Assignment4 Report

Part I

1. 数据处理

本次实验的数据处理使用 fastnlp 提供的 dataset 和 vocabulary 完成

⇒ 调用 fetch_20newsgroups 函数获取数据集

⇒ 构建 train 和 test 数据集的 dic，从而获取 fastnlp 的 dataset

⇒ 利用 dataset 提供的 apply 以及 apply_field 函数进行数据简单清洗

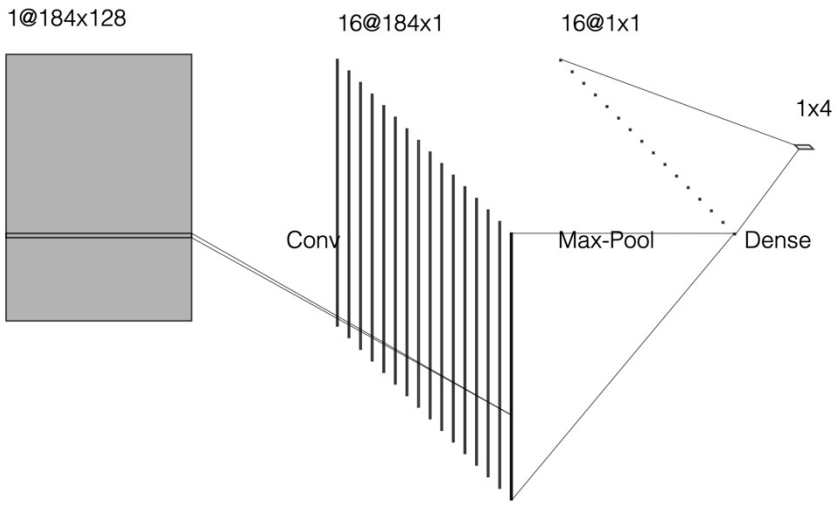
⇒ 切分获得 train_set 之后，构建 vocabulary，并对所有数据进行 index 处理

⇒ 对 input、target 进行 rename_field

2. CNN

2.1. 模型

本模型参照要求中提供的论文（CNN: Convolutional Neural Networks for Sentence Classification）用下图和文字简单示意（为了图片效果 channel 数比真实数目少）



◊ 对文章用长度不一，宽度为 embedding 的 kernel 进行卷积

◊ 对卷积结果进行 relu 和 dropout 操作

◊ 对文章在 seq 维度进行 maxpooling 获得长度为 channel 的向量

◊ 经过 fc 和 cross entropy 得到 loss

2.2. 训练结果

```
In Epoch:7/Step:448, got best dev performance:AccuracyMetric: acc=0.950893
Reloaded the best model.
[tester]
AccuracyMetric: acc=0.925134
```

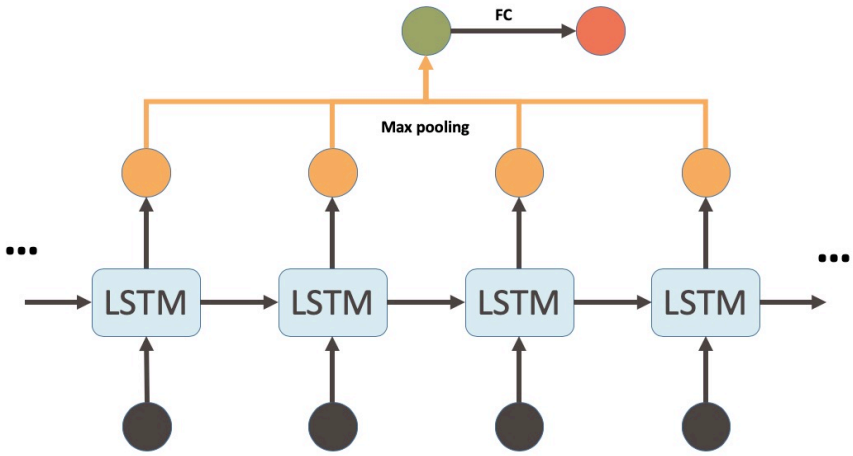
2.3. 实现细节

为了方便的实现多个同样位置但是不用参数的卷积，尝试使用了 pytorch 提供的 nn.ModuleList。具体见代码

3. LSTM 以及 BiLSTM

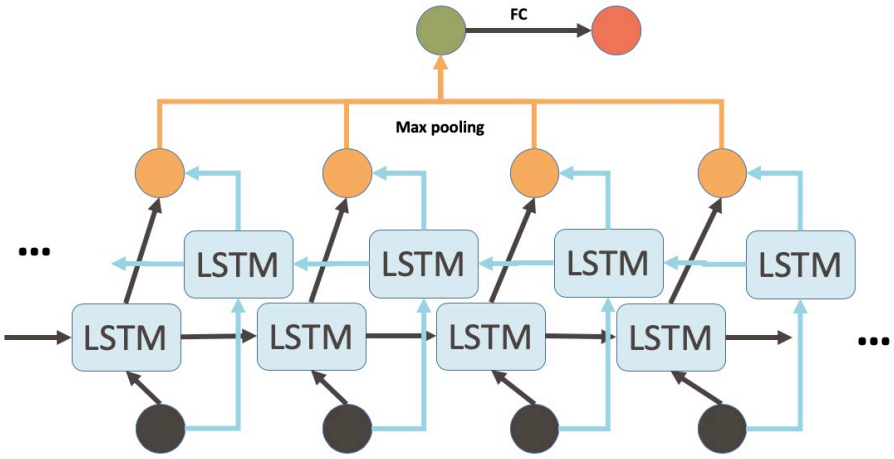
3.1. 模型

本模型同样参考论文（RNN: Recurrent Convolutional Neural Networks for Text Classification）



⇒ 将文章的 tokens 依次输入到 LSTM 中
⇒ 将所有位置上的 hidden 层取出，在 seq 维度进行 maxpooling
⇒ 对获得的一维向量进行 fc 和 cross entropy 获得 loss

因为 pytorch 已经进行了实现，所以很容易将代码变为双向 lstm，其结构如下



3.2. 训练结果

LSTM

```
In Epoch:7/Step:448, got best dev performance:AccuracyMetric: acc=0.964286
Reloaded the best model.
[tester]
AccuracyMetric: acc=0.941176
```

BiLSTM

```
In Epoch:3/Step:192, got best dev performance:AccuracyMetric: acc=0.96875
Reloaded the best model.
[tester]
AccuracyMetric: acc=0.927807
```

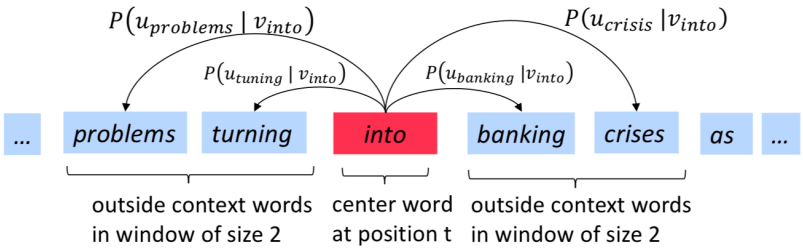
3.3. 实现细节

论文中提出最简单的方法为将最后一个 hidden 取出，获得相应的结果。但是在经过实践后发现，仅使用最后一个 hidden 的信息进行预测结果并不理想，**因此借鉴 cnn 中 maxpool 的方式将所有 hidden 都进行利用，效果较佳。**

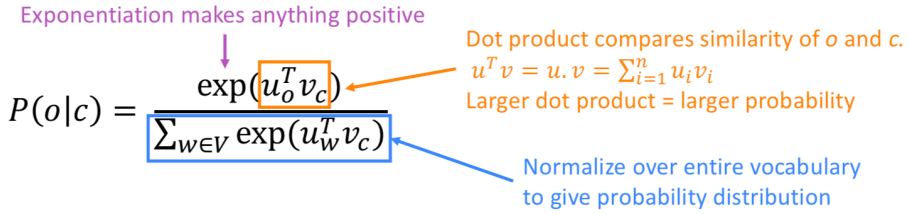
4. Pretrained Embedding

4.1. Word2vec 初识

◊ 基于 Skip-grams (SG)的方法，word2vec 训练的思路是给定 center word 之后，预测 context word，如下图使其条件概率最大。



◊ 借助 softmax 的思想，将 embedding 转换为该条件概率，并利用 SGD 的方式进行训练，获得每个词汇在文章中合适的 embedding



◊ Word2vec 能够较快速的获得较为合适的 embedding，但对于上层网络结构而言，并不一定是最合适的结果，**所以在实现时导入 embedding 之后依然继续进行训练。**

4.2. Word2vec 实现

⇒ 利用之前的 vocabulary 将 index 还原为带有 unk 的文章

⇒ 将文章作为语料放入 word2vec，构建 model

⇒ 对 model 进行训练，并获取 vocabulary 对应的 embedding weight

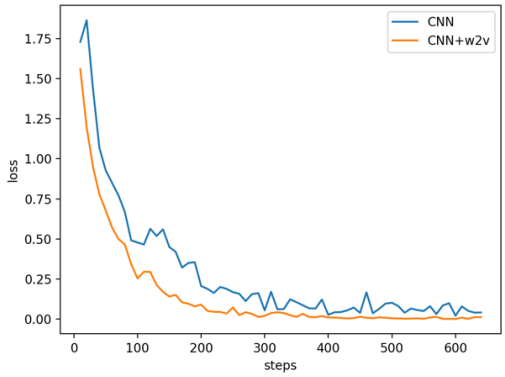
4.3. 实验探究

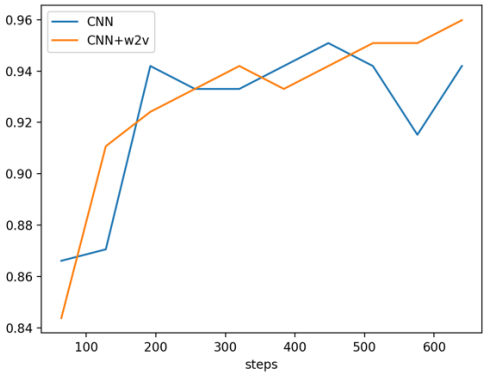
为了更好的体验 word2vec 的魅力，继续在之前三个模型中添加了 word2vec 预训练的 embedding，并与之前的 loss 和 accuracy 进行对比。

下图为所有测试结果，**添加 word2vec 之后的准确率都有了 1%到 2%不等的提升，而且可以看出，添加 word2vec 之后的 loss 特别在起始时都比较小，可以看出 word2vec 为模型的 embedding 提供了一个良好的初始参数空间。**

CNN+word2vec

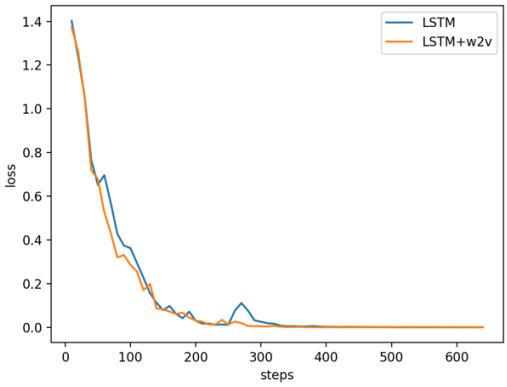
```
In Epoch:10/Step:640, got best dev performance:AccuracyMetric: acc=0.959821
Reloaded the best model.
[tester]
AccuracyMetric: acc=0.933155
```

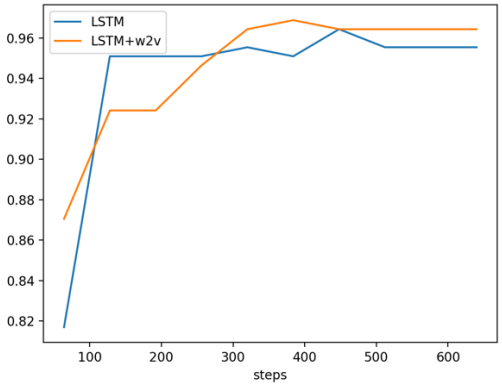




LSTM+ word2vec

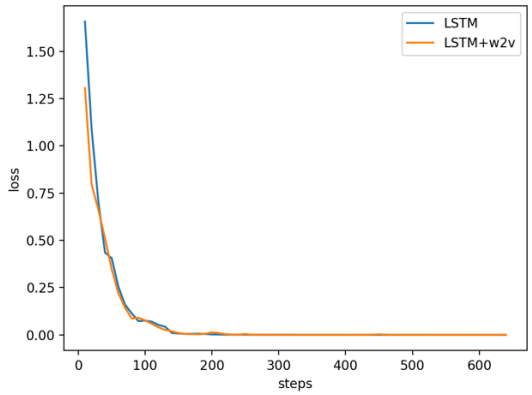
```
In Epoch:6/Step:384, got best dev performance:AccuracyMetric: acc=0.96875
Reloaded the best model.
[tester]
AccuracyMetric: acc=0.963904
```

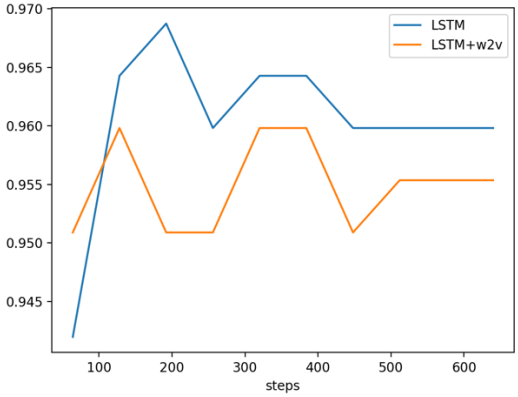




BiLSTM+ word2vec

```
In Epoch:2/Step:128, got best dev performance:AccuracyMetric: acc=0.959821
Reloaded the best model.
[tester]
AccuracyMetric: acc=0.951872
```





5. 全数据集测试

将以上三种模型以及 word2vec 在大数据集上进行了训练和测试，以下为最终结果

CNN+word2vec

```
In Epoch:5/Step:1480, got best dev performance:AccuracyMetric: acc=0.899204
Reloaded the best model.
[tester]
AccuracyMetric: acc=0.794875
```

LSTM+word2vec

```
In Epoch:5/Step:1290, got best dev performance:AccuracyMetric: acc=0.925729
Reloaded the best model.
[tester]
AccuracyMetric: acc=0.840016
```

BiLSTM+word2vec

```
In Epoch:10/Step:2950, got best dev performance:AccuracyMetric: acc=0.931034
Reloaded the best model.
[tester]
AccuracyMetric: acc=0.859267
```

Part II

对 fastnlp 的建议

Fastnlp 为自然语言处理任务提供了很多高层的工具封装 vocabulary，dataset，trainer 等，能够在 tutorial 的指导下很快速地完成一个 nlp 经典任务的全部流程，为 nlp 初学者提供了一个方便、全面的指导。

但是当使用者想要进行更加灵活和花样的探究时，受到的限制就较多，以下为我在使用 fastnlp 的一些思考与个人建议。

- ◊ 参数控制与获取。Trainer 在提供高层的封装时，让用户对很多的参数控制和获取受到了限制，例如无法分别设置 train batch 和 test batch 的大小，获取 loss_his，acc_his 等，以及无法获得训练过程中模型参数等状态，让一些探究实验、模型监控等变得稍微不便，希望能提供更加强大而灵活的接口
- ◊ 变量名问题。Fastnlp 几乎不需要我们手动控制数据在训练过程中的行为，但是变量名命名问题让我感到有些脑阔疼。当我在 dataset 将 input field 的名字进行修改后，我需要在 metric、loss 等等地方也需要考虑到 field name 的问题。虽然 fastnlp 已经提供了 Const 来提供统一的命名方式，但是如果能通过一个全局 dict 来进行配置的话将会是一劳永逸的行为，对于代码书写可能会更舒适灵活一些。
- ◊ 希望可以实现一些 word2vec、Glove 等一些 nlp 常用的算法。
- ◊ 有两个小细节想要吐槽一下，一个 trainer 对保存 model 的行为规定的太死，如果能实现 top k 的 model 保存行为应该会为模型试验提供更多的可能性；另外一个是代码的细节行为不太一致，比如说我在使用 vocabulary 的时候，想象中的使用场景是

```
vocab = Vocabulary(min_freq=10, padding="</s>").add_word_lst(wordlist).build_vocab()
```

而实际的使用场景是（有些函数没有返回左值）

```
vocab = Vocabulary(min_freq=10, padding="</s>")
vocab.add_word_lst(wordlist)
vocab.build_vocab()
```

最后祝 fastnlp 成为中文 nlp 框架的领头人！