北京航空航天大學
BEIHANG UNIVERSITY

自然语言处理第三次作业

用金庸小说及 LDA 模型进行文本分类

姓　名：夏秀博

学　号：BY2107030

学　院：机械工程及自动化学院

2022 年 5 月 6 日

金庸小说及 LDA 模型进行文本分类

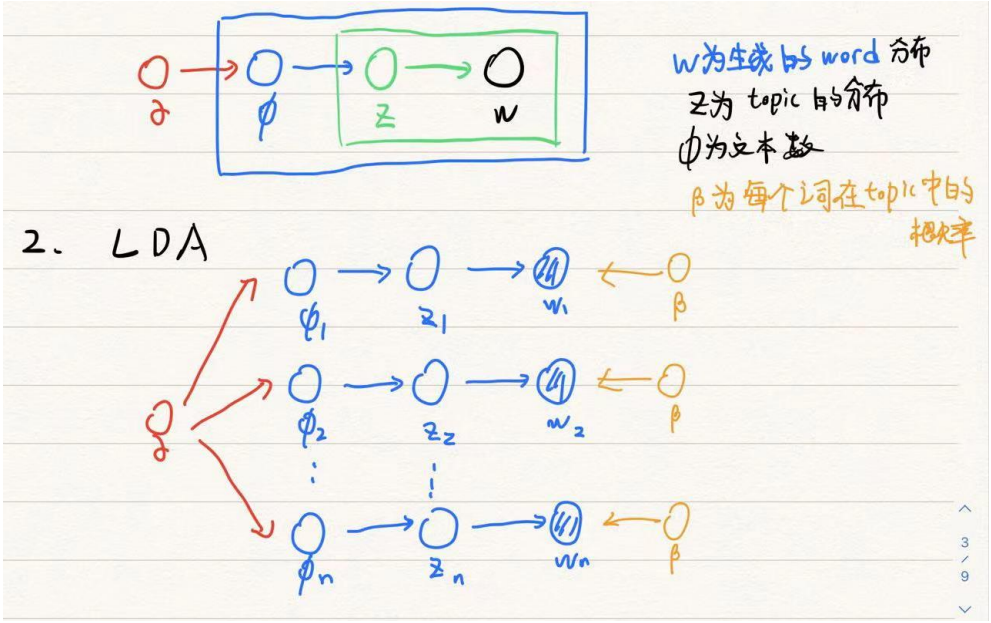
1 问题描述

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。

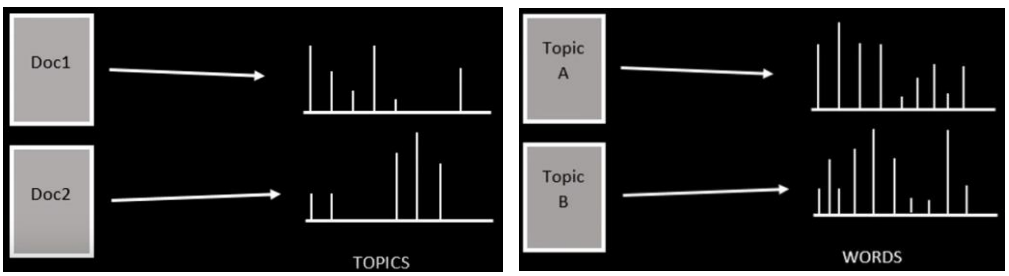
2 EM 算法

2.1 算法简介

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。



2.2 LDA 算法



LDA 就是一堆概率分布推出另一些概率分布

$$P(z,\theta,\varphi|w,\alpha,\beta)=\frac{P(z,\theta,\varphi|\alpha,\beta)}{P(w|\alpha,\beta)}$$

然后通过概率构建：主题中词的分布、文本中主题的部分等来实现对文本特征的抽象，实现分类。

3 实验及结果

首先对文本进行了停词的剔除，然后使用 gensim 进行 lda 主题分类，在 lda 后的文本分类使用了 sklearn 的贝叶斯分类器，重要代码如下：

```
# 利用lda包训练lda模型
dictionary = corpora.Dictionary(train_test_data)
lda_bow_data = [dictionary.doc2bow(x) for x in train_data]
lda = models.LdaModel(corpus=lda_bow_data, id2word=dictionary, num_topics=topic_num)
```

```
# 对测试集进行lda主题分布求解
lda_test_bow_data = [dictionary.doc2bow(x) for x in test_data]
test_topic_distribution = lda.get_document_topics(lda_test_bow_data)
test_topic_matrix = np.zeros((len(test_data), topic_num))
for i in range(len(test_topic_distribution)):
    for j in test_topic_distribution[i]:
        test_topic_matrix[i][j[0]] = j[1]
```

```
# 使用贝叶斯分类器对测试文本进行分类
nav = MultinomialNB()
nav.fit(train_topic_matrix, train_label)
y_predict = nav.predict(test_topic_matrix)
```

测试过程中发现超参数对实验结果影响较大，整体准确性较低，初步怀疑是 lda 模型的问题以及金庸小说中各文章关联性较强。

进行了多次测试，结果如下：

序号	参数			测试准确率/%			平均
	Topic 数	训练段数	每段字数	1	2	3	
1	100	40*16	1000	60.4	58.3	60.4	59.7
2	100	40*16	500	79.2	62.5	64.6	68.8
3	100	20*16	500	52.0	47.9	58.3	52.7
4	50	40*16	500	58.3	54.2	45.8	52.7

5	150	40*16	500	70.8	52.0	64.6	62.5

4　附录

```
import numpy as np
import jieba
from gensim import corpora, models
import os
from sklearn.naive_bayes import MultinomialNB


# 统计所有的标点符号和英文字符
punctuation = "!\"#$%&\'()*+,-./:;<=>?@[\\]^_`{|}~！？｡。＂＂\
            ＂＃＄％＆＇（）＊＋，－／：；＜＝＞＠［＼］＾＿｀｛｜｝‘’"\
            "～｟｠｢｣､、〃《》「」『』【】〔〕（）〖〗〘〙〚〛〜〝〞“”〟〰\n\u3000" \
              "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ" \
              "说道一个自己咱们什么不是他们一声心想心中知道只见还是却是甚么突然
"


def remove_punctuation(something):   # 移除列表中的标点符号
    new_l = []
    for s in something:
        if s not in punctuation:
            new_l.append(s)
    return new_l


def book_read():   # 将所有的 book 读取，使用 jieba 进行分词，再去除其中的标点
    if os.path.exists('books.npy'):
        a = np.load('books.npy', allow_pickle=True)
        books = a.tolist()
    else:
        books_name = open('source/inf.txt').read()
        book_list = books_name.split(",")

        books = []
        i = 0

        for book in book_list:
            i += 1
            f = open('source/'+book+'.txt', encoding="UTF-8")
            txt = f.read()
            # seg_list = jieba.lcut(txt, cut_all=False)
            seg_list = [w for w in jieba.cut(txt) if len(w) > 1]
            seg = remove_punctuation(seg_list)
            books.append(seg)
            print('读取进度{}/16'.format(i))
```

```
        m = np.array(books, dtype=object)
        np.save('books.npy', m)
    return books


if __name__ == '__main__':
    # 超参数
    para_lenth = 500
    para_num_each = 40
    para_num_for_test = 3
    topic_num = 150

    books = book_read()  # 读取所有书

    book_num = len(books)
    # 预处理数据
    train_test_data = []
    train_test_label = []
    for i in range(len(books)):
        for j in range(para_num_each):
            start = np.random.randint(0, len(books[i])-para_lenth-100)
            train_test_data.append(books[i][start:start+para_lenth])
            train_test_label.append(i)

    for i in range(len(books)):
        for j in range(para_num_for_test):
            start = np.random.randint(0, len(books[i])-para_lenth-100)
            train_test_data.append(books[i][start:start+para_lenth])
            train_test_label.append(i)

    train_data = train_test_data[:para_num_each * book_num]
    train_label = train_test_label[:para_num_each * book_num]
    test_data = train_test_data[para_num_each * book_num:]
    test_label = train_test_label[para_num_each * book_num:]

    # 利用 lda 包训练 lda 模型
    dictionary = corpora.Dictionary(train_test_data)
    lda_bow_data = [dictionary.doc2bow(x) for x in train_data]
    lda = models.LdaModel(corpus=lda_bow_data, id2word=dictionary, num_topics=topic_num)

    train_topic_distribution = lda.get_document_topics(lda_bow_data)
    train_topic_matrix = np.zeros((len(train_data), topic_num))
    for i in range(len(train_topic_distribution)):
        for j in train_topic_distribution[i]:
            train_topic_matrix[i][j[0]] = j[1]
    print(train_topic_matrix)

    print('模型训练结果：')
    print(lda.print_topics(num_topics=topic_num, num_words=10))
```

```
lda.save('lda.model')

# 对测试集进行 lda 主题分布求解

lda_test_bow_data = [dictionary.doc2bow(x) for x in test_data]
test_topic_distribution = lda.get_document_topics(lda_test_bow_data)
test_topic_matrix = np.zeros((len(test_data), topic_num))
for i in range(len(test_topic_distribution)):
    for j in test_topic_distribution[i]:
        test_topic_matrix[i][j[0]] = j[1]

nav = MultinomialNB()
nav.fit(train_topic_matrix, train_label)
y_predict = nav.predict(test_topic_matrix)

perfect = 0
for i in range(len(y_predict)):
    if y_predict[i] == test_label[i]:
        perfect += 1
print("预测的结果为: ", y_predict)
print("实际的结果为: ", test_label)
print("试验的准确率为:  ", perfect/len(y_predict))
```