National Taiwan University
Application of Deep Learning
Homework 1 Report

I-Ho Chiu (r11922189)

October 2022

# 1 Data processing (2%) Describe how do you use the data for intent_cls.sh, slot_tag.sh:

1. **How do you tokenize the data.**
2. **The pre-trained embedding you used.**

I tokenize the data with sample code. Which use **GloVe** as the pre-trained embedding.

For input sentence in dataset, we map each word to its specific serial ID. The ID of unknown token [**UNK**] is 1. And then we pad each sentence to specific length so that the input length will be the same. The ID of pad token [**PAD**] is 0. In model, we encode each ID to its vector with **GloVe**.

For output class in intent classification problem. We directly map each class to serial ID.

For output tags in slot tagging problem. We map each tag to serial ID. And then we pad each tag-set to specific length so that the output length will be the same. The ID of pad token [**PAD**] here is -100 as **ignore index**. Pytorch will ignore them when calculating loss function.

# 2 Describe your intent classification model. (2%)

1. your model
   $h_t = GRU(w_t, h_{t-1})$, $where$ $\mathrm{w}_t$ is the input word embedding token at time $t$, $h_t$ is the hidden state at time $t$, $h_{t1}$ is the hidden state of the layer at time $t-1$ or the initial hidden state at time 0. The shape of input is $(N, L, H_{in})$ The shape of output is $(N, L, D * H_{out})$, where $N$ is batch

size, $L$ is sequence length, $D$ is 2 if bidirectional otherwise $D$ is 1, $H_{in}$ is input size and $H_{out}$ is hidden size. The model is bidirectional and number of layers is 2

$x_1 = Flatten(h_t),\ where$ $\mathrm{h}_t$ containing the output features from the last layer of the GRU. $x_1$ is flattened $h_t$, it's shape is $(N, L * D * H_{out})$.
$x_2 = Dropout(x_1),\ Input\ shape$ (N, L* D*$\mathrm{H}_{out}$), Output shape $(N, L * D * H_{out})$
$x_3 = Linear(x_2),\ Input\ shape$ (N, L* D*$\mathrm{H}_{out}$), Output shape $(N, H_{out})$
$x_4 = BatchNorm1D(x_3),\ Input\ shape$ (N, $\mathrm{H}_{out}$), Output shape $(N, H_{out})$
$x_5 = LeakyReLU(x_4),\ Input\ shape$ (N, $\mathrm{H}_{out}$), Output shape $(N, H_{out})$
$x_6 = Dropout(x_5),\ Input\ shape$ (N, $\mathrm{H}_{out}$), Output shape $(N, H_{out})$
$x_{out} = Linear(x_6),\ Input\ shape$ (N, $\mathrm{H}_{out}$), Output shape $(N, C)$, where $C$ is number of class and $x_{out}$ is the final output of whole model.

2. performance of your model. (public score on kaggle)
   Public Score: 0.92355
   Private Score: 0.92888

3. the loss function you used.
   Cross Entropy

4. The optimization algorithm (e.g. Adam), learning rate and batch size.
   The optimization algorithm: Adam
   learning rate: 1e-3
   batch size: 384

# 3 Describe your slot tagging model. (2%)

1. your model
   $h_t = GRU(w_t, h_{t-1}), where$ $\mathrm{w}_t$ is the input word embedding token at time $t$, $h_t$ is the hidden state at time $t$, $h_{t1}$ is the hidden state of the layer at time $t-1$ or the initial hidden state at time 0. The shape of input is $(N, L, H_{in})$ The shape of output is $(N, L, D * H_{out})$, where $N$ is batch size, $L$ is sequence length, $D$ is 2 if bidirectional otherwise $D$ is 1, $H_{in}$ is input size and $H_{out}$ is hidden size. The model is bidirectional and number of layers is 2

   $x_1 = Dropout(h_t),\ Input\ shape$ (N, L, D*$\mathrm{H}_{out}$), Output shape $(N, L, D * H_{out})$
   $x_2 = Linear(x_1),\ Input\ shape$ (N, L, D*$\mathrm{H}_{out}$), Output shape $(N, L, H_{out})$
   $x_3 = BatchNorm1D(x_2),\ Input\ shape$ (N, L, $\mathrm{H}_{out}$), Output shape $(N, L, H_{out})$
   $x_4 = LeakyReLU(x_3),\ Input\ shape$ (N, L, $\mathrm{H}_{out}$), Output shape $(N, L, H_{out})$
   $x_5 = Dropout(x_4),\ Input\ shape$ (N, L, $\mathrm{H}_{out}$), Output shape $(N, L, H_{out})$
   $x_{out} = Linear(x_5),\ Input\ shape$ (N, L, $\mathrm{H}_{out}$), Output shape $(N, L, C)$, where $C$ is number of class and $x_{out}$ is the final output of whole model.

2. performance of your model. (public score on kaggle)
   Public Score: 0.78605
   Private Score: 0.77920

3. the loss function you used.
   Cross Entropy

4. The optimization algorithm (e.g. Adam), learning rate and batch size.
   The optimization algorithm: Adam
   learning rate: 1e-3
   batch size: 128

## 4 Sequence Tagging Evaluation (2%)

1. **Please use seqeval to evaluate your model in Q3 on validation set and report classification report(scheme=IOB2, mode='strict').**

2. **Explain the differences between the evaluation method in seqeval, token accuracy, and joint accuracy.**

joint accuracy $= c/n$, where c is correct sequence count and n is total sequence count.
token accuracy $= c/n$, where c is correct token count and n is total token count.
precision $= tp/(tp + fp)$, where tp is true positive and fp is false positive.
recall $= tp/(tp + fn)$, where tp is true positive, and fn is false negative.
f1-score $= 2 * precison * recall/(precision + recall)$

| | |
|---|---|
| joint accuracy | 0.788 |
| token accuracy | 0.9645165378279053 |

Table 1: classification report (scheme=IOB2, mode='strict')

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| date | 0.78 | 0.80 | 0.79 | 206 |
| first_name | 0.88 | 0.87 | 0.88 | 102 |
| last_name | 0.75 | 0.64 | 0.69 | 78 |
| people | 0.70 | 0.69 | 0.69 | 238 |
| time | 0.86 | 0.86 | 0.86 | 218 |
| | | | | |
| micro avg | 0.79 | 0.78 | 0.78 | 842 |
| macro avg | 0.79 | 0.77 | 0.78 | 842 |
| weighted avg | 0.79 | 0.78 | 0.78 | 842 |

# 5 Compare with different configurations (1% + Bonus 1%) Please try to improve your baseline method (in Q2 or Q3) with different configuration (includes but not limited to different number of layers, hidden dimension, GRU/LSTM/RNN) and EXPLAIN how does this affects your performance / speed of convergence / ...

## 5.1 Intent Classification

I use initial configurations in sample code. increase dropout from 0.1 to 0.5 to avoid over fitting.

First, I try to compare GRU/LSTM/RNN (dropout=0.5 lr=1e-3 batch_size=128 num_epoch=100 num_layers=2 hidden_size=512). Finally I select GRU. (See Table 2)

Second, I try to compare different batch_size (dropout=0.5 lr=1e-3 num_epoch=100 num_layers=2 hidden_size=512). Finally I select batch_size=384. (See Table 3)

Third, I try to compare different num_layers (dropout=0.5 lr=1e-3 num_epoch=100 batch_size=384 hidden_size=512). Finally I select num_layers=2. (See Table 4)

Last, I try to compare different hidden_size (dropout=0.5 lr=1e-3 num_epoch=100 batch_size=384 num_layers=2). Finally I select hidden_size=512. (See Table 5)

### 5.2 Slot Tagging

I use initial configurations in sample code. increase dropout from 0.1 to 0.5 to avoid over fitting.

First, I try to compare GRU/LSTM/RNN (dropout=0.5 lr=1e-3 batch_size=128 num_epoch=100 num_layers=2 hidden_size=512). Finally I select GRU. (See Table 6)

Second, I try to compare different batch_size (dropout=0.5 lr=1e-3 num_epoch=100 num_layers=2 hidden_size=512). Finally I select batch_size=128. (See Table 7)

Third, I try to compare different num_layers (dropout=0.5 lr=1e-3 num_epoch=100 batch_size=128 hidden_size=512). Finally I select num_layers=2. (See Table 8)

Last, I try to compare different hidden_size (dropout=0.5 lr=1e-3 num_epoch=100 batch_size=128 num_layers=2). Finally I select hidden_size=512. (See Table 9)

Table 2: Intent Classification: Compare GRU/LSTM/RNN (dropout=0.5 lr=1e-3 batch_size=128 num_epoch=100 num_layers=2 hidden_size=512).

| | accuracy | training time(m:s) |
|---|---|---|
| RNN | 0.896 | 05:04 |
| LSTM | 0.928 | 08:53 |
| GRU | 0.931 | 07:51 |

Table 3: Intent Classification: Compare different batch_size (dropout=0.5 lr=1e-3 num_epoch=100 num_layers=2 hidden_size=512).

| | batch size | accuracy | training time(m:s) |
|---|---|---|---|
| GRU | 128 | 0.931 | 07:51 |
| | 256 | 0.933 | 06:13 |
| | 384 | 0.942 | 05:27 |
| | 512 | 0.941 | 05:15 |

Table 4: Intent Classification: Compare different num_layers (dropout=0.5 lr=1e-3 num_epoch=100 batch_size=384 hidden_size=512).

| | number of layers | accuracy | training time(m:s) |
|---|---|---|---|
| GRU | 1 | 0.942 | 02:33 |
| | 2 | 0.942 | 05:27 |
| | 3 | 0.924 | 08:19 |

Table 5: Intent Classification: Compare different hidden_size (dropout=0.5 lr=1e-3 num_epoch=100 batch_size=384 num_layers=2).

| | hidden size | accuracy | training time(m:s) |
|---|---|---|---|
| GRU | 128 | 0.928 | 01:42 |
| | 256 | 0.935 | 02:39 |
| | 512 | 0.942 | 05:27 |
| | 1024 | 0.932 | 16:23 |

Table 6: Slot Tagging: Compare GRU/LSTM/RNN (dropout=0.5 lr=1e-3 batch_size=128 num_epoch=100 num_layers=2 hidden_size=512).

| | joint accuracy | training time(m:s) |
|---|---|---|
| RNN | 0.769 | 02:54 |
| LSTM | 0.797 | 04:43 |
| GRU | 0.8 | 04:02 |

Table 7: Slot Tagging: Compare different batch_size (dropout=0.5 lr=1e-3 num_epoch=100 num_layers=2 hidden_size=512).

| | batch size | joint accuracy | training time(m:s) |
|---|---|---|---|
| GRU | 128 | 0.8 | 04:02 |
| | 256 | 0.792 | 03:24 |
| | 384 | 0.791 | 03:04 |
| | 512 | 0.784 | 02:56 |

Table 8: Slot Tagging: Compare different num_layers (dropout=0.5 lr=1e-3 num_epoch=100 batch_size=128 hidden_size=512).

| | number of layers | joint accuracy | training time(m:s) |
|---|---|---|---|
| GRU | 1 | 0.787 | 01:57 |
| | 2 | 0.8 | 04:02 |
| | 3 | 0.789 | 06:15 |

Table 9: Slot Tagging: Compare different hidden_size (dropout=0.5 lr=1e-3 num_epoch=100 batch_size=128 num_layers=2).

| | hidden size | joint accuracy | training time(m:s) |
|---|---|---|---|
| GRU | 128 | 0.776 | 02:06 |
| | 256 | 0.786 | 02:34 |
| | 512 | 0.8 | 04:02 |
| | 1024 | 0.799 | 09:31 |