\ of\ sequences\ predicted\ correctly}{number\ of\ sequences}$$

$$Token\ Accuracy = \frac{number\ of\ tokens\ predicted\ correctly}{number\ of\ all\ tokens}$$

A token is predicted correctly if it has correct tag prediction.
A sequence is predicted correctly if all tokens in the sequence are predicted correctly.

	DEV	Kaggle Public	Kaggle Private
Joint Accuracy	0.823	0.8075	0.821

5. Extra Configuration

In slot tagging, besides using CRF, a simple fully connected linear layer + cross entropy loss is also experimented.

It turns out the performance using CRF doesn't ourperform simple fc layer much.

	CRF	FC
Kaggle Public	0.8075	0.81286
Kaggle Private	0.8210	0.81457

There are some possible reasons that CRF doesn't outperform simple fully connected much:

- The relation between tags are not strong enough, and the number of tags are few (only 9 tags).
- Many data are all labeled as all O tags. (thus doesn't have much context to train CRF).

Appendix

1. CRF reference:

https://zhuanlan.zhihu.com/p/44042528
https://pytorch.org/tutorials/beginner/nlp/advanced_tutorial.html