

Assignment 4 Report

2019/5/28

Assignment 4 Report

- Project Structure

- Data Preprocessing

 - Assignment 2 Dataset

 - Data Statistics

 - Text's length distribution (based on word)

 - Preprocess

- Algorithm Implement

 - CNN Text

 - RNN Text(LSTM Text)

 - Word2Vec

 - Glove

- Result

 - Assignment 2 dataset

 - Hyper Parameter

 - Train with different model

 - 20newgroups total classes

 - Hyper Parameter

 - Train with different model

 - Summary

 - The influence of data

 - The influence of pre-train

 - The influence of optimizer

- Reference

- Classification

 - Work

 - About FastNLP

 - Bug

 - Other problems

 - Suggestions

Project Structure

项目架构借鉴了比较常用的parser结构，在run.py中控制运行模式，通过参数指定调用model.py中的各个模型。

In /assignment4/学号

--- data : 数据、模型、及结果的存储

--- model : 存储各个模型的BestModel

--- result : 存储预测的结果

--- src : 程序代码

--- run.py : 运行控制
--- model.py : 本实验的模型实现
--- dataset.py : 数据预处理及存储
--- run.sh : 运行脚本

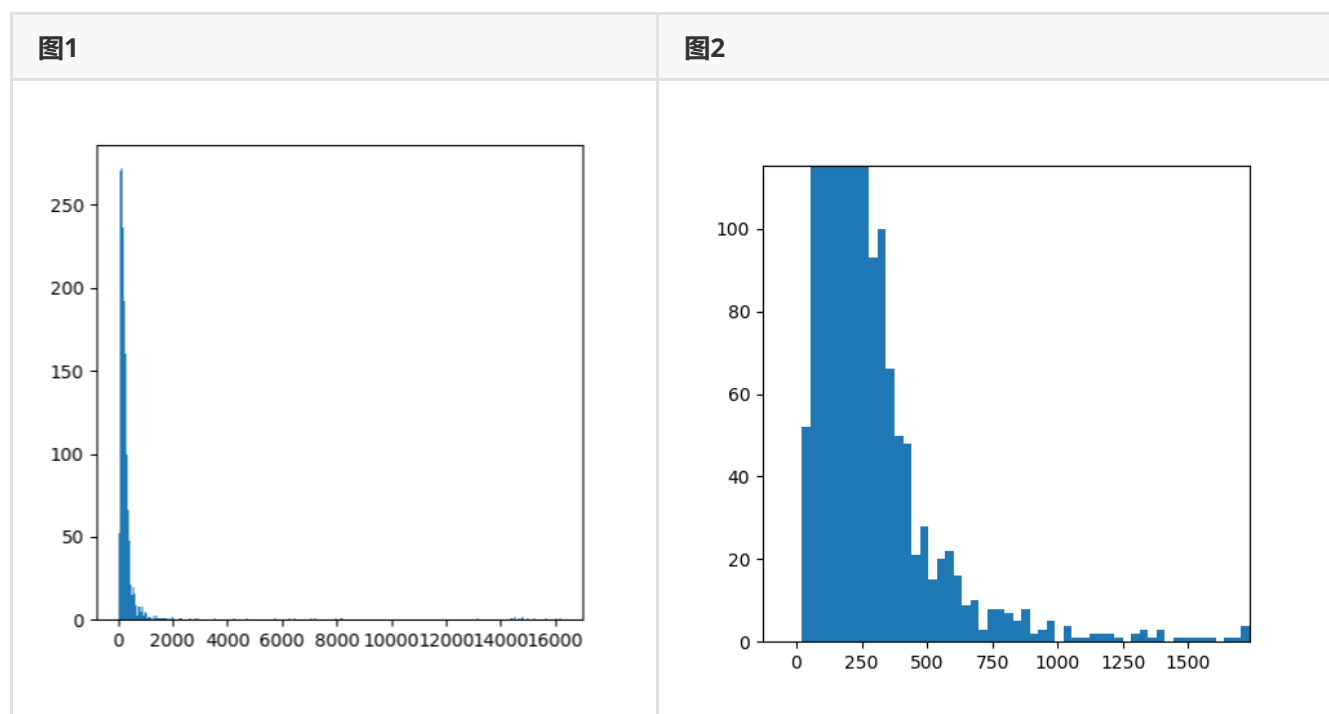
Data Preprocessing

Assignment 2 Dataset

对于之前assignment2中使用的数据集，进行了进一步的分析和处理，并转换到FastNlp的Dataset以供后续训练使用。(对应 dataset.py 中的 TextData 类)

Data Statistics

Text's length distribution (based on word)



可以发现，绝大多数的数据的长度在500以下，所以在实验中默认的最大长度设置为500(这一参数可以在运行时指定 `max_seq_len` 更改)

Preprocess

预处理和assignment2中的预处理基本相同，都是先分词tokenize，再构建数据集和词表。不同的地方在于由于使用 fastnlp，构建此表的工作大大简化了。同时dataset也可以不进行padding（不过此处还是进行了补零操作）。

预处理的过程可以通过如下命令进行：

```
1 python run.py --prepare \  
2 --min_count 10 --max_seq_len 10 \  
3 --data_src "20news" --vocab_dir ../data/vocab --vocab_data vocab.data
```

第二、三行的参数可以皆为默认值，可以不写，数据处理完毕后通过pickle讲TextData类存储在vocab_dir/vocab_data文件中。

Algorithm Implement

CNN Text

参考FastNlp的CNNTText实现用于文本分类任务的CNN模型，相比FastNLP的CNNTText，实现要更为简洁（因为没有用很多FastNlp包装好的东西，而是直接使用torch）实现的类可在model.py中的MyCNNTText类。运行方法如下：

```
1 python --train --model MyCNNTText --optim Adam \  
2     --cuda --gpu 0\  
3     --vocab_dir ../data/vocab --vocab_data vocab.data \  
4     --model_dir ../data/models --model_suffix default \  
5     --epochs 10 --batch_size 64 \  
6     --learning_rate 0.001 --dropout 0 --weight_decay 0
```

除第一行外，其他几行皆可不用写，上述脚本中的值皆为默认值。其中-cuda指示是否要使用GPU，若启用，则使用-gpu参数指定的GPU。训练过程中最好的模型会被存储在"model_dir/model/model_suffix/"目录下

RNN Text(LSTM Text)

本次实验的RNN直接使用了LSTM进行分类，讲assignment3中实现的LSTM直接拿来使用，讲输出层的维度改为类别数量即可。运行方法同上，只需将model参数改为LSTMText即可。

Word2Vec

word2vec是通过gensim进行训练而获取的。训练的参数如下代码所示：

```
1 word2vec_model = word2vec.Word2Vec( sentences.content,size=300,  
2                                     min_count=10,workers=8,iter=15)
```

训练好的模型直接存储在../data/prepare/w2v_model.txt中，并通过fastnlp提供的EmbedLoader类进行加载，生成pre-train的embedding，存储于../data/prepare/w2v_embeds.pkl中，在训练时通过指定--pretrain_model可以直接加载该与训练的embedding。

预训练脚本如下：

```
1 python run.py --pretrain --pretrain_model word2vec
```

训练时使用与训练模型的方法：

```
1 python run.py --pretrain_model word2vec --train \  
2     .....
```

Glove

Glove直接从官网下载了预训好的模型，连接ref: <http://nlp.stanford.edu/data/glove.6B.zip>，使用的是6B tokens，也就是最小的那个，解压后由多个文件，选择最大的glove.6B.300d.txt文件作为预训模型。

虽然是训练好的，但是也要经过以下处理，所以使用前也要运行预训脚本。

```
1 | python run.py --pretrain --pretrain_model glove
```

运行时:

```
1 | python run.py --pretrain_model glove --train \  
2 | .....
```

Result

Assignment 2 dataset

Hyper Parameter

epochs	patience	batch_size	lr	embed_size	hidden_size	weight_decay
<=100	10	64	0.001	128	128	0

patience 为early stop的参数, 使用了fastnlp的earlystop 的callback方法

Train with different model

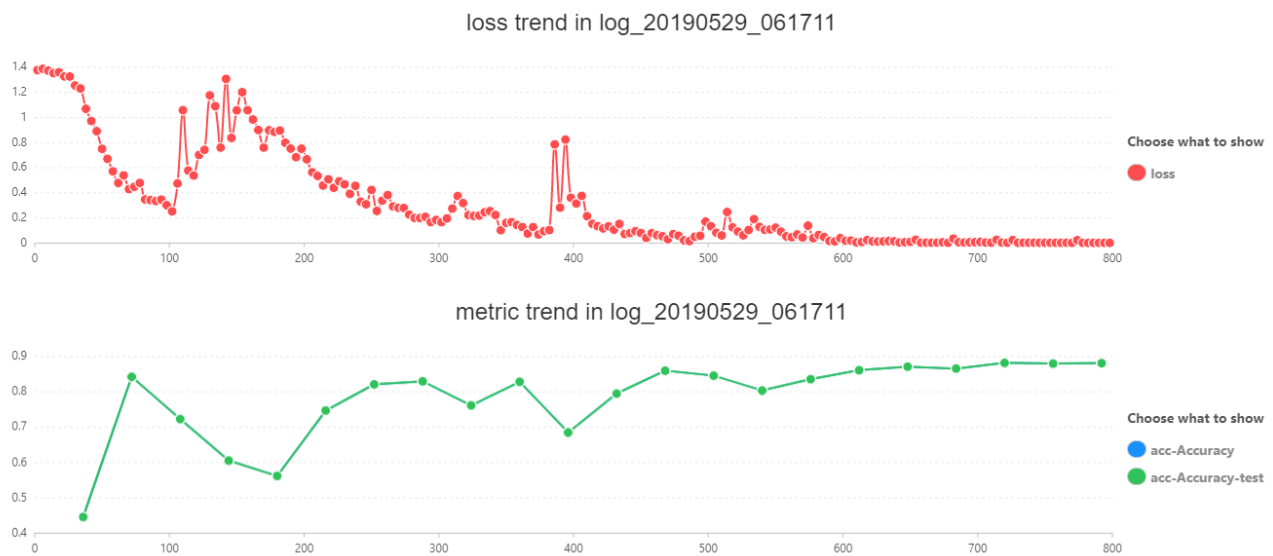
id	model	optimizer	pretrain	Dropout	acc
6	CNNText	Adam	None	0	0.841578
2	CNNText	None	w2v	0	0.951872
1	CNNText	Adam	w2v	0	0.954545
8	CNNText	Adam	Glove	0.3	0.95254
7	LSTMText	None	None	0	0.692513
5	LSTMText	Adam	None(BOW)	0	0.725936
3	LSTMText	None	w2v	0	0.915775
4	LSTMText	Adam	w2v	0	0.885027
9	LSTMText	Adam	Glove	0.3	0.837567

在使用w2v的情况下表现最好的是CNNText+Adam模型，其Loss的曲线和Accuracy曲线情况如下：

 1559137570750

图像由 `fitlog` 绘制

使用Glove的情况下，最好的依然是CNN+Adam，整体变换情况和使用word2vec基本没区别。下图为LSTM模型的Loss变化 (word2vec+Adam+LSTM)：



20newgroups total classes

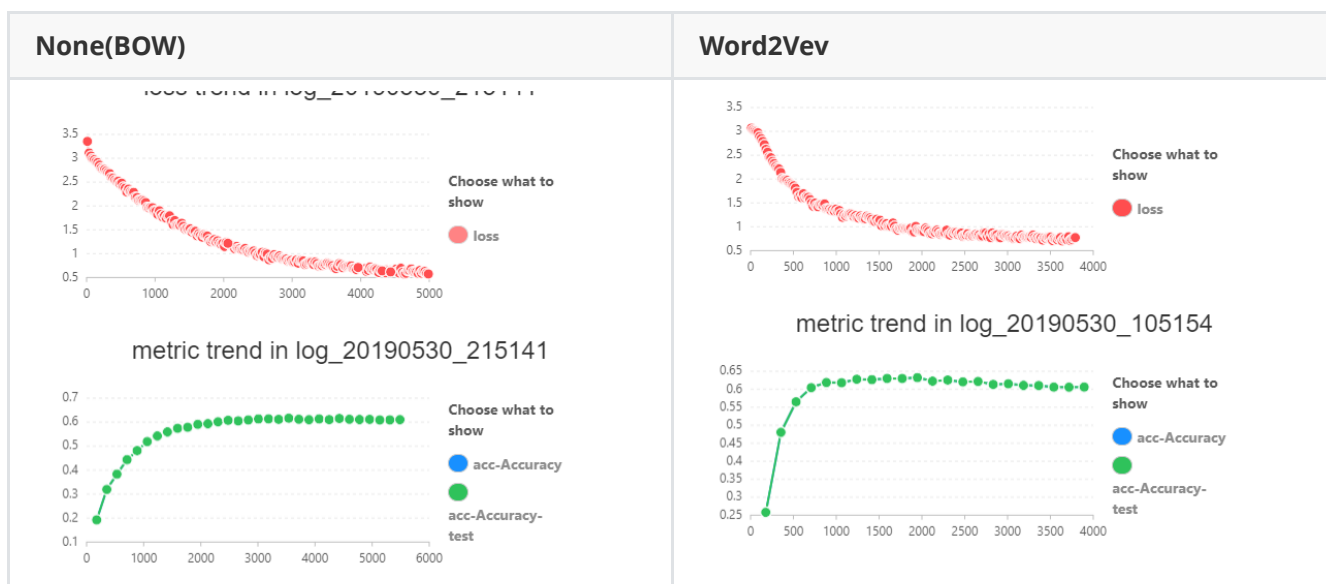
Hyper Parameter

epochs	patience	batch_size	lr	embed_size	hidden_size	weight_decay	drop
<=100	10	64	0.001	128	128	0	0.3

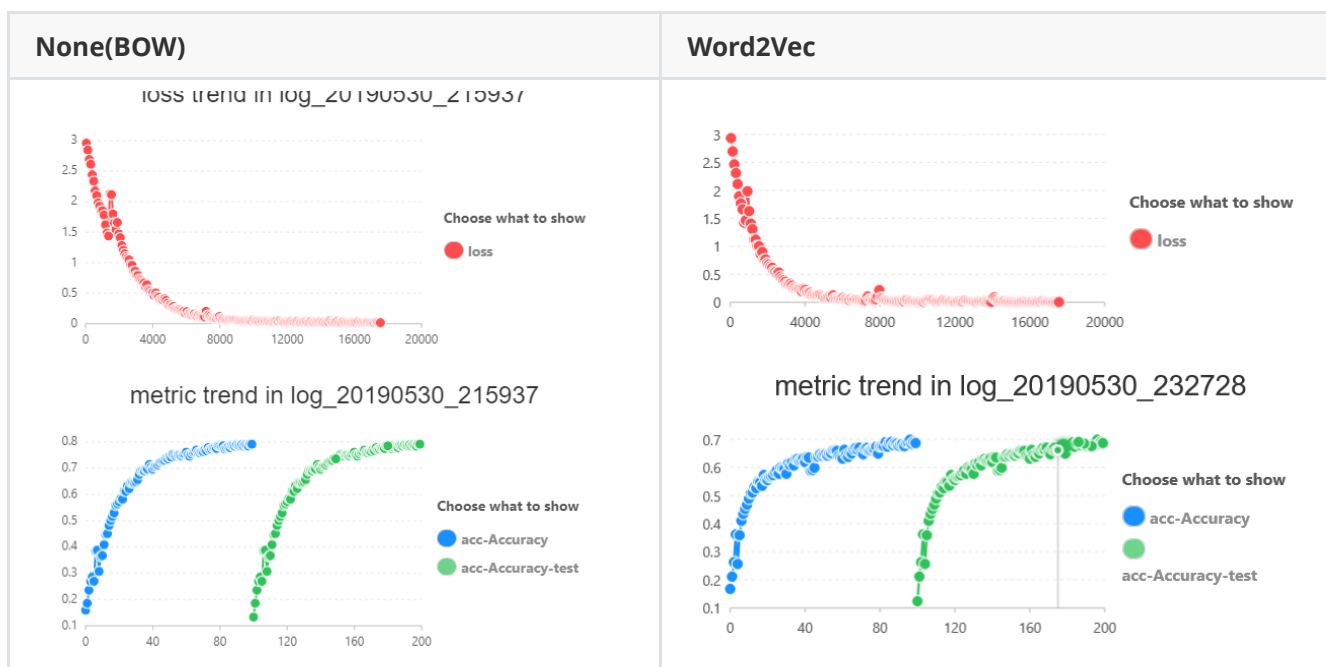
Train with different model

id	model	optimizer	pretrain	acc
1	CNNText	Adam	None(BOW)	0.616569
2	CNNText	Adam	w2v	0.632369
3	CNNText	Adam	Glove	0.72504
4	LSTMText	Adam	None	0.792751
5	LSTMText	Adam	w2v	0.698354
6	LSTMText	Adam	Glove	0.695964

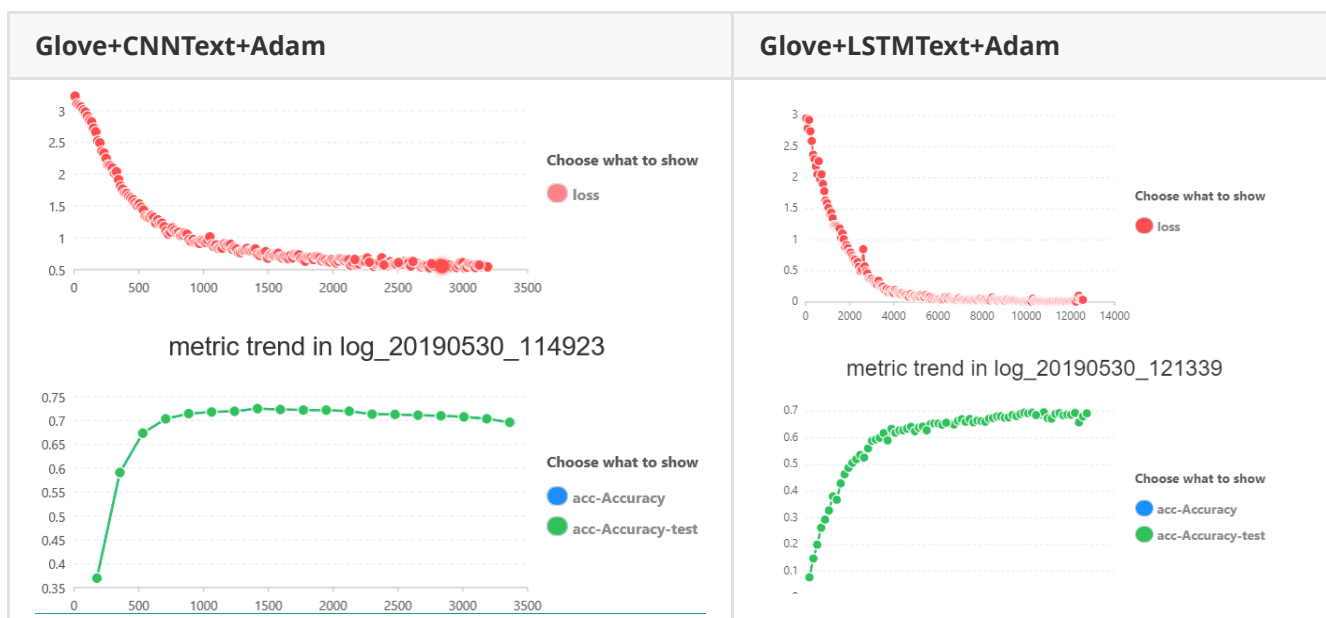
下图为CNNText+Adam在本数据集下的LOSS和准确率曲线。



下图为LSTM+Adam在本数据集下的LOSS和准确率曲线：



使用Glove预训练模型的结果：



一开始Glove的数据集没能训起来，下载来的Glove预训模型在这个数据集下总是报错（Index out of range），所以有关Glove在本数据集下使用时采用的方法时直接利用gensim提供的 `glove2word2vec` 函数转换成word2vec的形式，再使用。发现错误没有了。

```
File "run.py", line 291, in <module>
    run()
File "run.py", line 289, in run
    train(args)
File "run.py", line 252, in train
    callbacks=[FitlogCallback(test_data),earlystop])
File "/remote-home/competition/anaconda3/lib/python3.7/site-packages/fastNLP/core/trainer.py", line 446, in __init__
    batch_size=min(batch_size, DEFAULT_CHECK_BATCH_SIZE))
File "/remote-home/competition/anaconda3/lib/python3.7/site-packages/fastNLP/core/trainer.py", line 810, in _check_cod
e
    pred_dict = model(**refined_batch_x)
File "/remote-home/competition/anaconda3/lib/python3.7/site-packages/torch/nn/modules/module.py", line 493, in __call_
    result = self.forward(*input, **kwargs)
File "/remote-home/competition/anaconda3/lib/python3.7/site-packages/fastNLP/models/cnn_text_classification.py", line
53, in forward
    x = self.embed(words) # [N,L] -> [N,L,C]
File "/remote-home/competition/anaconda3/lib/python3.7/site-packages/torch/nn/modules/module.py", line 493, in __call_
    result = self.forward(*input, **kwargs)
File "/remote-home/competition/anaconda3/lib/python3.7/site-packages/fastNLP/modules/encoder/embedding.py", line 42, i
n forward
    x = super().forward(x)
File "/remote-home/competition/anaconda3/lib/python3.7/site-packages/torch/nn/modules/sparse.py", line 117, in forward
    self.norm_type, self.scale_grad_by_freq, self.sparse)
File "/remote-home/competition/anaconda3/lib/python3.7/site-packages/torch/nn/functional.py", line 1506, in embedding
    return torch.embedding(weight, input, padding_idx, scale_grad_by_freq, sparse)
RuntimeError: index out of range at /pytorch/aten/src/TH/generic/THTensorEvenMoreMath.cpp:193
```

Summary

本实验使用的小数据集和大数据集的区别主要在于类别数目的不同，大数据集有20类，小的只有4类，所以大数据集不仅训练时间要长，训练的难度也更大。在数据集增大后，可以加入较大的dropout来防止过拟合，提高泛化能力，所以本实验在大数据集上使用了0.3的dropout。

The influence of data

从实验结果上看，大数据集的结果显然要比小数据集的结果要差（类别增多，难度自然增大）。但是从结果上可以发现，不同模型在两个数据集下的表现不同。在小数据集上，CNN表现出了极好的特性，而LSTM就相对差很多（暂不考虑参数的影响）。但是在大数据集上，LSTM的表现明显好于CNN，不管是加不加预训练都是如此。而且需要注意的是，由于时间关系，在大数据集上的训练都限定了epochs不超过100，CNN在20-40便会early

stop (patience=10) , 但是LSTM直到第100epoch准确率依然在上升, 如果多训练几个epoch, 模型结果应该会更好。

The influence of pre-train

从结果可以看出, 预训练确实提高了模型的准确率 (10% - 20%) , 这说明语义信息在分类任务中有很大的作用。同时也可以看出在小集上Word2Vec的效果要优于Glove, 但是在大数据集上Glove要优于Word2Vec (但看CNN) 。其中的原因可能在于在数据较小时 (类别较少) , Word2Vec由于经过了针对训练集的训练, 而充分挖掘了训练集的信息, 而Glove相对小数据可能会有更多的噪音, 从而表现不佳。但是当数据集扩大后, Glove模型的信息显得更加充分, 从而表现要更好一些。 (还有一种可能是两个数据的测试集大小也不一样, 在使用Word2Vec时只针对训练集进行训练, 从而无法充分表示测试集的信息, 而Glove是用其他预料充分训练的, 可能会对测试集有更多的信息。即有可能不是Glove变好了, 而是Word2Vec变差了 (比如过拟合了) , 但是时间关系, 没法证实这一猜想了。)

但是, 从结果中也观察到LSTM在大数据集下加入与训练后效果变差了, 但是LOSS是很低的, 有可能的原因在于预训练的模型导致了过拟合, 从而导致最终结果变差, 由于LSTM训起来时间实在过长, 所以没有做进一步验证。

The influence of optimizer

由于本次实验没有特别关注优化器, 所以只在小数据集上测试了以下优化器的表现, 在大数据集上一律采用了Adam优化器。从小数据集的结果可以看出, 优化器对模型的影响与模型和参数皆有关系, 但是由于实验不够, 所以无法得出更细致的结论。

Reference

- [1]. https://fastnlp.readthedocs.io/zh/latest/user/tutorial_one.html# (fastnlp fastnlp)
- [2]. <https://www.aclweb.org/anthology/D14-1181> (CNN Text paper)
- [3]. <https://github.com/Shawn1993/cnn-text-classification-pytorch> (CNN Text implement with pytorch)
- [4]. <https://blog.csdn.net/nlpuser/article/details/83627709> (预训练模型的使用)
- [5]. <https://blog.csdn.net/u010417185/article/details/80651716> (nlp 大杂烩)
- [6]. <https://nlp.stanford.edu/projects/glove/> (Glove 官网)

Classification

Work

CNNText的实现参考了fastnlp的官方实现以及Assignment Discription中的连接。

LSTMText的实现参考了fastnlp的入门教程和Assignment3的任务。

扩大数据集后遇到很多奇奇怪怪的bug, 不过最终还是解决了。

最后一个assignment的, 感觉自己写的代码越来越漂亮了。

About FastNLP

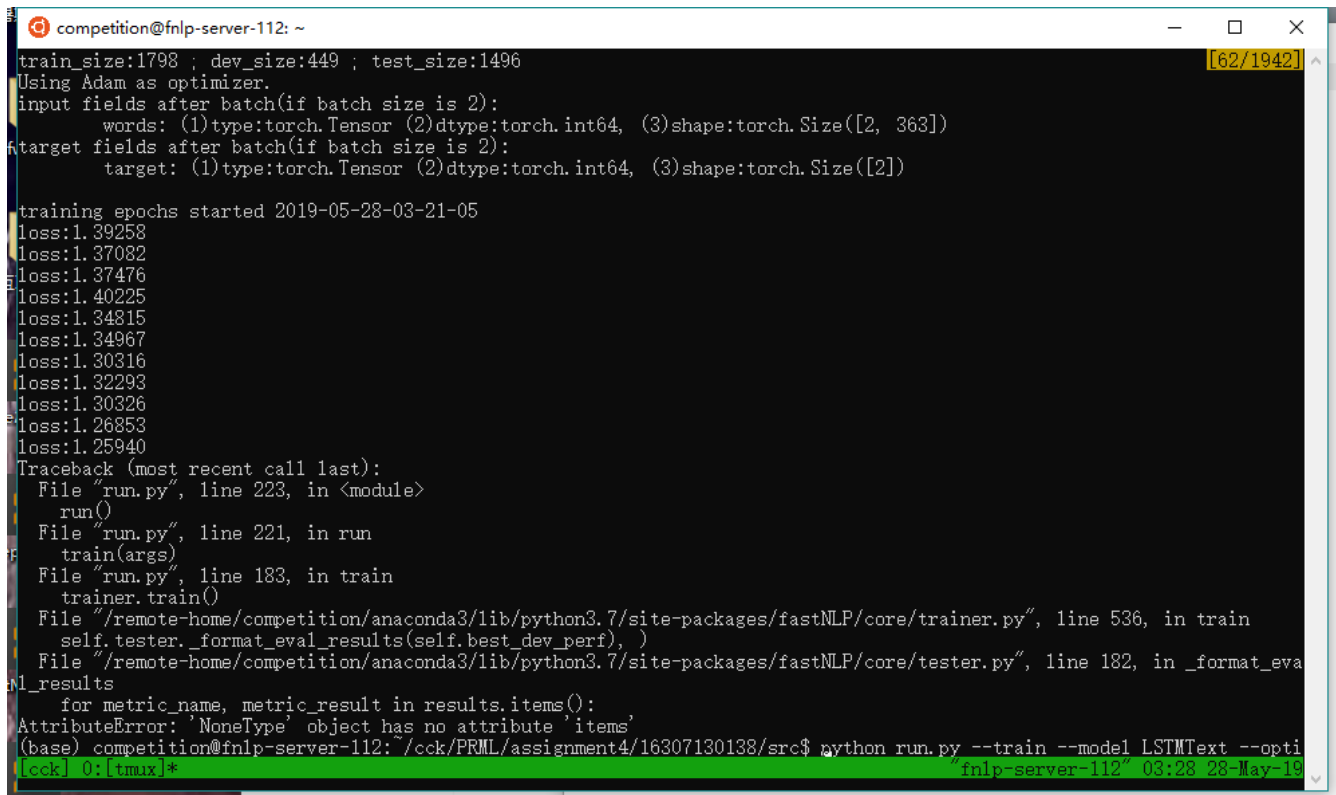
FastNLP作为一个高级工具, 搭建NLP模型是比较便捷的。相比于其他类似工具, fastnlp有几点优势:

1. 中文文档，方便中国人使用，而且对学生群体比较友好。
2. 代码很简洁，源码看起来还是很舒服的。
3. 有一个fitlog进行跟踪，并结合了git版本控制系统，给模型管理带来了极大的便利，而且结合callback使用起来特别爽。
4. 相比于torchtext,keras等高阶包，感觉fastnlp更专注于模型的训练和管理，前述的fitlog和callback就是这样，非常好用。同时fastnlp的vocabulary类个人很喜欢，非常方便。

但是fastnlp作为一个新秀，还是很多问题。比如最重要的，Bug。

Bug

目前遇到的一个比较谜的bug是在计算准确率时的错误。错误发生在LSTM使用AccuracyMetric()时（在使用CNNText的时候没有问题），错误发生在一个epoch的train完成后调用metric进行计算时。此时dev_set大小为449，batch_size=64总是有一个None Type的错误。Debug了好久也没找到问题的源头。调整batch_size不能解决问题，手动padding不能解决问题。最后我放弃使用dev_set,直接在test_set上进行验证，没有报错。dev_set时直接从train_set中split出来的(0.1)。报错图如下：



```
competition@fnlp-server-112: ~  
train_size:1798 ; dev_size:449 ; test_size:1496 [62/1942]  
Using Adam as optimizer.  
input fields after batch(if batch size is 2):  
  words: (1)type:torch.Tensor (2)dtype:torch.int64, (3)shape:torch.Size([2, 363])  
target fields after batch(if batch size is 2):  
  target: (1)type:torch.Tensor (2)dtype:torch.int64, (3)shape:torch.Size([2])  
  
training epochs started 2019-05-28-03-21-05  
loss:1.39258  
loss:1.37082  
loss:1.37476  
loss:1.40225  
loss:1.34815  
loss:1.34967  
loss:1.30316  
loss:1.32293  
loss:1.30326  
loss:1.26853  
loss:1.25940  
Traceback (most recent call last):  
  File "run.py", line 223, in <module>  
    run()  
  File "run.py", line 221, in run  
    train(args)  
  File "run.py", line 183, in train  
    trainer.train()  
  File "/remote-home/competition/anaconda3/lib/python3.7/site-packages/fastNLP/core/trainer.py", line 536, in train  
    self.test._format_eval_results(self.best_dev_perf, )  
  File "/remote-home/competition/anaconda3/lib/python3.7/site-packages/fastNLP/core/tester.py", line 182, in _format_eval_results  
    for metric_name, metric_result in results.items():  
AttributeError: 'NoneType' object has no attribute 'items'  
(base) competition@fnlp-server-112: /cck/PRML/assignment4/16307130138/src$ python run.py --train --model LSTMText --opti  
[cck] 0: [tmux]* fnlp-server-112 03:28 28-May-19
```

再一个我不知道是不是fastnlp导致的bug，官方文档表明 `EmbedLoader` 可以自动区分glove和word2vec的形式进行加载，看了源码似乎也确实是这样，但是我直接加载glove总是出问题（就是前面所提到的问题），经过n长时间的努力，还是没有de出来，直接使用gensim的转换工具进行转换了。

Other problems

除了这个bug，还有一些其他使用过程中的问题：

1. 文档还是不太全，部分地方还没有完成（我看到了TODO）。
2. 没有社区，有bug没地儿讨论。
3. 很多模型单单靠文档还是不知道怎么使用，建议在文档中添加更多实例。
4. fitlog 功能还比较单一，文档很不全。在网站上的log名字比较乱，如果可以指定一个名字就好了。

5. fitlog记录hyper parameter不太方便。因为我的程序结构使用了parser，所以至今没想到一个合适的方法把hyper parameter加进去，导致找模型还是靠人脑记录顺序。

Suggestions

1. 在文档中添加更多实例，建议放出一些fastnlp的Test
2. 完善以下fitlog的文档，感觉fitlog如果能和一些比较流行（或统一）的神经网络代码的架构（组织代码的习惯）结合起来就好了，这样大家使用的时候就更方便了。
3. fitlog记录的曲线好像不方便作比较，可以考虑在同一张图里画多个曲线，即加入一个曲线整合比较功能。