实验主题： 文本情感分析

# 实验目的

基于从IMDb用户影评中收集的真实语料，使用torchText搭建RNN模型，实现文本情感分析，即模型能够根据输入的电影评论语料判断其是正面还是负面评论。学生在将在基础实验中掌握数据预处理、RNN模型搭建以及训练的基础框架搭建，并在拓展实验中尝试基于LSTM模型优化细节，提升训练结果。

# 实验环境与数据准备

## 实验环境

* + 1. 硬件环境

PC机一台

* + 1. 软件环境

Python 3.7; PyTorch 1.5; torchtext 0.6; spacy

## 数据准备

* + 1. 数据介绍

IMDb数据集包含来自互联网的50000条严重两极分化的评论，该数据被分为用于训练的25000条评论和用于测试的25000条评论，训练集和测试集都包含50%的正面（Positive）评价和50%的负面（Negative）评价。

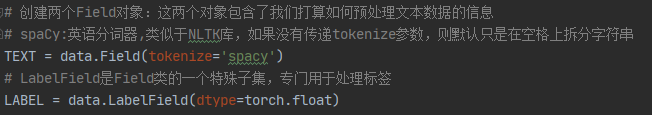
# 实验操作步骤（占60%）

## 相关准备

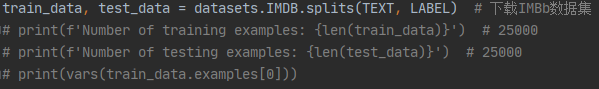
下载并解压工程：<https://github.com/AuMgLi/exp3_sentiment_analysis>

## 数据下载及预处理

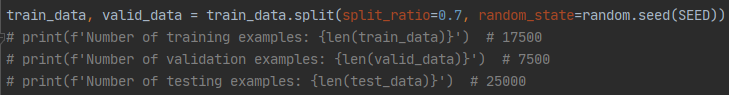
在torchText中的一个重要概念是Field，Field决定了你的数据会被怎样处理。首先，创建两个Field 对象：这两个对象包含了我们打算如何预处理文本数据的信息。



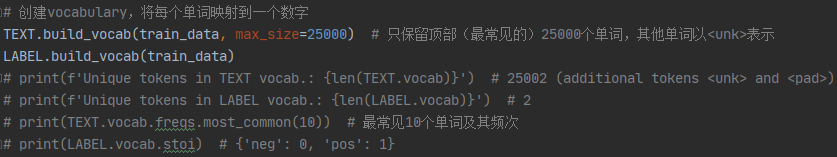
然后下载IMDb数据集，数据被前面的Fields处理。IMDb数据集一共有50000电影评论，每个评论都被标注为正面（“pos”）的或负面(“neg”)的。



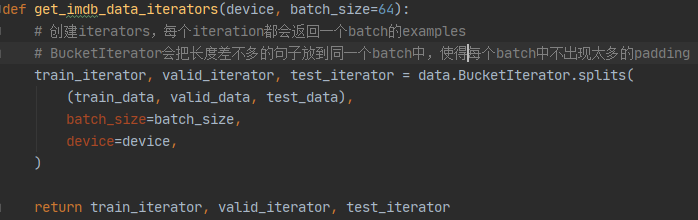
将训练集中的数据按照7:3的比例划分为训练集和验证集。



由于模型无法理解英文单词，需要创建vocabulary将每个单词一一映射到一个数字。我们使用最常见的25000个单词来构建我们的单词表，所有其他的单词都用<unk>来表示。词典中应当由25002个单词（len(TEXT.vocab)），额外的2个单词除了<unk>外还有一个<pad>，用来将一个batch中每个句子填充到该batch中最长句子的长度。



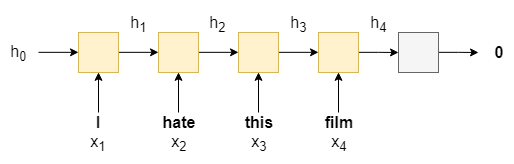
创建iterators，每个iteration都会返回一个batch的examples。我们会使用BucketIterator，它将长度差不多的句子放到同一个batch中，使得每个batch中不出现太多的<pad>。



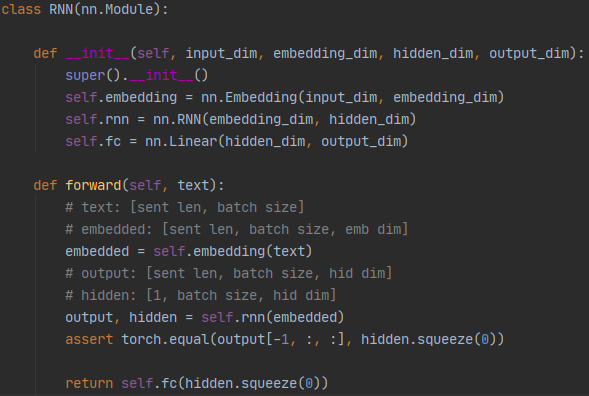
## 搭建模型

我们使用循环神经网络（RNN），其通常用于序列分析。RNN一次接收一组单词序列，并为每个单词产生一个隐状态（hidden state），并结合上一个单词的隐状态来产生下一个隐状态。

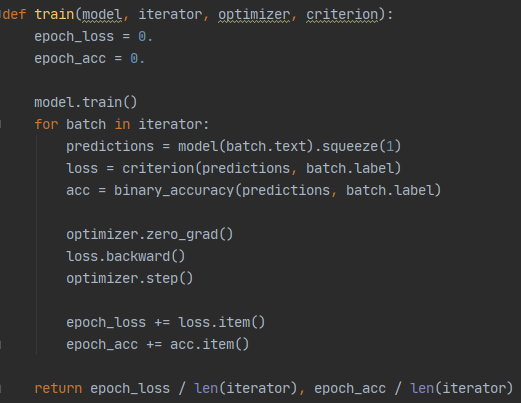
经过数层RNN，最终隐状态将经过一层全连接层以得到最后的预测值，如下图所示。



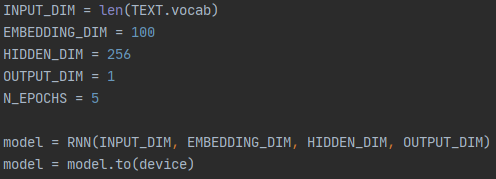
下面在PyTorch中搭建模型，主要分为嵌入层（embedding），RNN和全连接层。嵌入层用于将稀疏的one-hot向量转换为致密的嵌入向量，来减少输入到RNN的维数，并使得对评论的情感具有相似影响的单词更紧密地映射在一起。



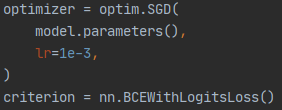
## 训练模型



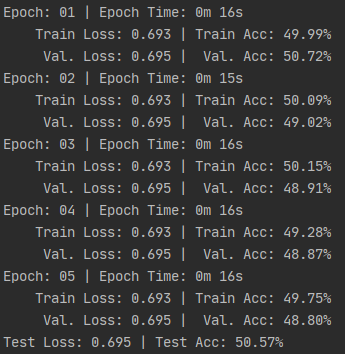
创建模型



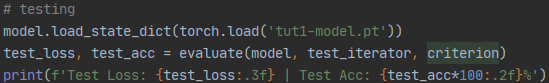
定义优化器和损失函数



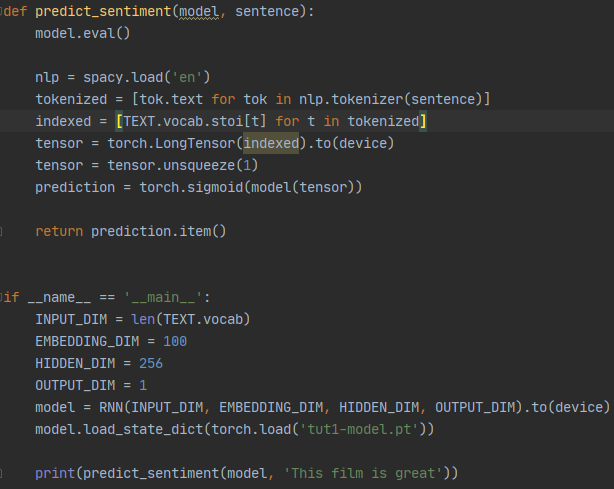
运行train.py开始训练



计算测试集精度



在predict.py中对自定义的输入语句进行预测，输出值越接近1表示正面评价，越接近0表示负面评价





# 拓展实验（占40%）

在上述实验步骤中我们基于标准RNN搭建了一个简单的情感分析任务框架，并使它成功运行起来，但是不难发现其训练效果并不够理想，接下来请在上述实验的基础上优化各处细节，包括使用GloVe算法计算词向量、搭建一个LSTM模型进行训练、使用Droupout避免过拟合现象等，具体实现可参考<https://github.com/bentrevett/pytorch-sentiment-analysis/blob/master/2%20-%20Upgraded%20Sentiment%20Analysis.ipynb>，并记录不同层数的LSTM和不同比例的Dropout对实验结果带来的影响。