基于 LDA 的段落分类

ZY2203402 冯士轩

1、作业内容

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：在不同数量的主题个数下分类性能的变化和以"词"和以"字"为基本单元下分类结果的差异。

2、原理介绍

LDA (Latent Dirichlet Allocation)模型是 Blei 等人基于 PLSI 模型提出的，LDA 是一个三层贝叶斯概率模型，包含词、主题和文档三层结构。在文本建模中每个文本都被建模为潜在主题集合的有限混合分布。每个主题又被建模为单词集的概率分布。文本就是用这种主题的概率分布来表达。LDA 的核心思想是寻找到最佳的投影方法，将高维的样本投影到特征空间(feature space)，使得不同类别间的数据“距离”最大，而同一类别内的数据“距离”最小。

LDA 作为完备的生成模型，对于文档-语义，语义-单词两个层次都进行了建模，并以概率模型充分描述了文本、语义、单词三层的生成。同时 LDA 在文档-语义一层以 Dirichlet 分布来描述主题的生成,使得要估计的模型参数数量不会随着文档集的增多而增多。因此目前 LDA 在潜在语义分析中得到越来越广泛的使用。作为一种产生式文档模型,用 LDA 提取文档的隐含语义结构和文档表征已经成功的应用到很多文本相关的领域。

具体地说，LDA 的建模过程如下：

（1）首先定义文章集合为 Doc，文章主题集合为 Topic，Doc 中的每个文档可以看作为一个单词序列<w_1 , w_2 , . . . , w_n >。LDA 模型以文档集合作为输入，最终训练出两个结果向量：文本的 Topic 和文本单词量。

每个文本中对应到不同主题的概率θ=<p_1，…，p_i >，其中 p 表示文本对应 Topic 中第 i 个主题的概率。

$$P_i = \frac{n_i}{n}$$

n_i 表示文本中对应的第 i 个主题的词的数目，n 表示文本中所有词的总数。

每个主题生成不同单词的概率φ=<q_1，…，q_i >。

$$q_i = \frac{N_i}{N}$$

N_i表示对应到 Topic 的文本中的第 i 个单词的数目，N 表示所有对应到 Topic 的单词总数。

（2）随机生成每个单词的主题标签。

我们利用当前θ，φ的值，我们可以为一个文档中的单词计算它对应任意一个主题时的概率，然后根据这些结果来更新这个词对应的 Topic。循环迭代每个单词，对其主题进行更新，并计算文档中各个主题的分布；

3、实验过程

（1）数据预处理。将文本中广告、标点符号进行删除，防止其影响文本分类。

```
def content_deal(content):  # 语料预处理，进行断句，去除一些广告和无意义内容
    ad = ['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', '----〖新语丝电子文库(www.xys.org)〗', '新语丝电子文
          '\u3000', '\n', '。', '？', '！', '，', '、', '；', '：', '﹃', '﹄', '〈', '〉', '“', '”', '‘', '’', '［', '］', '····', '……',
          '『', '』', '（', '）', '—', '［', '］', '\ue41b', '<', '>', '+', '\x1a', '\ue42b']
    for a in ad:
        content = content.replace(a, '')
    return content
```

（2）对文件夹中的 16 个文档进行读取，将每个文档均匀分成 13 份，选用每份的前 5000 个字符，生成题目要求均匀抽取 208 个段落和标签。每个段落的标签就是对应段落所属的小说。

```
def read_files(path):
    texts = []
    labels = []
    for filename in os.listdir(path):
        with open(os.path.join(path, filename), 'r', encoding='GB18030') as f:
            content = f.read()
            content = content_deal(content)
            total_len = len(content)
            size = int(total_len // 13)
            for i in range(13):
                start = i * size
                end = (i + 1) * size
                selected_text = content[start:end][:5000]
                if selected_text:
                    texts.append(content)
                    labels.append(filename)
    return texts, labels
```

（3）为主题建模准备数据。读取文件返回的标签和文本，对它们进行分字或分词的方法处理文本，然后使用词袋模型表示它们。使用库里自带的 Latent Dirichlet Allocation (LDA) 模型训练指定数量的主题，学习方法设置为批量学习方法，迭代次数设置为 20，并返回每个文档的主题分布以及相应的标签。

```
def prepare_data(path, n_topics, unit):
    texts, labels = read_files(path)
    tokenized_texts = tokenize(texts, unit)
    #文档的词袋向量表示
    count_vectorizer = CountVectorizer(tokenizer=lambda x: x, lowercase=False)
    X = count_vectorizer.fit_transform(tokenized_texts)
    #训练模型
    lda = LatentDirichletAllocation(n_components=n_topics, learning_method='batch', max_iter=20, random_state=0)
    lda.fit(X)
    # Transform texts to topic distributions
    X_topics = lda.transform(X)
    return X_topics, labels
```

（4）使用评估逻辑回归分类器判断主题分布和相应标签上的准确率。首先，数据分割成训练集和测试集，其中 10%的数据用于测试，并使用分层抽样来确保训练集和测试集中类别分布平衡。然后，它在训练集上训练逻辑回归分类器，并计算在测试集上的准确率得分来评估其准确率。

```
def evaluate_classification(X, y):
    # Split data into training and testing sets
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1, stratify=y, random_state=0)
    # Train and test a logistic regression classifier
    from sklearn.linear_model import LogisticRegression
    clf = LogisticRegression(random_state=0)
    clf.fit(X_train, y_train)
    y_pred = clf.predict(X_test)
    acc = accuracy_score(y_test, y_pred)
    return acc
```

4、结果分析

将主题数设置为 10、100、200，分别测试其在以字和以词的形式处理文本的准确率。

```
n_topics=10, unit=word, accuracy=0.143
n_topics=10, unit=character, accuracy=0.333
n_topics=100, unit=word, accuracy=0.286
n_topics=100, unit=character, accuracy=0.524
n_topics=200, unit=word, accuracy=0.286
n_topics=200, unit=character, accuracy=0.810
```

从结果可以看出当设置的主题数增加时，不论是以字还是以词的形式，段落所对应的标签准确率均有一定程度的提升。判断原因可能是在训练 LDA 模型时，增加主题数量会使得模型更加细致地刻画文本语义。

相同主题数，可以看出以字为分词对象时，准确率高于以词为分类对象。分析原因可能是中文的语义和结构都与单个字有关系。