Central University of Finance and Economics

中央财经大学

非结构化大数据分析

基于机器学习的《哈利波特》评论情感分类

吴宇翀

2021210793

统计与数学学院

15910852867

EMAIL@WUYUCHONG.COM

指导老师：刘苗

2022 年 3 月 10 日

摘要

通过网络爬虫的方式获取《哈利波特》全系列图书的评论进行文本分析，我们使用主题模型进行对评论进行文本挖掘，之后进行文本情感分类模型的训练。在文本预处理阶段，我们尝试使用词编码和词向量的方式，在训练阶段，我们构建了 DNN、LSTM、BERT 等多个深度学习模型进行训练，并进行了模型比较，最终达到了 88% 的准确率。最后，为了进一步实现在超大文本集上进行训练，我们使用基于 Spark 的分布式算法在集群服务器上进行训练测试。[1]

模型	计算配置	用时	准确率	可拓展性
tokenize + DNN	阿里云服务器 Xeon 8 核 CPU 32G 内存	3 分钟	55%	低-单机
Word2Vec + LSTM	阿里云服务器 Xeon 8 核 CPU 32G 内存	1 小时	70%	低-单机
bert - 小型	阿里云服务器 Xeon 8 核 CPU 32G 内存	24 分钟	78%	低-单机
bert - AL	阿里云服务器 Xeon 8 核 CPU 32G 内存	1.6 小时	82%	低-单机
bert - 标准	阿里云服务器 Xeon 8 核 CPU 32G 内存	1 小时	88%	低-单机
spark - logit	中央财经大学大数据高性能分布式集群	4 分钟	73%	高-集群
spark - 决策树	中央财经大学大数据高性能分布式集群	10 分钟	80%	高-集群
spark - 梯度助推树	中央财经大学大数据高性能分布式集群	15 分钟	82%	高-集群
spark - 随机森林	中央财经大学大数据高性能分布式集群	30 分钟	87%	高-集群

[1] 分布式模型在该小型数据集上没有优势，进行此项的意义在于对大型文本数据集可拓展性的技术储备，仅有在文本量级超过单机可承载上限时，分布式计算才具备意义

目录

1 数据爬取

我们使用 python 对豆瓣上《哈利波特》全系列图书的评论进行爬取。由于豆瓣的评论机制要求读者撰写评论时必须同时填写评价级别，因此我们可以同时获取到评价文本及其对应的情感级别。豆瓣将评价级别分为三类：好、坏、中等，这有利于我们进行通过文本的内容预测评价级别的有监督机器学习训练。

在爬取过程中，我们使用已登录的 cookie 信息和浏览器信息提交请求，并在每两次请求的间隔加入随机时间的休眠，以尽可能地模拟真实人的操作，避免 IP 被官方查封。

评论	评分
每翻一页便是与你的告别。我不想同你...	good
几年后重新看了最后的几章，在这热得...	good
所有爱唱反调的胖友们，你们都没试过...	good
基本上一如预想。罗琳笔下的主要角色...	good
别的还好，双胞胎死一个活一个实在太...	medium
这本稀碎情节太多，完全都是没必要的...	medium
反正总是要看完的嘛！	medium
罗琳忽悠了全球读者	medium
对不起，我实在是无法接受这个结局。	bad
从哈一到哈七。。。水平依次下降。。...	bad

图 1: 词云图

2 文本预处理

我们使用 jieba 对中文进行分词处理。

1. 去除标签
   - 将一些网页 HTML 特有的标签进行去除，如段落标记、换行标记 `p` `br` 等
2. 去除标点符号
   - 将常用标点符号进行去除，如 ！ ； 等
3. 去除多余的空格
   - 删除无意义的连续性空格
4. 去除数字
   - 由于数字对文本情感识别作用小，我们选择将其删去
5. 去除停用词
   - 对意义较小的常用词进行删除
6. 去除过短的词汇
   - 由于英文中过短的字符一般意义较小，我们选择将其删去
7. 大小写统一
   - 大小写代表同一词汇，需要进行统一

表 3: 关键词提取

text	rating	clean_text	label
对不起，我实在是无法接…	bad	对不起	1.0
从哈一到哈七。。。水平 …	bad	从哈一到免不了	1.0
这套书根本就是在宣扬扭曲…	bad	根本就是价值观	1.0
翻译太烂。和上一本一样…	bad	比不上	1.0
糟透了	bad	糟透了	1.0

3 主题模型

3.1 主题模型的构建

主题模型是一个相对泛化的概念，从实现了文本数据的结构化的文本模型都能称为主题模型，狭义上一般代指基于隐式 Dirichlet 先验概率分布模型。

基于主题模型，每一篇文档都是一个主题的多项分布，文档集合事实上可以看作是一个天然的软聚类过程，各个主题就是聚类中心，文档在各个主题上的概率就是它与这个聚类中心的距离。同时，主题模型可以得到各主题下词汇的概率分布，主题-词汇概率矩阵旋转后就可以得到词汇在各个主题的概率分布情况，得到词汇的软聚类结果。

LDA 模型通过将生成的随机文档与真实文档进行相似度对比来决定狄利克雷先验分布的参数，从而得到最优参数的模型。

3.2 结果分析

我们构建一个 3 个主题的主题模型，并绘制解释图，图中每个气泡代表一个主题，越大的气泡代表该主题涵盖的评论数越多。蓝色柱子代表特定词汇在整个语料库中的词频，红色柱子代表特定词汇在特定话题中的词频。

3.2.1 主题一

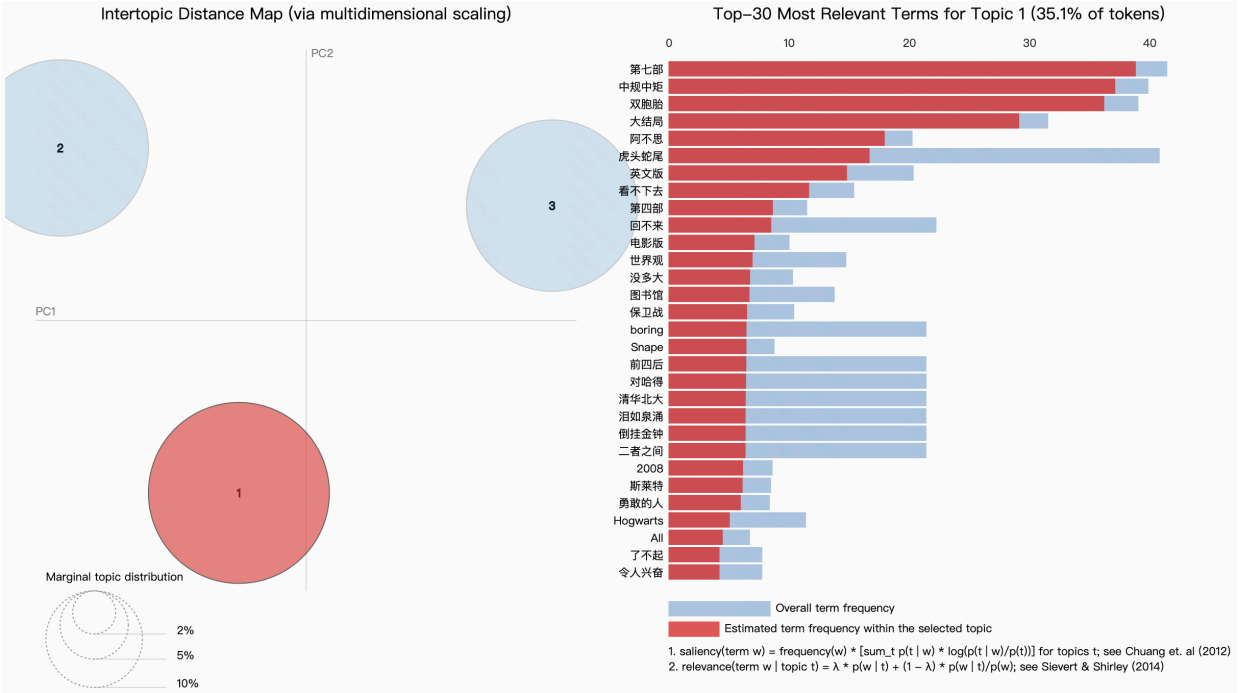

图 2: 主题一成分词汇权重图

第一个话题主要是对大结局的评论。图书馆保卫战、第七部、虎头蛇尾都是对结局的讨论。

3.2.2 主题二

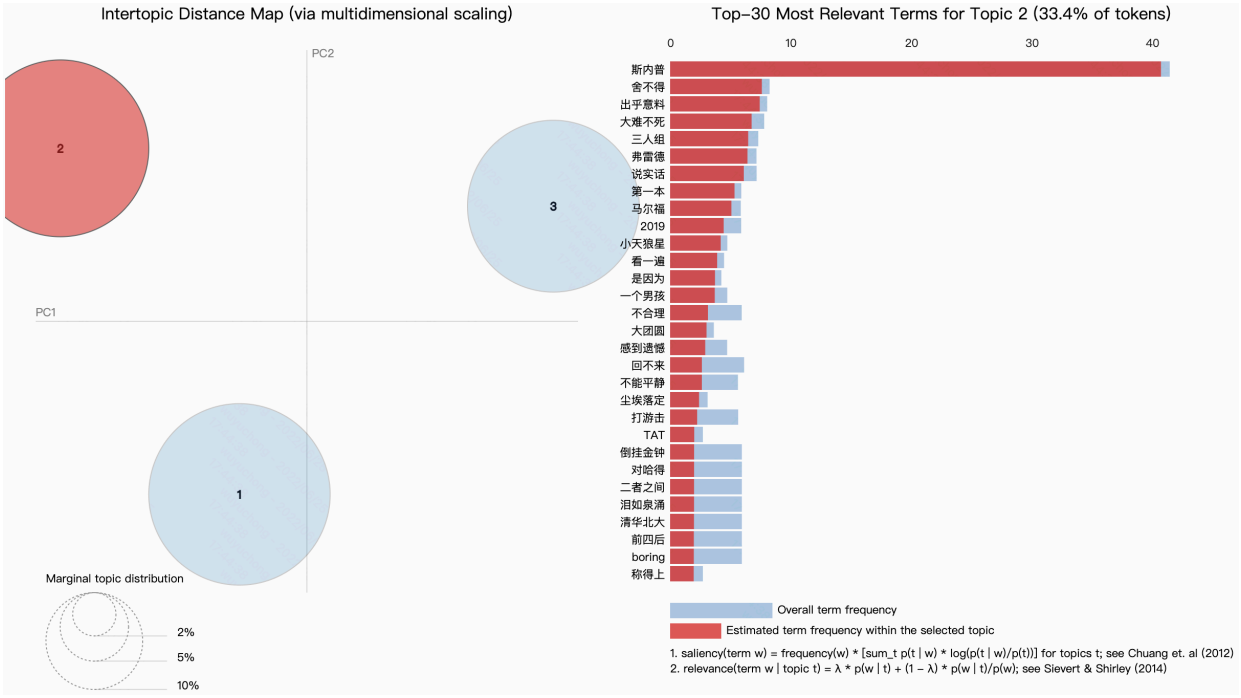

图 3: 主题二成分词汇权重图

第二个话题主是对书中各类较为主要的人物进行讨论。排名第一的词汇斯内普是书中以为颇具争议的反派人物，其性格方面的多重性给了读者非常大的讨论空间，弗雷德、小天狼星也都是书中较为典型的人物，他们与哈利波特具有密切地联系，

3.2.3 主题三

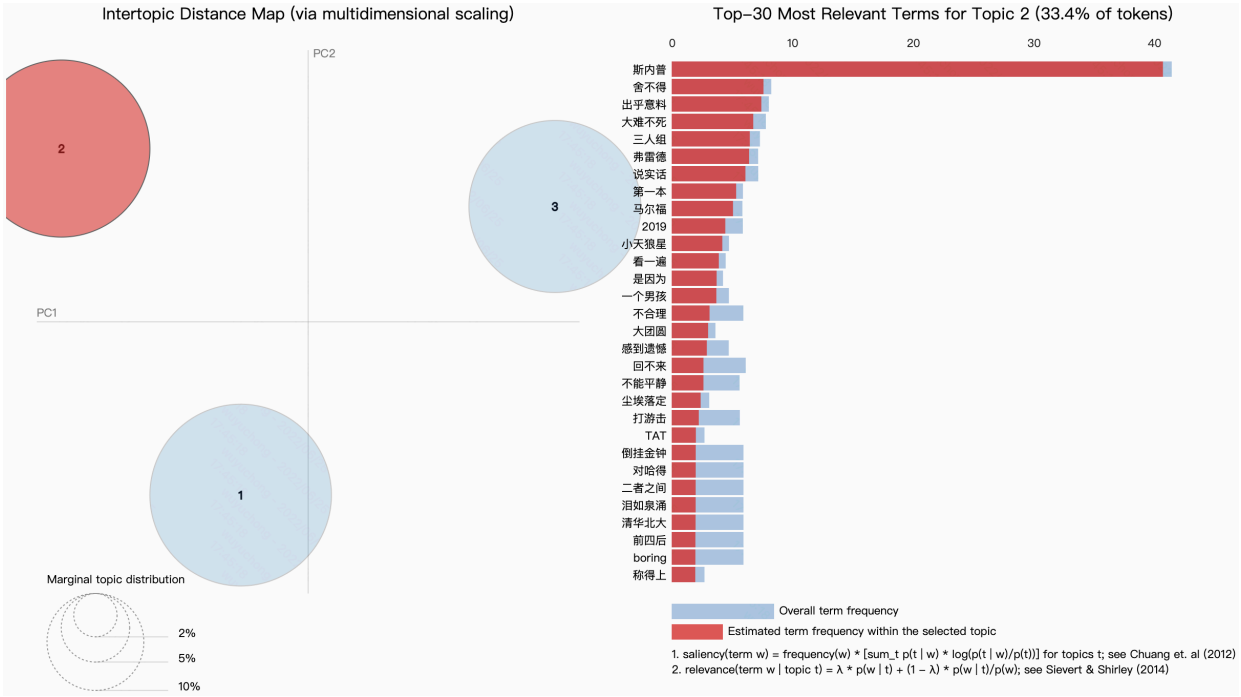

图 4: 主题三成分词汇权重图

第三个话题主要围绕最主要人物哈利波特和成长历程中的重要节点剧情进行讨论。英雄主义、火焰杯、打游击都体现了主角的主线剧情。

当然，读者的讨论并非总是局限于同一个话题，在特定的话题下有会参杂着许多对全书的感受、见解看法等等。

4 深度学习

4.1 数据处理

在进行文本情感分类的有监督训练中，我们将评级级别为差和中等的两类合并为消极一类，将好评作为积极一类。

1. 标签处理：分类标签由类别名称转为数字。
2. 数据集划分：在总体 5 万条文本中随机划分 20% 的测试集，再从训练集中划分 20% 的验证集。
   - 使用训练集的文本进行模型训练
   - 使用验证集的文本进行模型超参数的调整
   - 使用测试集的文本进行模型效果评价
3. 数据集格式转换：使用自动的缓冲区大小，使用 32 的 `batch size`。
   - batch size 为一次训练所抓取的数据样本数量
   - 分批训练相对于直接对全训练集训练的好处在于：提高了每次迭代的训练速度、利于多线程训练、使得梯度下降的方向更加准确
   - batch size 的大小与模型的收敛速度和随机梯度噪音有关
   - 当 batch size 过小时，在一定的迭代次数下，模型来不及收敛
   - 当 batch size 过大时，一方面容易出现内存紧缺，另一方面模型的泛化能力会变差

4.2 Tokenize + DNN

4.2.1 DNN 模型结构

我们搭建了一个三层神经网络用于训练。

表 4: 搭建的神经网络结构（总参数个数：775681）

神经网络层	神经元个数	参数个数
输入层	512	512512
drop out (50%)	0	0
中间层	512	262656
drop out (50%)	0	0
输出层	1	513

4.2.2 模型训练

正常情况下，随着训练迭代次数的增加，损失函数逐渐减小，对训练集的拟合越来越趋向于精细。然而过度精细的拟合容易导致模型的泛化能力变差，即当模型用于之前未曾训练过的数据时表现很差。为了观测这种情况，我们需要划分一部分数据与用于训练的数据隔开，这便是我们划分验证集的原因之一。

为了防止模型过拟合，我们设定在验证集准确率连续三次迭代不再上升时提前终止训练。

我们让学习率随着迭代次数递减：

$$\text{learn-rate} = \frac{\text{initial}}{1 + \frac{\text{decay-rate} \times \text{step}}{\text{decay-step}}}$$

其中：

- decay_rate 为衰减进行的频率：经过多次尝试调参，我们将衰减率定为 e^{-2}
- initial 为初始学习率：经过多次尝试调参，我们将初始学习率定为 e^{-5}

4.3 Word2Vec + LSTM

4.3.1 Word2Vec

在词编码的基础上，我们对文本进行 word2vec 处理。Word2Vec 模型将每个单词映射到一个唯一的固定大小向量，同时可用文档中所有单词的平均值将每个文档转换为向量；然后，此向量可用作预测、文档相似度计算等。

4.3.2 LSTM 模型结构

1. 第一层为 `Embedding` 层，我们使用 `word2vec` 方法将单词编码转换为词向量。这些词向量经过训练，对于意思相近的词，其向量夹角小。
2. 第二层使用双向的长短期记忆层。长短期记忆网络层是一种特殊的循环神经网络层，它能够减轻长序列训练过程中的梯度消失和梯度爆炸问题，适合此处词向量长度较长的情况。它遍历序列中的每个元素作为输入，按照时间顺序传递输出。由于我们使用双向结构，最终结果由输入的前向和后向传递共同决定，这使得最前端的输入不必通过漫长的处理步数才能影响到最终结果，有效的提高了训练在文本中的均匀度。
3. 第三层为全连接层，由于在多层神经网络中梯度容易在深层网络中变得极小，使得参数无法正常更新，所以我们使用 `RELU` 作为激活函数解决梯度消失问题。
4. 第四层为输出维度为 5 的输出层，为了得到多分类的概率值，使用 `softmax` 函数将输出值压缩至 0 - 1 的范围内。

表 5: 搭建的神经网络结构（总参数个数：775681）

神经网络层	神经元个数	参数个数
Embedding	64	64000
双向 LSTM	128	66048
全连接	64	8256

神经网络层	神经元个数	参数个数
输出层	1	65

4.3.3 模型训练

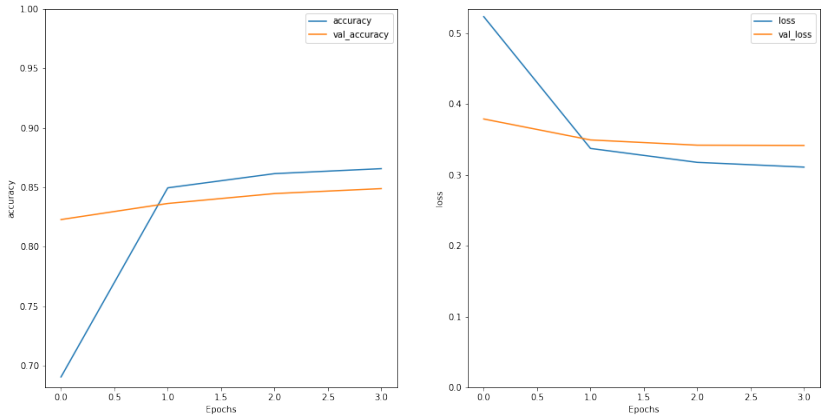

图 5: LSTM 模型训练过程损失函数和准确率趋势图

随着迭代次数的上升，模型在训练集和测试集上的损失函数下降，准确率上升。

4.4 BERT

4.4.1 BERT 介绍

BERT 是一系列双向文字编码转换模型的总称，用来结合上下文语义计算每个词的词向量，在自然语言处理中被广泛使用。

我们使用了前人在超大型语料库上训练的已有基础 BERT 模型，通过迁移学习的方式在我们的 BBC 文本数据集上进行微调。

4.4.2 预训练 BERT 模型

我们首先使用了一个参数量较少的 small-BERT 模型用于测试，在通过测试后，为了进一步提升模型的准确度，我们使用 al-BERT 和标准的 BERT 进行正式训练。

在 BERT 的输入层，对于原始的文字输入，我们需要将其转换成为数值编码。每一个 BERT 模型都有其严格对应的预处理模型来提升转换效果。

对于 small-BERT 模型预处理模型将输入的向量设为 128 的长度。

4.4.3 BERT 模型结构

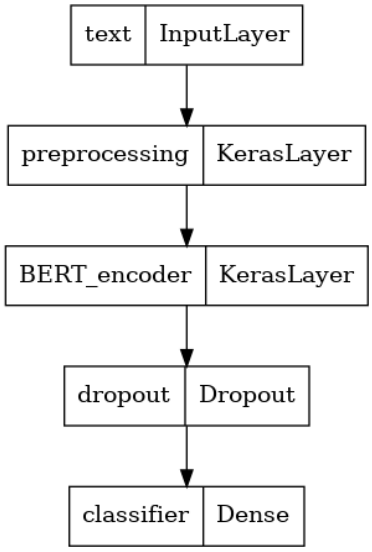

图 6: BERT 模型结构示意图

表 6: 搭建的神经网络结构（总参数个数：775681）

神经网络层	神经元个数	参数个数
Embedding	64	64000
双向 LSTM	128	66048
全连接	64	8256
输出层	1	65

4.4.4 模型训练

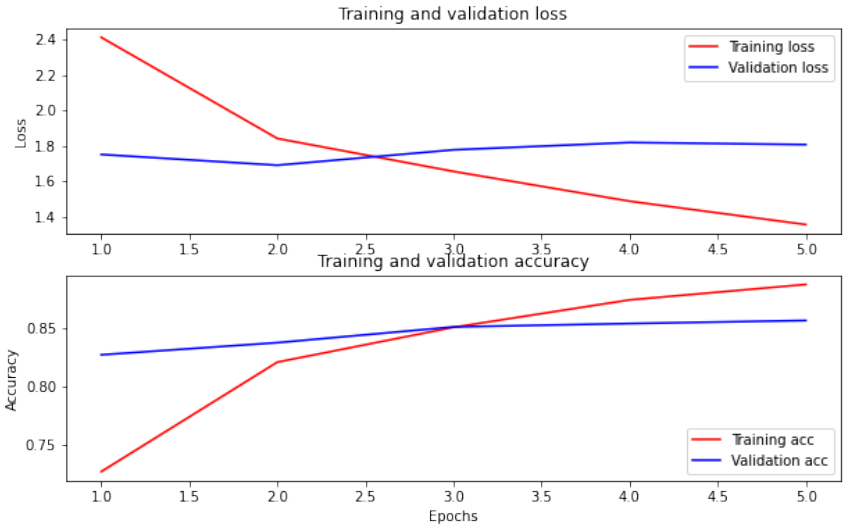

图 7: BERT 模型训练过程损失函数和准确率趋势图

随着迭代次数的上升，模型在训练集损失函数下降，但在验证集上损失函数基本保持平稳，准确率上升。

我们使用交叉熵作为我们的损失函数：

$$-\sum_{c=1}^{M} y_{o,c} \log(p_{o,c})$$

其中：

- M 是分类数
- y 是标签 c 在观测 o 下是否分类正确的 0/1 变量
- p 是预测概率

由于神经网络刚开始训练时非常不稳定，因此刚开始的学习率应当设置得很低很低，这样可以保证网络能够具有良好的收敛性。但是较低的学习率会使得训练过程变得非常缓慢，因此这里采用从较低学习率逐渐增大至较高学习率的方式实现网络训练前 10% 次迭代的“热身”阶段。一直使用较高学习率是不合适的，因为它会使得权重的梯度一直来回震荡，很难使训练的损失值达到全局最低谷。因此在 warm-up 结束后，我们使用线性减小的学习率。

在迁移学习时，我们选取的优化器与 BERT 在预训练时的 `Adamw` 优化器保持一致。

input : γ(lr), β_1, β_2(betas), θ_0(params), $f(\theta)$(objective), ϵ (epsilon)
λ(weight decay), $amsgrad$
initialize : $m_0 \leftarrow 0$ (first moment), $v_0 \leftarrow 0$ (second moment), $\widehat{v_0}^{max} \leftarrow 0$

for $t = 1$ **to** ... **do**
$g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$
$\theta_t \leftarrow \theta_{t-1} - \gamma\lambda\theta_{t-1}$
$m_t \leftarrow \beta_1 m_{t-1} + (1 - \beta_1) g_t$
$v_t \leftarrow \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$
$\widehat{m_t} \leftarrow m_t / \left(1 - \beta_1^t\right)$
$\widehat{v_t} \leftarrow v_t / \left(1 - \beta_2^t\right)$
if $amsgrad$
$\widehat{v_t}^{max} \leftarrow \max(\widehat{v_t}^{max}, \widehat{v_t})$
$\theta_t \leftarrow \theta_{t-1} - \gamma\widehat{m_t} / \left(\sqrt{\widehat{v_t}^{max}} + \epsilon\right)$
else
$\theta_t \leftarrow \theta_{t-1} - \gamma\widehat{m_t} / \left(\sqrt{\widehat{v_t}} + \epsilon\right)$

return θ_t

图 8: Adamw 算法示意图

Adam 的超收敛性质使其在训练学习率高的神经网络时可以达到节省迭代次数的效果。只要调整得当，Adam 在实践上都能达到 SGD+Momentum 的高准确率，而且速度更快。在几年前人们普遍认为 Adam 的泛化性能不如 SGD-Momentum，然而今年论文表明这通常是由于所选择的超参数不正确导致，通常来说 Adam 需要的正则化比 SGD 更多。

4.4.5 模型评价

我们在测试集上进行拟合，得到准确率为 86%。由于该模型仅仅为小型的 BERT，为了进一步提升模型的准确度，我们使用 al-BERT 和标准的 BERT 进行正式训练，分别达到了 88% 和 90% 的准确率。

4.4.6 模型应用

使用模型对输入的文本进行分类。我们输入一则测试新闻文本：“这部书很差劲”，该文本被模型分类为消极，符合预期。

5 分布式训练

我们使用 `pyspark` 进行分布式训练。分布式不同于单机训练，而是通过集群上许多的计算机节点同时进行训练。对于文本量很大的数据集而言，单机可能不具备足够的内存和 CPU 资源进行训练，借助于分布式系统，我们能调度集群计算资源进行计算。`pyspark` 是 `spark` 在 `python` 下的实现，它使用 Zookeeper、hadoop 作为底层，通过 MapReduce 的方式将大的计算任务拆解成为一个个小的任务，分发到每个计算机节点上进行计算。

5.1 环境启动

- 通过 YARN 资源调度系统提交到作业队列：`spark-submit --master yarn`
- 由于在 UDF（用户自定义）函数中使用了第三方包，需要将其发送至集群中的每个计算节点 `--py-files gensim.zip`
- 队列计算完成后将结果重定向输出 `> output.txt`

5.2 数据读取

由于数据为逗号分隔的 csv 格式，在文本列出现混淆。我们使用 pandas 进行读取后再转换为 spark DataFrame 格式

5.3 文本特征工程

词频-逆文档频率（TF-IDF）是一种广泛用于文本挖掘的特征向量化方法，它反映了单个词汇相对于语料库中文档的重要性。我们用表示 t 代表词汇，用 d 代表表示文档，用 D 表示语料库。词频 $TF(t,d)$ 是该词在文档 d 中出现的次数，而文档频率 $DF(t,D)$ 是包含该词的文档的数量。如果我们只使用词频来衡量重要性，很容易过分强调那些出现频率很高但几乎没有关于文档的信息的词，例如 “这”“的” 等词汇。如果一个术语在语料库中经常出现，则意味着它不包含有关特定文档的特殊信息。逆文档频率是一个术语提供多少信息的数值度量：

$$IDF(t,D) = \log \frac{|D|+1}{DF(t,D)+1}$$

其中 $|D|$ 是语料库中的文档总数。

由于使用对数，如果一个词出现在所有文档中，它的 IDF 值变为 0，因此使用平滑词以避免对语料库之外的词除以零。TF-IDF 是 TF 和 IDF 的乘积：

$$TFIDF(t,d,D) = TF(t,d) \cdot IDF(t,D)$$

在 TF 的基础上，我们使用改进版的 HashingTF 进行处理。HashingTF 将词汇转换为固定长度的特征向量。HashingTF 利用散列表应用哈希函数映射词汇到索引，之后通过映射的函数计算词频，能有效降低 TF 在大型语料库所需的时间。

我们从一组句子开始，将每个句子分成单词，构建词袋，使用 HashingTF 将句子散列成特征向量，使用 IDF 重新缩放特征向量，然后将我们的特征向量传递给学习算法。

5.4 训练模型

我们首先使用简单的 logistic 模型进行拟合，在训练集上进行拟合，之后在测试集上验证模型的效果。

表 7: 真是值与预测值示例

真实值	预测值
0.0	0.0
0.0	0.0
1.0	1.0
1.0	1.0
0.0	0.0
1.0	1.0
0.0	1.0
0.0	0.0

我们准备了两个测试用例来验证模型是否有效。

1. 我喜欢这部书
2. 它很差劲

模型对前一个句子的分类结果为积极，对一个句子的分类结果为消极。

5.5 模型比较

在 logistic 模型的基础上，我们还搭建了决策树模型、梯度助推树模型、随机森林模型。

决策树的特点是它总是在沿着特征做切分。随着层层递进，这个划分会越来越细。

梯度助推树模型使用 Boosting 的方式把基础模型组合起来。既然决策树基础模型可以做出不完美的预测，那么用第二的基础模型，把“不完美的部分”补上，不断地对“不完美的部分”进行完善，就可以得到效果足够好的集成模型。Boosting 的策略非常多，以 GBDT 为例，它会用第 K 个 CART 拟合前 k-1 个 CART 留下的残差，从而不断的缩小整个模型的误差。

相比于决策树模型，随机森林其实是一种集成算法。它首先随机选取不同的特征 (feature) 和训练样本 (training sample)，生成大量的决策树，然后综合这些决策树的结果来进行最终的分类。所以理论上随机森林相比单一的决策树模型一般来说准确性上有很大的提升，同时一定程度上改善了决策树容易被攻击的特点。

5.6 模型调参

我们使用网格搜索的方式对几个模型的超参数进行调整，选取最优的模型。

6 结论

在本研究中，我们通过网络爬虫的方式获取《哈利波特》全系列图书的评论进行文本分析，在文本预处理阶段，我们首先进行文本清洗，之后对中文文本进行分词，再进行文本特征工程。

6.1 主题模型

使用主题模型，我们对评论进行主题提取，提取出三个最主要的主题：

1. 第一个话题主要是对大结局的评论。图书馆保卫战、第七部、虎头蛇尾都是对结局的讨论。
2. 第二个话题主是对书中各类较为主要的人物进行讨论。排名第一的词汇斯内普是书中以为颇具争议的反派人物，其性格方面的多重性给了读者非常大的讨论空间，弗雷德、小天狼星也都是书中较为典型的人物，他们与哈利波特具有密切地联系，
3. 第三个话题主要围绕最主要人物哈利波特和成长历程中的重要节点剧情进行讨论。英雄主义、火焰杯、打游击都体现了主角的主线剧情。

6.2 评价分类训练

我们使用用户的评论评价进行文本情感分类模型的训练。在文本预处理阶段，我们尝试使用词编码和词向量的方式，在训练阶段，我们构建了 DNN、LSTM、BERT 等多个深度学习模型进行训练，并进行了模型比较，最终达到了 88% 的准确率。最后，为了进一步实现在超大文本集上进行训练，我们使用基于 Spark 的分布式算法在集群服务器上进行训练测试。[2]

模型	计算配置	用时	准确率	可拓展性
tokenize + DNN	阿里云服务器 Xeon 8 核 CPU 32G 内存	3 分钟	55%	低-单机
Word2Vec + LSTM	阿里云服务器 Xeon 8 核 CPU 32G 内存	1 小时	70%	低-单机
bert - 小型	阿里云服务器 Xeon 8 核 CPU 32G 内存	24 分钟	78%	低-单机
bert - AL	阿里云服务器 Xeon 8 核 CPU 32G 内存	1.6 小时	82%	低-单机
bert - 标准	阿里云服务器 Xeon 8 核 CPU 32G 内存	1 小时	88%	低-单机
spark - logit	中央财经大学大数据高性能分布式集群	4 分钟	73%	高-集群
spark - 决策树	中央财经大学大数据高性能分布式集群	10 分钟	80%	高-集群
spark - 梯度助推树	中央财经大学大数据高性能分布式集群	15 分钟	82%	高-集群
spark - 随机森林	中央财经大学大数据高性能分布式集群	30 分钟	87%	高-集群

[2]分布式模型在该小型数据集上没有优势，进行此项的意义在于对大型文本数据集可拓展性的技术储备，仅有在文本量级超过单机可承载上限时，分布式计算才具备意义

7 参考文献

[1] 张征杰, 王自强. 文本分类及算法综述 [J]. 电脑知识与技术,2012,8(04):825-828+841.

[2] 汪岿, 刘柏嵩. 文本分类研究综述 [J]. 数据通信,2019(03):37-47.

[3] 贾澎涛, 孙炜. 基于深度学习的文本分类综述 [J]. 计算机与现代化,2021(07):29-37.

[4] 王博, 刘盛博, 丁堃, 刘则渊. 基于 LDA 主题模型的专利内容分析方法 [J]. 科研管理,2015,36(03):111-117.DOI:10.19571/j.cnki.1000-2995.2015.03.014.

[5] 关鹏, 王曰芬, 傅柱. 不同语料下基于 LDA 主题模型的科学文献主题抽取效果分析 [J]. 图书情报工作,2016,60(02):112-121.DOI:10.13266/j.issn.0252-3116.2016.02.018.

[6] 黄佳佳, 李鹏伟, 彭敏, 谢倩倩, 徐超. 基于深度学习的主题模型研究 [J]. 计算机学报,2020,43(05):827-855.

[7] 胡吉明, 陈果. 基于动态 LDA 主题模型的内容主题挖掘与演化 [J]. 图书情报工作,2014,58(02):138-142.DOI:10.13266/j.issn.0252-3116.2014.02.023.

[8] 陈晓美, 高铖, 关心惠. 网络舆情观点提取的 LDA 主题模型方法 [J]. 图书情报工作,2015,59(21):21-26.DOI:10.13266/j.issn.0252-3116.2015.21.003.

8 代码

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

# --------------------> 准备工作 ---------------------------------------------
# base
import os
import re
import time
import nltk
import jieba
import pydot
import random
import gensim
import pickle
import shutil
import requests
import pyLDAvis
```

```
import itertools
import numpy as np
import pandas as pd
import tensorflow as tf
from pprint import pprint
from imageio import imread
import tensorflow_hub as hub
from bs4 import BeautifulSoup
import tensorflow_text as text
from wordcloud import WordCloud
import matplotlib.pyplot as plt
import gensim.corpora as corpora
from gensim.corpora import Dictionary
from official.nlp import optimization
from wordcloud import ImageColorGenerator
import pyLDAvis.gensim_models as gensimvis
from gensim.models.ldamodel import LdaModel
from sklearn.metrics import confusion_matrix
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from gensim.models.coherencemodel import CoherenceModel

# pyspark
import pyspark.ml.feature
from pyspark.ml import Pipeline
from pyspark.ml.tuning import CrossValidator,ParamGridBuilder
from pyspark.ml.evaluation import MulticlassClassificationEvaluator
from pyspark.ml.feature import Tokenizer,StopWordsRemover,CountVectorizer,IDF,StringIndexer,Word2V
from pyspark.ml.classification import LogisticRegression,RandomForestClassifier,GBTClassifier,Deci
from pyspark.sql import SparkSession,Row
from pyspark.sql.types import StringType
from pyspark.sql.functions import udf, col

# ------------------------------------------------------------------------------


# --------------------> 爬取设置 ------------------------------------------------
def getHtml(url, headers):

```

```
    try:
        r = requests.get(url, timeout=30, headers=headers)
        r.raise_for_status()
        return r.text
    except:
        return ''

# 获取评论
def getComment(html):
    soup = BeautifulSoup(html, 'html.parser')
    comments_list = []  # 评论列表
    comment_nodes = soup.select('.comment > p')
    for node in comment_nodes:
        comments_list.append(node.get_text().strip().replace("\n", "") + u'\n')
    return comments_list

# 获取并将评论保存到文件中
def saveCommentText(fpath, headers, pre_url, depth):
    with open(fpath, 'w', encoding='utf-8') as f:
        for i in range(1, depth):
            print('开始爬取第{}页评论...'.format(i))
            url = pre_url + 'start=' + str(20 * (i-1)) + '&limit=20&status=P&sort=new_score'
            html = getHtml(url, headers)
            f.writelines(getComment(html))
            # 设置随机休眠防止 IP 被封
            time.sleep(1 + float(random.randint(1, 20)) / 20)
    print('成功完成爬取任务')


# 浏览器信息 - 依据特定电脑信息 (https://blog.csdn.net/ysblogs/article/details/88530124)
headers = {'User-Agent': 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTM
           'Cookie': 'll="108288"; bid=Qm5EziAHieA; __utma=30149280.2109946700.1655390050.16553900

# ---------------------------------------------------------------------------


# ------------------->  爬取 ------------------------------------------
depth = 30
```

```
pre_url = "https://book.douban.com/subject/2295163/comments/?percent_type=h&"
fpath = './text/good.txt'
saveCommentText(fpath, headers, pre_url, depth)

pre_url = "https://book.douban.com/subject/2295163/comments/?percent_type=m&"
fpath = './text/medium.txt'
saveCommentText(fpath, headers, pre_url, depth)

pre_url = "https://book.douban.com/subject/2295163/comments/?percent_type=l&"
fpath = './text/bad.txt'
saveCommentText(fpath, headers, pre_url, depth)

print('-------------------------finish-----------------------------')
# ------------------------------------------------------------------------------


# --------------------> 整理数据 ------------------------------------------
text = []
rating = []

text_file = open("./text/bad.txt", "r")
unit = text_file.readlines()
text = text + unit
rating = rating + ['bad'] * len(unit)

text_file = open("./text/good.txt", "r")
unit = text_file.readlines()
text = text + unit
rating = rating + ['good'] * len(unit)

text_file = open("./text/medium.txt", "r")
unit = text_file.readlines()
text = text + unit
rating = rating + ['medium'] * len(unit)

print(len(text))
print(len(rating))

outcome = pd.DataFrame(list(zip(text, rating)), columns=['text', 'rating'])
```

```
outcome.to_csv('./text/text.csv', index=False)
# ------------------------------------------------------------------------------



# --------------------> LDA      ------------------------------------------
dat = pd.read_csv('./text/text.csv')
print(dat.query('rating == "bad"'))
dat['text'] = dat.text.apply(lambda x: ",".join(jieba.cut(x)))
tweets = [t.split(',') for t in dat.text]
f = open('./code/ChineseStopWords.txt')
stopwords = f.read().splitlines()
f.close()
stopwords += ['....', '.....', 'end']
for i in range(len(tweets)):
    tweets[i] = [w for w in tweets[i] if w not in stopwords and len(w)>2]
text = dat.text.values.tolist()
words_list = list(itertools.chain(*tweets))
print(tweets[0:5])
id2word = Dictionary(tweets)
corpus = [id2word.doc2bow(text) for text in tweets]
print(corpus[:1])
[[(id2word[i], freq) for i, freq in doc] for doc in corpus[:1]]
lda_model = LdaModel(corpus=corpus,
                     id2word=id2word,
                     num_topics=3,
                     random_state=0,
                     chunksize=100,
                     alpha='auto',
                     per_word_topics=True)
pprint(lda_model.print_topics())
doc_lda = lda_model[corpus]
pyLDAvis.enable_notebook()
p = gensimvis.prepare(lda_model, corpus, id2word)
```

```
# --------------------> LDA      ------------------------------------------
# ------------------> 加载数据
dat = pd.read_csv('./text/text.csv')
dat = dat.query('rating != "medium"')

import jieba
dat['text'] = dat.text.apply(lambda x: " ".join(jieba.cut(x)))

# ------------------> 标签处理
encoder = LabelEncoder()
encoder.fit(dat['rating'])
y = encoder.transform(dat['rating'])
text_labels = encoder.classes_
text_labels

X_train, X_test, y_train, y_test = train_test_split(
    dat['text'], y, test_size=0.2, random_state=1)
X_train, X_val, y_train, y_val = train_test_split(
    X_train, y_train, test_size=0.2, random_state=1)

AUTOTUNE = tf.data.AUTOTUNE
batch_size = 32

raw_train_ds = tf.data.Dataset.from_tensor_slices((X_train.values, y_train))
raw_val_ds = tf.data.Dataset.from_tensor_slices((X_val.values, y_val))
raw_test_ds = tf.data.Dataset.from_tensor_slices((X_test.values, y_test))

train_ds = raw_train_ds.cache().prefetch(buffer_size=AUTOTUNE).batch(batch_size)
val_ds = raw_val_ds.cache().prefetch(buffer_size=AUTOTUNE).batch(batch_size)
test_ds = raw_test_ds.cache().prefetch(buffer_size=AUTOTUNE).batch(batch_size)

for text_batch, label_batch in train_ds.take(1):
  for i in range(2):
    print(f'Review: {text_batch.numpy()[i]}')
    label = label_batch.numpy()[i]
    print(f'Label : {label} ({text_labels[label]})')
```

```
max_words = 1000
tokenize = tf.keras.preprocessing.text.Tokenizer(num_words=max_words,
                                                 char_level=False)
tokenize.fit_on_texts(X_train) # fit tokenizer to our training text data
x_train_token = tokenize.texts_to_matrix(X_train)
x_test_token = tokenize.texts_to_matrix(X_test)
y_train_token = y_train
y_test_token = y_test
print('x_train shape:', x_train_token.shape)
print('x_test shape:', x_test_token.shape)
print('y_train shape:', y_train_token.shape)
print('y_test shape:', y_test_token.shape)

batch_size = 32
epochs = 100
drop_ratio = 0.5


model = tf.keras.models.Sequential()
model.add(tf.keras.layers.Dense(512, input_shape=(max_words,)))
model.add(tf.keras.layers.Activation('relu'))
model.add(tf.keras.layers.Dropout(drop_ratio))
model.add(tf.keras.layers.Dense(512))
model.add(tf.keras.layers.Activation('relu'))
model.add(tf.keras.layers.Dropout(drop_ratio))
model.add(tf.keras.layers.Dense(1))
model.add(tf.keras.layers.Activation('relu'))


model.compile(loss='binary_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])

model.summary()

callback = tf.keras.callbacks.EarlyStopping(monitor='accuracy', patience=3)
history = model.fit(x_train_token, y_train_token,
```

```
                    batch_size=batch_size,
                    epochs=epochs,
                    verbose=1,
                    callbacks=[callback],
                    validation_split=0.1)

score = model.evaluate(x_test_token, y_test_token,
                       batch_size=batch_size, verbose=1)
print('Test loss:', score[0])
print('Test accuracy:', score[1])

VOCAB_SIZE = 1000
encoder = tf.keras.layers.TextVectorization(
    max_tokens=VOCAB_SIZE)
encoder.adapt(train_ds.map(lambda text, label: text))
vocab = np.array(encoder.get_vocabulary())
vocab[:20]


encoded_example = encoder(text_batch)[:3].numpy()
encoded_example


# 打印示例:
for n in range(3):
  print("Original: ", text_batch[n].numpy())
  print("Round-trip: ", " ".join(vocab[encoded_example[n]]))
  print()

model = tf.keras.Sequential([
    encoder,
    tf.keras.layers.Embedding(
        input_dim=len(encoder.get_vocabulary()),
        output_dim=64,
        # Use masking to handle the variable sequence lengths
        mask_zero=True),
    tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(64)),
    tf.keras.layers.Dense(64, activation='relu'),
    tf.keras.layers.Dense(1)
```

```
])
model.summary()

model.compile(loss=tf.keras.losses.BinaryCrossentropy(from_logits=True),
              optimizer=tf.keras.optimizers.Adam(1e-4),
              metrics=['accuracy'])

history = model.fit(train_ds, epochs=4,
                    validation_data=val_ds,
                    validation_steps=30)


def plot_graphs(history, metric):
  plt.plot(history.history[metric])
  plt.plot(history.history['val_'+metric], '')
  plt.xlabel("Epochs")
  plt.ylabel(metric)
  plt.legend([metric, 'val_'+metric])

plt.figure(figsize=(16, 8))
plt.subplot(1, 2, 1)
plot_graphs(history, 'accuracy')
plt.ylim(None, 1)
plt.subplot(1, 2, 2)
plot_graphs(history, 'loss')
plt.ylim(0, None)
plt.savefig('./figure/word2vec_lstm.png')
plt.show()

test_loss, test_acc = model.evaluate(test_ds)

print('Test Loss:', test_loss)
print('Test Accuracy:', test_acc)


tfhub_handle_encoder = 'https://storage.googleapis.com/tfhub-modules/tensorflow/small_bert/bert_en
tfhub_handle_preprocess = 'https://storage.googleapis.com/tfhub-modules/tensorflow/bert_en_uncased

bert_preprocess_model = hub.KerasLayer(tfhub_handle_preprocess)
```

```
text_test = ['The first sentence. The second sentence.']
text_preprocessed = bert_preprocess_model(text_test)

print(f'Keys       : {list(text_preprocessed.keys())}')
print(f'Shape      : {text_preprocessed["input_word_ids"].shape}')
print(f'Word Ids   : {text_preprocessed["input_word_ids"][0, :12]}')
print(f'Input Mask : {text_preprocessed["input_mask"][0, :12]}')
print(f'Type Ids   : {text_preprocessed["input_type_ids"][0, :12]}')


# ## BERT 模型
#
# 在进行迁移学习之前，我们先看预训练 BERT 模型的输出格式

bert_model = hub.KerasLayer(tfhub_handle_encoder)
bert_results = bert_model(text_preprocessed)

print(f'Pooled Outputs Shape:{bert_results["pooled_output"].shape}')
print(f'Pooled Outputs Values:{bert_results["pooled_output"][0, :12]}')
print(f'Sequence Outputs Shape:{bert_results["sequence_output"].shape}')
print(f'Sequence Outputs Values:{bert_results["sequence_output"][0, :12]}')


def build_classifier_model():
  text_input = tf.keras.layers.Input(shape=(), dtype=tf.string, name='text')
  preprocessing_layer = hub.KerasLayer(tfhub_handle_preprocess, name='preprocessing')
  encoder_inputs = preprocessing_layer(text_input)
  encoder = hub.KerasLayer(tfhub_handle_encoder, trainable=True, name='BERT_encoder')
  outputs = encoder(encoder_inputs)
  net = outputs['pooled_output']
  net = tf.keras.layers.Dropout(0.1)(net)
  net = tf.keras.layers.Dense(1)(net)
  return tf.keras.Model(text_input, net)
classifier_model = build_classifier_model()
bert_raw_result = classifier_model(tf.constant(text_test))
print(tf.sigmoid(bert_raw_result))

tf.keras.utils.plot_model(classifier_model)
classifier_model.summary()
```

```
loss = tf.keras.losses.BinaryCrossentropy()
metrics = tf.metrics.BinaryAccuracy()
epochs = 5
steps_per_epoch = tf.data.experimental.cardinality(train_ds).numpy()
num_train_steps = steps_per_epoch * epochs
num_warmup_steps = int(0.1*num_train_steps)


init_lr = 3e-5
optimizer = optimization.create_optimizer(init_lr=init_lr,
                                          num_train_steps=num_train_steps,
                                          num_warmup_steps=num_warmup_steps,
                                          optimizer_type='adamw')


get_ipython().run_cell_magic('time', '', "classifier_model.compile(optimizer=optimizer,\n
history_dict = history.history

acc = history_dict['binary_accuracy']
val_acc = history_dict['val_binary_accuracy']
loss = history_dict['loss']
val_loss = history_dict['val_loss']

epochs = range(1, len(acc) + 1)
fig = plt.figure(figsize=(10, 6))
fig.tight_layout()

plt.subplot(2, 1, 1)
# r is for "solid red line"
plt.plot(epochs, loss, 'r', label='Training loss')
# b is for "solid blue line"
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title('Training and validation loss')
# plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()

plt.subplot(2, 1, 2)
plt.plot(epochs, acc, 'r', label='Training acc')
```

```
plt.plot(epochs, val_acc, 'b', label='Validation acc')
plt.title('Training and validation accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend(loc='lower right')
plt.savefig('./figure/bert_train.png')
plt.show()



loss, accuracy = classifier_model.evaluate(test_ds)
print(f'Loss: {loss}')
print(f'Accuracy: {accuracy}')

def plot_confusion_matrix(cm, classes,
                          title='Confusion matrix',
                          cmap=plt.cm.Blues):

    cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]

    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title, fontsize=30)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45, fontsize=22)
    plt.yticks(tick_marks, classes, fontsize=22)

    fmt = '.2f'
    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, format(cm[i, j], fmt),
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else "black")

    plt.ylabel('True label', fontsize=25)
    plt.xlabel('Predicted label', fontsize=25)

predict_probability = classifier_model.predict(test_ds)
prediction = [np.argmax(i) for i in predict_probability]
```

```
cnf_matrix = confusion_matrix(y_test.tolist(), prediction)
plt.figure(figsize=(24,20))
plot_confusion_matrix(cnf_matrix, classes=text_labels, title="Confusion matrix")
plt.show()


classifier_model.save('./model/IMDB_bert', include_optimizer=False)

model = tf.saved_model.load('./model/IMDB_bert')
query = ['This movie is so bad']
result = tf.sigmoid(model(tf.constant(query)))
print('----- 评论积极的概率 -----')
dict(zip(text_labels, result.numpy()))

import pandas as pd
import pyspark.ml.feature
from pyspark.ml import Pipeline
from pyspark.ml.tuning import CrossValidator,ParamGridBuilder
from pyspark.ml.evaluation import MulticlassClassificationEvaluator
from pyspark.ml.feature import Tokenizer,StopWordsRemover,CountVectorizer,IDF,StringIndexer,Word2V
from pyspark.ml.classification import LogisticRegression,RandomForestClassifier,GBTClassifier,Deci
from pyspark.sql import SparkSession,Row
from pyspark.sql.types import StringType
from pyspark.sql.functions import udf, col

spark = SparkSession.builder.appName('text_classification').getOrCreate()

try:
    df = spark.read.csv('./text/text.csv', header = True, inferSchema = True)
except 'FileNotFoundError':
    # location on server
    df = spark.read.csv('file:///home1/cufe/students/wuyuchong/text.csv', header = True, inferSche
df.printSchema()
df.show()

pandasDF = pd.read_csv('./text/text.csv')
pandasDF = pandasDF.query('rating != "medium"')
pandasDF['text'] = pandasDF.text.apply(lambda x: " ".join(jieba.cut(x)))
pandasDF.isnull().sum() # 缺失值检查
```

```
df = spark.createDataFrame(pandasDF)
df.show()

cleaning = True
if cleaning == False:
    df = df.withColumn("clean_text", df.text)
else:
    try:
        # 在服务器上的分布式模式中，需要使用 --py-files 将 gensim 包传到每个子节点
        # 若该过程失败则跳过文本清洁过程
        import gensim.parsing.preprocessing as gsp
        from gensim import utils
        filters = [
            gsp.strip_tags,
            gsp.strip_punctuation,
            gsp.strip_multiple_whitespaces,
            gsp.strip_numeric,
            gsp.remove_stopwords,
            gsp.strip_short,
            gsp.stem_text
        ]
        def clean_text(x):
            x = x.lower()
            x = utils.to_unicode(x)
            for f in filters:
                x = f(x)
            return x

        cleanTextUDF = udf(lambda x: clean_text(x), StringType())
        df = df.withColumn("clean_text", cleanTextUDF(col("text")))
    except:
        df = df.withColumn("clean_text", df.text)

# ----------------------------> 标签数字转换
labelEncoder = StringIndexer(inputCol='rating', outputCol='label').fit(df)
labelEncoder.transform(df).show(5)
df = labelEncoder.transform(df)
# ------------------------------------------------------------------------
```

```
(trainDF,testDF) = df.randomSplit((0.7,0.3), seed=1)

# ----------------------------> 特征工程方法选项
#  processType = 'word2vec'
processType = 'vectorize-idf'
#  processType = 'tf-idf'

# ----------------------------> 文本特征工程
tokenizer = Tokenizer(inputCol='clean_text', outputCol='tokens')
add_stopwords = ["<br />","amp"]
stopwords_remover = StopWordsRemover(inputCol='tokens', outputCol='filtered_tokens').setStopWords(
vectorizer = CountVectorizer(inputCol='filtered_tokens', outputCol='rawFeatures')
hashingTF = HashingTF(inputCol="filtered_tokens", outputCol="rawFeatures")
idf = IDF(inputCol='rawFeatures', outputCol='vectorizedFeatures')
word2Vec = Word2Vec(vectorSize=50, minCount=2, inputCol="filtered_tokens", outputCol='vectorizedFe
if processType == 'word2vec':
    pipeline = Pipeline(stages=[tokenizer,stopwords_remover,word2Vec])
if processType == 'vectorize-idf':
    pipeline = Pipeline(stages=[tokenizer,stopwords_remover,vectorizer,idf])
if processType == 'tf-idf':
    pipeline = Pipeline(stages=[tokenizer,stopwords_remover,hashingTF,idf])
preprocessModel = pipeline.fit(trainDF)
trainDF = preprocessModel.transform(trainDF)
testDF = preprocessModel.transform(testDF)
trainDF.show()

lr = LogisticRegression(featuresCol='vectorizedFeatures',labelCol='label')
lr_model = lr.fit(trainDF)
prediction = lr_model.transform(testDF)
prediction.select(['label', 'prediction']).show()
evaluator = MulticlassClassificationEvaluator(labelCol='label',predictionCol='prediction',metricNa
accuracy = evaluator.evaluate(prediction)
print(accuracy)


inputText = spark.createDataFrame([("I like this movie",StringType()),
                                   ("It is so bad",StringType())],
                                  ["clean_text"])
```

```
inputText.show(truncate=False)
inputText = preprocessModel.transform(inputText)
inputPrediction = lr_model.transform(inputText)
inputPrediction.show()
inputPrediction.select(['clean_text', 'prediction']).show()

def logisticCV(trainDF, testDF):
    lr = LogisticRegression(featuresCol='vectorizedFeatures',labelCol='label')
    model = lr.fit(trainDF)
    prediction = model.transform(testDF)
    evaluator = MulticlassClassificationEvaluator(labelCol='label',predictionCol='prediction',metr
    accuracy = evaluator.evaluate(prediction)
    print('Accuracy of logistic regression: %g' % accuracy)

def RandomForest(trainDF, testDF):
    rf = RandomForestClassifier(featuresCol='vectorizedFeatures',labelCol='label')
    model = rf.fit(trainDF)
    prdiction = model.transform(testDF)
    evaluator = MulticlassClassificationEvaluator(labelCol='label',predictionCol='prediction',metr
    accuracy = evaluator.evaluate(prediction)
    print('Accuracy of random forest: %g' % accuracy)

def GBT(trainDF, testDF):
    gbt = GBTClassifier(featuresCol='vectorizedFeatures',labelCol='label')
    model = gbt.fit(trainDF)
    prdiction = model.transform(testDF)
    evaluator = MulticlassClassificationEvaluator(labelCol='label',predictionCol='prediction',metr
    accuracy = evaluator.evaluate(prediction)
    print('Accuracy of gbt: %g' % accuracy)

def DecisionTree(trainDF, testDF):
    dt = DecisionTreeClassifier(featuresCol='vectorizedFeatures',labelCol='label')
    model = dt.fit(trainDF)
    prdiction = model.transform(testDF)
    evaluator = MulticlassClassificationEvaluator(labelCol='label',predictionCol='prediction',metr
    accuracy = evaluator.evaluate(prediction)
    print('Accuracy of decision tree: %g' % accuracy)

logisticCV(trainDF, testDF)
```

```
RandomForest(trainDF, testDF)
GBT(trainDF, testDF)
DecisionTree(trainDF, testDF)
# ## 模型调参
#
# 我们使用网格搜索的方式对几个模型的超参数进行调整，选取最优的模型
def logisticCV(trainDF, testDF):
    lr = LogisticRegression(featuresCol='vectorizedFeatures',labelCol='label')
    pipeline = Pipeline(stages=[lr])
    paramGrid = ParamGridBuilder() \
        .addGrid(lr.regParam, [0, 0.5, 2.0]) \
        .addGrid(lr.elasticNetParam, [0.0, 0.5, 1.0]) \
        .addGrid(lr.maxIter, [50, 100, 200]) \
        .build()
    evaluator = MulticlassClassificationEvaluator(labelCol='label',predictionCol='prediction',metr
    crossValidator = CrossValidator(estimator=pipeline,
                                    evaluator=evaluator,
                                    estimatorParamMaps=paramGrid,
                                    numFolds=5)
    cv = crossValidator.fit(trainDF)
    best_model = cv.bestModel.stages[0]
    prediction = best_model.transform(testDF)
    accuracy = evaluator.evaluate(prediction)
    print('Accuracy in Cross Validation of logistic regression: %g' % accuracy)

def RandomForestCV(trainDF, testDF):
    rf = RandomForestClassifier(featuresCol='vectorizedFeatures',labelCol='label')
    pipeline = Pipeline(stages=[rf])
    paramGrid = ParamGridBuilder() \
        .addGrid(rf.maxDepth, [5, 10]) \
        .addGrid(rf.maxBins, [16, 32]) \
        .addGrid(rf.minInfoGain, [0, 0.01]) \
        .addGrid(rf.numTrees, [20, 60]) \
        .addGrid(rf.impurity, ['gini', 'entropy']) \
        .build()
    evaluator = MulticlassClassificationEvaluator(labelCol='label',predictionCol='prediction',metr
    crossValidator = CrossValidator(estimator=pipeline,
                                    evaluator=evaluator,
                                    estimatorParamMaps=paramGrid,
```

```
                                    numFolds=5)
    cv = crossValidator.fit(trainDF)
    best_model = cv.bestModel.stages[0]
    prediction = best_model.transform(testDF)
    accuracy = evaluator.evaluate(prediction)
    print('Accuracy in Cross Validation of random forest: %g' % accuracy)

def GBTClassifierCV(trainDF, testDF):
    gbt = GBTClassifier(featuresCol='vectorizedFeatures',labelCol='label')
    pipeline = Pipeline(stages=[gbt])
    paramGrid = ParamGridBuilder() \
        .addGrid(gbt.maxDepth, [5, 10]) \
        .addGrid(gbt.maxBins, [16, 32]) \
        .addGrid(gbt.minInfoGain, [0, 0.01]) \
        .addGrid(gbt.maxIter, [10, 20]) \
        .addGrid(gbt.stepSize, [0.1, 0.2]) \
        .build()
    evaluator = MulticlassClassificationEvaluator(labelCol='label',predictionCol='prediction',metr
    crossValidator = CrossValidator(estimator=pipeline,
                                    evaluator=evaluator,
                                    estimatorParamMaps=paramGrid,
                                    numFolds=5)
    cv = crossValidator.fit(trainDF)
    best_model = cv.bestModel.stages[0]
    prediction = best_model.transform(testDF)
    accuracy = evaluator.evaluate(prediction)
    print('Accuracy in Cross Validation of GBT: %g' % accuracy)

def DecisionTreeCV(trainDF, testDF):
    dt = DecisionTreeClassifier(featuresCol='vectorizedFeatures',labelCol='label')
    pipeline = Pipeline(stages=[dt])
    paramGrid = ParamGridBuilder() \
        .addGrid(dt.maxDepth, [5, 10]) \
        .addGrid(dt.maxBins, [16, 32]) \
        .addGrid(dt.minInfoGain, [0, 0.01]) \
        .addGrid(dt.minWeightFractionPerNode, [0, 0.5]) \
        .addGrid(dt.impurity, ['gini', 'entropy']) \
        .build()
    evaluator = MulticlassClassificationEvaluator(labelCol='label',predictionCol='prediction',metr
```

```
    crossValidator = CrossValidator(estimator=pipeline,
                                    evaluator=evaluator,
                                    estimatorParamMaps=paramGrid,
                                    numFolds=5)
    cv = crossValidator.fit(trainDF)
    best_model = cv.bestModel.stages[0]
    prediction = best_model.transform(testDF)
    accuracy = evaluator.evaluate(prediction)
    print('Accuracy in Cross Validation of GBT: %g' % accuracy)

logisticCV(trainDF, testDF)
RandomForestCV(trainDF, testDF)
GBTClassifierCV(trainDF, testDF)
DecisionTreeCV(trainDF, testDF)
# ------------------------------------------------------------------------
```

9 爬取评论展示

“阿不思•西弗勒斯，你的名字中含有霍格沃茨两位校长的名字。其中一个就是斯莱特林的，而他可能是我见过的最勇敢的人。
哈利波特原本就是一套适合从11，12岁读到20出头的系列，我们有幸赶上这个时代，和小主人公们一起沉淀，一起成长；我们
这本书是我高三的时候在Amazon买的预售，和一枚小小的金色飞贼一起到来，用了3天读完，就再也没有勇气看第二遍，当时
邓布利多死得值了，结尾够曲折。邓布利多与格林德沃……一个世纪的爱与恨，这么有料的CP，发展个番外吧。
休憩33rd.当初刚出七，我买来英文版艰辛啃完，泪流满面，今天我第一次读中文版，于是在魔都旅馆床上，放声大哭
所有爱唱反调的胖友们，你们都没试过给疯姑娘卢娜的《唱唱反调》投稿吗?
基本上一如预想。罗琳笔下的主要角色——除了铁三角是标准成长故事的逐步成长外——其他的都经历了一个从认识表象到探
//阿布思·西弗勒斯，你的名字中含有霍格沃茨两位校长的名字，其中一位就是斯莱特林的，而他可能是我见过的最勇敢的人
走吧走吧 人总要学着自己长大
最感动的情节是多比的死亡和哈利的面对死亡的表现.
无论何时何地，爱与勇气都不可忘却。
多年以后才想起来看结局，马上就沦陷了。和初中时候一样，一开始看就满眼幻觉，四点爬起来看书这种事情只有在这时候才
all as well，一套看了近十年的书，JK罗琳用十年的时间让我们忘记了自己永远只是一个麻瓜的事实。
每翻一页便是与你的告别。我不想同你告别。那瑰丽堂皇的永不言弃的世界。我不想同你们分开。| 消失的东西去了哪里。化
几年后重新看了最后的几章，在这热得要死的梅雨终焉，在东瀛的小小房子中依旧看得热泪盈眶，哈利是真正的领袖，真正的
再也沒有哈利波特了.多讓人傷心.
我觉得2018年1月，我打开豆瓣读书的年终总结页面，可能会发现17年我一共读过七本书，哈利波特1-7。。
哈利还是那个哈利，纳威已经不是那个纳威了。
赶在去大阪的前一天重温完了这套书。为了把大布局收尾，罗琳硬生生编出两个“N大”设定，这是让线索合理化的最简单技巧
我用六个小时读完了《哈利波特与死圣》。中文版的。中间只站起来喝了一口水。合上书时人都要虚脱了。

我所看过的最杰出的一部通俗文学。
卢平，教授，一路走好
一个时代的完结
2020年重读08:故事结束了，难过的时候我会拿出来读读，因为还相信某些就像书里所说早已被耻笑的东西，是因为这些才想
想写的太多来不及交代完的感觉，还有还我的双胞胎!
终于看完了，斯内普是最大的剧情了，总体来说，越来越烦哈利波特本人。。
呜呜呜哭了好几次。
在哈利三人组开始毫无头绪的逃亡之后，作者又试图不断用圣器魂器这类名词混淆读者的思路，但最终，还是以哈利在决斗中
毕竟是大结局……
很高兴我们能和全世界的孩子一起见证我们童年的快乐和回忆，以及一起画上的还算圆满的句号。
要是所有的书我都能读得那么快就好了：去大马出差了四天，等飞机看，吃饭看，坐车看，等人看，工作完了回宾馆继续看，
最后知道真相的我眼泪掉下来
2009.02.12.和这套书里的主人公一起长大了。在小说全部读完之后，有一种终于长大和童年告别的不舍感。故事里的大家都
结束了
我怎么会没标过已读……（我永远爱哈利波特）
总感觉伏地魔太蠢了，也罢，谁让他是儿童文学里的坏人呢。。。7本哈利波特，到此，终于全部读完了
啊居然就完了就完了=- -!
扣一分是因为赫敏居然和罗恩在一起了，为什么不是克鲁姆
结束了青春童话的一部份。
十一月重读哈利波特计划，顺利完成！看完只想说：哈利波特万岁！五年后再见!
终于，终于。
7
这本就俗套了呢
二刷。听到快大决战的部分已经半夜，想着不看完今晚是睡不着了，就拿起手机开始接着看。真的很感谢罗琳给的这个结局，
All was fine. And it's my final final.
纠结的Snape啊。。。
Severus
阿姨您太同人了
超帅超漂亮超好看..
结尾有点仓促..
哎
陪伴我成长的故事……真的完结了……
七本都看鸟。。
！！！还我弗雷特！！！…….
十年后才看到大结局。失望。
狗尾~
呀。终于把7本哈利波特看完了。
结局不喜欢
大概算是一个俗套而……美好的结局

没有原先疯狂的感觉了，只是顺着原先的惯性吧
小时候读到废寝忘食的哇
星云雨果合集（下）
人物形象终于完整，可惜情节安排凌乱。
看过英文版之后就对中文版不那么感兴趣了……
悲情的斯内普。“那我的灵魂呢？邓布利多？我的呢”当属全书最有分量的话，沉重地压在我心上。
闹闹哄哄的，终于收场了，如释重负。
闭着目承认故事看完
2010年我最期盼的一部电影就是哈7，因为这么多年，他们长大了，我也长大了，我只想看看他们三个现在怎样了，我去到电
终于结束了。。
我读得太快了，完全是忽略细节在追剧情。这本书是该系列信息量最大的一本，罗琳阿姨写这本书算是一种大爆发，把这么多
益

全家是情圣
原作文字的力量远远超过电影的影像。爱与勇气能够战胜死亡，这种仿佛说腻了的陈腔滥调在经历了生离死别之后，随着少年
大结局
这本哈利波特是我唯一一本花了一年时间才读完的……因为中途是高考，加上我对哈利波特的感情在坠机般的下降
@20110802
每到暑假就忍不住重温一下。。
书确实比电影精彩多了，不过偶们不能拿书要求电影不是。。。后三部很好，链接成一个宏大的故事～
这就结束了啊...太感慨了。。。呃看完英文版再看中文就忍不住一直在挑刺。。。
补标。十年前吧。当时特别喜欢霍格沃茨保卫战，里面有太多太多的细节了，韦斯莱家真的是珍宝一样的存在。唯一的不解就
十九年后。
邓布列多是个计划通啊 和斯内普部署的计划一步一步实现 甚至牺牲自己去打败伏地魔 而哈利在这个过程中也着实很痛苦 因
15岁以前每年最开心事的就是等待哈利波特!
就这么 就这么 完结了...
小学运动会，YZF同意周末借给我回家看，我超开心。午后读完，不舍合卷。
意犹未尽。几个比较意外的点：与德思礼一家分别时刻，达力意外的感恩；赫敏的S.P.E.W终于在最后一刻发挥了作用，多比
终于看完了，弥补青春的回忆，做一个勇敢正直的人。
终于重读完了，啊，小时候记忆深刻的点（which is哭点），仍然让人热泪盈眶啊，就是纳威，我是真的很高兴看到平凡的小
2000年出版，七年后完结。我却故意把整个故事的结局又拖了六年。不过，一切的冒险还是成长都终于结束了。Blessing u
补上这堂魔法课
终于看完了
出一本，买一本，每年暑假拿出来重温一遍。儿童文学的老师说，它不具有文学性，但那又怎样呢？斯内普真是大逆转啊~
作为一个结局，罗琳阿姨有一点用力过度了……
童年结束了。虽然第七部构思很奇妙，布局也很优秀，但是总觉得很遗憾，哪里缺了一点，反而没有小时候的感动。可能变的
补，我读书很少哭，但第一遍读到海格抱着哈利走在草地上的情节，不禁流泪。书上还留着十几年前的泪痕。
伟大之处在于西弗勒斯的塑造，魂器的呼应，魔幻界最精彩的战争之一，不输魔戒
=v=谢谢帮我买到它的人。熬夜也要看完的，很舍不得的故事。

死亡圣器的故事是完满的结局，那么多鲜活可爱的角色，那么多九曲回肠的传奇，伴随着自高自大丑态尽显的伏地魔找到了自
尼玛再吼一次，罗琳你TM少弄死个人，你书会卖不出去么！！
宏大的多卷魔咒书终于落幕了，每个人物都有详细且合理的交待安排，哈利也成家生子了。。。不知道会不会有续集呢?
|4497:2456|
我居然没有标记这本我已经读过?!
花了7年终于完结　JK罗琳把一个童话故事变成了现实小说...顺便缅怀我的青春小鸟一去不复还
大爱
过了这么多年，终于读了。
陪伴着我长大的一个系列,看完之后一时间有点惆怅
终于狠下心鼓起勇气把最后的大决战看了，果然电影还是只能自己在屋里下下来看，看书时候哭得那个造孽哟~
用重读缅怀流逝的十年
和青春一起的回忆划上句号～～
故事的终结，哈利的十七岁和我的十七岁，一起结束了。我还记得中译版没出来我不自量力啃了几页原文，因为当时英文不济
跟了这么多年，哈利.波特系列总算尘埃落定。最后一本少了些悬念，最后的决战也比较仓促，不知是期望过高还是怎地。反正
厄里斯魔镜——献给霍格沃兹最伟大的校长。
不希望结束
从童年一直延续至今的追踪终于有了一个答案～
最后的最后，好大团圆哪。那么多人不在了，但他们三个还是团圆了。
结局个人认为不好…不过作为流行青少小说，这样就够了。
直到最后的最后
这个结局在侮辱我的智商
读过了之后有一点小失望。也许原版的能给人更美好的阅读感? 所以还是需要好好修炼英语==粉丝们猜对了结尾的大半，而过
结束
我不得不承认，那一只牝鹿把我感动了。
邪恶终究没有办法战胜正义
（T）一塌糊涂，无论读者抑或作者还是都需放平心态，将其当做儿童文学吧
尽管结局你不能免俗哇
地铁上啃完，俨然没了十年前初见的marvelous 的感觉。
谁来和我聊聊这个故事……看完没人交流很痛苦
还好，最后有点拖拉，在自圆其说。
终于完结了，看的心头一揪一揪。我心爱的双胞胎和斯内普。。。人物众多，看的有点晕，想构架起庞大的人物框架但是对人
终于让我给看到了..
Never end
向罗琳致敬。
英文版看得真爽!
斯内普
受不了那谁和那谁还有那谁的死..
虽然颇有非议，终于是这样结束了。
为什么非要写点评，疼讯好烦。

终于读完最后一部了,可以说是一个了结.值.
对于赫敏没有和马尔福在一起，我耿耿于怀。而且最后对马尔福德归宿寥寥带过，让我有点伤心哪~
读了英文的，不准备读中文的了
再折磨一次要疯了
结局..我我我不算太喜欢
Mark高三时偷偷看? 还有个同学看哭了
结局有点仓促，不过还好
8年时间，终于终结
果然没有看错斯内普。像亮司一样的男人。
什么都不用说了吧
曾看了三遍　（虽然还是记不住名字= =
高进同学给背回来的原版.....我还是没有等到小天狼星复活.....sigh
哈利波特系列的最终升华!
这是一个读过青春的童话啊~
哈利都谢顶了~
高三看完，结局还是很温暖，导向正确。能写出来是多么幸福的事情。
结尾颠覆了儿童读物的感觉……
最惊险最传奇的一本
精彩，动人，忠于哈利的人会得到特别的感动
最后的一部，最爱的一部，最光明的一部，最不舍的一部。
终于写完了~很好看~我觉得后面写的比电影的好看~
那个十九年。。。
伏地魔死的很简单啊。。大概是罗琳想就此停手吧 其实还有点为之惋惜呢~
狠心的作者最讨厌了
最后一本啦……当时第一时间冲到书店去买，买回来还不敢看，因为看完，就再也没有续集可期待了…现在连电影的最后一集
爱上了斯内普!
信息量最大的一本，很多内容突然展开，匆匆交代，不如之前严谨。不过依旧是个不错的结局，文字比电影激动人心。
最後被愛與力量包圍。熱淚盈眶。
全部看完了~~最喜欢第三部和这个最后一部。第三部胜在结构，最后这部胜在节奏。总算让我感觉不拖沓了呀~~看完对邓布利
又翻出来看了一遍，斯内普的回忆真是最后一本的精髓所在。虽然谈不上哈利迷，但是七本书我也完整看了下来，有四本我买
补标
我给一百颗星都不满足啊
中文译的一般
着魔
差不多十年，终于读完它。
结局就是 邓布利多下了盘很大的棋 自己都没了 还能把伏地魔将死
..永远的Harry
怎么说..结束了呢...就五星吧...
看了前面的就应该这么看下来。

我爱斯里普!
其实我希望喜欢的书在结尾之前戛然而止
圆满了，升天了!
郁闷……接受现实……
最近，很哈迷
再强大的人都有弱点。十三年，终于看完哈利波特系列的所有电影和小说。
暑假势必要赶在八月前再温一边罗
又回到当初看武侠小说那时候的紧张刺激，以及浪费时间的堕落感。
家庭式结局。
圆满了~ 一切圆满了~
六年的等待....完结了....
一部不错的儿童读物，不过我现在已经超过了看这部书的年龄了，哈利波特七部我都看了，很精彩。可惜jk罗琳已经走了，不
西弗勒斯
哭得一塌糊涂``哦 斯内普
精彩的结局，邓布利多
我这个伪哈迷终于看完了整套HP。
双胞胎居然死了一个。泪流。
==
good!!
完结篇。羽翼丰满的结局。
最后一部了。仍然大爱。哈利和我一起长大了，哈哈
真不舍啊，就这么结束了。
喧哗散尽，最终还是归于流水般的日子里
其实那个结局我并不是很喜欢，我只是庆幸我从第一本开始还是小学生的时候就心心念念喜欢的斯内普他真的从头到脚都是让
结束这本书。结束一段生活。
可怜我的那么多人都死了.....
敬Severus Snape。临死前对Harry轻声说“看着我”，是为了再看一眼那双和Lily一样的眼睛。
落幕。
9/150 深深地感觉伏地魔太弱了。我要是伏地魔早把这三个死小孩弄死了。
之前看都觉得哈利波特是有趣的作品，看到完就觉得，这是一部伟大的作品
光阴的故事
7
终于看完了，第七部太沉重了~结尾太草率了~
Farewell, Harry
不让主角死亡，不让读者失望。
从初中读到大学
不管怎样，都是我们结束的青春
一九九七至二零零七，十年成長。
早早地在99上预订了它。寄来的时候却等了好多天才打开来看，，因为一直骗自己：只要我没看完它，HP就没有完结。还好，

第七部!!盼星星,盼月亮,总算是盼来了，不过感觉第七部的翻译有点搞笑!"你太有才了!说时迟,那时快!"
这也许是我翻过次数最多的一套书籍
虽然有点俗。。。不过相对还是有精彩的~
原来不精彩的前几本儿都是铺垫!
就这样结束了。
我忘记了所有人 却只记得你
我背叛了所有人 却只忠于你

等待 不是为了你能回来
而是让自己找个借口 不离开
----致永远的混血王子 斯内普
哈7。结尾没太弄明白啊，哈利波特和伏地魔弄成了你中有我，我中有你的局面，那不是应该同归于尽吗？为什么最后又是光明
决战伏地魔···为了斯内普···
重温完成！真好看!
也许是我长得太大了 也许只是因为他完结了 也许是因为我引以为傲的想象力在一点点的丧失 这本没有前面六本精彩
最后一部了啊～不知道是不是以后都没得看了～
哈哈，恨了七年的斯大人终于可以一抿恩仇了~想想就爽!
青春岁月就这么结束了
就等电影上映了！！
最终回
其实一套都读了，但，不满意结局。妈的，跟金妮结婚算什么？！
大爱 我是波特迷 完结篇让我心里难过
隔了这么多年才看了最后一本。2014年开篇
其实我只是怀念有你相伴的初中高中大学。
第7部，最后一部。
和哈里波波一起成长的我们。
仅以这一部代表我所爱过的十年.
·····
看得电子书，与时俱进了一回！~发现了哈利波特父母都是水瓶座呢……那个罗琳大妈最后的最后还行吧……
5年的守侯
就像一个朋友一样陪我长大.....我終於完成我的諾言了,我要和哈利一起長大
好吧，结束。
谋害时间。
怅然若失.....但所有的时光都必须有终结的一天。
结尾弱了，整体还不错
这最后一本，真的很好看！不过因为比较长，好多人物的名字老记不住。。。
第七部
12.8.2010 六点醒来一直看到下午三点.....10年终于结束了
头尾都不错，精彩刺激，唯独败在中段冗长拖沓，有明显拖戏嫌疑。

话说真是舍不得就这么结束了
烂尾了，多年后简直多余。不过即使烂尾，也是真爱的一套书。觉得罗琳不会写爱情，哈利和金妮完全不相配，金妮人设也立
哈利强行不死感觉很别扭
终于看完了 2014看了整整一个春节 这集死了好多人
我的青春在我22岁时候终结。
尽管大结局其实可以从此前的很多蛛丝马迹猜测到，但最后的大场面还是挺震撼。此外这一部里我很喜欢一些很小很细节的情
best of the seven
两天看完的 非常好看。
好了好了，终于一个时代结束了。
这部居然没标 当年一部部追下来的小说啊
一直爱它。终于结束。
无限好看~
十年历程完美谢幕。
四颗星星献给斯内普教授
郑州购书中心。总算还好，比5、6强。
终于结束了……虽然有些感慨，但总算是结束了
看完了才知道心痛
这本拖的时间太长 我都不记得自己是不是完整的读完了 作为十年的结尾 哈利波特和我们一起成长刻
这是一代人共同的回忆``
这本明显就是大结局前的铺垫
最好的一部。终于结束了。
哈哈，这也是问那朋友借的，但我想这本就不打算还她了，因为这本正好凑成全套^_^满足我小小的私心
终于结束了.还好.
终于看完了，这一路走了这么多年。
给5星，并不为内容……
我开始后悔我只看了一半了。。。
这本我没点么- -
最近闲着没事干又读了第五遍
结束了。总是有一点点解脱，又有一点点失落，甚至失望。总的来说，收尾收的还可以吧，虽然在高标准上来说一般般。。。
终于完结了…我都快大学毕业了
精彩
马马虎虎总算看完这个故事了，写的可以
终于完结了。还是偏于流俗了。
洗白了的斯内普，再不会回来的人们。也许从凤凰社开始，故事就已经渐渐不再吸引我了。P.S.“他们说他在之后的几年里显
罗琳YYDS！虽然杀了我好多次！
词条变了还是怎么着，之前标记的竟然是台版。。。。
罗琳大妈的各种逻辑性啊~~
二刷：上次看觉得最后一部太黑暗，然后二刷时却格外喜欢这最后一本了，只因最后每一个小角色带来的感动，和经历了那么
[当然是发生在你脑子里的事。但为什么那就意味着不是真的呢。] 【阿不思·西弗勒斯·波特】

总感觉有点bug
哭到无法自已，就连伏地魔死都觉得伤心
忍不住抢在小朋友前面看完了 :) 很好的「英雄教育」！从古典套路里塞进了人文主义和后现代的独立思维。回想从第一本开
相当精彩！结尾是颗老鼠屎·····
my love
即便是各种故弄玄虚，各种人物杂乱，各种没事儿找事儿。。。还是结束了，故事在最后也达到了高潮，现在回头想更多的是
两天才看完，确实是很长，结局没有出乎意料但是我也没全部猜出来。一直不忍心看这最后一本，但真的开看还是蛮吸引的。
还是觉得最后收不住了，斯内普的记忆那章感情过于集中，好像一下子要把他带入好人的境界，而忘却了他本来就是个好人。
哈利彻底变成了不死小强，孩子的名字很有意义却十分难听。
熬夜看完,但很后悔不能慢慢地仔细地看，好像一段长达六年的婚姻突然结束的感觉
最近在讀
完结。 从此不再有Harry Potter
Harry长大了，我也长大了。
真不希望结局。。。
我们的时代印记
十分仓促的一个结局
大结局！还算圆满吧
翻译好差
我只是觉得把前几本都加上来很累。。。
没理由。我喜欢那个小男孩。
伴我度过童年
双胞胎的结局和教授，卢平的死看得我太难受了。想起死圣出来的那天，整个下午都在书店那里等着。拿到书就迫不及待地回
我是哈利波特迷：）
1997-2007
细算起来，从初中到现在，已经是哈7部一个系列了，真快，时间走的真快这牵强的结局，给人狗尾续貂的感觉，错觉么？还是
最可怕的事情永远不是死亡，还有太多太多比死亡可怕的东西，面对死亡所展示出的勇气，才是hp最为动人的品质。
终于来到最后一本。出版时我都高三啦，第一次装逼的买了英文原版。“This is war.”5555我的卢平唐克斯……这么算起
电影很少能超越书籍啊
初中结束时的暑假仅仅读完了前几章，到现在我电影也没有看 就等待着有一天把这本书读完 现在哈利波特的故事在我的世界
被多比 克利切 西弗勒斯 感动！
这下是最后本了。。。
还是喜欢以前的故事，那个时候哈利没有变得这样高大全，罗恩赫敏可以常常拌嘴，卢平还是忧郁得帅呆，邓布利多依然睿智
这套书承载了我成长的记忆。之前零散的买，新出了就看，和现在追美剧好像挺像的。不记得多少岁生日的时候，小姨送了我
迷失在霍格沃茨。
All stories come to an end. Like never before, like never again.
終わった
向J.K.罗琳致敬！
不学习光看小说！
補標記

本科借pz送给badger的那本
我是哈迷，怎样~
七本结尾能收成这样算不错了.只是翻译差了点，许多词句只是生硬的直译，没有美感。虽然封面上依然印了那两个译者的名字
斯内普~~~~~~
我的青春结束了
好故事
大结局是如此让人感动！看了7年！好像做完了一项艰辛的功课！终于不用再读了！！很好
后悔看了剧透
终结。其实不太推荐这类魔幻灵异小说，只是从第一部开始，便产生了惯性。终于，结束了。
整个系列最精彩的一本
圆满了
终于完结了
爱了十年。
看完这本书时的感受我已经完全忘记了，但我永远记得这个系列和你。
五星
最后的大战全员参加实在是太！帅！了！连霍格沃兹的石像都冲了出去实在是太赚眼泪了。。但合起书来想想 哈利从头到底的
大结局，不过不喜欢罗恩和赫敏好
最爱
因为自己翻译过这本书
所以觉得 还是读原版的好啊~!
原版的 可以自己揣摩作者的意思~嗯
结局就已经是22成双了 诶~ 死的人多了点 现在想想 觉得斯内普真不容易。。。
死的死，伤的伤，最终却是这样的结局很狗血不是？！高中一年级读~
我爱她的创造力
我擦！斯内普才是男主角啊！
强烈期待电影版大结局!!!整个系列看了三个月终于全看完了
我是哈迷
写死了这么多人 减你一 哼 不过还是看哭了吧 还是空落落的吧 陪伴了自己这么多年的让人憧憬的世界
终于到了最后。。看不下去的伤痛。还是看了下去。
万人瞩目下终于完成了，为什么我还是很想哭呢。
任俊妍借的,哈利.波特真是太棒了!
2009.12.16
致斯殿以敬意。
大二了，看到倒数第二章痛哭一晚
完美的结局,北京晚报无耻的剧透,王府井书店无理的当众朗读,儿时梦想的完结
毕竟还是童话斯内普完美的诠释了伟大二字
最终版，好舍不得
哈里波特就是好就是好就是好
向伟大的西弗勒斯·斯内普致以我最崇高的无产阶级文化大敬礼!

10年“魔法时代”的终结
我属于最早读过的一批人，这个系列对我们这代人已经宣告终结，但影响始终在~~~~~~~~
不管怎样。。陪伴我少年时光。。一个尘埃落定
够煽。
越读越失望。阅读兴趣明显下滑
斯内普最高！金妮路人去死去死
结局还比较合我的意 只是我还需要在从1好好看几遍 恩
完结了。
只是有些恋爱关系真的很雷……
好喜欢~~~
第一次都哈利波特，那是七年前了吧，今天终于读完了最后一本，才发现，它根本不是什么魔幻小说，它不是在说魔法，而是
最终！！！童年结束！！！
以前忘标记了。
很混沌的一个架构，然而从这里出发讨论什么都行，不过要是写成《追踪1789》就没意思了。 14.12.12 补：其实现在想来
居然没有标注...
曲终。
从初中到大学~
生命的意义不在追求长生，而在接受死亡。陪伴一代人成长的哈利波特，感谢书中呈现的爱与温暖
最后一本居然看哭了。
终于有机会读了。十年啊十年!
长大后看的。终于把哈利波特都看完了。很高兴罗恩和赫敏最终走在一起，想不到斯内普竟然隐藏最深。谢谢小说内所有角色
最后还是没有买正版，在网友那要来的电子版。通宵达旦的看完，最后期待电影版的结尾。貌似罗琳还要续貂？道听途说而已
原来初中看到高中！！ 最爱的魔法小说
【如果说这是种信仰。】
【完结篇。】
忠于哈利到最后的最后。
我读它，是为了回忆起那个温暖的感觉，为了想起你们，想起我们的纯真年代。大妈卢娜胡萝卜头，大爹莉莉，小妈，赫敏，
哎？读过竟然连封面都不记得了.....大概因为不像前五部都被我翻烂了，第六、七部只看了一遍，总得承认，every stor
好多年了
哈利波特老了……
俺的整个青春期啊
伴随我的少年时代
买来第二天就看完了~赞!
小斯...你为什么那么傻...
啊啊啊啊啊啊啊啊啊啊！！！？？ 得赶紧去买咯……
其实我当初看的网翻版，想看下纸质的了。
总算终结了
精彩!爱不释手啊
就这么结束了……有点失落，有点不舍，罗琳用了一个极长的铺垫，结束还是没有预想的精彩，阅读在洛夫古德家关于圣器那

从第一本就坚信斯内普是卧底!
看哭了呢
那个叫做斯内普的男人
就这样结束了吗
原来除了坏人之外都是好人,极坏的只是少数人而已呢.
剧情要素十分丰富，做为最后一本十分妥当
结局了，像是硬生生断掉了过去的十多年时光。刚看完情绪还没缓和时写了长评（其实算是批评啦，汗）如今再看，又是一种
还算满意的结局
真相是无间道式的XD
终于结束了·~~~结局虽然在意料之中，中间却蛮出彩的
破烂电影改了好多啊啊啊啊啊! 是要多节省成本啊!!!
不能用眼的日子听了混血王子和死亡圣器的有声小说，爱，正直，勇敢，以后要当作睡前故事读给宝宝听。更新。201911重读
就是喜欢、
哈利波特，多比来救你了。
觉得这本没有前几本翻译的好了
看手机盗版的时候我还觉得离它出版的日子很遥远,一眨眼它就来了,好象不曾等待一样
再看一遍还是觉得疼，斯内普教授啊斯内普教授。。
找不到中文版了OTZ
看完这本书，——虽然是在这么久这么久之后，我还是莫名有种从死里复活的感觉……
终于读完这本，屡有玩RPG之感
逃亡旅行颇让我想起琼瑶奶奶的大作=v=
没有想到的太多了
横跨我整个学生时代的系列 终于在我大一时期结束了
再见，波特。
还是到了结局，爱，始终都在。。。。
1.邓不利多费了那么大劲才搞到的灵魂碎片，但RAB却如此轻松。2.看来老邓死后，会出现新的老大，来继续帮助哈利波特。
拖了一年，才终于看了，唉。
每个孩子心里都住着一个Hogwards，几乎满足了我们对魔幻世界所有的想象。读完第七部，是件令人无限伤感的事。
看完的最大感慨就是斯内普太悲剧了。。
系列的收关之作。系列的精华之作——前提是你用了8年的时间来读它们……
各种热泪盈眶，斯内普说，拿走全部拿走，我就流泪了5555555555555
这两天把HP补完了，算了一下离上次看HP居然有八年了。。
十颗星。
这是关于孩子们成长的故事的一个里程碑，可以是结局，但我不希望是结局……
电影终于上映了，现在回想当初预定哈VII时的激动和对电影的期盼，始终都觉得好幸福哦~ 看完电影后，还是有些细节记不
我读完了，无论谁牺牲了谁无畏了谁战胜了，我都不感动。果然这个拖的时间太长了。
童年回忆啊，昨天在zan咖啡和燕燕小聚的时候看的，因为被剧透得差不多了所以没啥新鲜感，感觉这个系列还是密室最最棒
No story can be great without closure.
歸根結底，七部「哈利·波特」寫的不過是一個“愛”字。

所有的绳结都在哈7中完美的缠绕着，哈利波特系列里最迷人的一部……
哈利·波特十年历程完美谢幕，死亡圣器终揭神秘面纱
一九九七至二零零七，十年成長。
爱这整个系列。
纠结万分的大结局
J.K.写得最好的一本，没有辜负广大人民的厚望，哈哈
魔幻的盛宴，引人入胜。
消除了我对这本书的偏见。以前听别人讲起哈利波特的时候，总会觉得幼稚，我再也不这么看了。这是一本关于爱、勇气和恐
再看起来好快啊~
其实看到最后，看到斯内普，我们才发现，真正的主角全死光了。
事隔多年，我依然在读它，还好有结局。
整个故事告诉我们，做女人要行善积德，这样万一挂了，也还有人看在与你旧日的情分上罩着你儿子……
相当有阅读快感的一本书

觉得写得很棒，好多地方都有前后相衔接的，不容易。
童年
这本书里面我最喜欢的台词—“ 但是他知道的东西一直都少得可怜，哈利，这是伏地魔最没用的地方，他从来不费神去理解去
每部电影和书都是五星，没什么理由。这是十年。
T T终于看完了，真是一梦十年呐
童年记忆系列
完结了，有点失落：{
長追不捨~~
无论如何 我还是喜欢它的
结尾挺感人 尤其是 那个男孩的名字是阿不思 西弗勒思
第七本 让人摇头的结局
我最爱的鄧布利多死了。。
噢..读得好猛
哈利也完啦～～
《哈利波特与死亡圣器》应该是整个系列中写得最好的一部，这并不因为这是这个系列的终结，也不因为这是所有伏笔的谜底
终于几年之后还是看了哈七，当时拒绝相信热爱的人物死亡，现在也是。跳读完了这一本，也算一个结束。
看几遍哭几遍
Like never before.Like never again.In my heart,it never ends.
我不甚喜欢第七部QWQ
要想真正了解哈利波特，还是看书吧，电影的局限性让它不得不阉掉太多太多的精髓了...
这是一个关于成长的伟大故事!
高一读物，坐最后一排，和长期哈迷好友徐月濛一人抱一本，物理课读，第一次物理考50分。
哈利波特是伴随一代人成长的书吧。
一个不成熟男人的标志是能为了伟大的事业死去，而一个成熟男人的标志是恩呢该为了伟大的事业坚韧地活着。即使被误解、
我是否注定要知道，而不是去谋求？你是否会知道我会觉得这有多难？是否正因为如此，你才把它安排的如此困难，让我有时

不算童年时期了，好吧，青少年时期，重复看的次数最多的一个系列的书啊...魔幻大爱!
我用听的
陪我长大的外国童话
我竟然没添加这一项，太奇怪了……
看了我一个学期才看完的!
基本10年了。看前6本书的时候，我还不是一个英语专业的学生，我还处在懵懂的中学时期，好多事还没有发生。看完第7本，
中文翻译超烂。
想说终于结局了。
我从第二本开始，直到这一本结束，都没有等到德拉科的幸福。
多比 一个自由的小精灵
有人说它黑暗。可是，对于一个长大的孩子，世界仍然灿烂，便是美好。
最后一本，我看完了，我25岁了，第一本我还是从小学五年级开始看的，十四年过去了……
8.29 凌晨 一梦十二年
读过而且爱死了却没有标记的书实在是 太 多 啦
7:死亡圣器
居然没添加
重温的赶脚。
终于 落幕了 一切为了更伟大的……!!!
最感动的莫过于斯内普了。为他甚至有点讨厌邓不利多
第1部：从前有一个男孩叫哈利•波特，有一天，他抢了伏地魔最想要的石头。第2部：从前有一个男孩叫哈利•波特，有一天，
不得不說哈利·波特系列之所以如此成功的原因之一就是他不完美.有時候真的想給他一拳讓他清醒一下少自以爲是了,但這才
泪点来得太早也不够持久><但是结束了呀。最后10章保持在深夜档看的，在看之前被透露了下要死好多人。
当一切都结束，我爱斯内普，look...at...me
斯内普很爱莉莉。弗雷德死了。多比死了。邓布利多和哈利在车站的谈话。看到以上的片段情节，哭了。
比我想象中好很多~~~~
阿不思·西弗勒斯·波特
为了终结，画上一个句号，不管这句号圆不圆满。前半段的逃亡真的很压抑，后面的爆发还不够厉害，不过人倒是死了很多。
说实话没让我失望~
挨了很久不敢看哈7，就是担心所有的童年都会在看完的那一刻结束。
终究是逃不过自己的好奇，弄来了电子书疼着眼睛看完。

最终只能不去想，不去想。

关于结尾我猜过很多，所谓的正义是王道大概就是这个意思。

我祝所有所有的哈迷，一切都好。
这么多年了·~
最爱的逃亡故事。
即使看再多遍 依然会感动 它包含所有我需要学会的爱 勇气 力量 友谊 感谢RWOLING阿姨 使很多孩子相信 真的有魔法存在

时隔九年终于圆满。
罗琳卯足了劲啊