

成 绩

****

深度学习与自然语言处理

第3次大作业

LDA主题模型进行文本分类

|  |  |
| --- | --- |
| 院（系）名称 | 自动化科学与电气工程学院 |
| 专业名称 | 电子信息 |
| 学号 | ZY2103113 |
| 姓名 | 孙茗逸 |
| 指导教师 | 秦曾昌 |

2022年5月6日

1. **任务描述**

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。

1. **实验原理**
2. **LDA主题模型**

在文本挖掘领域中大量的数据都是非结构化的，难以从信息中直接获取相关和期望的信息。主题模型（Topic Model）能够识别在文档里的主题，并且挖掘语料里隐藏信息，在主题聚合、特征选择等场景有广泛的用途。

LDA（Latent Dirichlet Allocation）是一种文档主题生成模型，也称为一个三层贝叶斯概率模型，包含词、主题和文档三层结构。所谓生成模型，就是说，我们认为一篇文章的每个词都是通过“以一定概率选择了某个主题，并从这个主题中以一定概率选择某个词语"”这样一个过程得到。文档到主题服从多项式分布，主题到词服从多项式分布。

LDA采用了“词袋”的方法，这种方法将每一篇文档视为一个词频向量，从而将文本信息转化为了易于建模的数字信息。但是没有考虑词与词之间的顺序，这简化了问题的复杂性。每一篇文档代表了一些主题所构成的一个概率分布，而每一个主题又代表了很多单词所构成的一个概率分布。

1) 按照先验概率选择一篇文档

2) 从狄利克雷分布中取样生成文档的主题分布

3) 从主题的多项式分布中取样生成文档第j个词的主题

4) 从狄利克雷分布β中取样生成主题对应的词语分布

5) 从词语的多项式分布，中采样最终生成词语

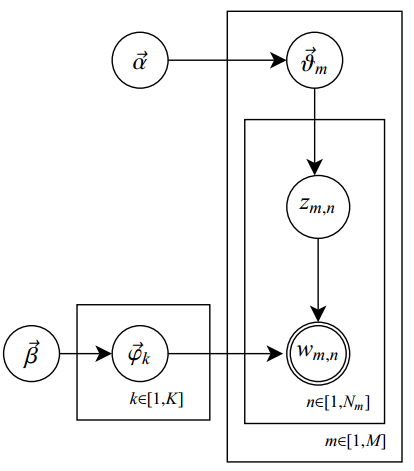


图 1 LDA模型

1. **利用LDA主题模型进行文本分类**

本文采用以下步骤/思路对金庸的小说集进行文本分类：

1. 从给定的16本金庸小说数据集中，随机、均匀地抽取个段落，每个段落的标签为对应小说的小说名，每个段落包含个字（），每个段落作为一个样本；
2. 将随机抽取的个段落（即个样本）中的作为训练样本，剩余作为测试样本。训练样本数为，测试样本数为；
3. 指定主题数为，利用上述个训练样本训练LDA模型；
4. 利用训练好的LDA模型得到上述个训练样本的主题分布。由于主题数为，因此每个训练样本得到的主题分布为一个的向量；所有训练样本的主题分布则为一个的特征向量；
5. 利用上述训练样本的的特征向量以及对应的个标签训练一个线性SVM分类器；
6. 上述训练样本的的特征向量通过训练好的SVM分类器，得到训练样本的预测标签，与真实的标签进行比较，计算训练样本文本分类准确率；
7. 利用训练好的LDA模型得到个测试样本的主题分布。同理，由于主题数为，因此每个测试样本得到的主题分布为一个的向量；所有测试样本的主题分布则为一个的特征向量；该特征向量通过训练好的SVM分类器，得到测试样本的预测标签，与真实的标签进行比较，计算测试样本文本分类准确率。

其中，上述步骤（1）~（3）为数据准备、预处理和训练LDA模型；步骤（4）（5）为训练线性SVM分类器；步骤（6）（7）为计算训练和测试样本的文本分类准确率。

1. **实验结果**

本次实验测试了不同的段落（文档）数、每个段落的字数、不同主题数对文本分类准确率的影响。此外，还考察了是否去除停用词对分类准确率的影响。没有什么实际含义的功能词，或用十分广泛但对这样的词搜索引擎无法保证能够给出真正相关的搜索结果、难以帮助缩小搜索范围的词。实验中停用词表由百度、哈工大等创造的停用词表给出。去除停用词有助于数据清洗得更干净、获得的文本更具有实际含义。

实验结果如下表所示。

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **序号** | **主题数** | **段落数** | **每段字数** | **是否去除停用词** | **训练集**  **准确率（%）** | **测试集**  **准确率（%）** |
| 1 | 50 | 200 | 500 | 否 | 34.38 | 2.50 |
| 2 | 50 | 200 | 500 | 是 | 41.25 | 12.50 |
| 3 | 50 | 1000 | 500 | 否 | 23.25 | 15.50 |
| 4 | 50 | 1000 | 500 | 是 | 30.63 | 25.00 |
| 5 | 50 | 1000 | 5000 | 否 | 44.00 | 46.00 |
| 6 | 50 | 1000 | 5000 | 是 | 52.75 | 55.00 |

1. **结果分析**

从实验结果可以看出：

1. 对比1和2，或3和4，或5和6的结果，可以看出，去除无意义的停用词，可以增强样本中文本的实际含义，显著增强训练集和测试集的文本分类正确率；
2. 对比1和3，或2和4的结果，可以看出，增加抽取的段落（文档）数，可以显著提高测试集文本分类准确率，但训练集文本分类准确率有所降低，可能是因为训练样本（段落数）太少的时候，训练集上容易引起过拟合导致；
3. 对比3和5，或4和6的结果，可以看出，增加抽取段落的每段话字数，可以显著增加训练集和测试集上的文本分类准确率；

附录：实验代码

import jieba

import os

import re

import time

import math

import numpy as np

import random

from gensim import corpora, models

from collections import defaultdict

from sklearn.svm import SVC

def data\_preprocessing(data\_roots, abandon\_stop\_words):

listdir = os.listdir(data\_roots)

char\_to\_be\_replaced = "\n `1234567890-=/\*-~!@#$%^&\*()\_+qwertyuiop[]\\QWERTYUIOP{}|asdfghjkl;" \

"'ASDFGHJKL:\"zxcvbnm,./ZXCVBNM<>?~！@#￥%……&\*（）——+【】：；“‘’”《》？，。" \

"、★「」『』～＂□ａｎｔｉ－ｃｌｉｍａｘ＋．／０１２３４５６７８９＜＝＞＠Ａ" \

"ＢＣＤＥＦＧＨＩＪＫＬＭＮＯＰＱＲＳＴＶＷＸＹＺ［＼］ｂｄｅｆｇｈｊｋｏｐｒｓ" \

"ｕｖｗｙｚ￣\u3000\x1a"

char\_to\_be\_replaced = list(char\_to\_be\_replaced)

txt\_corpus = []

label\_idxes = []

label\_words = []

label\_idx = 0

label\_idx\_to\_words = dict()

stop\_words\_list = []

for tmp\_file\_name in os.listdir("/data1/dxy/codes/NLP\_homework/NLP\_homework3/stopwords/"): # replace this path with the stopwords path

with open("/data1/dxy/codes/NLP\_homework/NLP\_homework3/stopwords/"+tmp\_file\_name, "r", encoding="utf-8", errors="ignore") as f:

stop\_words\_list.extend([word.strip('\n') for word in f.readlines()])

for tmp\_file\_name in listdir:

if tmp\_file\_name == "inf.txt":

continue

path = os.path.join(data\_roots, tmp\_file\_name)

if os.path.isfile(path):

with open(path, "r", encoding="gbk", errors="ignore") as tmp\_file:

tmp\_file\_context = tmp\_file.read()

for tmp\_char in char\_to\_be\_replaced:

tmp\_file\_context = tmp\_file\_context.replace(tmp\_char, "")

tmp\_file\_context = tmp\_file\_context.replace("本书来自免费小说下载站更多更新免费电子书请关注", "")

if abandon\_stop\_words:

for tmp\_char in stop\_words\_list:

tmp\_file\_context = tmp\_file\_context.replace(tmp\_char, "")

txt\_corpus.append(tmp\_file\_context)

label\_idxes.append(label\_idx)

label\_words.append(tmp\_file\_name.split(".txt")[0])

label\_idx\_to\_words[label\_idx] = tmp\_file\_name.split(".txt")[0]

label\_idx += 1

return txt\_corpus, label\_idxes, label\_words, label\_idx\_to\_words

if \_\_name\_\_ == '\_\_main\_\_':

num\_topics = 50

num\_docs = 200

len\_per\_doc = 500

abandon\_stop\_words = True

print("主题数：{}，段落(文档)数：{}，每段话字数：{}，是否去除停用词：{}".format(num\_topics, num\_docs, len\_per\_doc, "yes" if abandon\_stop\_words else "no"))

print("Preparing data...")

data\_roots = '/data1/dxy/codes/NLP\_homework/NLP\_homework1/txt\_files/' # replace this path with the txt files path

txt\_corpus, label\_idxes, label\_words, label\_idx\_to\_words = data\_preprocessing(data\_roots, abandon\_stop\_words)

whole\_samples = []

#### get training samples and testing samples

for i in range(len(txt\_corpus)):

for j in range(num\_docs//len(txt\_corpus) + 1):

tmp\_start = random.randint(0, len(txt\_corpus[i])-len\_per\_doc-1)

tmp\_sample = list(jieba.cut(txt\_corpus[i][tmp\_start:tmp\_start + len\_per\_doc]))

whole\_samples.append((label\_idxes[i], tmp\_sample))

random.shuffle(whole\_samples)

whole\_samples = whole\_samples[:num\_docs]

train\_data, train\_label = [], []

test\_data, test\_label = [], []

for i in range(int(len(whole\_samples) \* (1 - 0.2))):

train\_data.append(whole\_samples[i][1])

train\_label.append(whole\_samples[i][0])

for i in range(int(len(whole\_samples) \* (1 - 0.2)), len(whole\_samples)):

test\_data.append(whole\_samples[i][1])

test\_label.append(whole\_samples[i][0])

#### train lda

dictionary = corpora.Dictionary(train\_data)

lda\_corpus\_train = [dictionary.doc2bow(tmp\_doc) for tmp\_doc in train\_data]

print("Trainng LDA model...")

lda = models.LdaModel(corpus=lda\_corpus\_train, id2word=dictionary, num\_topics=num\_topics)

#### train svm classifier for correct label

train\_topic\_distribution = lda.get\_document\_topics(lda\_corpus\_train)

train\_features = np.zeros((len(train\_data), num\_topics))

for i in range(len(train\_topic\_distribution)):

tmp\_topic\_distribution = train\_topic\_distribution[i]

for j in range(len(tmp\_topic\_distribution)):

train\_features[i][tmp\_topic\_distribution[j][0]] = tmp\_topic\_distribution[j][1]

print("Training SVM classifier...")

assert len(train\_label) == len(train\_features)

train\_label = np.array(train\_label)

classifier = SVC(kernel='linear', probability=True)

classifier.fit(train\_features, train\_label)

print("Prediction accuracy of training samples is {:.4f}.".format(sum(classifier.predict(train\_features) == train\_label) / len(train\_label)))

#### testing

lda\_corpus\_test = [dictionary.doc2bow(tmp\_doc) for tmp\_doc in test\_data]

test\_topic\_distribution = lda.get\_document\_topics(lda\_corpus\_test)

test\_features = np.zeros((len(test\_data), num\_topics))

for i in range(len(test\_topic\_distribution)):

tmp\_topic\_distribution = test\_topic\_distribution[i]

for j in range(len(tmp\_topic\_distribution)):

test\_features[i][tmp\_topic\_distribution[j][0]] = tmp\_topic\_distribution[j][1]

assert len(test\_label) == len(test\_features)

test\_label = np.array(test\_label)

print("Prediction accuracy of testing samples is {:.4f}.".format(sum(classifier.predict(test\_features) == test\_label) / len(test\_label)))