深度学习与自然语言处理
第三次大作业

LDA 主题分布

院（系）名称	自动化科学与电气工程学院
学 生 学 号	ZY2103803
学 生 姓 名	李鑫磊

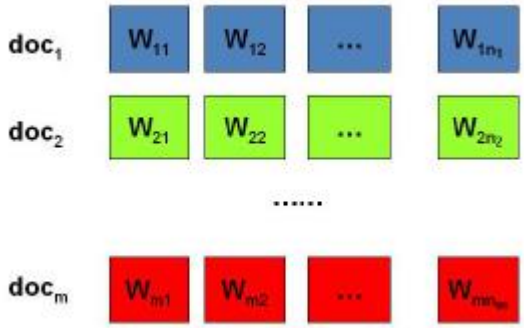
2022 年 04 月

一、问题描述

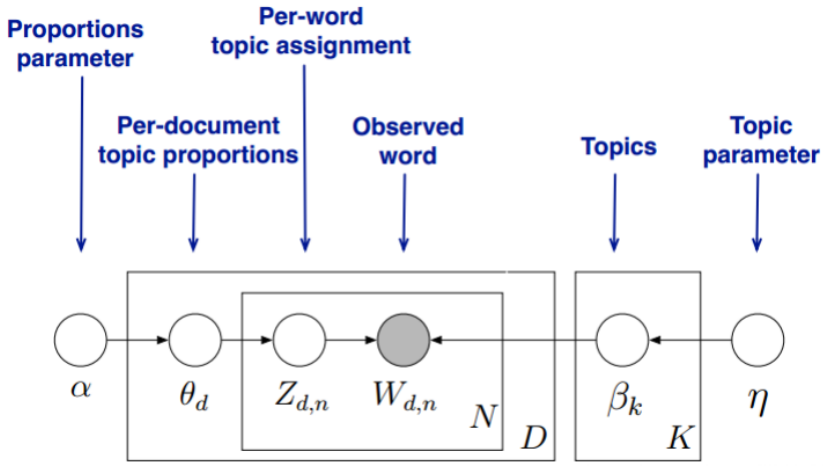
从给定的语料库中均匀抽取 200 个段落（每个段落大于 500 个词）， 每个段落的标签就是对应段落所属的小说。利用 LDA 模型对于文本建模，并把每个段落表示为主题分布后进行分类。验证与分析分类结果。

二、LDA 模型

假设有 M 篇文档，对应第 d 个文档中有 N_d 个词。



目标是找到每一篇文档的主题分布和每一个主题中词的分布。在 LDA 模型中，我们需要先假定一个主题数目 K，这样所有的分布就都基于 K 个主题展开。



LDA 假设文档主题的先验分布是 Dirichlet 分布，即对于任一文档 d，其主题分布 θ_d 为：

$$\theta_d = Dirichlet(\vec{\alpha})$$

其中，α 为分布的超参数，是一个 K 维向量。

LDA 假设主题中词的先验分布是 Dirichlet 分布，即对于任意主题 k，其词分布 β_k 为：

$$\beta_k = Dirichlet(\vec{\eta})$$

其中，η 为分布的超参数，是一个 V 维向量。V 代表词汇表里所有词的个数。

对于数据中任一一篇文档 d 中的 n 个词，我们可以从主题分布 θ_d 中得到它的主题编号 z_{dn} 的分布为：

$$z_{dn} = multi(\theta_d)$$

而对于该主题编号，得到我们看到的词 w_{dn} 的概率分布为：

$$w_{dn} = multi(\beta_{z_{dn}})$$

这个模型里，我们有 M 个文档主题的 Dirichlet 分布，而对应的数据有 M 个主题编号的多项分布，这样($\alpha \to \theta_d \to \vec{z}_d$)就组成看 Dirichlet-multi 共轭，可以使用贝叶斯推断的方法得到基于 Dirichlet 分布的文档主题后验分布。

如果第 d 个文档中，第 k 个主题的词的个数为：$n_d^{(k)}$，则对应的多项分布的计数可以表示为：

$$\vec{n}_d = (n_d^{(1)}, n_d^{(2)}, ...n_d^{(K)})$$

利用 Dirichlet-multi 共轭，得到 θ_d 的后验分布为：

$$Dirichlet(\theta_d \mid \vec{\alpha} + \vec{n}_d)$$

对于主题与词的分布，我们有 K 个主题与词的 Dirichlet 分布，而对应的数据有 K 个主题编号的多项式分布，这样($\eta \to \beta_k \to \vec{w}_{(k)}$)就组成看 Dirichlet-multi 共轭，可以使用贝叶斯推断的方法得到基于 Dirichlet 分布的文档主题后验分布。

如果在第 k 个主题中，第 v 个词的个数为：$n_k^{(v)}$，则对应的多项分布的计数可以表

示为：

$$\vec{n}_k = (n_k^{(1)}, n_k^{(2)}, ...n_k^{(V)})$$

利用 Dirichlet-multi 共轭，得到 β_k 的后验分布为：

$$Dirichlet(\beta_k \mid \vec{\eta} + \vec{n}_k)$$

由于主题产生词不依赖具体某一个文档，因此文档主题分布和主题词分布是独立的。

三、程序实现

3.1 生成段落数据库

在 16 篇小说中选取 7 篇小说在 DatabaseChinese 中，随机选取 3 篇，每篇随机抽取不小于 500 字的段落 150 段，存在 DataExcel 中。

```
def txt_convert_2_excel(file_path, data_path, K=3):
    """
    :param file_path: 小说集存储的路径
    :param data_path: excel路径
    :param K: 随机选取的小说篇数
    :return: 将txt变成excel数据，返回excel路径
    """
    logging.info('Converting txt to excel...')
    files = []
    for x in os.listdir(file_path):
        files.append(x)
    selected_files = random.sample(files, k=3)

    txt = []
    txtname = []
    n = 150  # 每篇选取150段

    for file in selected_files:
        filename = os.path.join(file_path, file)
        with open(filename, 'r', encoding='ANSI') as f:
            full_txt = f.readlines()
            lenth_lines = len(full_txt)
            i = 200
            for j in range(n):
                txt_j = ''
                while(len(txt_j) < 500):
                    txt_j += full_txt[i]
                    i += 1
                txt.append(txt_j)
                txtname.append(file.split('.')[0])
                i += int(lenth_lines / (3 * n))

    dic = {'Content': txt, 'Txtname': txtname}
    df = pd.DataFrame(dic)
```

3.2 分词

创建 Paragraph 类，断句分词。

```
class Paragraph:
    def __init__(self, txtname='', content='', sentences=[], words=''):
        self.fromtxt = txtname
        self.content = content
        self.sentences = sentences
        self.words = words
        global punctuation
        self.punctuation = punctuation
        global stopwords
        self.stopwords = stopwords

    def sepSentences(self):
        line = ''
        sentences = []
        for w in self.content:
            if w in self.punctuation and line != '\n':
                if line.strip() != '':
                    sentences.append(line.strip())
                    line = ''
            elif w not in self.punctuation:
                line += w
        self.sentences = sentences

    def sepWords(self):
        words = []
        dete_stopwords = 1
        if dete_stopwords:
            for i in range(len(self.sentences)):
                words.extend([x for x in jieba.cut(
                    self.sentences[i]) if x not in self.stopwords])
        else:
            for i in range(len(self.sentences)):
                words.extend([x for x in jieba.cut(self.sentences[i])])
        reswords = ' '.join(words)
        self.words = reswords
```

3.3 LDA 模型求解

3 个主题，迭代 1000 次

```
# LDA
logging.info('Training LDA model...')
cntVector = CountVectorizer(max_features=40)  # 特征词数设置为40个
cntTf = cntVector.fit_transform(corpus)
lda = LatentDirichletAllocation(
    n_components=3, learning_offset=50., max_iter=1000, random_state=0)  # 主题3个，迭代1000次
docres = lda.fit_transform(cntTf)
```

3.4 SVM 分类

参数：确定训练集个数。

```
# SVM
logging.info('SVM classify...')
X = docres
y = [data_list[i].fromtxt for i in range(len(data_list))]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)   # 训练集150*0.2=30个段落
svm_model = LinearSVC()  # model = SVC()
svm_model.fit(X_train, y_train)
y_pred = svm_model.predict(X_test)
```

四、运行结果

随机抽取到《鹿鼎记》《神雕侠侣》《书剑恩仇录》三本小说，段落主题分类结果部分如下图所示，左侧为真实值，右侧为预测值。整体分类的准确率为 91.1%，分类效果较为准确。

```
Topic real:       Topic predict:

鹿鼎记      鹿鼎记

鹿鼎记      鹿鼎记

鹿鼎记      鹿鼎记

鹿鼎记      鹿鼎记

鹿鼎记      鹿鼎记

书剑恩仇录        书剑恩仇录

书剑恩仇录        书剑恩仇录
```