Applied Deep Learning, Fall 2022

Assignment 1 - Report

R11922133 Cheng-Hua, Liao

1 Data Processing

Tokenization. The input sentences are simply tokenized with space character as delimiter. These tokens are collected along with two special tokens, [PAD] and [UNK], to form a vocabulary.

Lookup table. Next, a lookup table is built to map each token to its corresponding GloVe[1] word embedding. For tokens not presented in GloVe, their embeddings will be randomly initialized with values drawn from a uniform distribution with range [-1, 1].

Preprocessing for slot tagging task follows the same procedures, except that the tokenization has already been done in the given dataset.

2 Intent Classification

2.1 Model

The model architecture follows the work of [2] and [3], as illustrated in Figure 1. Given an input sentence, it is first padded or truncated to a sequence of 25 tokens, where each token is represented by a 300 dimensional vector. Each input sequence is treated as a 1-d data with 300 channels, in which 1-d convolution operations are applied with kernels of width 2, 3, and 4. This produces 100 feature maps from each kernel. We then apply max-over-time pooling over feature map, resulting in total 300 feature points. Finally, these features are concatenated to output a vector of 300 dimensions, then fed to a linear layer with softmax to obtain output probabilities.

2.2 Training

parameter	**value**
optimizer	adam
loss function	cross entropy
batch size	128
dropout	0.3
learning rate	1e-3 $\rightarrow$ 5e-5

Table 1: Hyperparameters for intent classification task.

The configurations are as Table 1. The model is trained with learning rate 1e-3 for the first 50 epochs, and then with learning rate of 5e-5 for the last 50 epochs.

2.3 Result

Public score: 0.93644.

[1] Common Crawl (840B tokens, 2.2M vocab, cased, 300d vectors)

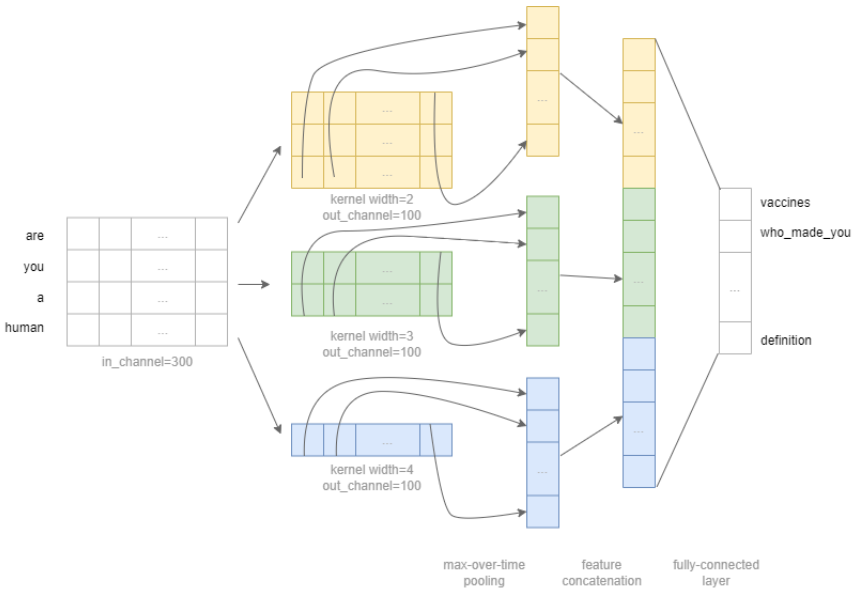

Figure 1: The textCNN pipeline in intent classification task.

3 Slot Tagging

3.1 Model

The model architecture follows the work of [1], as illustrated in Figure 2. Note that the fully-connected layers share the same parameters. To extract character-level information for each token, each character is represented by a 25-dimensional embedding, randomly initialized with values drawn from a uniform distribution with support [-0.5, 0.5]. Each token is padded to the same length, followed by a convolution operation and max-pooling to output a vector of 10 dimension. This 10-dimension features is concatenated to the word embedding to be the input of the recurrent layer.

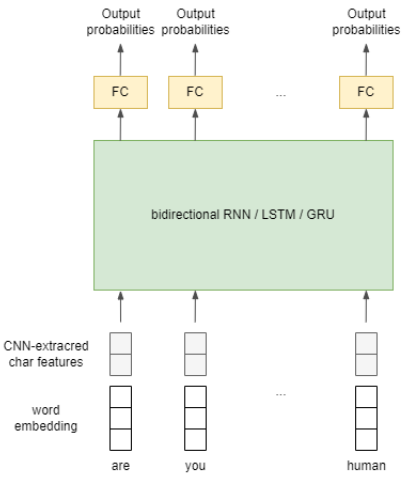

Figure 2: The cnn-biLSTM architecture in slot tagging task.

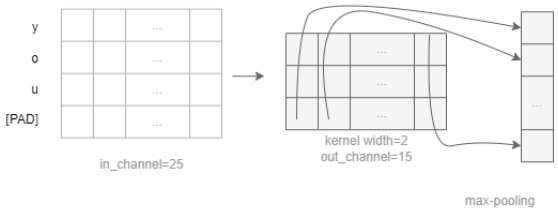

Figure 3: The convoutional layer extracting character-level features from each token.

3.2 Training

parameter	value
optimizer	adam
loss function	cross entropy
batch size	128
dropout	0.7
learning rate	1e-3 → 5e-5

Table 2: Hyperparameters for slot tagging task

The configurations are as Table 2. The model is trained with learning rate 1e-3 for the first 30 epochs, and then with learning rate of 5e-5 for the last 30 epochs.

3.3 Result

Public score: 0.76032

4 Sequence Tagging Evaluation

- joint accuracy: 0.776
- token accuracy: 0.988

	precision	recall	f1-score	support
date	0.71	0.71	0.71	206
first_name	0.92	0.83	0.87	101
last_name	0.82	0.65	0.73	78
people	0.72	0.68	0.70	238
time	0.89	0.85	0.87	218
micro avg	0.79	0.75	0.77	841
macro avg	0.81	0.75	0.78	841
weighted avg	0.79	0.75	0.77	841

Table 3: Evaluation results of `seqeval` on the validation set.

The differences between joint accuracy, token accuracy, and `seqeval` are as follows: **precision**: the rate of true positives among the predicted positive samples. **recall**: the rate of true positives among the actual positive samples. **f1-score**: the harmonic mean of precision and recall. **micro avg**: first pool the contingency tables over categories (i.e. date, first_name, ...), then calculate the metrics. **macro avg**: calculate the metrics for each categories individually, then average them.

In brief, token accuracy assess accuracy token-wisely; joint accuracy evaluates the overall performance; `seqeval` gives more detailed analysis on how the model perform on different categories.

5 Compare with different configurations

5.1 Intent Classfication

This subsection reports findings from recurrent-network-based methods. The model architecture is illustrated as Figure 4. Each experiments is trained for 50 epochs, 1e-3 learning rate, and the rest hyperparameters the same as Section 2.2.

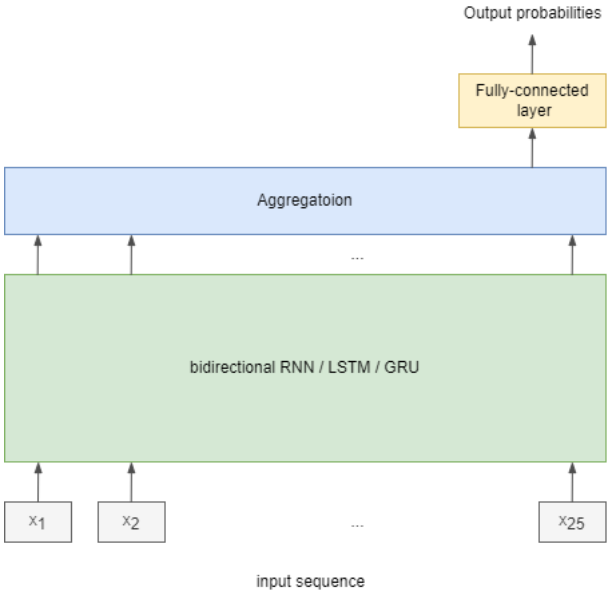

Figure 4: Model architecture for rnn-based intent classification.

RNN variants. RNN_RELU, RNN_TANH (the suffix indicates the nonlinearity it applies), LSTM, GRU are 4 common recurrent network architecures, they are compared in the following experiments. Results comparing these variants are shown in Figure 5. First, LSTM and GRU performs comparably well, followed by RNN_RELU, while RNN_TANH performs the worst. However, GRU converges much faster than the others, making it more desirable than LSTM in this task. Finally, exploding gradients are more frequently observed in RNN_RELU and RNN_TANH, which might indicate that we can improve their performance with gradient norm clipping.

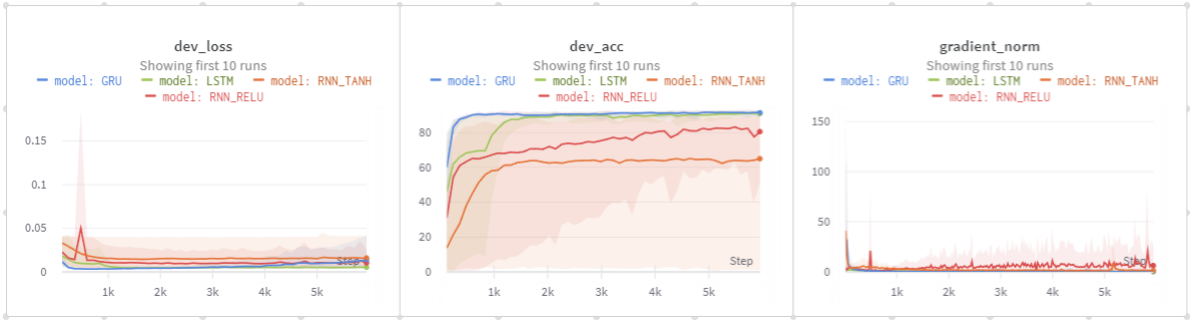

Figure 5: The training curves for different rnn variants.

Aggregation of output embeddings. Let $o_1, ..., o_{25}$ be the output embeddings of the recurrent layer. The aggregation layer in Figure 4 performs a weighted sum $\sum_{i=1}^{25} \alpha_i o_i$. We compare 4 aggregation schemes:

- **last**: $\alpha_i = 1$ if $i = 25$ or $\alpha_i = 0$ otherwise
- **mean**: $\alpha_i = 1/25$
- **sum**: $\alpha_i = 1$
- **learnt**: α_is are learnable parameters

Results are shown in Table 4 and Figure 6. Although in previous researches the last output embedding is believed to aggregate information from the whole sequence, we found that this scheme leads to not only slower convergence but poorer performance. Instead, aggregating the whole sequence explicitly might gain richer information. It is interesting that LSTM and GRU are less affected by the aggregation scheme.

RNN variants	Aggregation method			
	last	mean	sum	learnt
RNN_RELU	.51267	.89700	**.92767**	.88333
RNN_TANH	.01000	.83233	.87067	**.88600**
LSTM	.87900	.91033	.92333	**.92367**
GRU	.91767	.92333	.88967	**.92433**

Table 4: Prediction accuracy for each rnn-variant-aggregation-method pairs.

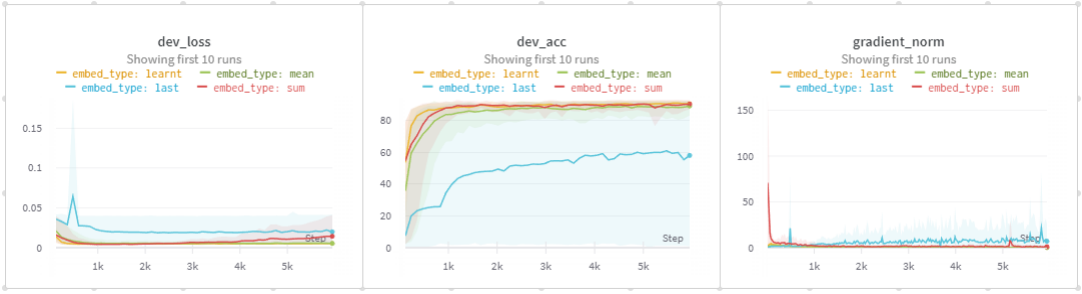

Figure 6: The training curves under different aggregation method.

Gradient norm clipping. Results of applying gradient norm clipping are show in in Table 5, where the max-norm values are heuristically chosen from observing the gradient-norm curves. Although it seemed that GRU is the only to benefit from gradient clipping, the effects should require further analysis with different strategies like adaptive gradient clipping.

	w/o clipping (accuracy)	w clipping (accuracy/max_norm)
RNN_RELU	**.92767**	.92667 (2)
RNN_TANH	**.87067**	.84067 (10)
LSTM	**.92333**	.91367 (2)
GRU	.88967	**.90500 (10)**

Table 5: Prediction accuracy for intent classification task with or without gradient norm clipping.

5.2 Slot Tagging

This subsection compares performances of different rnn variants. The model architecture is the same as Section 3.1. Each experiments is trained for 30 epochs, 1e-3 learning rate, and the rest hyperparameters the same as Section 3.2. Results are shown in Table 6 and Figure 7. We can see that introducing cnn-extracted character features do improve performance.

	token accuracy	joint accuracy
RNN_RELU	.9879	.776
RNN_TANH	.9864	.749
LSTM	.9880	.769
GRU	.9884	.782
CNN-biLSTM	**.9890**	**.791**

Table 6: Prediction accuracy for slot tagging task under different rnn variant.

Figure 7: Caption

References

[1] Chiu, J. P., and Nichols, E. Named entity recognition with bidirectional lstm-cnns. *Transactions of the association for computational linguistics 4* (2016), 357–370.

[2] Kim, Y. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (Doha, Qatar, Oct. 2014), Association for Computational Linguistics, pp. 1746–1751.

[3] Zhang, Y., and Wallace, B. A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. *arXiv preprint arXiv:1510.03820* (2015).