LDA 建模以及文本分类

一、实验要求

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。

二、实验原理

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 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上述某个主题，再以一定的概率选取那个主题下的某个单词，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：

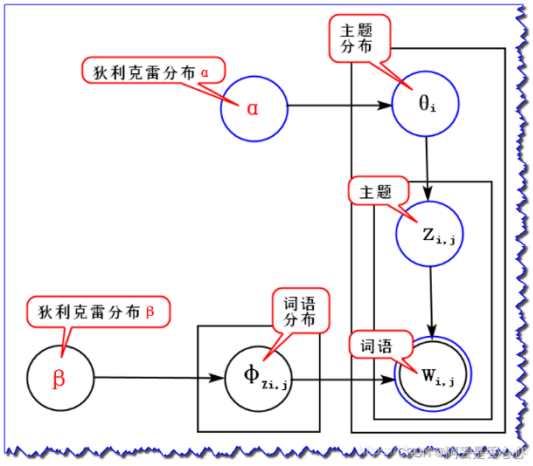

图 1 LDA 模型中文档的生成方式

从狄利克雷分布中取样生成文档 i 的主题分布θ_i

从主题的多项式分布中取样生成文档 i 第 j 个词的主题$z_{i,j}$

从狄利克雷分布中取样生成主题的词语分布$\emptyset_{z_{i,j}}$

从词语的多项式分布中采样最终生成词语$w_{i,j}$

三、实现方法

3.1 文本预处理

该步骤包括特殊标点符号、停词、无效编码以及与书籍内容无关的广告信息

等。读取语料后，通过预处理可以滤除这些和文本语义信息无关的内容。

3.2 段落的截取

考虑训练集需要 200 个段落，训练集和测试集的比例定为 7：3，所以共需要采集 300 个段落左右。因为共有 16 本小说，为了构建一个样本分配均衡的数据集，每本小说采集 20 个段落。实验中发现，单本小说中词数超过 500 的段落数量很有限，因此采用小段落拼组成大段落的方法进行段落的随机选取。

3.3 LDA 模型的构建

将第二步选取出的段落构建词袋模型，并且滤除掉出现频率过少以及过多的词语。最终得到的语料采用 gensim 库中的 LDA 模型构建。

设定 topic 数为 10，训练次数为 400。每 100 个迭代对模型进行一次评价。在后续过程中，通过不断调节 topic 的数目，并观察分类器的结果，来得到一个最佳的 topic 数。

3.4 分类器的构建

以主题分布作为样本，对应的书籍作为标签，使用支持向量机作为关于主题分布的文本分类器，采用多项式核函数。训练完成后采用测试集进行测试，采用平均准确率作为评价指标。

四、实验结果

当 topic 设为 10 的情况下主题分布如下图所示：

```
(Pdb) topics_
array([[2.5968409e-05, 4.7755722e-05, 3.3219403e-05, ..., 9.9962640e-01,
        5.7402056e-05, 2.7041720e-05],
       [2.5978408e-05, 4.7774109e-05, 3.3232194e-05, ..., 3.1026324e-05,
        5.7424157e-05, 2.7052132e-05],
       [2.5980666e-05, 4.7778260e-05, 3.3235083e-05, ..., 3.1029020e-05,
        9.9965262e-01, 2.7054482e-05],
       ...,
       [2.5976669e-05, 4.7770911e-05, 9.9962848e-01, ..., 3.1024247e-05,
        5.7420311e-05, 2.7050321e-05],
       [2.5975871e-05, 4.7769445e-05, 3.3228949e-05, ..., 3.1023294e-05,
        5.7418547e-05, 2.7049489e-05],
       [2.5969999e-05, 4.7758647e-05, 9.9962860e-01, ..., 3.1016283e-05,
        5.7405570e-05, 2.7043377e-05]], dtype=float32)
```

对应的书籍编码(0~15 对应 16 本小说)为

```
(Pdb) bookid
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1
, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
3, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5,
 5, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7
, 7, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9,
9, 9, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 11, 11, 11, 11, 11, 11,
 11, 11, 11, 11, 11, 11, 11, 11, 11, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12,
12, 12, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 14, 14, 14, 14, 14, 1
4, 14, 14, 14, 14, 14, 14, 14, 14, 14, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15
, 15, 15]
```

LDA 训练完成后，不同主题间的差异为 0.09。可以看到不同主题对应的词语的分布情况：

topic #0	*皇上*	*宝刀*	*天下*	*卫士*	*皇帝*	*杀*	*少林*	*自然*	*做*
概率	0.05	0.02	0.02	0.02	0.02	0.01	0.01	0.01	0.01
topic #4	*有人*	*死*	*走*	*长剑*	*两人*	*出*	*脸上*	*武功*	*会*
概率	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01
topic #1	*做*	*事*	*时*	*死*	*之后*	*会*	*派*	*后来*	*英雄*
概率	0.02	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01

采用主题分布作为样本，对应的书籍作为标签，训练 svm。在训练集上测试，测试的 mean_acc=23.9%

改用不同的 topic 数进行训练，得到的准确率如下表所示：

	Topic=5	Topic=10	Topic=15
Topic_difference	0.08	0.09	0.12
mean_acc	22.9%	23.9%	28.1%

五、实验总结

本次实验尝试通过构建 LDA 模型以及支持向量机对文本段落进行分类。让我更熟悉了 LDA 中两步采样的思想，并学会了如何调用 gensim 包实现 LDA 模型。

但是通过观察最后的分类准确率，还不是很理想。通过观察 LDA 模型训练完成后不同主题间的差异，发现差异较小，说明还需要滤除不同语料中反复出现的词语，增加不同 topic 间的差异性，从而得到更好的分类特征。

六、代码实现

```
from gensim.models import LdaModel
from gensim import corpora
import jieba
from sklearn import svm
import numpy as np
from sklearn.model_selection import train_test_split as ts

import os
import re
import logging
logging.basicConfig(format='%(asctime)s : %(levelname)s : %(message)s',
level=logging.INFO)

class TraversalFun():

    # 1 初始化
    def __init__(self, rootDir):
        self.rootDir = rootDir

    def TraversalDir(self):
        return self.getCorpus(self.rootDir)

    def getCorpus(self, rootDir):
        corpus = []
        r1 = u'[a-zA-Z0-9’!"#$%&\'()*+,-./:: ;<=>?@，。?★、…【】《》？“”‘’！
[\\]^_`{|}~]+'  # 用户也可以在此进行自定义过滤字符
        listdir = os.listdir(rootDir)
        count=0
        for file in listdir:
            path  = os.path.join(rootDir, file)
            if os.path.isfile(path):
                with open(os.path.abspath(path), "r", encoding='gbk',
errors="ignore") as file:
                    filecontext = file.read()
                    filecontext = re.sub(r1, '', filecontext)
                    filecontext = filecontext.replace("\n", '')
                    filecontext = filecontext.replace(" ", '')
                    filecontext = filecontext.replace("\u3000", '')
```

```
                    filecontext = filecontext.replace("本书来自 www.cr173.com
免费 txt 小说下载站\n 更多更新免费电子书请关注 www.cr173.com", '')
                    filecontext = filecontext.replace("本书来自免费小说下载站
\n 更多更新免费电子书请关注", '')
                    #seg_list = jieba.cut(filecontext, cut_all=True)
                    #corpus += seg_list
                    count += len(filecontext)
                    corpus.append(filecontext)
        return corpus,count

def construct_dataset(corpus, paragraphs_per_book=15,
words_per_paragraph=500):
    paragraphs, bookid = random_select(corpus, paragraphs_per_book,
words_per_paragraph)

    dictionary = corpora.Dictionary(paragraphs)
    dictionary.filter_extremes(no_below=20, no_above=0.5)
    dictionary.compactify()
    corpus = [dictionary.doc2bow(text) for text in paragraphs]

    # Set training parameters.
    num_topics = 15
    chunksize = 2000
    passes = 20
    iterations = 400
    eval_every = 100  # Don't evaluate model perplexity, takes too much
time.

    lda = LdaModel(
        corpus=corpus,
        id2word=dictionary,
        chunksize=chunksize,
        alpha='auto',
        eta='auto',
        iterations=iterations,
        num_topics=num_topics,
        passes=passes,
        eval_every=eval_every
```

```
    )

    topics_ = lda.get_document_topics(corpus, minimum_probability=0)
    topics = [[t[1] for t in topic] for topic in topics_]
    topics = np.array(topics)
    bookid = np.array(bookid)
    return topics, bookid


def random_select(corpus, paragraphs_per_book, words_per_paragraph):
    paragraphs = list()
    bookid = list()
    for i, text in enumerate(corpus):
        stopwords = get_stopwords()
        text = [p for p in jieba.cut(text) if p not in stopwords]
        for _ in range(paragraphs_per_book):
            p = np.random.randint(0, len(text)-words_per_paragraph)
            paragraphs.append(text[p:p+words_per_paragraph])
            bookid.append(i)
    return paragraphs, bookid


def get_stopwords():
    with open("./cn_stopwords.txt", "r") as f:
        stopwords = f.readlines()

    return [word.replace('\n', '') for word in stopwords]

def main():
    # 准备数据
    tra = TraversalFun("./datasets")
    corpus,count = tra.TraversalDir()
    data, target = construct_dataset(corpus, 20, 500)
    X_train,X_test,y_train,y_test = ts(data, target, test_size=0.3)
    # kernel = 'rbf'
    clf_rbf = svm.SVC(kernel='poly')
    clf_rbf.fit(X_train,y_train)
    score_rbf = clf_rbf.score(X_test,y_test)
```

```
    print("The score of rbf is : %f"%score_rbf)



if __name__ == '__main__':
    main()
```