## 推荐系统第二次作业——基于内容的推荐

### 算法简介

#### **简介**

在上次作业中，基于UID-IID-评分的关系构建了基于用户的协同过滤推荐系统，但是这个推荐系统效果仍可提升，因为在上次的推荐系统中，并未考虑用户的评论。在本次作业中，我将用户的评论用词向量的序列表示，用word2vec算法构建词向量，然后首先用神经网络学习用户的评论与评分之间的关系，再构建一个线性回归模型，将基于内容的评分与协同过滤的评分进行结合，得到最后的结果。

#### **实现过程描述**

首先根据上一次实验的结果，确定最优的协同过滤算法，即为k=5情况下以余弦相似度为相似函数。在确定协同过滤部分最优化以后，开始构建文本-得分模型。首先对所有英文文本用NLTK自然语言处理库进行分词、去除停用词以及词型变化还原，并统一大小写。然后用这些预处理后的英文文本训练word2vec模型，得到将单词向量化的方法。

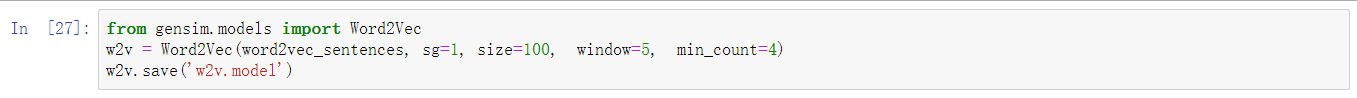
### 核心代码注解

#### **2.1 文本预处理与word2vec训练语料库准备**



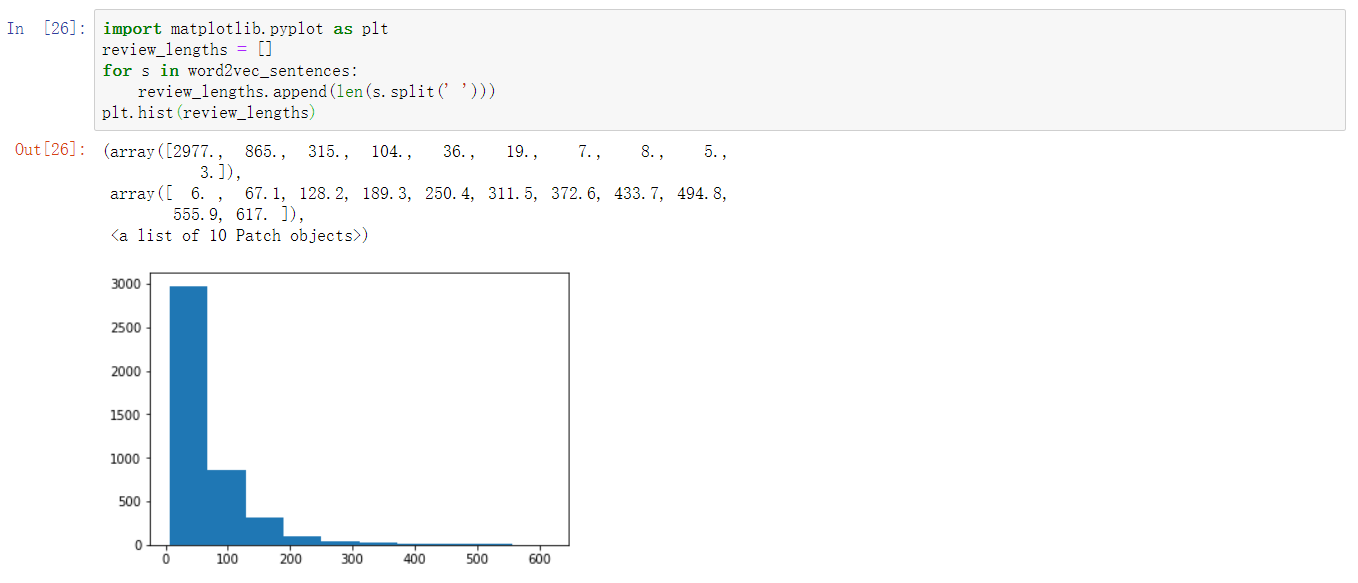
对英文文本进行分词、去除停用词、词型还原后，用处理后的review特征代替原来的review特征，并用这些句子构成语料库，准备训练word2vec。保存处理过的数据集以便以后使用。

#### **2.2 训练word2vec**



训练word2vec模型，考虑到语料库较小，采用skip-gram而非cbow算法，并调低低频词的判断标准，增强对于低频词的处理能力。训练后将模型保存，可以发现得到的模型并不大。

#### **2.3 训练神经网络**



首先构造训练数据，考虑评论之间长短不一，先规定一个统一的评论长度。用matplotlib绘制频次分布直方图，可以发现长度在200词以上的评论较少，因此可统一截短/增长长度到200词。

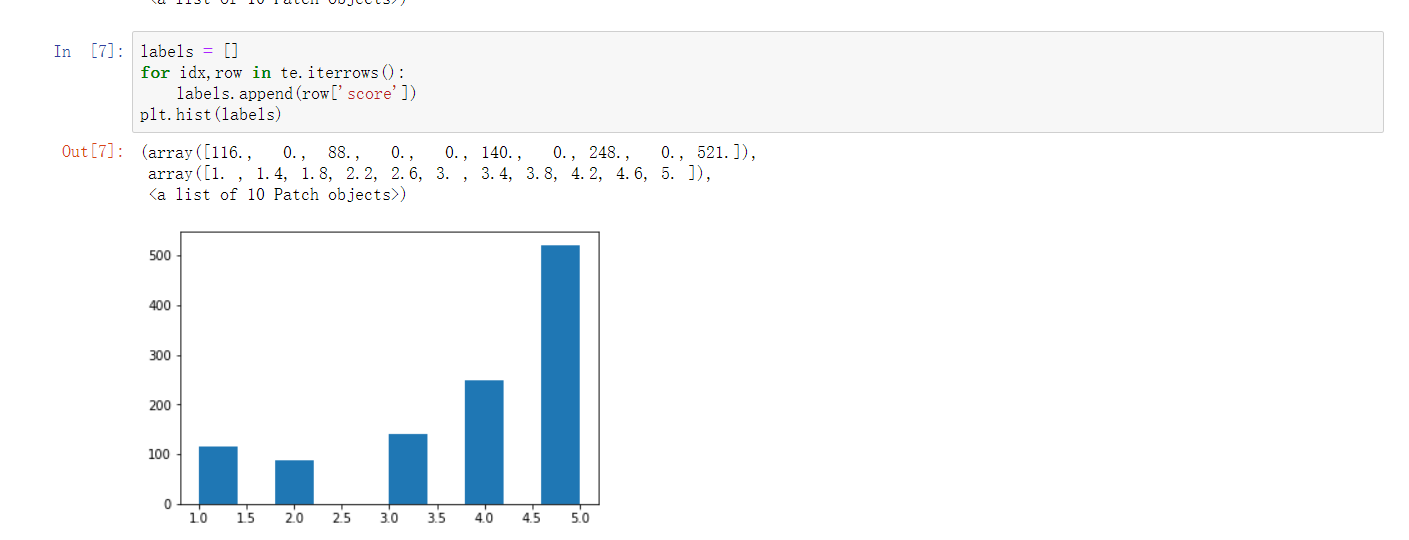


规定了统一的评论长度以后就可以构造训练数据了，特征为大小为词向量长度\*评论长度的矩阵，标签为得分，对于长度不够的补0，对于长度超出的直接截断，对于不存在模型中的词补0。对于五种不同的评分，转化为one-hot向量，这样预测评分就变成了一个分类问题。把数据都转化成numpy.array的形式，有利于keras进行下一步运用。



使用如图所示的网络，与LeNet-5近似，训练30个epoch观察训练过程，考虑是否进行早停。验证集Loss持续上升的问题应该是由于模型对于结果过于“自信”导致的，和训练样本过小有关系。经过多次测试，发现batch\_size选择16，收敛速度较为合适，选择epochs为30，进行模型训练。

#### **2.4 测试神经网络**

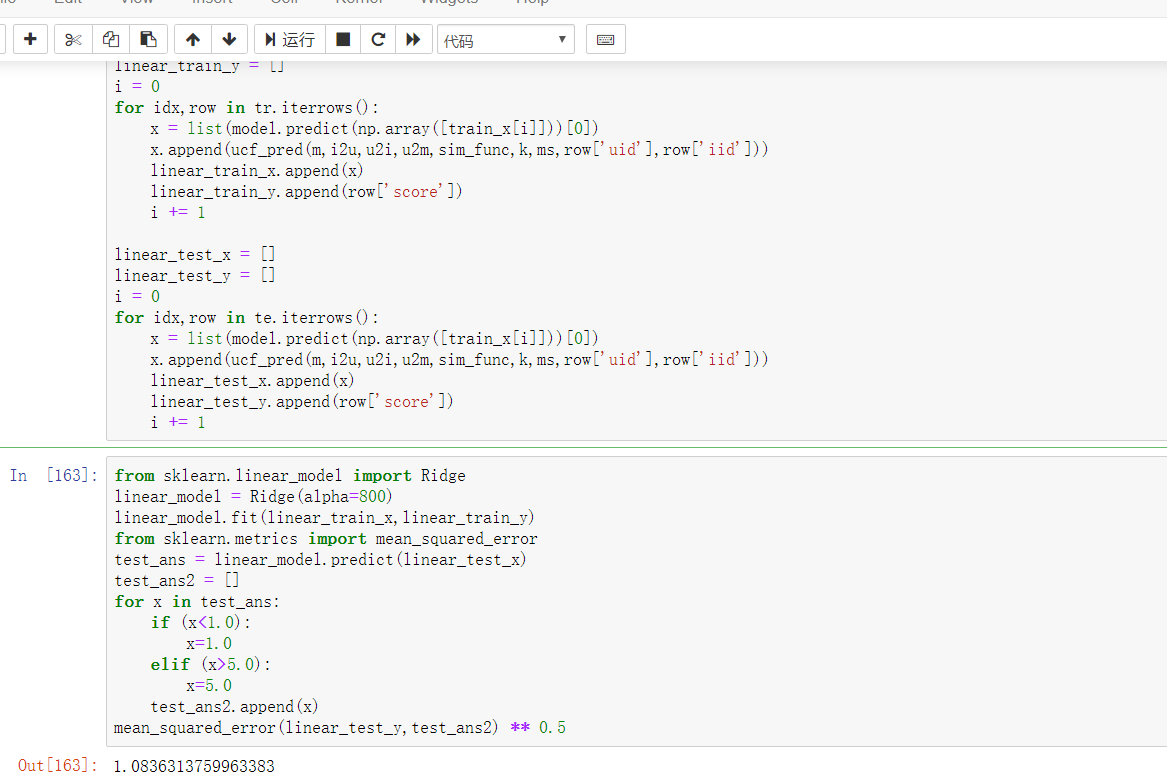


用测试集对神经网络进行测试。首先画出测试集中标签的频率分布直方图，发现测试集中的众数占比约0.521，也就是说，训练的模型其精度必须超过0.521，否则比直接猜测众数效果要差，明显是不能解决实际问题的。



比直接猜测众数上升了10%左右，模型有一定效果，下一步把模型和原来的模型用线性回归集成后再使用。

#### **2.5 线性回归集成**



以神经网络预测的五个类的分类以及协同过滤的评分为特征，考虑到特征之间可能存在多重共线性问题，故采用岭回归，得到最终的模型，可以发现集成后的岭回归模型在验证集上的表现比原先要好。

#### **2.6 填写答案**



仿照上文的代码，将岭回归模型的结果输出到my\_answer.csv中。

### 结果分析

集成后的岭回归模型在验证集上的表现优于原模型10%左右。

### 代码运行

#### **4.1 环境**

Anaconda 2.6.0

Python 3.7.1

#### **4.2 依赖**

除运行实验1所需要的的依赖外，新增依赖如下：

Keras 2.2.4

Tensorflow 1.13.1

NLTK 3.4

gensim 3.8.3

#### **4.3 命令行命令**

命令行运行jupyter notebook 后打开本文件

### 提交文件

RS\_HW2.ipynb 基于实验1完成的本次作业

my\_answer.csv 我的预测答案

train.csv test.csv pred.csv 数据集