LDA 模型文本分类

黄渲淇
2320634327@qq.com

Abstract

LDA（Latent Dirichlet Allocation）模型是一种无监督学习算法，常用于文本分类和主题建模。它可以将文本数据集中的文档表示为一组主题的概率分布，而每个主题又由一组单词的概率分布表示。通过这种方式，LDA 模型可以帮助我们发现文本数据集中隐藏的主题结构，并根据文档中单词的出现情况进行分类。

LDA 模型在文本分类中的应用是通过将文本看作由多个主题组成的混合物，并将每个主题看作一个类别或标签。然后，模型根据文本中每个单词所属的主题，计算每个文本属于每个类别的概率。最终，模型将文本分配给概率最高的类别或标签。 LDA 模型文本分类的可以处理大量的文本数据，并且可以自动识别和学习文本中的主题和类别。它可以用于许多应用程序，例如文本分类、情感分析、信息检索等。

Introduction

1.LDA 算法流程

1) 初始化参数

K：主题数，即需要从文本中发现的主题数量。

α：文档-主题分布的超参数，用于控制每个文档中主题的权重。

β：主题-单词分布的超参数，用于控制每个主题中单词的权重。

V：词汇表的大小，即不同单词的数量。

2) 初始化主题分布和单词分布

对于每个主题 k，从 Dirichlet 分布中生成一个长度为 V 的概率向量 ϕ_k，表示该主题中每个单词的概率分布。

对于每个文档 d，从 Dirichlet 分布中生成一个长度为 K 的概率向量 θ_d，表示该文

档中每个主题的概率分布。

3) 遍历文档中的每个单词

对于文档中的每个单词 $w_{d,n}$，计算该单词属于每个主题 k 的概率 $p(z_{d,n}=k|w_{d,n},\phi,\theta)$，即该单词的主题分布。根据这些概率，从多项式分布中选择一个主题 $z_{d,n}$，表示该单词的主题。更新文档-主题分布 θ_d 和主题-单词分布 ϕ_k：

对于文档 d 中的单词 $w_{d,n}$，更新 θ_d 和 $\phi_{z_{d,n}}$，使得它们更好地反映当前单词的主题：

$$\theta_{d,k} \propto \alpha + \sum_{n=1}^{N_d} I(z_{d,n}=k)$$

$$\phi_{k,w_{d,n}} \propto \beta + \sum_{d=1}^{D}\sum_{n=1}^{N_d} I(z_{d,n}=k)I(w_{d,n}=w)$$

其中 $I(x)$ 为指示函数，当 x 成立时为 1，否则为 0。

4) 重复步骤 3，直到模型收敛。

5) 输出主题

对于每个主题 k，输出与之相关的一组单词，即 ϕ_k 中概率最高的若干个单词。可以使用这些主题进行文本分类、信息检索和推荐等应用。总之，LDA 模型通过迭代计算文档-主题分布和主题-单词分布，从而发现文本中的隐藏主题。在这个过程中，LDA 模型逐渐调整主题分布和单词分布，以便每个主题包含一组相关的单词，并且每个文档可以由一组主题描述。

2.K-Means 聚类

Kmeans 是一种聚类算法，它将数据集划分为 k 个簇，其中每个簇包含具有相似特征的数据点。该算法的核心思想是将所有数据点分配给距离它们最近的簇中心，并将簇中心移动到簇中所有数据点的平均值的位置。重复这个过程，直到簇中心不再发生变化或者达到预设的迭代次数。

Methodology

1: 数据读取

读取文档并存储段落，将段落对应的文档主题进行存储

对于所给语料库进行读取，将文本按照段落进行存储，同时按照要求只存储长度大于500的段落，将每个段落对应的标题及段落做一个一维二元数组。

```
with open('./stopwords.txt', 'r', encoding='utf-8') as f:
    stopwords = set([line.strip() for line in f])
# 读取小说语料库中的所有文本
corpus_path = 'txt'
texts = []
for file in os.listdir(corpus_path):
    with open(os.path.join(corpus_path, file), 'r', encoding='ansi') as f:
        text = f.read()
        # 将每个文本按句子分割成段落
        paragraphs = text.split('\n')
        for p in paragraphs:
            if len(p) > 500:  # 只选取长度大于500的段落
                texts.append((p, file[:-4]))  # 每个段落的标签即为所属小说的文件名（去除后缀）
```

随机选择200个段落

使用random.shuffle()函数进行段落打乱，后选取前200个实现随机选取段落。

```
# 随机选择200个段落
random.shuffle(texts)
texts = texts[:200]
texts_chars = []
topic_chars = []
for text, topic in texts:
    words = list(word for word in jieba.cut(text) if word not in stopwords)
    wod = []
    for word in words:
        if '\u4e00' <= word <= '\u9fa5':
            wod.append(word)
    wod = [char for word in wod for char in word]  # 将每个词转化为其包含的字
    texts_chars.append(wod)
    topic_chars.append(topic)
```

2: Lda 模型算法

构建词典

```
corpus = texts_chars
# 构建词典
vocab = set(word for doc in corpus for word in doc)
word2id = dict((v, idx) for idx, v in enumerate(vocab))
# 将文本数据转换为词频矩阵
M = len(corpus)
V = len(vocab)
X = np.zeros((M, V))
for i, doc in enumerate(corpus):
    for word in doc:
        X[i][word2id[word]] += 1
num_topics_ = len(set(topic_chars))
```

lda 模型训练

```
# 训练 LDA 模型
model = lda.LDA(n_topics=16, n_iter=2000, random_state=1)
X = X.astype(int)
model.fit(X)
```

3: K-Means 聚类

K-Means 算法的流程可以分为以下几个步骤:

1) 初始化聚类中心

在数据集中随机选择 K 个数据点作为初始聚类中心，$u_1^0, u_2^0, u_3^0 \ldots u_k^0$

2) 分配数据点到聚类中心

对于每个数据点，计算其与每个聚类中心之间的距离，将其归为距离最近的聚类中心所在的类别。聚类中心的位置可以通过计算每个簇中所有数据点的平均值得到，称为簇的中心点。 K-Means 算法的优化目标是最小化所有数据点与其所属簇的聚类中心之间的距离的平方和，称为簇内误差平方和（SSE）。SSE 的计算公式为:

$$SSE = \sum_{i=1} \sum_{x \in C_i} \| x - u_i^t \|^2$$

其中，C_i表示第i个簇，u_i^t表示第i个簇的第 t 次迭代中心点，x表示数据点，$\| x - u_i^t \|$表示数据点x与簇中心u_i^t之间的距离。

3) 更新聚类中心

对于每个聚类，计算其中所有数据点的平均值，以此更新聚类中心。

4) 重复步骤 2 和步骤 3，直到聚类中心不再发生变化或达到预定的迭代次数 t。

5) 输出聚类结果

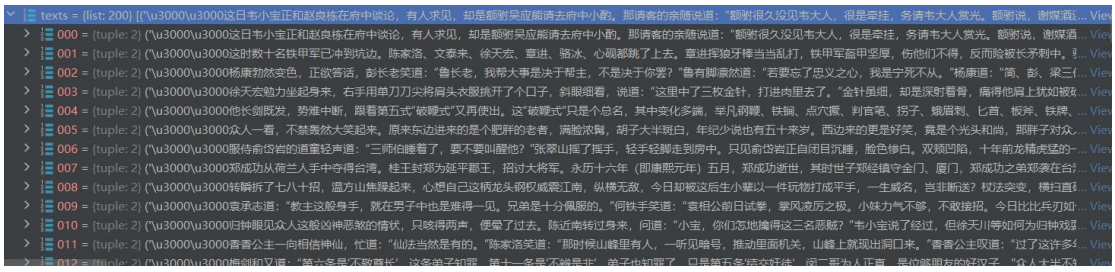
将所有数据点按照所属类别进行分类，输出聚类结果。 需要注意的是，K-Means 算法中的聚类中心是可以动态更新的，即每次更新聚类中心之后，数据点与聚类中心之间的距离也会发生变化，因此需要多次迭代进行聚类。此外，在实际应用中，为了避免陷入局部最优解，需要对初始聚类中心进行多次随机初始化，选择最优的聚类结果。

```
# 定义 KMeans 聚类模型，假设要将文本集合聚为 3 类
kmeans = KMeans(n_clusters=num_topics_, random_state=0)
# 训练 KMeans 模型
kmeans.fit(doc_topic)
# 获得每个文本所属的簇
labels = kmeans.labels_
for i in range(len(labels)):
    print('文本%d 所属簇为：%d' % (i, labels[i]))
```

通过聚类可以将性质相似的段落聚类在同一 topic 下，将聚类结果输出在 excel 文档中显示所挑选所属同一篇文档段落的整体所属 topic 进行分析。

Experimental Studies

段落选取结果



按照词进行 LDA 模型文本分类

以词为单位进行文本分类，并采取不同数量的 topics 进行聚类，对训练结果和聚类结果进行分析。

主题分布示意

```
文本0的主题分布： [1.96582067e-02 1.71829078e-01 9.00433976e-05 1.37366006e-04
 1.46097986e-01 6.19562730e-05 4.20080262e-03 4.66976497e-05
 2.86396223e-05 4.60772123e-05 1.09489025e-01 1.98642130e-05
 4.49143839e-02 2.08940522e-01 1.70829348e-04 4.93468034e-05
 2.45569515e-01 5.29512775e-05 1.25095854e-04 4.84716131e-02]
文本1的主题分布： [2.41360480e-02 1.66687981e-02 1.14363509e-04 8.70195107e-05
 1.14010092e-02 1.56560158e-02 2.67311179e-01 1.40954249e-01
 3.78988519e-05 4.27431439e-05 7.23492587e-05 1.36538359e-02
 1.83731198e-04 2.23210176e-01 1.19306547e-04 1.33760045e-03
 6.48210941e-05 9.19365706e-05 1.87625331e-01 9.72315882e-02]
文本2的主题分布： [1.01358087e-01 1.09272247e-04 3.87253306e-05 1.96607783e-04
 4.76960458e-02 1.88140503e-01 2.13439244e-04 6.06387509e-02
 6.91597128e-05 1.02238527e-03 3.91900324e-04 1.13537013e-04
 4.32173246e-02 1.74281651e-01 3.37156257e-02 1.04726450e-04
 4.92760044e-03 7.31042742e-02 2.70585181e-01 7.52030701e-05]
```

Topics 数量为抽取类别数

首先进行主题词抽取:

```
主题0的单词分布： ['中' '一个' '便' '见' '听' '心中' '说道' '之中' '不知' '心想']
主题1的单词分布： ['黄蓉' '郭靖' '周伯通' '武功' '黄药师' '见' '裘千仞' '欧阳克' '侯通海' '洪七公']
主题2的单词分布： ['杨过' '小龙女' '麽' '於' '公孙谷主' '後' '著' '韦小宝' '剑' '金轮法王']
主题3的单词分布： ['令狐冲' '便' '剑' '道' '虚竹' '长剑' '林平之' '岳不群' '派' '剑法']
主题4的单词分布： ['陈家洛' '见' '胡斐' '文泰来' '铁木真' '张召重' '哲别' '杀' '两人' '马春花']
主题5的单词分布： ['一声' '身子' '左手' '一个' '中' '兵刃' '出' '两人' '突然' '抓住']
主题6的单词分布： ['韦小宝' '康熙' '说' '吴三桂' '皇帝' '皇上' '做' '张咏' '哥哥' '侍卫']
主题7的单词分布： ['道' '说' '说道' '笑' '听' '事' '瞧' '二人' '没' '不敢']
主题8的单词分布： ['袁承志' '见' '青青' '众人' '何铁手' '小人' '只见' '乞丐' '小蛇' '汉子']
主题9的单词分布： ['张无忌' '张翠山' '谢逊' '殷素素' '少女' '教主' '便' '武功' '慕容复' '丐帮']
```

可以看到主题词差异性, 进一步进行聚类后将各文档主题进行标记输出 excel 做进一步分析。

鹿鼎记	0
射雕英雄传	0
射雕英雄传	0
射雕英雄传	0
射雕英雄传	0
射雕英雄传	0
射雕英雄传	0
射雕英雄传	0
射雕英雄传	0
射雕英雄传	0
射雕英雄传	0
射雕英雄传	0
射雕英雄传	0
射雕英雄传	0
射雕英雄传	0
射雕英雄传	0
射雕英雄传	0
射雕英雄传	0
射雕英雄传	0
射雕英雄传	0
射雕英雄传	0
射雕英雄传	0
射雕英雄传	0
射雕英雄传	0
射雕英雄传	0
射雕英雄传	0
射雕英雄传	0
神雕侠侣	0
神雕侠侣	0
神雕侠侣	0
神雕侠侣	0
神雕侠侣	0
神雕侠侣	0
神雕侠侣	0
神雕侠侣	0

飞狐外传	1
笑傲江湖	1
笑傲江湖	1
笑傲江湖	1
笑傲江湖	1
笑傲江湖	1
笑傲江湖	1
笑傲江湖	1
笑傲江湖	1
笑傲江湖	1
笑傲江湖	1
笑傲江湖	1
笑傲江湖	1
笑傲江湖	1
笑傲江湖	1
笑傲江湖	1
笑傲江湖	1
笑傲江湖	1
笑傲江湖	1
笑傲江湖	1
笑傲江湖	1
笑傲江湖	1
倚天屠龙记	1
倚天屠龙记	1
倚天屠龙记	1
倚天屠龙记	1
倚天屠龙记	1
倚天屠龙记	1
倚天屠龙记	1
倚天屠龙记	1
倚天屠龙记	1
倚天屠龙记	1
倚天屠龙记	1
倚天屠龙记	1
倚天屠龙记	1
倚天屠龙记	1

碧血剑	2
碧血剑	2
碧血剑	2
碧血剑	2
碧血剑	2
碧血剑	2
碧血剑	2
碧血剑	2
碧血剑	2
碧血剑	2
碧血剑	2
碧血剑	2
碧血剑	2
飞狐外传	2
飞狐外传	2
飞狐外传	2
飞狐外传	2
飞狐外传	2
飞狐外传	2
飞狐外传	2
飞狐外传	2
飞狐外传	2
飞狐外传	2
飞狐外传	2

碧血剑	3
三十三剑客图	3
三十三剑客图	3
书剑恩仇录	3
书剑恩仇录	3
书剑恩仇录	3
书剑恩仇录	3
书剑恩仇录	3
书剑恩仇录	3
书剑恩仇录	3
书剑恩仇录	3
书剑恩仇录	3
书剑恩仇录	3
书剑恩仇录	3
倚天屠龙记	3

碧血剑	4
碧血剑	4
碧血剑	4
碧血剑	4
射雕英雄传	4
射雕英雄传	4
射雕英雄传	4
射雕英雄传	4
射雕英雄传	4
射雕英雄传	4
书剑恩仇录	4
书剑恩仇录	4
书剑恩仇录	4
书剑恩仇录	4
书剑恩仇录	4
书剑恩仇录	4
书剑恩仇录	4
书剑恩仇录	4
天龙八部	4

鹿鼎记	5
鹿鼎记	5
鹿鼎记	5
鹿鼎记	5
鹿鼎记	5
鹿鼎记	5
鹿鼎记	5
鹿鼎记	5
鹿鼎记	5
鹿鼎记	5
鹿鼎记	5
鹿鼎记	5
鹿鼎记	5
鹿鼎记	5
三十三剑客图	5
书剑恩仇录	5

碧血剑	6
碧血剑	6
碧血剑	6
碧血剑	6
碧血剑	6
鹿鼎记	6
鹿鼎记	6
射雕英雄传	6
射雕英雄传	6
射雕英雄传	6
书剑恩仇录	6

射雕英雄传	7
射雕英雄传	7
天龙八部	7
天龙八部	7
天龙八部	7
天龙八部	7
天龙八部	7
天龙八部	7
天龙八部	7
天龙八部	7
笑傲江湖	7
笑傲江湖	7
笑傲江湖	7
笑傲江湖	7
笑傲江湖	7
笑傲江湖	7
笑傲江湖	7
倚天屠龙记	7
倚天屠龙记	7

射雕英雄传	8
书剑恩仇录	8
笑傲江湖	8
笑傲江湖	8
笑傲江湖	8
笑傲江湖	8
倚天屠龙记	8
倚天屠龙记	8
倚天屠龙记	8
倚天屠龙记	8
倚天屠龙记	8
倚天屠龙记	8
倚天屠龙记	8
倚天屠龙记	8
倚天屠龙记	8

碧血剑	9
鹿鼎记	9
鹿鼎记	9
书剑恩仇录	9
书剑恩仇录	9
笑傲江湖	9
笑傲江湖	9
笑傲江湖	9
笑傲江湖	9
笑傲江湖	9

由表格统计可以看到迭代 1000 次的情况下，对于主题的提取有效，根据大致占比能看出各个文档所属主题。但仍存在误差，部分主题中成分较为平均不具有典型性。

Topics 数量为 20

首先进行主题词抽取:

```
主题0的单词分布： ['说' '道' '师父' '说道' '事' '死' '心中' '不能' '没有' '做']
主题1的单词分布： ['令狐冲' '老者' '英雄' '前辈' '晚辈' '铁门' '丹青生' '一条' '无人' '老爷子']
主题2的单词分布： ['胡斐' '卫士' '任通武' '二人' '胡斐道' '众' '那书生' '王维扬' '上官铁生' '福康安']
主题3的单词分布： ['清兵' '众' '兵' '北京' '蒙古' '罗刹' '说' '侍卫' '鞑子' '没有']
主题4的单词分布： ['听' '一声' '突然' '身子' '出' '左手' '眼见' '之下' '敌人' '手中']
主题5的单词分布： ['袁承志' '青青' '崔秋山' '见' '两个' '孙仲君' '大厅' '师哥' '水' '总兵']
主题6的单词分布： ['洪七公' '欧阳锋' '黄蓉' '二人' '郭靖' '吃' '听' '两人' '师父' '欧阳克']
主题7的单词分布： ['武功' '一招' '功夫' '便' '内力' '剑法' '剑' '劲力' '非' '义父']
主题8的单词分布： ['张无忌' '赵敏' '虚竹' '大师' '教主' '便' '和尚' '头陀' '周芷若' '高手']
主题9的单词分布： ['令狐冲' '道' '师父' '剑' '岳不群' '仪琳' '尼姑' '岳灵珊' '华山派' '派']
主题10的单词分布： ['道' '陆菲青' '张召重' '文泰来' '群雄' '众人' '皇帝' '请' '李沅芷' '陈家洛']
主题11的单词分布： ['袁崇焕' '刘挥' '吕用之' '愿' '故事' '许寂' '写' '毛文龙' '皇帝' '时']
主题12的单词分布： ['张翠山' '谢逊' '二人' '父亲' '殷素素' '长剑' '几句话' '一块' '卫璧' '剑尖']
主题13的单词分布： ['道' '中' '见' '一个' '便' '想' '走' '没' '只见' '之中']
主题14的单词分布： ['派' '一个' '慕容复' '剑' '本门' '弟子' '声音' '洞' '下去' '功夫']
主题15的单词分布： ['杨过' '武三通' '麽' '後' '武修文' '著' '於' '樊一翁' '甚' '武敦儒']
主题16的单词分布： ['陈家洛' '霍青桐' '徐天宏' '乾隆' '少女' '卫春华' '心砚' '木卓伦' '周' '绮']
主题17的单词分布： ['韦小宝' '银子' '里' '出' '骰子' '白衣' '掷' '马' '儿子' '干']
主题18的单词分布： ['道' '说' '说道' '听' '武功' '不敢' '请' '便' '这位' '兄弟']
主题19的单词分布： ['郭靖' '周伯通' '黄药师' '梁子翁' '何红药' '完颜洪烈' '王妃' '小子' '完颜康' '画']
```

可以看到主题词差异性，进一步进行聚类后将各文档主题进行标记输出 excel 做进一步分析。

Name	类别								
碧血剑	0	笑傲江湖	1	天龙八部	2	碧血剑	3	碧血剑	4
碧血剑	0	笑傲江湖	1	天龙八部	2	射雕英雄	3	碧血剑	4
碧血剑	0	笑傲江湖	1	天龙八部	2	书剑恩仇	3	碧血剑	4
射雕英雄	0	笑傲江湖	1	笑傲江湖	2	书剑恩仇	3	三十三剑	4
射雕英雄	0	笑傲江湖	1	笑傲江湖	2	书剑恩仇	3	三十三剑	4
射雕英雄	0	笑傲江湖	1	倚天屠龙	2	书剑恩仇	3	三十三剑	4
射雕英雄	0	笑傲江湖	1	倚天屠龙	2	书剑恩仇	3	三十三剑	4
射雕英雄	0	笑傲江湖	1	倚天屠龙	2	书剑恩仇	3	天龙八部	4
射雕英雄	0	笑傲江湖	1	倚天屠龙	2	书剑恩仇	3	倚天屠龙	4
射雕英雄	0	笑傲江湖	1	倚天屠龙	2	书剑恩仇	3		
射雕英雄	0	笑傲江湖	1	倚天屠龙	2	书剑恩仇	3		
射雕英雄	0	笑傲江湖	1	倚天屠龙	2	书剑恩仇	3		
射雕英雄	0	笑傲江湖	1	倚天屠龙	2	书剑恩仇	3		
射雕英雄	0	笑傲江湖	1	倚天屠龙	2	书剑恩仇	3		
射雕英雄	0	笑傲江湖	1	倚天屠龙	2	书剑恩仇	3		
射雕英雄	0	笑傲江湖	1	倚天屠龙	2	书剑恩仇	3		
射雕英雄	0	笑傲江湖	1			书剑恩仇	3		
射雕英雄	0	笑傲江湖	1			书剑恩仇	3		
射雕英雄	0	笑傲江湖	1			书剑恩仇	3		
射雕英雄	0	笑傲江湖	1			书剑恩仇	3		
射雕英雄	0					笑傲江湖	3		
射雕英雄	0					笑傲江湖	3		
射雕英雄	0					笑傲江湖	3		
射雕英雄	0					笑傲江湖	3		
射雕英雄	0					笑傲江湖	3		
射雕英雄	0					笑傲江湖	3		
射雕英雄	0					倚天屠龙	3		
神雕侠侣	0								
书剑恩仇	0								

飞狐外传	6	碧血剑	7	碧血剑	8	飞狐外传	9
鹿鼎记	6	鹿鼎记	7	碧血剑	8	飞狐外传	9
鹿鼎记	6	鹿鼎记	7	碧血剑	8	飞狐外传	9
鹿鼎记	6	鹿鼎记	7	碧血剑	8	飞狐外传	9
鹿鼎记	6	鹿鼎记	7	碧血剑	8	飞狐外传	9
鹿鼎记	6	射雕英雄	7	碧血剑	8	飞狐外传	9
鹿鼎记	6	射雕英雄	7	碧血剑	8	飞狐外传	9
鹿鼎记	6	书剑恩仇	7	碧血剑	8	飞狐外传	9
射雕英雄	6	书剑恩仇	7	碧血剑	8	飞狐外传	9
		倚天屠龙	7	碧血剑	8	书剑恩仇	9
		倚天屠龙	7	碧血剑	8	天龙八部	9
				碧血剑	8	天龙八部	9
				碧血剑	8		
				碧血剑	8		
				碧血剑	8		
				飞狐外传	8		
				飞狐外传	8		
				射雕英雄	8		

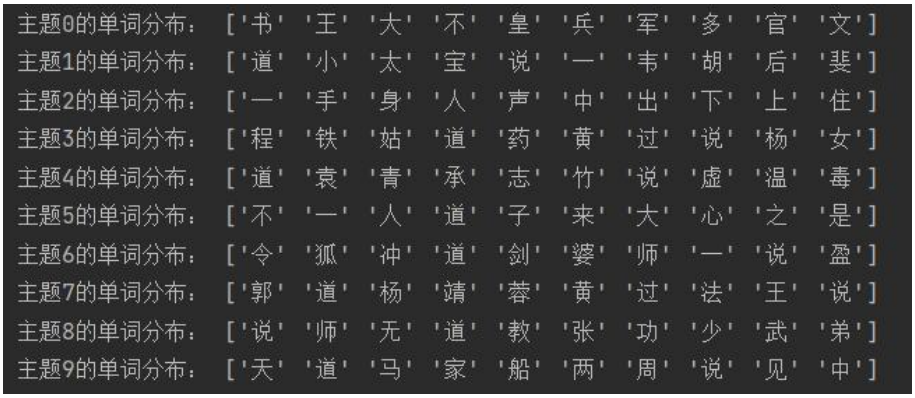
碧血剑	5
碧血剑	5
飞狐外传	5
飞狐外传	5
飞狐外传	5
飞狐外传	5
鹿鼎记	5
鹿鼎记	5
射雕英雄	5
射雕英雄	5
神雕侠侣	5
神雕侠侣	5
书剑恩仇	5
书剑恩仇	5
书剑恩仇	5
书剑恩仇	5
书剑恩仇	5
书剑恩仇	5
书剑恩仇	5
书剑恩仇	5
书剑恩仇	5
书剑恩仇	5
天龙八部	5
天龙八部	5
天龙八部	5
天龙八部	5
天龙八部	5
天龙八部	5
天龙八部	5
天龙八部	5
笑傲江湖	5
笑傲江湖	5
笑傲江湖	5
笑傲江湖	5
倚天屠龙	5
倚天屠龙	5
倚天屠龙	5
倚天屠龙	5
倚天屠龙	5
倚天屠龙	5
倚天屠龙	5
倚天屠龙	5
倚天屠龙	5
倚天屠龙	5
倚天屠龙	5
倚天屠龙	5
倚天屠龙	5
倚天屠龙	5
倚天屠龙	5

由表格统计可以看到迭代 1000 次的情况下，对于主题的提取有效，根据大致占比能看出各个文档所属主题。效果优于 topics 数为抽取类别数时的情况，当主题数划分更细时会得到更准确地主题聚类结果。

按照字进行 LDA 模型文本分类

Topics 数量为抽取类别数

首先进行主题字抽取：



可以看到主题字差异性，相较于词，主题字的特征更不直观，进一步进行聚类后将各文档主题进行标记输出 excel 做进一步分析。

Name	类别				
射雕英雄	0	碧血剑	1	飞狐外传	2
射雕英雄	0	碧血剑	1	飞狐外传	2
射雕英雄	0	鹿鼎记	1	飞狐外传	2
射雕英雄	0	射雕英雄	1	飞狐外传	2
射雕英雄	0	射雕英雄	1	飞狐外传	2
射雕英雄	0	射雕英雄	1	鹿鼎记	2
射雕英雄	0	神雕侠侣	1	射雕英雄	2
射雕英雄	0	神雕侠侣	1	书剑恩仇	2
射雕英雄	0	神雕侠侣	1	书剑恩仇	2
射雕英雄	0	书剑恩仇	1	书剑恩仇	2
射雕英雄	0	书剑恩仇	1	书剑恩仇	2
神雕侠侣	0	书剑恩仇	1	书剑恩仇	2
神雕侠侣	0	书剑恩仇	1	书剑恩仇	2
神雕侠侣	0	书剑恩仇	1	天龙八部	2
神雕侠侣	0	书剑恩仇	1	笑傲江湖	2
		书剑恩仇	1	倚天屠龙	2
		书剑恩仇	1	倚天屠龙	2
		书剑恩仇	1	倚天屠龙	2
		书剑恩仇	1		
		书剑恩仇	1		
		倚天屠龙	1		
		倚天屠龙	1		
		倚天屠龙	1		
		倚天屠龙	1		

碧血剑	3
碧血剑	3
碧血剑	3
碧血剑	3
飞狐外传	3
飞狐外传	3
飞狐外传	3
飞狐外传	3
飞狐外传	3
飞狐外传	3
飞狐外传	3
飞狐外传	3
射雕英雄	3
射雕英雄	3
射雕英雄	3
射雕英雄	3
射雕英雄	3
射雕英雄	3
射雕英雄	3
射雕英雄	3
射雕英雄	3
射雕英雄	3
书剑恩仇	3
书剑恩仇	3
书剑恩仇	3
天龙八部	3
天龙八部	3
笑傲江湖	3
笑傲江湖	3
笑傲江湖	3
笑傲江湖	3
笑傲江湖	3
笑傲江湖	3
笑傲江湖	3
笑傲江湖	3
倚天屠龙	3
倚天屠龙	3
倚天屠龙	3
倚天屠龙	3

碧血剑	4	笑傲江湖	5	碧血剑	6	碧血剑	7	飞狐外传	8	飞狐外传	9
碧血剑	4	笑傲江湖	5	飞狐外传	6	碧血剑	7	天龙八部	8	飞狐外传	9
碧血剑	4	笑傲江湖	5	飞狐外传	6	碧血剑	7	天龙八部	8	飞狐外传	9
飞狐外传	4	笑傲江湖	5	鹿鼎记	6	碧血剑	7	笑傲江湖	8	飞狐外传	9
鹿鼎记	4	笑傲江湖	5	射雕英雄	6	碧血剑	7	笑傲江湖	8	鹿鼎记	9
鹿鼎记	4	笑傲江湖	5	射雕英雄	6	碧血剑	7	笑傲江湖	8	鹿鼎记	9
鹿鼎记	4	笑傲江湖	5	书剑恩仇	6	碧血剑	7	笑傲江湖	8	鹿鼎记	9
鹿鼎记	4	笑傲江湖	5	天龙八部	6	碧血剑	7	倚天屠龙	8	鹿鼎记	9
鹿鼎记	4	笑傲江湖	5	笑傲江湖	6	碧血剑	7	倚天屠龙	8	鹿鼎记	9
鹿鼎记	4	笑傲江湖	5	笑傲江湖	6	碧血剑	7	倚天屠龙	8	鹿鼎记	9
鹿鼎记	4	笑傲江湖	5	笑傲江湖	6	碧血剑	7	倚天屠龙	8	鹿鼎记	9
鹿鼎记	4	笑傲江湖	5	笑傲江湖	6	碧血剑	7	倚天屠龙	8	鹿鼎记	9
鹿鼎记	4	笑傲江湖	5	笑傲江湖	6	碧血剑	7	倚天屠龙	8	鹿鼎记	9
三十三剑	4	笑傲江湖	5	倚天屠龙	6	碧血剑	7	倚天屠龙	8	射雕英雄	9
书剑恩仇	4	笑傲江湖	5	倚天屠龙	6	天龙八部	7	倚天屠龙	8		
		笑傲江湖	5			天龙八部	7	倚天屠龙	8		
		笑傲江湖	5			天龙八部	7	倚天屠龙	8		
		笑傲江湖	5			天龙八部	7	倚天屠龙	8		
						倚天屠龙	7	倚天屠龙	8		
								倚天屠龙	8		
								倚天屠龙	8		
								倚天屠龙	8		
								倚天屠龙	8		

由表格统计可以看到迭代 1000 次的情况下，对于主题的提取有效，根据大致占比能看出各个文档所属主题。但仍存在误差，部分主题中成分较为平均不具有典型性，且词为单位进行分词的结果优于字为单位，词语更能体现文档属性。

Conclusions

Lda 模型能较好的实现文档分类的功能，随着迭代的次数上升，分词的精度上升，在对比实验的过程中可以得到以下结论：主题的设计也会影响分词精度，一定程度上提升主题数能够更好的表征文档，将文档进行更细致的分类；在使用词为单位时进行分类会有更高的精度，词语所能表征的含义更具体和丰富。

附录

```
import numpy as np
import os
import jieba
import random
from gensim.models.ldamodel import LdaModel
from gensim.corpora.dictionary import Dictionary
from collections import Counter
import lda
from sklearn.cluster import KMeans
import openpyxl


with open('./stopwords.txt', 'r', encoding='utf-8') as f:
```

```
    stopwords = set([line.strip() for line in f])
# 读取小说语料库中的所有文本
corpus_path = 'txt'
texts = []
for file in os.listdir(corpus_path):
    with open(os.path.join(corpus_path, file), 'r', encoding='ansi') as f:
        text = f.read()
        # 将每个文本按句子分割成段落
        paragraphs = text.split('\n')
        for p in paragraphs:
            if len(p) > 500:  # 只选取长度大于 500 的段落
                texts.append((p, file[:-4]))  # 每个段落的标签即为所属小说的文件名（去除后缀）

# 随机选择 200 个段落
random.shuffle(texts)
texts = texts[:200]
texts_chars = []
topic_chars = []
for text, topic in texts:
    words = list(word for word in jieba.cut(text) if word not in stopwords)
    wod = []
    for word in words:
        if '\u4e00' <= word <= '\u9fa5':
            wod.append(word)
    wod = [char for word in wod for char in word]  # 将每个词转化为其包含的字
    texts_chars.append(wod)
    topic_chars.append(topic)

corpus = texts_chars
# 构建词典
vocab = set(word for doc in corpus for word in doc)
word2id = dict((v, idx) for idx, v in enumerate(vocab))
# 将文本数据转换为词频矩阵
M = len(corpus)
V = len(vocab)
X = np.zeros((M, V))
for i, doc in enumerate(corpus):
    for word in doc:
        X[i][word2id[word]] += 1
num_topics_ = len(set(topic_chars))
# 训练 LDA 模型
model = lda.LDA(n_topics=16, n_iter=2000, random_state=1)
X = X.astype(int)
model.fit(X)
```

```
# 输出每个主题的单词分布
topic_word = model.topic_word_
for i, topic_dist in enumerate(topic_word):
    topic_words = np.array(list(vocab))[np.argsort(topic_dist)][:-(10+1):-1]
    print('主题%d 的单词分布：' % i, topic_words)
# 获得训练文本所属主题信息
doc_topic = model.transform(X)
for i in range(len(corpus)):
    print('文本%d 的主题分布：' % i, doc_topic[i])
# 定义 KMeans 聚类模型，假设要将文本集合聚为 3 类
kmeans = KMeans(n_clusters=num_topics_, random_state=0)
# 训练 KMeans 模型
kmeans.fit(doc_topic)
# 获得每个文本所属的簇
labels = kmeans.labels_
for i in range(len(labels)):
    print('文本%d 所属簇为：%d' % (i, labels[i]))
workbook = openpyxl.Workbook()
worksheet = workbook.active
worksheet.title = 'Sheet1'
worksheet['A1'] = 'Name'
worksheet['B1'] = '类别'
i = 0
for topic in topic_chars:
    row = i + 2
    worksheet.cell(row=row, column=1, value=topic)
    worksheet.cell(row=row, column=2, value=labels[i])
    i += 1
workbook.save('data1.xlsx')
```