Assignment 4 Report

In this assignment, I have tried the whole dataset in using `sklearn.dataset.fetch_20newsgroups`

File structure

```
.
├── checkpoints: ignored by git
├── models
├── preprocess.py: fetch dataset from sklearn and pass it to dataSet of fastNLP
├── test_cnn.py: reload saved model and test accuracy
├── train_cnn.py
└── train_rnn.py
```

Usage

Train RNN model:

```
python3 train_rnn.py
```

Train CNN model:

```
python3 run_cnn.py
```

RNN Text Classification

In general, the classification is mainly done by the last layer's hidden state of LSTM after feeding the sentence (article).

The model is

- Embedding: `(seq_len, batch_size) -> (seq_len, batch_size, embedding_dim)`
- LSTM: `(seq_len, batch_size, embedding_dim) -> hidden:(num_layer*num_directions, batch_size, hidden_size)`
    - `num_directions` is 2, a bidirectional LSTM
- Concatenation (of hidden state of last layer of LSTM): `(batch_size, hidden_size) -> (batch_size, hidden_size*2)`
- Dropout: input shape is the same as output shape
- Linear: `(batch_size, hidden_size*2) -> (batch_size, output_size=num_classes)`

CNN Text Classification

My CNN model is just a simple reproduction of the paper(refer to implementation from fastNLP). CNN is much faster than RNN since it can make full use of CPU.

The CNN model for text classification is as follows:

- Embedding: `(batch_size, seq_len) -> (batch_size, seq_len, embedding_dim)`
- Convolution: `(batch_size, height_in=seq_len, width_in=embedding_dim) -> (kernel_types, batch_size, out_channels, height_out, width_out=1)`
    - `kernel_types`: 3 kinds of kernels with height 3, 4, 5. I.e. 3-gram, 4-gram and 5-gram.
    - `out_channels`: 100 kernels of each kind.
- Relu: input shape is the same as output shape
- Max-pooling: `(kernel_types, batch_size, out_channels, height_out) -> (kernel_types, batch_size, out_channels)`
- Concatenation: `(kernel_types, batch_size, out_channels) -> (batch_size, kernel_types*out_channels)`
- Dropout: input shape is the same as output shape
- Linear: `(batch_size, kernel_types*out_channels) -> (batch_size, output_size=num_classes)`

After one night, **the accuracy on test set is `0.69`**, and I think it may due to the large dataset, which make it unable to converge in several hours.

Suggestions for fastNLP

When getting familiar with fastNLP, I find `DataSet` really a handy one and happily get rid of handwriting trivial preprocessing data. Still, something can improve to make it more efficient and user-friendly.

- More on padding sequence:

  It seems that padded data provided by DataSet just simply pad some 0 over the variant length sequences. Supports are recommended to make it more memory efficient by **using `pad_sequence` in Pytorch**. Then the inbuilt `rnn` in pytorch will handle our padded sequence quite well (This requires add some wrapper around `rnn`, which is shown in `./models/custom_rnn.py`). That's padding value in the padded sequence padded by `pad_sequence` won't be taken into consideration in RNN, which can greatly improve efficiency with the sequences' length varies differently.

- Save/load model and optimizer:

  I'm afraid that the saving of optimizer hasn't been implemented, which is required for optimizers with inner adaptive parameters such as Adam. And it would be much better if `Trainer` can automatically (or ask the user to determine) **restart from the previously saved checkpoint** if exists (then I don't have to code for loading the model and optimizer).