pytorch word2vec+LSTM 文本分类

SA22011050 吕瑞

环境准备

requirements:

```
python == 3.8.8
pytorch == 1.8.0
torchtext == 0.9.0
# pip install torchtext==0.9.0
```

数据集准备

prepare dataset: 本次实验统一使用指定的 IMDB 公开数据集"Large Movie Review Dataset"。 该数据集分别包含 25,000 条电影评论作为训练集和测试集。

```
mkdir dataset
cd dataset
wget https://ai.stanford.edu/~amaas/data/sentiment/aclImdb_v1.tar.gz
tar -zxvf aclImdb_v1.tar.gz

# catalogue
dataset
    aclImdb
        test
        train
            neg
            pos
```

数据预处理

prepare_data.py

遍历文件夹读取数据：

```
def read_imdb(path='../dataset/aclImdb', is_train=True):
    ...
```

清洗数据集并分词，构建数据集：

```
def load_imdb(path="../dataset/aclImdb",data_path="../dataset",train_set_path =
"../dataset/train_set_word2vec.txt"):
    print('load imdb dataset ...')
    reviews, labels = read_imdb(path,True)
    ...
```

利用 word2vec 模型预训练训练集的词向量

```
def pretrain_dataset(path="../dataset/aclImdb",train_set_path =
"../dataset/train_set_word2vec.txt",w2v_path="../model/word2vec"):
    print('pretrain dataset ... ')
    sentences = word2vec.LineSentence(train_set_path)
    # train vector model, set word vector size=200,  train model is skip-gram, and
save to .bin
    model = gensim.models.Word2Vec(sentences, vector_size=200, sg=1)
    model.wv.save_word2vec_format(w2v_path+".txt", binary=False)
    print('pretrain done! Word2vec model have saved in '+ w2v_path+ '.txt'+' as
type of txt.')
```

语言模型

model.py

利用 torch 自带模型库，构建 LSTM 模型：

基于训练好的语言模型（固定参数），编写一个情感分类模型，包含一个 LSTM 模型和一个分类器 MLP。 首先，将一个句子中的每个单词对应的词向量输入 LSTM，得到句子的向量表征。然后将句向量作为分类器的输入，输出二元分类预测，同样进行 loss 计算和反向梯度传播训练，这里的 loss 是分类 loss，交叉熵 loss。

```
# Define model
class LSTM(nn.Module):
    def __init__(self, vocab_size, pad_idx, embed_size = 200,
batch_first=False,hidden_size = 128, num_layers=1,bidirectional=False,
dropout=0.4,labels=2):

        super().__init__()
        self.embedding =
nn.Embedding(num_embeddings=vocab_size,embedding_dim=embed_size,padding_idx=pad_id
x)
        self.encoder = nn.LSTM(input_size=embed_size,batch_first=batch_first,
hidden_size=hidden_size,num_layers=num_layers,bidirectional=bidirectional,
dropout=dropout)
        if bidirectional:
            num_directions = 2
        else:
            num_directions = 1

        self.decoder = nn.Linear(num_directions * hidden_size, labels)
```

```
        self.dropout = nn.Dropout(dropout)

    def forward(self, x):

        x = self.embedding(x)

        encoder_out, (_,_) = self.encoder(x)
        """
        encoder_output shape : [seq_length, batch_size, num_directions *
hidden_size]
        """

        decoder_out = self.decoder(encoder_out[:,-1,:])
        """
        we choice the last step of lstm output as the sentence representation.
        """
        output = self.dropout(decoder_out)
        """
        use the dropout for the full-connnected layer output
        """

        return output
```

测试性能

```
# Set parameters
batch_size = 1024 # batchsize: 16,64,128,256
hiddim_list = [128,256] # hidden_dimension of LSTM: 128,256
deepth_list = [1, 2] # one-layer LSTM or two-layer LSTM: 1, 2
lr_list = [1e-3,1e-4]
epochs = 100
bidirectional = False # unidirectional LSTM or bidirectional LSTM: True, False
```

```
-------------------------------------------------------
numlayers1_hiddensize128_lr0.001
Avg valid loss 0.438995, Accuracy: 83.1%
Training done!
Avg test loss 0.275545, Accuracy: 89.1%

-------------------------------------------------------
numlayers1_hiddensize128_lr0.0001
Avg valid loss 0.410931, Accuracy: 81.2%
Training done!
Avg test loss 0.396939, Accuracy: 81.8%

-------------------------------------------------------
numlayers1_hiddensize256_lr0.001
Avg valid loss 1.118040, Accuracy: 79.2%
```

```
Training done!
Avg test loss 0.351807, Accuracy: 93.4%

----------------------------------------------------
numlayers1_hiddensize256_lr0.0001
Avg valid loss 0.416183, Accuracy: 80.4%
Training done!
Avg test loss 0.405870, Accuracy: 81.0%

----------------------------------------------------
numlayers2_hiddensize128_lr0.001
Avg valid loss 0.568931, Accuracy: 82.7%
Training done!
Avg test loss 0.262185, Accuracy: 91.4%

----------------------------------------------------
numlayers2_hiddensize128_lr0.0001
Avg valid loss 0.405948, Accuracy: 81.1%
Training done!
Avg test loss 0.393842, Accuracy: 82.0%

----------------------------------------------------
numlayers2_hiddensize256_lr0.001
Avg valid loss 0.394764, Accuracy: 81.6%
Training done!
Avg test loss 0.336288, Accuracy: 85.1%

----------------------------------------------------
numlayers2_hiddensize256_lr0.0001
Avg valid loss 0.404977, Accuracy: 81.8%
Training done!
Avg test loss 0.384591, Accuracy: 82.8%
```

最优参数:

```
numlayers = 1
hiddensize = 256
lr = 0.001
Avg valid loss 1.118040, Accuracy: 79.2%
Avg test loss 0.351807, Accuracy: 93.4%
```