R07922118 資工碩一 翁子騰

Assignment 1 Report- Dialogue Modeling

Q1. Data processing

Describe how do you use the data for rnn.sh, attention.sh, best.sh:

a. How do you tokenize the data.

```
from nltk.tokenize import word_tokenize
word_tokenize(sentence)
```

b. Number of negative samples used to train your model.

4 negative samples

c. Truncation length of the utterances and the options.

Utterances: `padded_len = min(self.context_padded_len=300, max(batch['context_lens']))`
Options: `padded_len = min( self.option_padded_len=50,max(sum(batch['option_lens'], [])))`

d. The pre-trained embedding you used.

FastText 的 crawl-300d-2M.vec.zip

Q2: Describe your RNN w/o attention model.

Describe

a. your RNN without attention model

Just use a single layer of bidirectional LSTM

```
self.rnn = nn.LSTM(dim_embeddings, hidden_size=128,
self.num_layers=1 ,bidirectional=True, batch_first=True)
```

b. performance of your model. (on the public leaderboard, at least)

Kaggle: 9.61999

Recall@10: 0.6596 on validation set in epoch 6

c. the loss function you used.

```
torch.nn.BCEWithLogitsLoss()
```

d. The optimization algorithm (e.g. Adam), learning rate and batch size.

Adam optimizer

Learning rate= 0.001

Batch size= 10

Q3: Describe your RNN w/ attention model

Describe

a. your RNN with attention model

I still set hidden size to 128 and use max-pooling to deal with the output of first RNN layer for both context and each option. Then I calculate the attention energies by torch.bmm(opt_output, context_outputs.transpose(1,2)), which means that I use a word in the context to calculate attention energy with every word in each option and will form a attention energy matrix of shape(batch, opt_len, context_len). Then I applied a softmax along the context dimension to get the attention weights of context. Apply the attention weight to the context and then concatenate the resulting vector with 1st RNN output of each option. Then I put the concatenated big vector into 2nd layer of RNN to get the output_2.

I calculate the score of each option by dot product of output of the context and output_2(both outputs were dealt with max-pooling after the RNN.).

b. performance of your model.
(on the public leaderboard, at least)

Kaggle: 9.453
Recall@10 : 0.7078 on validation set in epoch 6

c. the loss function you used.

```
torch.nn.BCEWithLogitsLoss()
```

d. The optimization algorithm (e.g. Adam), learning rate and batch size.
Adam optimizer
Learning rate= 0.001
Batch size= 10

Q4: Describe your best model.

1.Describe (1%)
a. your RNN with attention model
In addition to the RNN with attention model decribed in Q3. I set the num_layers of the first RNN and second RNN to 2 layers, and set dropout rate to 0.2. And I randomly took 9 negative samples when training. All the other parts remain the same.

b. performance of your model.
(on the public leaderboard, at least)
Kaggle score: 9.43333
Recall@10 on validation set in epoch 9: 0.7196

c. the loss function you used.

```
torch.nn.BCEWithLogitsLoss()
```

d. The optimization algorithm (e.g. Adam), learning rate and batch size.
Adam optimizer
Learning rate= 0.001
Batch size= 10

2.Describe the reason you think why your best model is better than your RNN w/ and w/o attention model. (1%)

I have observed that when I used the RNN with attention model and RNN without attention model, the loss stop decreasing and recall@10 stop increasing after 3-4 epochs. I think it's because the model is overfitting. So I increase the layer of RNN to increase the parameters in the model to prevent overfitting. The result showed that I am probably correct.

Increasing the number of negative samples may also help the model learn to prevent overfitting.

Q5: Compare GRU and LSTM

Compare GRU and LSTM models for the following properties

I use the same parameters as RNN without attention to compare GRU and LSTM.

a. the recall@10 score (on validation set or public leaderboard).

	LSTM	GRU
Recall@10	Recall@10: 0.6596 on validation set in epoch 6	Recall@10: 0.6426 on validation set in epoch 6

b. required GPU memory.

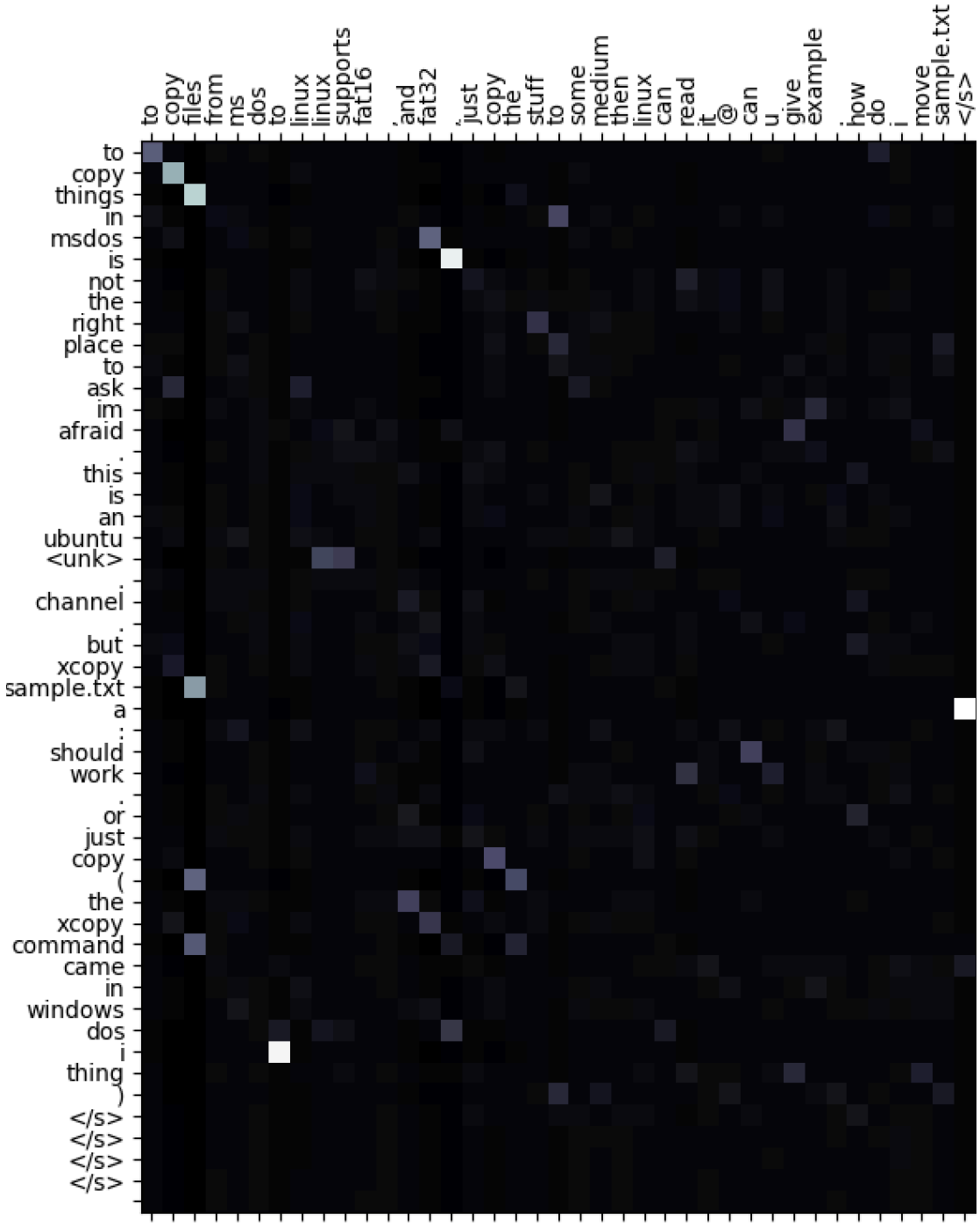
	LSTM	GRU
Required GPU memory	693MiB	691MiB

c. training, testing speed.

	LSTM	GRU
Training time for 1 epoch, batch size=10	4 min 3 sec	3 min 53 sec
Testing time for 1 epoch, batch size=10	12 sec	12 sec

Q6: Visualize the attention weights (2%).

1. Take one example in the validation set and visualize the attention weights (after softmax)



2. Describe your findings. (1%)

越白代表 attention 越強，橫軸是 context utterances，縱軸是 options for correct answer。
Attention 越強代表模型比較關注該 word 上。

可以觀察到比較明顯的是 things 和 files 對應在一起，然後 sample.txt 和 files 與</s>對應在一起。

由於在我設計的 model 中，這個 attention 後來要被用在 context 上，就可以觀察到機器在讀 context utterances 時他所關注的 word。

機器在這個 sample 裡面比較關注的 context word 是 to, copy, files, dos , fat32, </s>等。算是有抓到一點 context utterance 的意思。

原來的 context utterance 是(粗體代表機器比較注意的部分)

how **to copy files** from ms **dos** to linux**, linux supports** fat16**,** and **fat32 ,** just copy the stuff **to** some medium then linux can read it**, @ can** u give example**.** how do i move sample**.**txt

options for correct answer 是

how to copy things in msdos is not the right place to ask im afraid**.** this is an ubuntu hcanne**.** channel**.** but xcopy sample**.**txt a**:** should work**.** or just copy (the xcopy command came in windows dos i thing)

Q7: Compare different settings.

- Compare training with different settings:
  - different number of negative samples (1%).

negative samples	4	6	9
Recall@10 on validation set in epoch 9 of RNN w/ att	0.689	0.6944	0.698
Loss in epoch 9	0.12952	0.11182	0.10986