



同濟大學
TONGJI UNIVERSITY

《机器学习》课程
作业项目 1 说明文档
Description Document

项目名称：基于逻辑斯蒂回归的影评情感分类

学 院： 电子与信息工程学院

专 业： 计算机科学与技术

组 员：（第 3 组） 1652270 冯 舜

1651629 黄中昱

1651574 贾昊霖

1553449 王志业

授课老师： 李 洁

2019 年 04 月 08 日

目录

一、	引言.....	3
1.1	项目背景（IMDb）.....	3
1.2	定义.....	4
1.3	项目内容.....	5
二、	前期探索——使用决策树方法进行分类.....	5
2.1	原理概述.....	5
2.2	算法流程 ^[1]	6
2.3	判别决策树 ^[1]	6
2.4	扩展改进算法.....	7
2.5	局限性.....	7
三、	特征提取——基础：词袋模型.....	8
3.1	模型概述.....	8
3.2	实现过程.....	8
3.3	源程序代码.....	9
3.4	局限性.....	10
四、	特征提取——改进：Word2Vec.....	10
4.1	原理概述.....	11
4.2	实现过程.....	11
4.3	源程序代码.....	12
4.4	局限性.....	14
五、	特征提取——提升：Doc2Vec.....	14
5.1	原理概述.....	14
5.2	实现过程.....	15
5.3	源程序代码.....	16
5.4	结果展示.....	17
六、	逻辑斯蒂回归过程.....	19
6.1	原理概述.....	19
6.2	实现过程.....	20
6.3	源程序代码.....	21
6.4	结果展示.....	23
七、	最终结果.....	23
7.1	使用词袋法进行特征提取的结果.....	23
7.2	使用 Word2Vec 法进行特征提取的结果.....	26
7.3	使用 Doc2Vec 法进行特征提取的结果.....	28
7.4	使用 Doc2Vec 方法进行任意输入影评文本的情感分类.....	29
八、	可能的改进点.....	30
8.1	更好的向量化模型.....	30
8.2	Word2Vec 中更好的文档向量生成方法.....	30
8.3	逻辑斯蒂回归中的超参调整.....	31
九、	心得体会.....	31
一〇、	参考资料.....	31

一、 引言

1.1 项目背景（IMDb）

互联网电影资料库（Internet Movie Database，简称 IMDb）是一个关于电影演员、电影、电视节目、电视明星和电影制作的在线数据库。IMDb 的资料中包括了影片的众多信息、演员、片长、内容介绍、分级、评论等。对于电影的评分目前使用最多的就是 IMDb 评分。

IMDb 影评情感分类数据集（Large Movie Review Dataset，aclIMDb，<http://ai.stanford.edu/~amaas/data/sentiment/>）[2]是斯坦福大学收集的、用来进行影评情感二分类研究的数据集。在本次项目中，我们使用该数据集，通过逻辑斯蒂回归方法进行情感二分类。

		id	sentiment	review
1	0	0_3	0	might think good cinematography future great vilmos zsigmond future stars sally kirkland frederic forrest seen briefly
2	1	10000_4	0	hind even bit silly production bland direction help though film sunken plane boring lethargic followed concorde airport
3	2	10001_4	0	uthful exploits anybody anything else never convinced love princess disappointed movie forget nominated oscar judge
4	3	10002_1	0	een stapleton red dress dancing scene otherwise ripoff bergman fan f either think anyone says enjoyed hours well lying
5	4	10003_1	0	ce highly recommend one feeling little happy need something remind death otherwise let pretend film never happened
6	5	10004_3	0	palma excoriated ripping hitchcock suspense horror films robin wood view strange form cultural snobbery would agree
7	6	10005_3	0	c etc allen later serious films less embarrassing also far less entertaining take interiors woody rarely made funnier movie
8	7	10006_4	0	ighs cast whoopi goldberg judy tenuta instead predictable surprised find director five year old waste viewers actors well
9	8	10007_1	0	iscathed horrible anti comedy positive thing come mess charlie denise marriage hopefully effort produces better results
10	9	10008_2	0	like say twenty years ago great likely never get recognized lauren lily kristin boring see painful unless true harrelson fan
11	10	10009_1	0	y one could ever good enough redeem endless series flaws like three actresses watch something else movie worth time
12	11	1000_4	0	naked even nc version saw couple hours throw away want watch descent take nap instead probably interesting dreams
13	12	10010_3	0	t performance fight plot inadequacy job pretty good rest bearable though predictable whole watchable better tv shows
14	13	10011_3	0	iff upper lip higher social class sad walker becomes boring mess despite strong cast blame poor plot even worse pacing
15	14	10012_1	0	ent movie boy horrible time like getting ripped go rent movie people actually enjoyed movie like watch movie meaning
16	15	10013_1	0	ure regrettably dreadful movie lauren bacall pick good one expected better kristin scott thomas one definitely one miss
17	16	10014_2	0	medy horror teenage angst hot rod must sired mother car maybe deserves second viewing see accurate reflection time
18	17	10015_2	0	ide try old haunted house saving grace film ghost paul blaisdell creature suit turns work movie monster played blaisdell
19	18	10016_4	0	parrot well let rephrase talk like character quentin tarantino movie said things expect bird movie pure hokum harmless
20	19	10017_4	0	ar race actually takes place screen pre scooby doo styled unmasking makes sense however nostalgia buffs mindless fun
21	20	10018_3	0	characters unlikeable script awful waste talents deneuve auteuil
22	21	10019_3	0	half need many pages fill total im trying avoid seem extreme make mistake movie unwatchable matter decent first half
23	22	1001_4	0	interest besides point ends fairly bland movie overall invests everything idea basic story shocking compelling really pay
24	23	10020_3	0	li khan get due sanjay mishra manoj phawa yashpal sharma wasted tashan boring film films failure box office keep away
25	24	10021_2	0	ght remember watching anorexic kareena bikini reason give rating every time think seen worst bollywood proves wrong
26	25	10022_4	0	blockbuster start understand crowds already shunned one despite akshay kumar stealing show tashan could better whole

图一-1 IMDb 影评情感分类数据集（一）

```

ml@riv-Homeserver:~/jupyter/data/aclImdb/train/neg$ cat 0_3.txt ; echo
Story of a man who has unnatural feelings for a pig. Starts out with a opening scene that is a terrific example of absurd comedy. A formal orchestra audience is turned into an insane, violent mob by the crazy chanting of its singers. Unfortunately it stays absurd the WHOLE time with no general narrative eventually making it just too off putting. Even those from the era should be turned off. The cryptic dialogue would make Shakespeare seem easy to a third grader. On a technical level it's better than you might think with some good cinematography by future great Vilmos Zsigmond. Future stars Sally Kirkland and Frederic Forrest can be seen briefly.
ml@riv-Homeserver:~/jupyter/data/aclImdb/train/neg$ cat 1_1.txt ; echo
Robert DeNiro plays the most unbelievably intelligent illiterate of all time. This movie is so wasteful of talent, it is truly disgusting. The script is unbelievable. The dialog is unbelievable. Jane Fonda's character is a caricature of herself, and not a funny one. The movie moves at a snail's pace, is photographed in an ill-advised manner, and is insufferably preachy. It also plugs in every cliché in the book. Swoozie Kurtz is excellent in a supporting role, but so what?<br /><br />Equally annoying is this new IMDB rule of requiring ten lines for every review. When a movie is this worthless, it doesn't require ten lines of text to let other readers know that it is a waste of time and tape. Avoid this movie.
ml@riv-Homeserver:~/jupyter/data/aclImdb/train/neg$ cat 2_1.txt ; echo
I saw the capsule comment said "great acting." In my opinion, these are two great actors giving horrible performances, and with zero chemistry with one another, for a great director in his all-time worst effort. Robert De Niro has to be the most ingenious and insightful illiterate of all time. Jane Fonda's performance uncomfortably drifts all over the map as she clearly has no handle on this character, mostly because the character is so poorly written. Molasses-like would be too swift an adjective for this film's excruciating pacing. Although the film's intent is to be an uplifting story of curing illiteracy, watching it is a true "bummer." I give it 1 out of 10, truly one of the worst 20 movies for its budget level that I have ever seen.
ml@riv-Homeserver:~/jupyter/data/aclImdb/train/neg$

```

图一-2 IMDB 影评情感分类数据集（二）

1.2 定义

- 1) 互联网电影资料库（Internet Movie Database，简称 IMDb）
- 2) IMDB 影评情感分类数据集（Large Movie Review Dataset，简称 aclIMDb）
[2]
- 3) ID3 算法：一种基本的贪心算法，用来构造决策树。
- 4) C4.5 算法：由 Ross Quinlan 开发的用于产生决策树的算法，该算法是对 ID3 算法的一个扩展。
- 5) 逻辑回归（Logistic Regression，简称 LR）：一种广义的线性回归分析模型，常用于进行机器学习分类算法。
- 6) 词袋模型（Bag of Words，简称 BOW）：一种在自然语言处理和信息检索 (IR) 下被简化的表达模型。
- 7) Word2Vec：亦称作 Word Embeddings，指一群用来产生词向量的相关模型。
- 8) Doc2Vec：亦称作 Paragraph2Vec、Sentence Embeddings，可以获得 sentences/paragraphs/documents 的向量表达，是对 Word2Vec 的拓展。
- 9) 散点图（Scatter Plot）：回归分析中，数据点在直角坐标系平面上的分布图。本文使用不同颜色的点代表不同类别的样本，以期从主观的区分程度判断特征提取方法的好坏。

- 10) 混淆矩阵 (Confusion Matrix): 亦称误差矩阵, 以 n 行 n 列的矩阵形式表示精度评价的一种标准格式。
- 11) 受试者工作特征曲线 (Receiver Operating Characteristic curve, 简称 ROC 曲线): 受试者在特定刺激条件下由于采用不同的判断标准得出的不同结果画出的曲线。在机器学习分类领域, 纵轴常为真正率、横轴常为假正率, 参数为阈值。
- 12) AUC (Area Under the Curve): 本文中指 ROC 曲线与横坐标轴围成的区域。该区域在机器学习分类领域常用于度量模型能在多大程度上区分开不同类的实例。AUC 越接近 1, 模型表现越好。
- 13) 词云: 对网络文本中出现频率较高的“关键词”予以视觉上的突出, 过滤掉大量的文本信息, 使浏览网页者一眼扫过文本就可以领略文本的主旨

1.3 项目内容

- 使用不同的已有黑盒模型, 对于 `aclIMDb` 数据集的文本进行特征提取。
- 使用逻辑斯蒂回归, 对于提取后的特征向量进行机器学习分类, 做到对于 IMDb 评论的正面、负面情感二分类。

注: 可以在 Github 仓库 (<https://github.com/Harrypotterrrr/Machine-learning-course/tree/master/1-logistic-imdb>) 的三个笔记本内查看所有代码和输出。

二、 前期探索——使用决策树方法进行分类

2.1 原理概述

决策树 (Decision Tree) 是在已知各种情况发生概率的基础上, 通过构成决策树来求取净现值的期望值大于等于零的概率, 评价项目风险, 判断其可行性的决策分析方法, 是直观运用概率分析的一种图解法。由于这种决策分支画成图形很像一棵树的枝干, 故称决策树。在机器学习中, 决策树是一个预测模型, 其代表的是对象属性与对象值之间的一种映射关系。

决策树是一种树形结构, 其中每个内部节点表示一个属性上的测试, 每个分支代表一个测试输出, 每个叶节点代表一种类别。

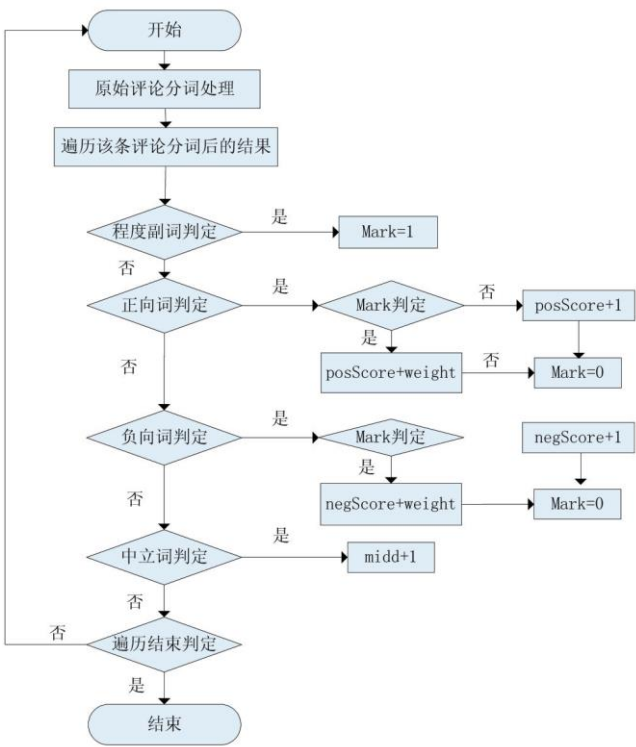
ID3 算法是一种用来构造决策树的最基础的贪心算法。ID3 算法起源于概念

学习系统 (CLS)，以信息熵的下降速度为选取测试属性的标准，即在每个节点选取还尚未被用来划分的具有最高信息增益的属性作为划分标准，然后继续这个过程，直到生成的决策树能完美分类训练样例。

2.2 算法流程^[1]

将网民的一条评价作为一个基准点，切词后进行情感词识别，在流程处理前每条评论的正向情感分数，负向情感分数，以及总分数都为 0。在遍历过程中，如果程度副词之后出现了情感词，则该次匹配到的情感词分数为原始情感分数乘以程度副词权值，比如“观赏此部电影真是一个十分享受的过程”，其中程度副词“十分”之后出现了正向情感词“享受”，则该次匹配的正向情感分数将从原来的+1 变为+2。

图二-1 是根据扩词后的情感词典，中立词词典以及程度副词词表进行情感属性提取的流程图。其中 mark 代表加权标志位，posScore 代表正向情感分数，negScore 代表负向情感分数，weight 代表加权后的分数。

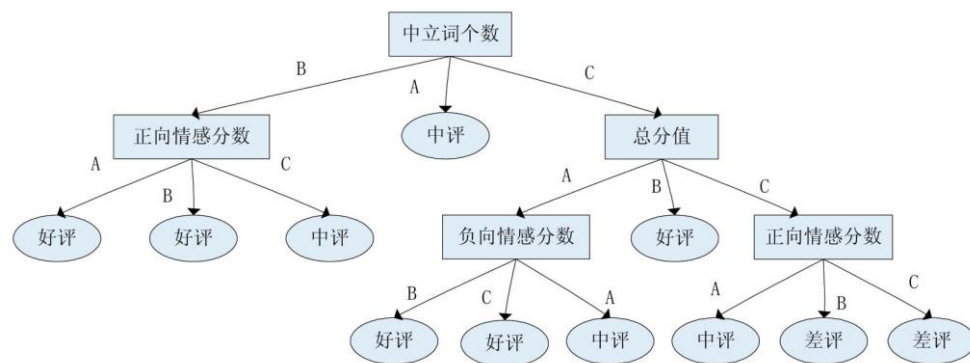


图二-1 属性提取流程图

2.3 判别决策树^[1]

从文献[1]中 2.3 节摘取使用 ID3 算法构造的评论情感判别决策树如图二-2

所示：



图二-2 评论情感判别决策树

2.4 扩展改进算法

- C4.5 算法
- 启发式搜索
- 错误率降低修剪、规则后修剪
- 增益比率、分裂信息

2.5 局限性

综合以上内容，我们认为决策树算法改进空间非常大，而相关方面参考资料较少。若希望构造广泛适用的决策树，针对一些特殊情况，比如反讽、欲扬先抑等风格类型，最终形成的决策树将会非常庞大，结果也难以可视化，因此最终**放弃选用决策树模型**。

考虑到决策树的诸多局限性，最终我们决定采用从文本中提取特征后，使用逻辑斯蒂回归的方法进行影评情感分类。



图二-3 项目方法改进

三、 特征提取——基础：词袋模型

3.1 模型概述

将所有词语装进一个袋子里，不考虑其词法和语序的问题，即每个词语都是独立的。例如下面 2 个例句，就可以构成一个词袋，袋子里包括 Jane、wants、to、go、Shenzhen、Bob、Shanghai。

Jane wants to go to Shenzhen.

Bob wants to go to Shanghai.

假设建立一个数组（或词典）用于映射匹配。

[Jane, wants, to, go, Shenzhen, Bob, Shanghai]

那么上面两个例句就可以用以下两个向量表示，对应的下标与映射数组的下标相匹配，其值为该词语出现的次数

[1,1,2,1,1,0,0]

[0,1,2,1,0,1,1]

这两个词频向量就是词袋模型，这种模型的缺点是语序关系已经完全丢失

3.2 实现过程

在 Anaconda Prompt 下，通过 pip 命令安装 sklearn 库。

我们将所有的影评按照 sklearn 读取数据的协议放置到 neg 和 pos 文件夹中。调用 load_files 读取所有的影评。利用 sklearn 中专门处理 NLP 问题的库

CountVectorizer，以上面的训练数据为参数，调用 CountVectorizer 对象，可以以每个单词的词频统计信息为基础，得到一个特征向量构成的矩阵。其中每列表示一词（实为 2-grams，即每两个连续单词，看作一个独特的单词），每行表示一训练实例。矩阵的元素，则表示了某实例中，某词的词频。矩阵对应的词汇表中共有 129549 个单词（词组）。我们后续的工作在这个矩阵上展开。

```
词汇表大小: 129549
X_train(训练用词袋特征):
<25000x129549 sparse matrix of type '<class 'numpy.int64'>'
  with 3607330 stored elements in Compressed Sparse Row format>
X_test(测试用词袋特征):
<25000x129549 sparse matrix of type '<class 'numpy.int64'>'
  with 3392376 stored elements in Compressed Sparse Row format>
特征数量: 129549
```

图三-1 从语料库生成的词袋特征

实际上，每条评论按照上述简介中的方法，将会变为一个句子向量，可以想象到这是一个及其稀疏的向量。将该向量作为特征输入逻辑斯蒂模型来训练我们的模型。同时绘制 ROG 曲线，混淆矩阵，降维后的分类图像。

3.3 源程序代码

由于代码量较大，因此只展示部分代码，完整代码详见链接地址：

<https://github.com/Harrypotterrrr/Machine-learning-course/tree/master/1-logistic-imdb>

```
[2]: from sklearn.feature_extraction.text import CountVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.datasets import load_files
from sklearn.model_selection import GridSearchCV
import numpy as np
import mglearn
import matplotlib.pyplot as plt
reviews_train = load_files("data/aclImdb/train/")
unsup_index = reviews_train.target_names.index('unsup')
text_train = []
y_train = []
for t, y in zip(reviews_train.data, reviews_train.target):
    if y != unsup_index:
        text_train.append(t)
        y_train.append(y)

print("训练用评论: {}".format(len(text_train)))
print("训练用评论(按pos/neg分类): {}".format(np.bincount(y_train)))
```

训练用评论: 25000
训练用评论(按pos/neg分类): [12500 12500]

```
[3]: reviews_test = load_files("data/aclImdb/test/")

text_test = reviews_test.data
y_test = reviews_test.target

print("测试用评论: {}".format(len(text_test)))
print("测试用评论(按pos/neg分类): {}".format(np.bincount(y_test)))
```

测试用评论: 25000
测试用评论(按pos/neg分类): [12500 12500]

```
[4]: vect = CountVectorizer(min_df=5, ngram_range=(2, 2))
X_train = vect.fit(text_train).transform(text_train)
X_test = vect.transform(text_test)

print("词汇表大小: {}".format(len(vect.vocabulary_)))
print("X_train(训练用词袋特征): \n{}".format(repr(X_train)))
print("X_test(测试用词袋特征): \n{}".format(repr(X_test)))
```

3.4 局限性

- 没有考虑单词的顺序
- 忽略了单词的语义信息
- 忽略了上下文语境
- 对于短文本效果很差，对于长文本效果一般

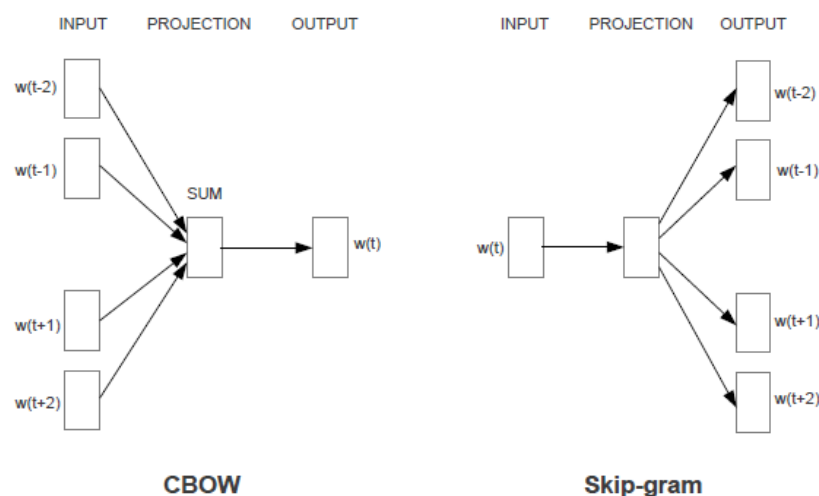
四、 特征提取——改进：Word2Vec

词袋模型的局限性在于忽略了上下文语境，单个单词仅仅能表示该单词本身。2013 年，Google 团队开发了 Word2Vec 新工具，既能获取词的语境，同时又减少了数据大小，通过利用基本的代数式来挖掘词的关系（例如：“king” - “man” + “woman” = “queen”）。词向量可以作为分类算法的输入来预测情感，相当程度上

解决了词袋模型存在的缺点。

4.1 原理概述

Word2Vec 主要包含两个模型：CBOW（Continuous Bag of Words，连续词袋模型）和 Skip-gram（跳字模型）。对于 CBOW，通过给定一组邻近的单词来预测单独的单词。而 Skip-gram 则相反，通过给定一个单独的单词来预测某个范围的单词。两个方法都使用人工神经网络（Artificial Neural Networks）作为它们的分类算法，词汇表中的每个单词都是随机的 N 维向量。在训练过程中，算法会利用 CBOW 或者 Skip-gram 来学习每个单词的最优向量。



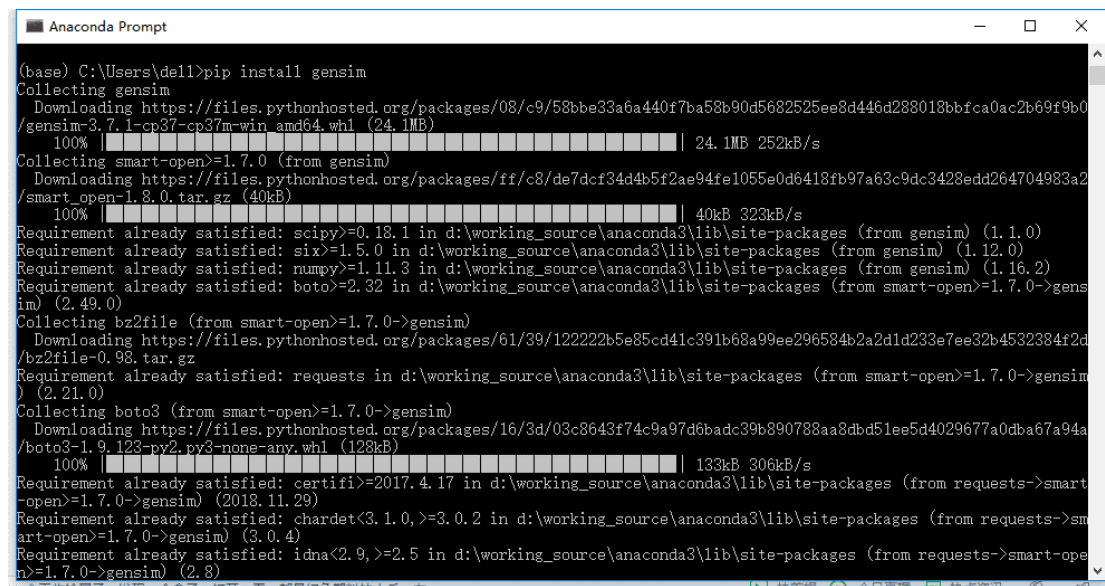
图四-1 CBOW 与 Skip-gram 结构图^[4]

4.2 实现过程

在 Anaconda Prompt 下，通过 pip 命令安装 gensim 库，并利用 gensim 库在影评数据集中训练词向量：

1. 我们直接使用现有的基于 Google News 训练好的 Word2Vec 模型（实际只使用其词向量）。
2. 将输入文本的每个词，使用 Word2Vec 模型映射到词向量；再采用 Average Word Vectors 方法，简单的对句子中的所有词向量取平均。
3. 对我们的数据集进行缩放，大于均值则为积极，小于则为消极。
4. 开始训练分类器，对逻辑回归使用随机梯度下降算法。
5. 通过计算测试数据的预测精度来验证分类器，同时绘制混淆矩阵、测试它们的 ROC 曲线。

由于这些向量有很多个维度，因此使用 PCA 降维算法在 2D 平面上可视化。



```
(base) C:\Users\del1>pip install gensim
Collecting gensim
  Downloading https://files.pythonhosted.org/packages/08/c9/58bbe33a6a440f7ba58b90d5682525ee8d446d288018bbfca0ac2b69f9b0/gensim-3.7.1-cp37-cp37m-win_amd64.whl (24.1MB)
100% |#####| 24.1MB 252kB/s
Collecting smart-open>=1.7.0 (from gensim)
  Downloading https://files.pythonhosted.org/packages/ff/c8/de7dcf34d4b5f2ae94fe1055e0d6418fb97a63c9dc3428edd264704983a2/smart_open-1.8.0.tar.gz (40kB)
100% |#####| 40kB 323kB/s
Requirement already satisfied: scipy>=0.18.1 in d:\working_source\anaconda3\lib\site-packages (from gensim) (1.1.0)
Requirement already satisfied: six>=1.5.0 in d:\working_source\anaconda3\lib\site-packages (from gensim) (1.12.0)
Requirement already satisfied: numpy>=1.11.3 in d:\working_source\anaconda3\lib\site-packages (from gensim) (1.16.2)
Requirement already satisfied: boto>=2.32 in d:\working_source\anaconda3\lib\site-packages (from smart-open>=1.7.0->gensim) (2.49.0)
Collecting bz2file (from smart-open>=1.7.0->gensim)
  Downloading https://files.pythonhosted.org/packages/61/39/122222b5e85cd41c391b68a99ee296584b2a2d1d233e7ee32b4532384f2d/bz2file-0.98.tar.gz
Requirement already satisfied: requests in d:\working_source\anaconda3\lib\site-packages (from smart-open>=1.7.0->gensim) (2.21.0)
Collecting boto3 (from smart-open>=1.7.0->gensim)
  Downloading https://files.pythonhosted.org/packages/16/3d/03c8643f74c9a97d6badc39b890788aa8dbd51ee5d4029677a0dba67a94a/boto3-1.9.123-py2.py3-none-any.whl (128kB)
100% |#####| 133kB 306kB/s
Requirement already satisfied: certifi>=2017.4.17 in d:\working_source\anaconda3\lib\site-packages (from requests->smart-open>=1.7.0->gensim) (2018.11.29)
Requirement already satisfied: chardet<3.1.0,>=3.0.2 in d:\working_source\anaconda3\lib\site-packages (from requests->smart-open>=1.7.0->gensim) (3.0.4)
Requirement already satisfied: idna<2.9,>=2.5 in d:\working_source\anaconda3\lib\site-packages (from requests->smart-open>=1.7.0->gensim) (2.8)
```

图四-2 Gensim 包的安装过程

4.3 源程序代码

由于代码量较大，因此只展示部分代码，完整代码详见链接地址：

<https://github.com/Harrypotterrr/Machine-learning-course/tree/master/1-logistic-imdb>

```
%%time

from sklearn.datasets import load_files

import numpy as np

import re

# 整理 aclImdb 目录下的数据

print("整理数据中...")

# 清洗文本

def wash_text(text):

    text = text.lower().replace(u'\x85', '').replace(u'\n', '').replace(u'<br/>', '').replace('<br />', '')
```

```

# 把标点符号算作一个词
text = re.sub(r"([\.\\"",\(\)!?;:]", " \\1 ", text)

return text

# 使用 sklearn 的 load_files 导入 category/data.txt 格式的目录, 并且去掉 unsup 分类
reviews_train = load_files("data/aclImdb/train/")
unsup_index = reviews_train.target_names.index('unsup')
text_train = []
y_train = []
text_unsup = []

for t, y in zip(reviews_train.data, reviews_train.target):
    if y != unsup_index:
        text_train.append(wash_text(t.decode('utf-8')))
        y_train.append(y)
    else:
        text_unsup.append(wash_text(t.decode('utf-8')))

print("训练用评论: {}".format(len(text_train)))
print("训练用评论 (按 pos/neg 分类): {}".format(np.bincount(y_train)))

print("非监督 (聚类) 用评论: {}".format(len(text_unsup)))

reviews_test = load_files("data/aclImdb/test/")
text_test = [wash_text(t.decode('utf-8')) for t in reviews_test.data]
y_test = reviews_test.target

print("测试用评论: {}".format(len(text_test)))
print("测试用评论 (按 pos/neg 分类): {}".format(np.bincount(y_test)))

```

```
%%time

# 加载词向量映射表文件

from gensim.models import KeyedVectors

current_model = KeyedVectors.load_word2vec_format('data/GoogleNews-vectors-negative300.bin', binary=True)
```

```
%%time

# 尝试用一下这个词向量映射表

current_model.most_similar(positive=['computer', 'portable', 'charger'], topn=5)
```

4.4 局限性

- a. 没有考虑单词的顺序
- b. 提供了高质量的词汇向量，但未能将它们结合成一个高质量的文档向量

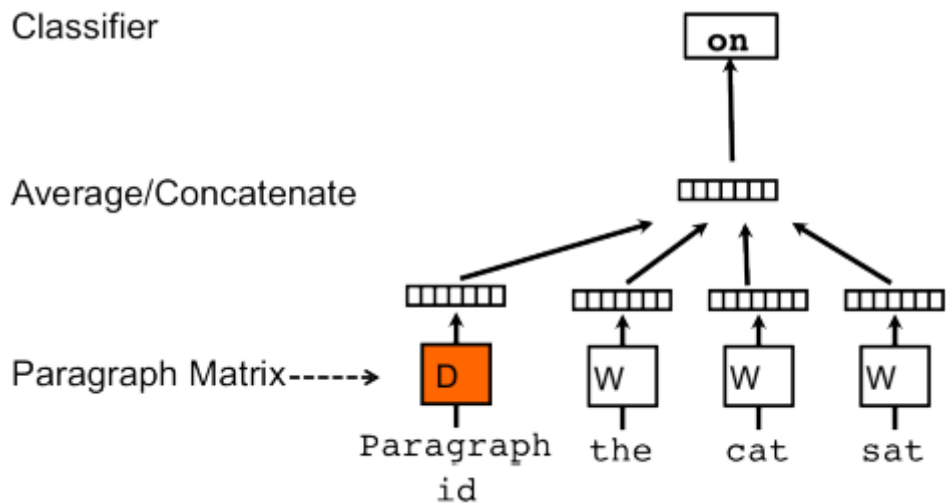
五、 特征提取——提升：Doc2Vec

5.1 原理概述

Doc2Vec 由 Mikilov 和 Le 在 2014 年提出 (<https://arxiv.org/abs/1405.4053>)。

相比于词袋模型和 Word2Vec 的方法，Doc2Vec 着眼于超越“词”的结构，直接构造任意长度文档的数字表示。由于自然语言写就的文档长度任意，且词的组合没有固定的结构，因此增加了向量化的难度，降低了准确性；Doc2Vec 则致力于解决此问题。

Doc2Vec 的一种简单实现——PV-DM 实际上是 Word2Vec 方法在 CBOW 模型上的一个小小的扩展：在词向量的基础上，增加了一个表示文档独特特征的“段向量”，仍使用 CBOW 的训练方法（使其预测下一个可能的词汇）进行训练。整个网络训练完毕后，段向量也就训练完毕，形成了可以对任意文档进行向量化的层。[3]

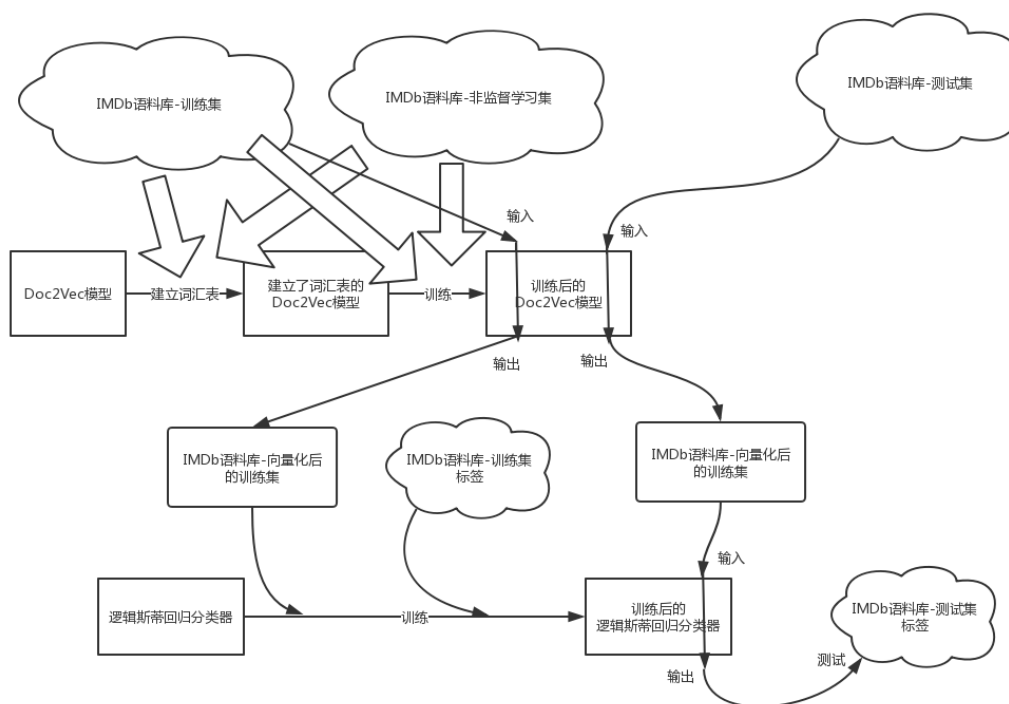


图五-1 PV-DM 结构图^[3]

但在本次作业中，我们将 Doc2Vec 作为一个“黑盒”：不考虑其内部实现，仅使用 Gensim 中的 Doc2Vec 类训练、生成文档向量。

5.2 实现过程

如图，使用训练集和非监督学习集的文本，对 Doc2Vec 模型进行无监督学习训练，使其能够对某一文本产生向量化的输出；接着，使用向量化的训练集及其标签对逻辑斯蒂回归模型进行训练。再将测试集文本向量化，输入到逻辑斯蒂回归模型中，将其输出与测试集标签对比，进行评估。



图五-2 训练 Doc2Vec 模型与逻辑斯蒂模型，并用以测试集分类的流程

5.3 源程序代码

Doc2Vec 的核心代码如下。由于代码量较大，因此只展示部分代码，完整代码详见链接地址：<https://github.com/Harrypotterrrr/Machine-learning-course/tree/master/1-logistic-imdb>

```

pv_dbow_plain = Doc2Vec(dm=0, vector_size=100, negative=5, hs=0,
    min_count=2, sample=0,

    epochs=20, workers=cores)

pv_dbow_plain.build_vocab(chain(docs_train, docs_unsup))
# 打散所有文档，然后开始训练
from random import shuffle
docs_train_and_unsup_shuffled = list(chain(docs_train, docs_unsup))
shuffle(docs_train_and_unsup_shuffled)
current_model = pv_dbow_plain
  
```

```

current_model.init_sims(True)

current_model.train(docs_train_and_unsup_shuffled,
                    total_examples=len(docs_train_and_unsup_shuffle
d),
                    epochs=pv_dbow_plain.epochs)

```

5.4 结果展示

下面验证 Doc2Vec 模型的有效性：挑选一段 Doc2Vec 训练集的文本，使用 Doc2Vec 对象的 `infer_vector` 方法，生成其文档向量；同时，因为该文本在 Doc2Vec 训练集中，对象中早已包含过该文档的向量。对比这两个向量的相似性，可得知 Doc2Vec 模型的有效性。

代码：

```

# 来看看这个模型靠不靠谱

demon_no = 1233

demon_doc = filter(lambda x: x.tags==[demon_no], docs_train).__n
ext__()

demon_y = demon_doc.y

print("Doc %d: (%d)" % (demon_no, demon_y), ' '.join(demon_doc.wo
rds))

print()

print("The vector of this doc:", np.around(current_model.docvecs
[demon_no], 2))

print()

demon_new = "a clever script from (……长文本已省略，与随机挑选出来的文
本相同) de roubaix music has lots of punch ! highly recommended ."
demon_new = demon_new.split()

print("A new doc: ", ' '.join(demon_new))

print()

print("The vector of this new doc:", np.around(current_model.infe
r_vector(demon_new), 2))

print()

```

```
print("Much alike! aren't they?")
```

输出:

Doc 1233: (1) a clever script from ... highly recommended .

The vector of this doc: [-0.92 -0.4 0.13 0.51 -0.43 -0.19 -0.78
-0.02 0.68 -0.05 0.11 -0.15

0.06 -0.01 -0.43 0.1 0.15 -0.03 -0.39 -0.33 0.16 0.1 0.04
-0.1

-0.06 0.26 -0.54 -0.19 0.2 -0.44 0.19 -0.27 0.46 -0.08 0.15
0.31

-0.22 -0.23 -0.64 -0.42 -0.44 -0.38 -0.63 -0.13 -0.45 0.13 0.33
-0.1

0.26 -0.26 0. 0.15 -0.2 -0.16 0.66 -0.14 -0.39 0.09 0.77
0.26

0.62 -0.03 0.32 -0.25 -0.27 0.21 0.63 -0.26 -0.6 -0. 0.15
-0.59

-0.41 -0.53 0.05 -0.16 -0.09 0.45 -0.3 -0. -0.21 -0.16 0.49
0.25

0.19 -0.11 -0.21 0.33 -0.47 -0.29 0.66 0.65 0.11 -0.06 -0.59
-0.19

0.14 -0.82 -0.36 -0.15]

A new doc: a clever script from ... highly recommended .

The vector of this new doc: [-1.04 -0.64 0.09 0.51 -0.65 -0.04 -
0.72 -0.13 0.81 0.18 0.28 -0.11

0.16 0.16 -0.66 0.24 0.1 0.07 -0.41 -0.34 -0.01 0.1 0.1
-0.13

-0.16 0.43 -0.61 -0.23 0.18 -0.37 0.39 -0.3 0.38 -0.02 0.23
0.35

-0.26 -0.24 -0.68 -0.48 -0.51 -0.38 -0.82 -0.03 -0.35 0.01 0.47
-0.02

```

0.24 -0.36 0.15 0.31 -0.14 -0.16 0.77 -0.26 -0.47 0.03 0.76
0.33

0.8 0.01 0.33 -0.36 -0.43 0.23 0.73 -0.19 -0.62 -0.07 0.14
-0.67

-0.46 -0.63 0.06 -0.39 -0.05 0.48 -0.26 -0.04 -0.3 -0.05 0.37
0.33

0.33 -0.09 -0.36 0.24 -0.62 -0.25 0.72 0.68 0.09 0.08 -0.65
-0.16

0.11 -0.87 -0.47 -0.03]

Much alike! aren't they?

```

两个向量极其相似，说明 Doc2Vec 模型训练成功。

六、 逻辑斯蒂回归过程

6.1 原理概述

逻辑斯蒂回归是一种用于分类的回归算法。本文仅探讨逻辑斯蒂回归在二分类问题上的应用。

我们定义样本数为 m ，每一个样本经过特征提取后成为 $(1+n)$ 维纵向量，其中第一个元素始终为 1，以提供偏置。于是对于第 i 个样本，其特征输入向量可以表示为

$$x^{(i)} = \begin{pmatrix} 1 \\ x_1^{(i)} \\ x_2^{(i)} \\ \vdots \\ x_n^{(i)} \end{pmatrix}$$

特别地，不妨记 $x_0^{(i)} = 1$ 。

与其对应地，其线性组合权重、偏置（参数）亦可表示为一 $(1+n)$ 维向量

$$\theta = \begin{pmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \\ \theta_n \end{pmatrix}$$

则，逻辑斯蒂回归分类器的预测输出（某一实例 x 为正例的可能性）为

$$\hat{y} = P_{\theta}(C_{pos}|x) = \frac{1}{1 + e^{-\theta^T x}}$$

用（二项）交叉熵损失作为它的性能度量

$$CrossEntropy = \sum_{i=1}^m - [y^{(i)} \times \ln(\hat{y}^{(i)}) + (1 - y^{(i)}) \times \ln(1 - \hat{y}^{(i)})]$$

其中 m 为样本数量。

则使用梯度下降法更新 θ ：

$$\begin{aligned} \frac{\partial CrossEntropy}{\partial \theta_j} &= - \sum_{i=1}^m (y^{(i)} - \hat{y}^{(i)}) x_j^{(i)} \\ \theta_j &:= \theta_j - \eta \frac{\partial CrossEntropy}{\partial \theta_j} \\ &= \theta_j + \eta \sum_{i=1}^m (y^{(i)} - \hat{y}^{(i)}) x_j^{(i)} \end{aligned}$$

其中 η 为学习率。

将总体样本随机划分为小批量（Mini-batch），使用随机梯度下降（Stochastic Gradient Descend, SGD）的方法迭代调整参数 θ ；当所有小批量均参与一次调整后，即为一个 Epoch 结束。运行多个 Epoch 后，模型在某一极小值进入收敛状态，损失值稳定在一低值，此时认为训练完毕。

6.2 实现过程

我们用了两种方法实现逻辑斯蒂回归：

- 1) 使用 Python 里的数学工具 numpy 数组，自行用 Python 代码实现逻辑斯蒂回归的逻辑。
该方法为最简单的逻辑斯蒂回归实现，采用随机梯度下降的优化方法，且不加任何正则化。
- 2) 使用 Python sklearn.linear_model 包内的 LogisticRegression 类，自动完成

逻辑斯蒂回归的计算。此为一个封装好的复杂的逻辑斯蒂回归实现。指定的优化方法是“随机平均梯度”（SAG）方法，并且考虑到正则化及其正则化参数 C 可取多值，使用了 `GridSearch` 对象来封装这一个过程。`GridSearch` 可从一个预定义的 C 值列表中，尝试出能使逻辑斯蒂回归模型表现最好的 C 值。

6.3 源程序代码

以上两种方法的部分代码如下。由于代码量较大，因此只展示部分代码，完整代码详见链接地址：<https://github.com/Harrypotterrrr/Machine-learning-course/tree/master/1-logistic-imdb>

```
import time

# 开始 Logistic 部分 (纯 Python)
# 输入: features_train
minibatch_size = 50

def get_minibatch(x_train, y_train, minibatch_size):
    for i in range(0, sample_num, minibatch_size):
        yield x_train[i:i+minibatch_size], y_train[i:i+minibatch_size]

def sigmoid(x):
    return 1. / (1 + np.exp(-x))

learning_rate = 0.01

theta = np.random.randn(features_num)

for epoch in range(30):
    total_abs_loss = 0
    total_loss = 0
```

```

    for i, (x_batch, y_batch) in enumerate(get_minibatch(features_train_shuffled, y_train_shuffled, minibatch_size)):
        outputs = sigmoid(np.dot(x_batch, theta))

        individual_cel = - y_batch * np.log(outputs) - (1 - y_batch) * np.log(1 - outputs)

        cross_entropy_loss = individual_cel.sum()

        total_abs_loss += np.abs(outputs - y_batch).sum()

        total_loss += cross_entropy_loss

        partial_common = (outputs - y_batch)

        delta_theta = (partial_common[:, np.newaxis] * x_batch).sum(axis=0)

        theta -= learning_rate * delta_theta

    avg_loss = total_loss

    print("Epoch %d : Avg Loss %f" % (epoch, avg_loss))

```

```

# sklearn 的 LogisticRegression
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import GridSearchCV

param_grid = {'C': [0.001, 0.01, 0.1, 1, 10]}

grid = GridSearchCV(LogisticRegression(solver='sag'), param_grid, cv=5)

grid.fit(features_train_shuffled, y_train_shuffled)

print("Best cross-validation score: {:.2f}".format(grid.best_score_))

print("Best parameters: ", grid.best_params_)

```

```
print("Best estimator: ", grid.best_estimator_)

lr = grid.best_estimator_
```

6.4 结果展示

对于第一种方法，其结果为不断打印每个 Epoch 在训练集上的损失函数：

```
Epoch 0 : Avg Loss 10537.789932
Epoch 1 : Avg Loss 6713.383619
Epoch 2 : Avg Loss 6683.776234
Epoch 3 : Avg Loss 6682.213666
Epoch 4 : Avg Loss 6682.108047
Epoch 5 : Avg Loss 6682.106338
Epoch 6 : Avg Loss 6682.109085
Epoch 7 : Avg Loss 6682.110340
Epoch 8 : Avg Loss 6682.110789
Epoch 9 : Avg Loss 6682.110942
Epoch 10 : Avg Loss 6682.110993
Epoch 11 : Avg Loss 6682.111010
Epoch 12 : Avg Loss 6682.111016
Epoch 13 : Avg Loss 6682.111018
Epoch 14 : Avg Loss 6682.111019
Epoch 15 : Avg Loss 6682.111019
```

图六-1 自己实现逻辑的逻辑斯蒂回归算法的每 Epoch 损失

可见，在 4 个 Epoch 后，逻辑斯蒂回归的参数已经趋于稳定。

对于第二种方法，结果展示方法为打印出最好的模型对应分数、最好的 C 参数。

```
Best cross-validation score: 0.89
Best parameters: {'C': 10}
Best estimator: LogisticRegression(C=10, class_weight=None, dual=False, fit_intercept=True,
    intercept_scaling=1, max_iter=100, multi_class='warn',
    n_jobs=None, penalty='l2', random_state=None, solver='sag',
    tol=0.0001, verbose=0, warm_start=False)
```

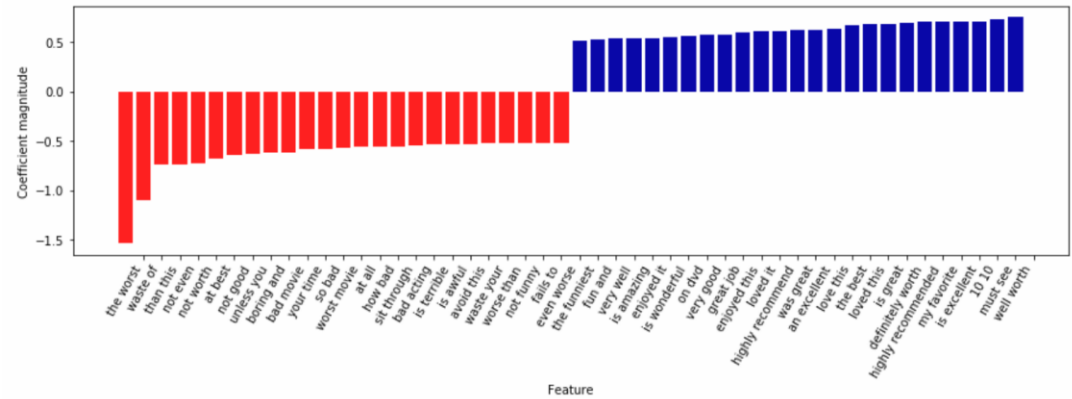
图六-2 使用 Sklearn 库的训练结果

可见，经尝试，最好的模型取得了 0.89 的准确度（经交叉验证），且最好的 C 值为 10。

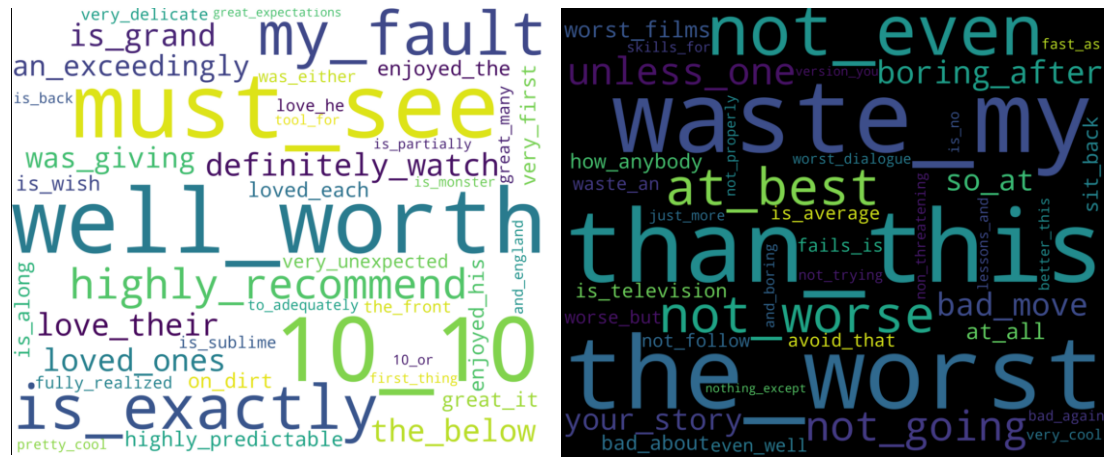
七、 最终结果

7.1 使用词袋法进行特征提取的结果

经过训练，提取出逻辑斯蒂回归模型的各参数中绝对值最大者，结合各参数对应输入特征的真实含义（2-grams），得到对于分类结果影响最大的正面、负面 2-grams 特征，以柱状图和“词云”方式展示如下：

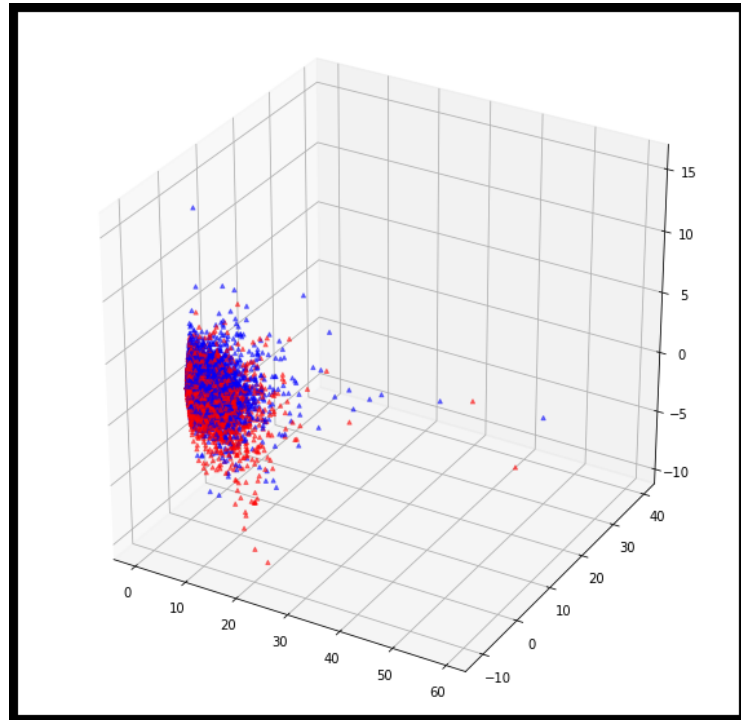


图七-1 词袋方法-特征影响程度柱状图

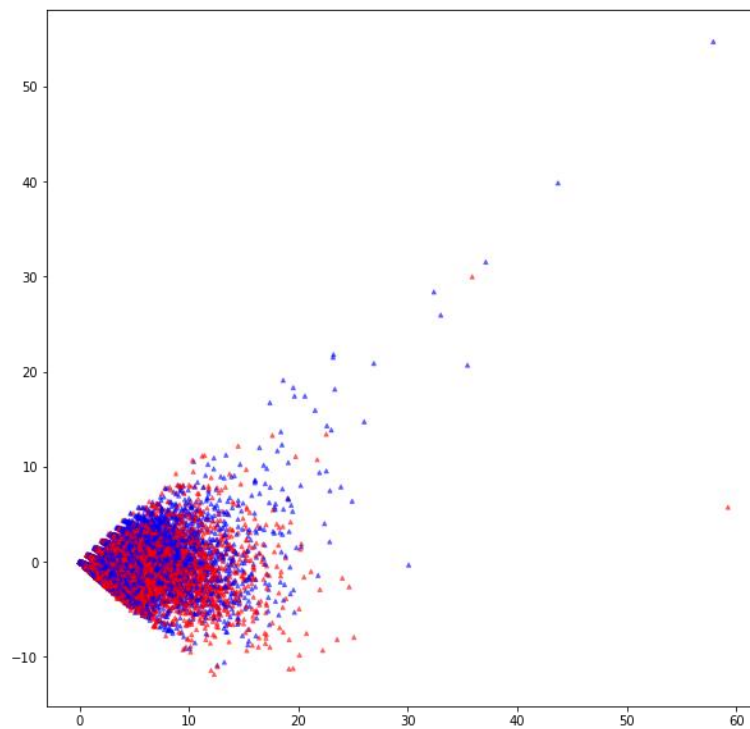


图七-2 词袋方法-特征词云

将这些 2-grams 词袋特征进行 SVD 降维后，加以模型输出分类的颜色表示绘制出散点图，可以发现，虽然模型分类结果优秀，但特征降维后的可视化并不直观：

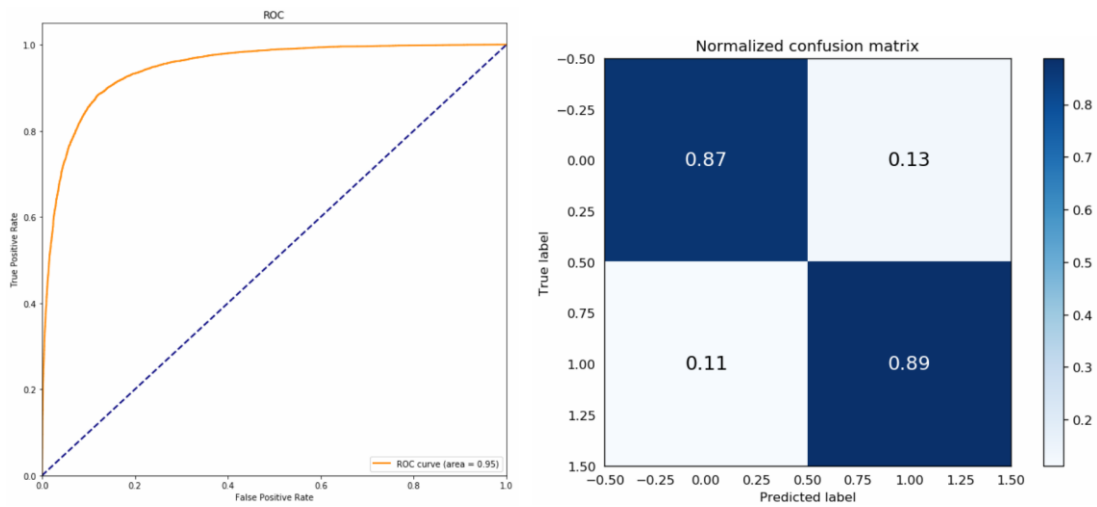


图七-3 词袋方法-SVD 降到 3 维时散点图



图七-4 词袋方法-SVD 降到 2 维时散点图

其 ROC 曲线与 AUC 值、混淆矩阵如下所示：

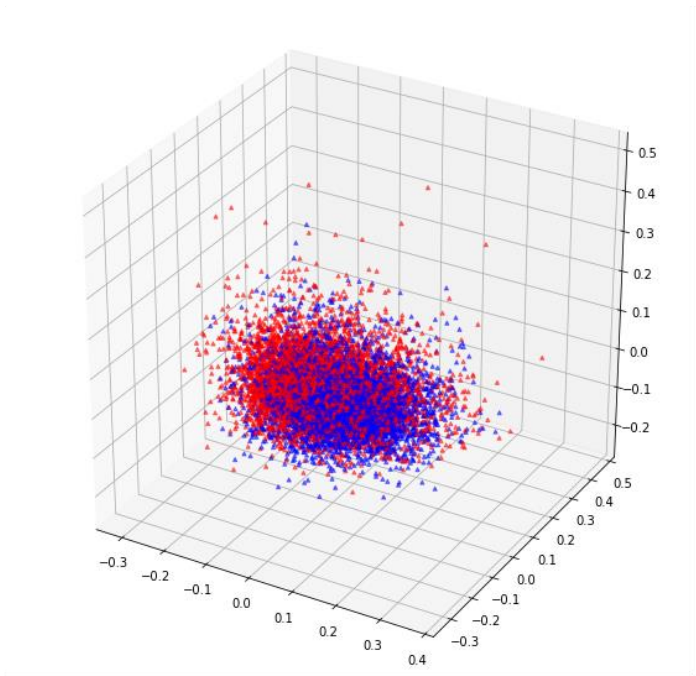


图七-5 词袋方法-ROC、AUC (0.95)、混淆矩阵

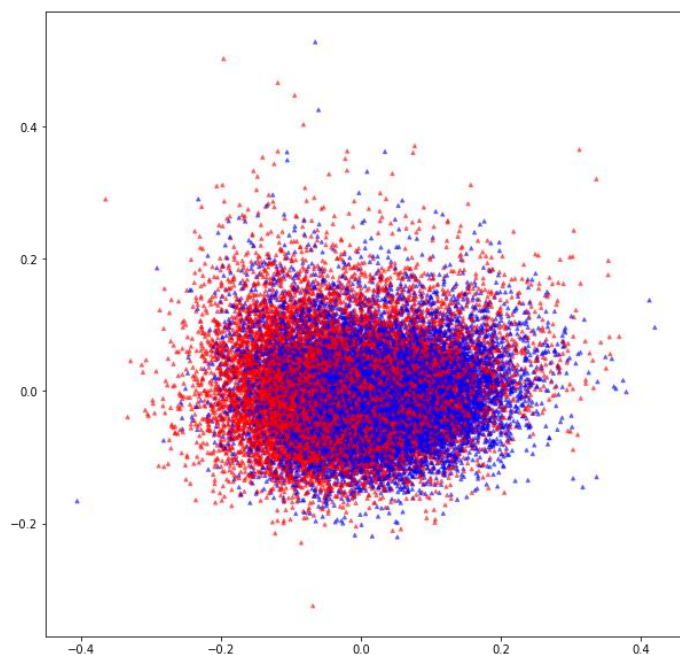
7.2 使用 Word2Vec 法进行特征提取的结果

由于文本使用 Word2Vec 法转为词嵌入向量后，失去了可解释性，因此无法基于词的原本含义给出最有影响力的词特征柱状图以及“词云”。

以下为 PCA 降维后的散点图：

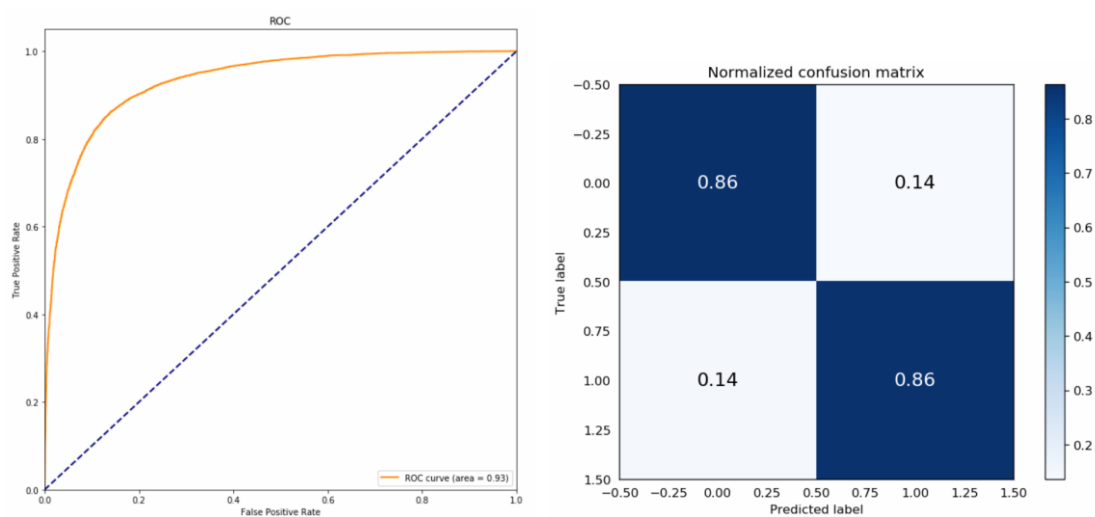


图七-6 Word2Vec 方法-PCA 降到 3 维时散点图



图七-7 Word2Vec 方法-PCA 降到 2 维时散点图

以下为 ROC 曲线及其对应 AUC 值，以及混淆矩阵：

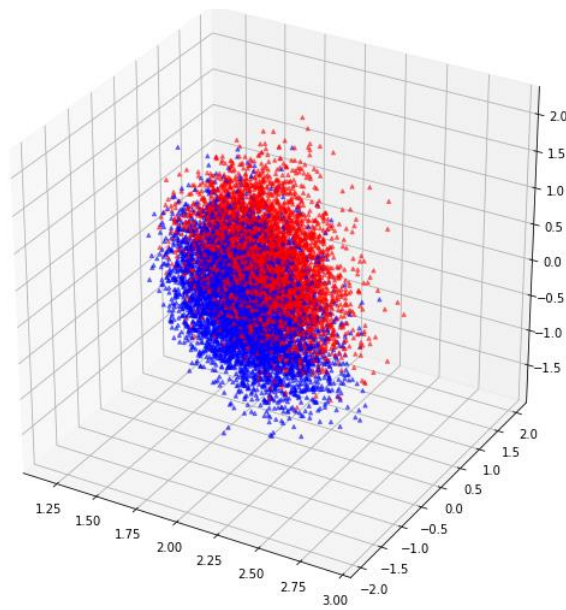


图七-8 Word2Vec 方法-ROC、AUC (0.93)、混淆矩阵

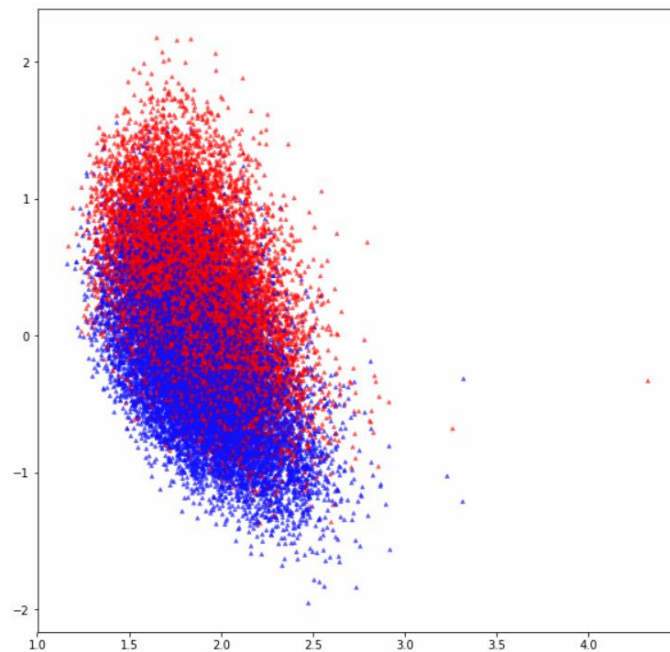
可以发现，虽然从散点图看，特征的分度程度较词袋方式而言明显了一些，但从 ROC 曲线和混淆矩阵来看，模型的表现要稍差。

7.3 使用 Doc2Vec 法进行特征提取的结果

以下为 SVD 降维后的散点图：

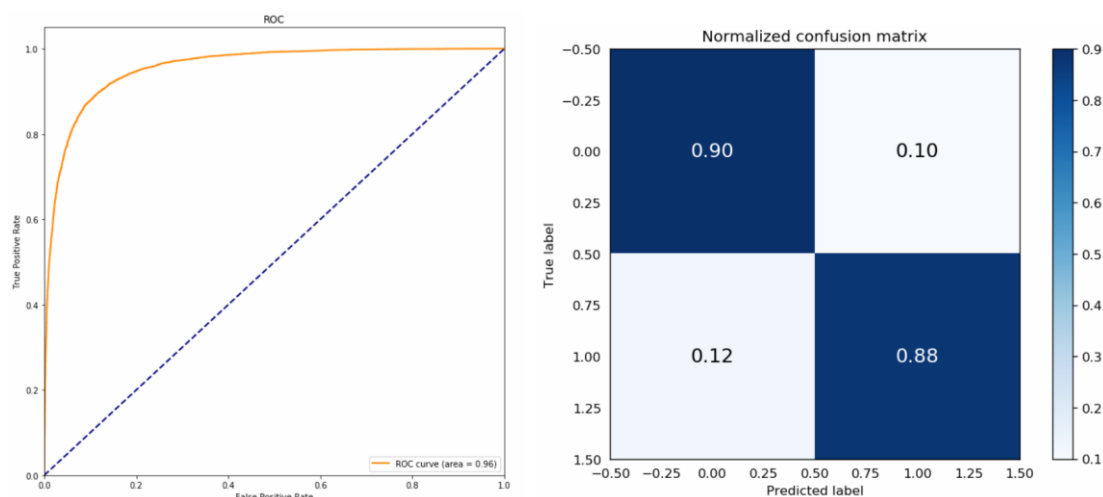


图七-9 Doc2Vec 方法-SVD 降至 3 维时散点图



图七-10 Doc2Vec 方法-SVD 降至 2 维时散点图

以下为 ROC 曲线及其对应的 AUC 值，以及混淆矩阵：



图七-11 Doc2Vec 方法-ROC、AUC (0.96)、混淆矩阵

可以发现，无论是从散点图的区别程度来看，还是从 ROC 曲线、混淆矩阵来看，使用 Doc2Vec 特征提取的方法在三个方法中是最好的。

模型	得分
词袋模型	0.88
Word2vec	0.86
Doc2vec	0.90

表七-1 三种方法的准确度比较

模型的得分见上图。我们上文提到过，词袋模型的主要缺点是忽略了单词的顺序、语义以及语境，在 Word2vec 模型中我们弥补了单词的语义所带来的偏差，通过神经网络将单词向量扩张成成千上万个维度来丰富它的语义信息。最后在 doc2vec 的部分，我们将文档张成一个向量，这样我们的特征中就包含了评论的单词顺序和语境信息，得到的结果最好。

7.4 使用 Doc2Vec 方法进行任意输入影评文本的情感分类

我们使用当前表现最好的基于 Doc2Vec 方法的情感分类器，对于用户任意输入的影评文本给出情感分类。

- 1) The movie rocks! (“这电影赞爆!”，判断正确)

```
[33]: # Demonstration
my_doc = TaggedDocument("The movie rocks !".split(), ['my'])
my_vector = current_model.infer_vector(my_doc.words)
print(reviews_train.target_names[apply_logistic_purelr(my_vector)])

pos
```

图七-12 Doc2Vec 方法-任意文本测试（一）

2) The movie sucks! (“这电影太烂了!”，判断正确)

```
[34]: # Demonstration
my_doc = TaggedDocument("The movie sucks !".split(), ['my'])
my_vector = current_model.infer_vector(my_doc.words)
print(reviews_train.target_names[apply_logistic_purelr(my_vector)])

neg
```

图七-13 Doc2Vec 方法-任意文本测试（二）

3) I like this movie! (“我喜欢这电影!”，判断错误)

```
[35]: # Demonstration
my_doc = TaggedDocument("I like this movie !".split(), ['my'])
my_vector = current_model.infer_vector(my_doc.words)
print(reviews_train.target_names[apply_logistic_purelr(my_vector)])

neg
```

图七-14 Doc2Vec 方法-任意文本测试（三）

可见，对于任意输入的影评文本，分类器仍有判断失误的现象发生，这可能归咎于影评文本的数据集不够全面，也可能与模型有待改进有关。

八、可能的改进点

8.1 更好的向量化模型

Word2Vec 向量化模型是基于 Google News 的数据集训练而成。如果能使用更为广泛的数据集训练而成的向量化模型，Word2Vec 的表现将会更好，分类准确率也会提升。

词袋模型形成的词向量是 2-grams 剔除高频停止词后，每个位置的值表示该位置对应词的出现次数的稀疏向量。调整 grams 的数目、调整高频停止词的判断阈值可能带来更好的效果。

8.2 Word2Vec 中更好的文档向量生成方法

Word2Vec 中，将文档中所有词的词向量取平均得到文档（影评）的向量。这忽略了词序和词在文档中的位置等大量信息。如果能将影评作为语料库建立语言模型，结合各词（词向量）出现的相关性，藉此形成文档向量，可能改善

Word2Vec 法提取文档特征向量的表现。

8.3 逻辑斯蒂回归中的超参调整

逻辑斯蒂回归中，对于正则化细化 C 值的调整、选择更优秀的优化算法，可能改善逻辑斯蒂回归的表现。

九、 心得体会

自然语言处理（NLP）是当今十分热门的数据科学研究项目，情感分析则是自然语言处理方法中常见的应用，尤其是以提炼文本情绪内容为目的的分类。将成千上万的文本数据在短时间内分析出情感类型，可以说是 NLP 中最有意思的挑战了。

本次作业的内容为使用已学过的机器学习模型做应用，我们便挑选了这个方向。在确定使用的模型时，我们对于已学过的模型进行了有效程度、可能实现方式以及实现复杂性的评估，放弃了决策树的模型，使用了逻辑斯蒂回归的线性模型来完成影评分类的任务。正确、审慎地选择方向，是项目能顺利推进的重要保证。

确定使用的模型后，我们针对文本信息应如何输入逻辑斯蒂回归模型这一问题，广泛搜索解决方法，得到了较为基础的词袋方法、Word2Vec 方法以及先进的 Doc2Vec 方法这三种提取特征的方法，并决定基于这些提取特征的方法在其上使用逻辑斯蒂回归模型。

基于上面的清晰思路，我们开始编写三种方法的代码。在其中，我们学习了 Jupyter Notebook 平台的使用，以及 Pandas、Sklearn、Pytorch、Gensim 等 Python 工具包，并基于它们提供的服务，写出了高效的代码。其中，还手动实现了逻辑斯蒂回归的基本逻辑。在之后，我们尝试着写出了漂亮的数据可视化代码。在其中，我们大量使用搜索工具解决问题，并参考了已有代码。作为机器学习框架的初学者，我们代码的编写过程还算顺利；代码成功运行后，成就感很高，大家也感到欣慰。

本次项目实现了基于逻辑斯蒂回归的影评情感分类，完美诠释了让机器学习服务于生活的理念。从基本算法决策树到逻辑斯蒂回归的转变，小组成员之间也配合得相当默契。大家各司其职，最终出色地完成了我们的第一次项目，并顺利地进行了课堂展示。

整个项目进展过程中，我们将机器学习理论运用到实践中，巩固掌握了前几周课内概念空间、逻辑回归、决策树的基本知识，交叉融合了自然语言处理、模式识别等学科领域，拓展延伸了逻辑创造思维，锻炼了团队合作意识与合作精神。短短几周之内我们收获了许多，在机器学习的课程经历中留下了难忘的回忆。

一〇、 参考资料

[1] 刘钢,张维石.基于决策树的网民评价情感分析[J].现代计算机(专业

版),2017(32):15-19.

[2] MaasL.Andrew, DalyE.Raymond, PhamT.Peter, HuangDan, NgY.Andrew, & PottsChristopher. (2011 年 6 月). Learning Word Vectors for Sentiment Analysis. Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (页 142-150). Portland: Association for Computational Linguistics. 检索来源: <http://www.aclweb.org/anthology/P11-1015>

[3] Gidi Shperber ,A gentle introduction to Doc2Vec, 2017.
<https://medium.com/scaleabout/a-gentle-introduction-to-doc2vec-db3e8c0cce5e>

[4] T. Mikolov, K. Chen, G. Corrado, J. Dean. Efficient Estimation of Word Representations in Vector Space[J]. 2013.

[5] Q. Le, T. Mikolov. Distributed Representations of Sentences and Documents[C]. 2014.

[6] 伯乐在线. 现代情感分析方法[EB/OL]. <http://python.jobbole.com/87811/>.

[7] Stanford University. Machine Learning: Sentiment analysis of movie reviews using Logistic Regression[EB/OL]. <https://itnext.io/machine-learning-sentiment-analysis-of-movie-reviews-using-logisticregression-62e9622b4532?gi=d85dec735d45>

[8] CSDN. NLP 之影评情感分类[EB/OL].
<https://blog.csdn.net/cskywit/article/details/79268146>.

[9] CSDN. doc2vec 原理及实践[EB/OL].
<https://blog.csdn.net/u010417185/article/details/80654558>.