成 绩 _____

北京航空航天大學

BEIHANG UNIVERSITY

深度学习与自然语言处理
第三次大作业

LDA模型

院（系）名称	自动化科学与电气工程学院
专业名称	电子信息
学生学号	ZY2103812
学生姓名	朱远哲
指导教师	秦曾昌

2022 年 5 月

目　　录

1 问题描述

从给定的语料库中均匀抽取200个段落（每个段落大于500个词）， 每个段落的标签就是对应段落所属的小说。利用LDA模型对于文本建模，并把每个段落表示为主题分布后进行分类。验证与分析分类结果。

2 问题表达

2.1 LDA 模型

LDA主题模型主要用于推测文档的主题分布，可以将文档集中每篇文档的主题以概率分布的形式给出根据主题进行主题聚类或文本分类。LDA主题模型不关心文档中单词的顺序，通常使用词袋特征（bag-of-word feature）来代表文档。

LDA模型认为主题可以由一个词汇分布来表示，而文章可以由主题分布来表示。所以想要生成一篇文章，可以先以一定的概率选取某个主题，再以一定的概率选取主题下的某个单词，不断重复这两步就可以生成最终文章。

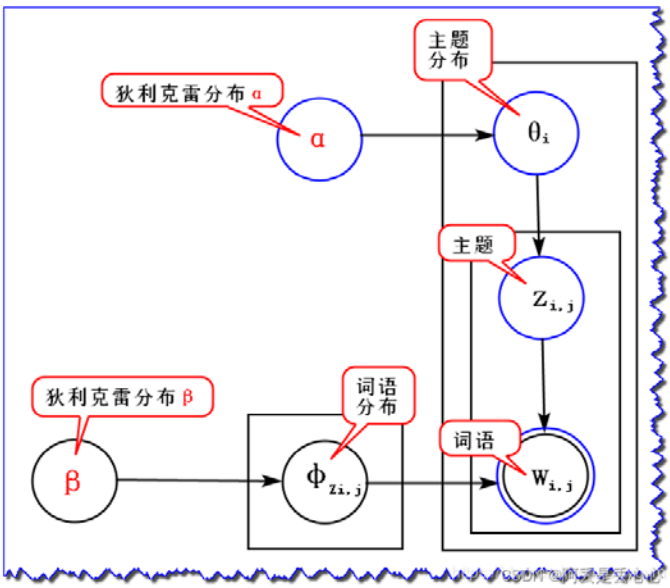
在LDA模型中，一篇文档生成的方式如下：

从狄利克雷分布中取样生成文档i的主题分布

从主题的多项式分布中取样生成文档i第j个词的主题

从狄利克雷分布中取样生成主题对应的词语分布

从词语的多项式分布中采样最终生成词语



如果我们要生成一篇文档，它里面的每个词语出现的概率为：

$$P(\text{词语}|\text{文档}) = \sum_{\text{主题}} P(\text{词语}|\text{主题}) \times P(\text{主题}|\text{文档})$$

2.2 利用 LDA 模型进行文本分类

文本分类步骤如下

1. 从给定的16本金庸小说数据集中，随机、均匀地抽取k个段落，每个段落的标签为对应小说的小说名，每个段落包含n个字（$n \geq 500$），每个段落作为一个样本；对数据进行处理，去掉广告和无意义的停用词，并利用jieba工具进行分词，生成样本段落；选定主题数量，利用训练样本训练LDA模型。

2. 构造测试集，从第一步中划分区间的不同位置取相应段落，计算每个段落的主题概率分布， 并计算该段落主题概率分布与每本小说主题概率分布的误差平方和，即欧式距离，选取欧式距离最小的一本小说，作为该段落的来源判别结果，并计算总的判别准确率。

3 具体算法实现

3.1 数据处理

对给定语料库进行分析可知，语料库内共有16篇文章，从每一篇文章内抽取13个段落，共有208个段落；对每篇文章分词之后，将一篇文章的总词数除以13，即每篇文章共有13个区间，所选取的段落为每个区间抽取前500个词作为训练集，501-1000个词作为测试集。同时对语料进行预处理，去除广告和无意义停词，采用中文停词表cn_stopwords.txt中的内容。

```
def read_novel(path):  # 读取语料内容
    content = []
    content1 = []
    names = os.listdir(path)
    for name in names:
            con_temp = []
            con_temp1 = []
            novel_name = path + '\\' + name
            with open(novel_name, 'r', encoding='ANSI') as f:
                con = f.read()
                con = content_deal(con)
                con = jieba.lcut(con)  # 结巴分词
                con = content_stopword(con)
                con_list = list(con)
                pos = int(len(con)//13) ####16篇文章，分词后，每篇均匀选取13个500词段落进行建模
                for i in range(13):
                    con_temp = con_temp + con_list[i*pos:i*pos+500]
                    con_temp1 = con_temp1 + con_list[i*pos+501:i*pos+1000]
                content.append(con_temp)
                content1.append(con_temp1)
            f.close()
    return content, content1, names
```

停用词处理

```
def content_stopword(content):
    stopwords = load_stopword()
    content_new = []
    for words in content:
        if words not in stopwords:
            content_new.append(words)
    return content_new
```

3.2 模型训练

```
Doc_count = []  # 每篇文章中有多少个词
Doc_fre = []  # 每篇文章有多少各个topic的词
i = 0
for data in data_txt:
    topic = []
    docfre = {}
    for word in data:
        a = random.randint(0, len(data_txt)-1)  # 为每个单词赋予一个随机初始topic
        topic.append(a)
        if '\u4e00' <= word <= '\u9fa5':
            Topic_count[a] = Topic_count.get(a, 0) + 1  # 统计每个topic总词数
            docfre[a] = docfre.get(a, 0) + 1  # 统计每篇文章的词频
            exec('Topic_fre{}[word]=Topic_fre{}.get(word, 0) + 1'.format(i, i))  # 统计每个topic的词频
    Topic_All.append(topic)
    docfre = list(dict(sorted(docfre.items(), key=lambda x: x[0], reverse=False)).values())
    Doc_fre.append(docfre)
    Doc_count.append(sum(docfre))  # 统计每篇文章的总词数
    # exec('print(len(Topic_fre{}))'.format(i))
    i += 1
Topic_count = list(dict(sorted(Topic_count.items(), key=lambda x: x[0], reverse=False)).values())
# print(Topic_All[0])
Doc_fre = np.array(Doc_fre)  # 转为array方便后续计算
Topic_count = np.array(Topic_count)  # 转为array方便后续计算
Doc_count = np.array(Doc_count)  # 转为array方便后续计算
```

```
while stop == 0:
    i = 0
    for data in data_txt:
        top = Topic_All[i]
        for w in range(len(data)):
            word = data[w]
            pro = []
            topfre = []
            if '\u4e00' <= word <= '\u9fa5':
                for j in range(len(data_txt)):
                    exec('topfre.append(Topic_fre{}.get(word, 0))'.format(j))  # 读取该词语在每个topic中出现的频数
                pro = Doc_pro[i] * topfre / Topic_count  # 计算每篇文章选中各个topic的概率乘以该词语在每个topic中出现的概率，得到该词出现的概率向量
                m = np.argmax(pro)  # 认为该词是由上述概率之积最大的那个topic产生的
                Doc_fre[i][top[w]] -= 1  # 更新每个文档有多少各个topic的词
                Doc_fre[i][m] += 1
                Topic_count[top[w]] -= 1  # 更新每个topic的总词数
                Topic_count[m] += 1
                exec('Topic_fre{}[word] = Topic_fre{}.get(word, 0) - 1'.format(top[w], top[w]))  # 更新每个topic该词的频数
                exec('Topic_fre{}[word] = Topic_fre{}.get(word, 0) + 1'.format(m, m))
                top[w] = m
        Topic_All[i] = top
        i += 1
```

3.3 模型测试

```
Doc_count_test = []  # 每篇文章中有多少个词
Doc_fre_test = []  # 每篇文章有多少各个topic的词
Topic_All_test = []  # 每篇文章中的每个词来自哪个topic
i = 0
for data in test_txt:
    topic = []
    docfre = {}
    for word in data:
        a = random.randint(0, len(data_txt) - 1)  # 为每个单词赋予一个随机初始topic
        topic.append(a)
        if '\u4e00' <= word <= '\u9fa5':
            docfre[a] = docfre.get(a, 0) + 1  # 统计每篇文章的词频
    Topic_All_test.append(topic)
    docfre = list(dict(sorted(docfre.items(), key=lambda x: x[0], reverse=False)).values())
    Doc_fre_test.append(docfre)
    Doc_count_test.append(sum(docfre))  # 统计每篇文章的总词数
    i += 1
# print(Topic_All[0])
Doc_fre_test = np.array(Doc_fre_test)
Doc_count_test = np.array(Doc_count_test)
```

```
loopcount = 1  # 迭代次数
while stop == 0:
    i = 0
    for data in test_txt:
        top = Topic_All_test[i]
        for w in range(len(data)):
            word = data[w]
            pro = []
            topfre = []
            if '\u4e00' <= word <= '\u9fa5':
                for j in range(len(data_txt)):
                    exec('topfre.append(Topic_fre{}.get(word, 0))'.format(j))  # 读取该词语在每个topic中出现的频数
                pro = Doc_pro_test[i] * topfre / Topic_count  # 计算每篇文章选中各个topic的概率乘以该词语在每个topic中出现的概率，得到该词出现的概率向
                m = np.argmax(pro)  # 认为该词是由上述概率之积最大的那个topic产生的
                Doc_fre_test[i][top[w]] -= 1  # 更新每个文档有多少各个topic的词
                Doc_fre_test[i][m] += 1
                top[w] = m
        Topic_All_test[i] = top
        i += 1
```

3.4 结果输出

```
[('曰', 58), ('杀', 43), ('李靖', 36), ('举人', 34), ('杨行密', 34), ('秦桧', 33), ('骈', 32), ('故事', 31), ('高宗', 25), ('李勉', 25)]
[('陈家洛', 190), ('张召重', 70), ('文泰来', 62), ('骆冰', 51), ('见', 50), ('徐天宏', 47), ('周仲英', 38), ('众人', 37), ('会', 35), ('红花', 35)]
[('便', 651), ('石破天', 136), ('不能', 80), ('时', 74), ('爷爷', 65), ('雪山', 63), ('夫妇', 61), ('十分', 60), ('白万剑', 51), ('老人', 47)]
[('张无忌', 178), ('便', 101), ('张翠山', 50), ('武功', 40), ('宋青书', 40), ('见', 38), ('谢逊', 35), ('殷素素', 34), ('赵敏', 34), ('那小鬟', 28)]
[('虚竹', 58), ('契丹', 54), ('段誉', 43), ('黄眉僧', 38), ('听', 36), ('王夫人', 34), ('段延庆', 32), ('南海', 31), ('叶二娘', 31), ('没', 30)]
[('见', 318), ('郭靖', 106), ('当下', 85), ('黄蓉', 78), ('欧阳克', 74), ('洪七公', 71), ('铁木真', 63), ('喝道', 59), ('哲别', 55), ('听', 54)]
[('李文秀', 235), ('麽', 194), ('知道', 185), ('没', 179), ('走', 139), ('死', 133), ('苏鲁克', 110), ('突然', 107), ('出去', 101), ('杀', 100)]
[('袁承志', 268), ('青青', 56), ('袁崇焕', 51), ('大汉', 48), ('见', 45), ('洪胜海', 44), ('走', 31), ('总兵', 31), ('心想', 30), ('原来', 27)]
[('杨过', 175), ('不知', 152), ('甚', 115), ('武功', 97), ('小龙女', 93), ('二人', 78), ('便是', 64), ('饿', 62), ('之下', 55), ('麽', 53)]
[('中', 528), ('想', 217), ('令狐冲', 174), ('之中', 122), ('派', 116), ('弟子', 65), ('仪琳', 64), ('剑法', 58), ('不会', 50), ('心下', 49)]
[('范蠡', 256), ('剑士', 225), ('一声', 180), ('青衣', 105), ('剑', 100), ('勾践', 100), ('阿青', 99), ('长剑', 85), ('身子', 82), ('少女', 80)]
[('师父', 125), ('倒', 119), ('起来', 94), ('狄云', 88), ('更', 83), ('万圭', 74), ('万震山', 62), ('水笙', 61), ('不错', 55), ('不住', 55)]
[('听', 350), ('只见', 192), ('众人', 133), ('两人', 132), ('心中', 119), ('麽', 118), ('曹云奇', 108), ('竟', 105), ('脸上', 98), ('站', 88)]
[('说道', 404), ('一个', 336), ('瞧', 185), ('笑', 143), ('胡斐', 127), ('两个', 94), ('出', 67), ('程灵素', 62), ('银子', 57), ('低声', 56)]
[('萧中慧', 101), ('周威信', 84), ('麽', 83), ('卓天雄', 76), ('著', 65), ('瞎子', 64), ('任飞燕', 61), ('萧半和', 61), ('袁冠南', 60), ('林玉龙', 59)]
[('道', 1871), ('说', 748), ('韦小宝', 272), ('做', 173), ('康熙', 113), ('话', 105), ('心想', 99), ('皇帝', 92), ('吃', 88), ('女子', 83)]
```

4 运行结果

4.1 运行结果

训练集经过21次迭代收敛，完成各段落主题的构建；测试集经过20次迭代收敛，完成各段落文本的分类。结果存放在result.txt文件中。

选取了前十个主题的前十高频词构建表格如下

主题	高频词1	高频词2	高频词3	高频词4	高频词5	高频词6	高频词7	高频词8	高频词9	高频词10
Topic1	'曰', 58	'杀', 43	'李靖', 36	'举人', 34	'杨行密', 34	'秦桧', 33	'骈', 32	'故事', 31	'高宗', 25	'李勉', 25
Topic2	'陈家洛', 190	'张召重', 70	'文泰来', 62	'骆冰', 51	'见', 50	'徐天宏', 47	'周仲英', 38	'众人', 37	'会', 35	'红花' 35
Topic3	'便', 651	'石破天', 136	'不能', 80	'时', 74	'爷爷', 65	'雪山', 63	'夫妇', 61	'十分', 60	'白万剑', 51	'老人', 47
Topic4	'张无忌', 178	'便', 101	'张翠山', 50	'武功', 40	'宋青书', 40	'见', 38	'谢逊', 35	'殷素素', 34	'赵敏', 34	'那小鬟', 28
Topic5	'虚竹', 58	'契丹', 54	'段誉', 43	'黄眉僧', 38	'听', 36	'王夫人', 34	'段延庆', 32	'南海', 31	'叶二娘', 31	'没', 30
Topic6	'见', 318	'郭靖', 106	'当下', 85	'黄蓉', 78	'欧阳克', 74	'洪七公', 71	'铁木真', 63	'喝道', 59	'哲别', 55	'听', 54
Topic7	'李文秀', 235	'麽', 194	'知道', 185	'没', 179	'走', 139	'死', 133	'苏鲁克', 110	'突然', 107	'出去', 101	'杀', 100
Topic8	'袁承志', 268	'青青', 56	'袁崇焕', 51	'大汉', 48	'见', 45	'洪胜海', 44	'走', 31	'总兵', 31	'心想', 30	'原来', 27
Topic9	'杨过', 175	'不知', 152	'甚', 115	'武功', 97	'小龙女', 93	'二人', 78	'便是', 64	'牋', 62	'之下', 55	'麽', 53
Topic10	'萧中慧', 101	'周威信', 84	'麽', 83	'卓天雄', 76	'著', 65	'瞎子', 64	'任飞燕', 61	'萧半和', 61	'袁冠南', 60	'林玉龙', 59

通过观察可以发现结果符合文本主题的分布

最终测试集分类结果为

```
[0, 1, 2, 3, 4, 5, 6, 0, 8, 9, 10, 11, 12, 13, 14, 15]
```

从结果可以看出，对绝大部分段落能够完成正确分类。

4.2 结果讨论与分析

从结果可以看出，利用LDA模型得到了较好的训练和分类效果，代码参考了已有程序，添加了段落划分并去除了停用词，并对各主题以及高频词进行了输出，对代码进行了成功的

改进，但是在运行速度方面还有比较大的提升空间。

5 个人总结与体会

本次大作业实现了利用LDA模型训练段落主题分布，通过本次程序编写，我对python编程变得更加熟悉，同时也学习了解了LDA模型，对迭代式方法的理解更加深刻。，通过算法的学习程序的编写，又让我对自然语言处理有了更加深刻的理解，也通过大作业有了很大的收获。