DL-NLP 第三次作业

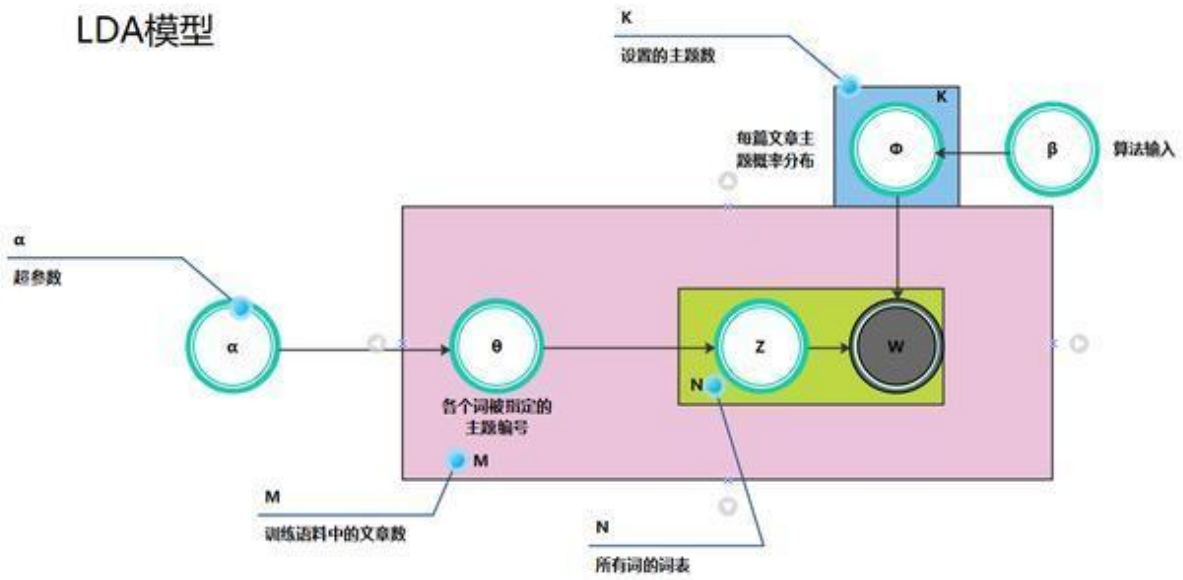
一、问题描述

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、背景介绍

2.1 LDA模型

LDA是自然语言处理中非常常用的一个主题模型，全称是隐含狄利克雷分布（Latent Dirichlet Allocation），简称LDA。LDA 在主题模型中占有非常重要的地位，常用来文本分类。LDA由Blei, David M.、Ng, Andrew Y.、Jordan于2003年提出，用来推测文档的主题分布。它可以将文档集中每篇文档的主题以概率分布的形式给出，从而通过分析一些文档抽取出它们的主题分布后，便可以根据主题分布进行主题聚类或文本分类。LDA 是一种矩阵分解技术，在向量空间中，任何语料（文档的集合）可以表示为文档（Document - Term，DT）矩阵。



主题模型是对文本中隐含主题的一种建模方法，每个主题其实是词表上单词的概率分布；主题模型是一种生成模型，一篇文章中每个词都是通过“以一定概率选择某个主题，并从这个主题中以一定概率选择某个词语”这样一个过程得到的。LDA为话题模型的典型代表，其在文本挖掘领域，如文本主题识别、文本分类以及文本相似度计算方面有者广泛的应用。这里我们定义语料 D 由 M 篇文档组成，$D=\{W_1,W_2,\dots,W_M\}$ ，其中一篇文档 W 又包含 N 个词， $W=\{w_1,w_2,\dots,w_N\}$ ，则从语料 D 中生成文档 W 的过程可表示为：

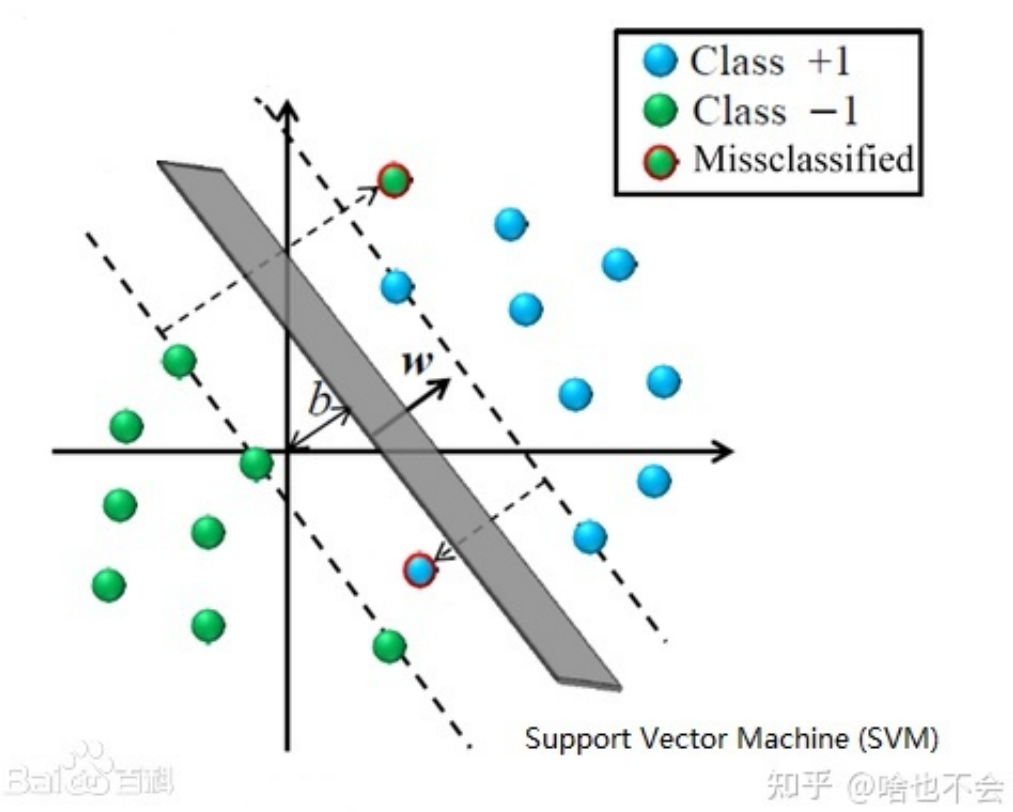
- Step1. 从狄利克雷分布中随机采样一个话题分布 θ ， $\theta\sim Dir(\alpha)$ ；
- Step2. 对文档 W 中的第 n 个词 w_n 进行话题指派，且 $z_n\sim Multionmial(\theta)$ ；
- Step3. 根据指派话题 z_n 所对应的词频分布 β 按 $P(w_n|z_n,\beta)$ 的概率分布随机采样生成词（ $\beta\in\mathbb{R}^{k\times V}$ ， k 为话题数目， V 为词表大小，且 $\beta_{ij}=p(w^j|z^i=1)$ ）。

2.2 SVM

支持向量机（Support Vector Machine, SVM）是一类按监督学习（supervised learning）方式对数据进行二元分类的广义线性分类器（generalized linear classifier），其决策边界是对学习样本求解的最大边距超平面（maximum-margin hyperplane）。

SVM使用铰链损失函数（hinge loss）计算经验风险（empirical risk）并在求解系统中加入了正则化项以优化结构风险（structural risk），是一个具有稀疏性和稳健性的分类器 。SVM可以通过核方法（kernel method）进行非线性分类，是常见的核学习（kernel learning）方法之一 。

SVM所要解决的问题普遍意义上的SVM是一个二分类线性分类器，它的学习策略是："在分类超平面的正负两边各找到一个离分类超平面最近的点（也就是支持向量），使得这两个点距离分类超平面的距离和最大"。



EM算法流程：

Step 1：选择EM算法的初值 （初值对结果由影响，EM算法只能达到局部最优）

Step 2：E步：就是求Q函数，

Step 3：M步：使参数 [公式] 的似然函数增大，达到局部最大值；

三、实验过程

3.1 数据介绍

- txt_data 文件夹下存放了：16本金庸武侠小说
- baidu_stopwords：baidu提供的禁用词（stopwords）
- dataset.json：进行随机选取语料段落后的分词数据
- preprocessor.py：数据读取及预处理程序

- main.py：模型训练及预测分类主程序

3.2 数据预处理

读入小说文件的过程中，要将非中文字符处理掉，只考虑中文字符。为了便于后续训练过程，每一份语料所对应的标签为该小说名，将小说名索引化，id为0-15。

```
def get_texts():
    import_stopwords()
    corpus_context_dict = {}
    id_corpus_dict = {}
    id = 0
    for file in get_files():
        simple_name = str(file).split(os.sep)[1].split('.')[0]
        with open(file, 'rb') as f:
            context = f.read()
            real_encode = chardet.detect(context)['encoding']
            context = context.decode(real_encode, errors='ignore')
            new_context = ''
            for c in context:
                if is_chinese(c):
                    new_context += c
            # for sw in stopwords_list:
            #     new_context = new_context.replace(sw, '')
            corpus_context_dict[simple_name] = new_context
            id_corpus_dict[id] = simple_name
            id += 1
        print(id)
    return corpus_context_dict, id_corpus_dict
```

之后需要对其进行禁用词过滤。首先要均匀的从16个小说中进行语料采样，采集250个段落，每种语料的词数约为500（本实验初始值设置为600，考虑到禁用词过滤会过滤掉一部分），采用jieba库对每一种语料进行分词，分词之后采用baidu_stopwords进行禁用词过滤。最后将语料与索引化的标签合并成元组，进行下一步训练操作。

```
def get_dataset():
    data = []
    for i in range(para_num):
        context = corpus_context_dict[id_corpus_dict[i % 16]]
        rand_value = randint(0, len(context) - per_para_words)
        new_context = context[rand_value:rand_value + per_para_words]
        cut_list = list(jieba.cut(new_context, cut_all=False))
        filter_list = []
        for c in cut_list:
            if stopwords_list.__contains__(c):
                continue
            filter_list.append(c)
        data.append((i % 16, filter_list))
    return data
```

3.3 LDA模型训练

采用gensim中的corpora和其中的lda模型来进行训练。corpora能够构建词频矩阵。corpus是把每本小说ID化后的结果，每个元素是新闻中的每个词语，在字典中的ID和频率。

```
print("Trainng Dictionary model...")
    dictionary = corpora.Dictionary(train_data)
    lda_corpus_train = [dictionary.doc2bow(tmp_doc) for tmp_doc in train_data]
    # print(dictionary)
    # print(lda_corpus_train)
    print("Trainng LDA model...")
    lda = gensim.models.LdaModel(corpus=lda_corpus_train, id2word=dictionary,
num_topics=topic_num)
```

为了更好的表示语料的特征，所以将num_topic的设置为200。其中训练的top topic如下所示：

```
[(25, '0.011*"文书" + 0.010*"军官" + 0.010*"韦小宝" + 0.008*"小玄子" + 0.007*"矮
子" + 0.006*"李沅芷" + 0.006*"张老爷" + 0.005*"请" + 0.005*"马敬侠" + 0.004*"总
爷"'),
 (180, '0.015*"马春花" + 0.010*"胡斐" + 0.009*"孩子" + 0.008*"两个" + 0.007*"喜欢"
+ 0.006*"说道" + 0.005*"记得" + 0.005*"丈夫" + 0.004*"弟子" + 0.004*"陈家洛"'),
 (168, '0.012*"石破天" + 0.011*"吃" + 0.008*"花万紫" + 0.007*"决不" + 0.006*"爱" +
0.006*"花姑娘" + 0.005*"陈香主" + 0.004*"监牢" + 0.004*"杀" + 0.004*"陈冲之"'),
 (75, '0.008*"小蛇" + 0.006*"见" + 0.004*"大" + 0.004*"石破天" + 0.004*"剑士" +
0.003*"只见" + 0.003*"关东" + 0.003*"不知" + 0.003*"一声" + 0.003*"举人"'),
 (185, '0.006*"李文秀" + 0.006*"大" + 0.005*"苏普" + 0.005*"狼" + 0.004*"手帕" +
0.004*"老人" + 0.004*"汉人" + 0.003*"食指" + 0.003*"张无忌" + 0.003*"强盗"'),
 (100, '0.006*"见" + 0.005*"卓天雄" + 0.005*"范蠡" + 0.005*"萧中慧" + 0.005*"袁冠
南" + 0.005*"说道" + 0.004*"姓" + 0.004*"袁承志" + 0.004*"一人" + 0.004*"著"'),
 (22, '0.006*"见" + 0.006*"袁承志" + 0.006*"钢杖" + 0.005*"杖" + 0.005*"张无忌" +
0.004*"木剑" + 0.004*"喝" + 0.004*"杨不悔" + 0.004*"草菌" + 0.003*"简捷"'),
 (159, '0.007*"尼摩星" + 0.005*"大" + 0.005*"长剑" + 0.005*"英雄" + 0.005*"耶律齐"
+ 0.005*"徐天宏" + 0.005*"李莫愁" + 0.004*"二" + 0.004*"脸上" + 0.004*"姊"'),
 (56, '0.007*"见" + 0.007*"苗若兰" + 0.007*"爹" + 0.006*"花铁干" + 0.006*"狄云" +
0.006*"妈" + 0.005*"胡斐" + 0.005*"说道" + 0.005*"我妈" + 0.005*"妈妈"'),
 (128, '0.008*"骈" + 0.007*"文书" + 0.005*"曰" + 0.005*"军官" + 0.004*"说道" +
0.003*"见" + 0.003*"聂隐娘" + 0.003*"皮盒" + 0.003*"请" + 0.003*"黄药师"'),
 (103, '0.009*"康熙" + 0.008*"建宁公主" + 0.007*"哥哥" + 0.007*"皇帝" + 0.006*"公
主" + 0.006*"云南" + 0.005*"陪" + 0.005*"佛经" + 0.004*"小桂子" + 0.004*"姑娘"'),
 (62, '0.005*"谢逊" + 0.005*"只见" + 0.004*"瞧" + 0.004*"走" + 0.004*"众人" +
0.004*"比武" + 0.004*"心中" + 0.004*"说道" + 0.004*"武功" + 0.004*"逃走"'),
 (74, '0.020*"乌老大" + 0.006*"众人" + 0.005*"一声" + 0.005*"女童" + 0.005*"手下"
+ 0.005*"却是" + 0.005*"说道" + 0.005*"胡斐" + 0.004*"女娃娃" + 0.004*"羞"'),
 (105, '0.007*"草菌" + 0.007*"简捷" + 0.007*"张无忌" + 0.006*"杨不悔" + 0.006*"喝"
+ 0.005*"吃" + 0.005*"薛公远" + 0.005*"热汤" + 0.004*"见" + 0.003*"张"'), (158,
'0.008*"说道" + 0.008*"袁承志" + 0.007*"心想" + 0.004*"青青" + 0.004*"公孙绿萼" +
0.004*"弟子" + 0.004*"金子" + 0.004*"走" + 0.004*"温方达" + 0.004*"高宗"'),
```

```
 (106, '0.011*"范蠡" + 0.010*"文种" + 0.008*"周威信" + 0.007*"勾践" + 0.006*"吴国"
+ 0.005*"大夫" + 0.005*"两人" + 0.004*"吴王" + 0.003*"说道" + 0.003*"见"')]
```

3.4 SVM分类器训练

将上一步所采集到的语料数据以9:1的训练/测试比进行实验，即225份用于训练，25份用于测试。为了便于调试，可以讲语料以json格式存储在硬盘中，避免每次运行程序都需要进行数据预处理。内核采用linear模式，并且允许分类器进行概率估计。

```
ratio = 0.9
    train_data, train_label, test_data, test_label = [], [], [], []
    if GEN_DATA:
        corpus_context_dict, id_corpus_dict = get_texts()
        dataset = get_dataset()
        # print(dataset)
        for i in range(int(len(dataset) * ratio)):
            train_data.append(dataset[i][1])
            train_label.append(dataset[i][0])
        for i in range(int(len(dataset) * ratio), len(dataset)):
            test_data.append(dataset[i][1])
            test_label.append(dataset[i][0])
        # print(train_data)

        with open('dataset.json', 'w', encoding='utf-8') as f:
            json.dump(dataset, fp=f, indent=4)
```

```
    lda_corpus_test = [dictionary.doc2bow(tmp_doc) for tmp_doc in test_data]
    topics = lda.get_document_topics(lda_corpus_test)
    for i in range(len(test_data)):
        print(topics[i])

    train_topic_results = lda.get_document_topics(lda_corpus_train)
    train_features = np.zeros((len(train_data), topic_num))
    for i in range(len(lda_corpus_train)):
        for topic_no, freq in train_topic_results[i]:
            train_features[i][topic_no] = freq
    print("训练SVM分类器")
    train_label = np.array(train_label)
    classifier = SVC(kernel='linear', probability=True)
    classifier.fit(train_features, train_label)
    print("训练集的精确度为:
{:.4f}.".format(sum(classifier.predict(train_features) == train_label) /
len(train_label)))
```

四、实验结果

- 对测试集进行实验代码

```
    lda_corpus_test = [dictionary.doc2bow(tmp_doc) for tmp_doc in
test_data]
    test_topic_results = lda.get_document_topics(lda_corpus_test)
    test_features = np.zeros((len(test_data), topic_num))
    for i in range(len(test_topic_results)):
        for topic_no, freq in test_topic_results[i]:
            test_features[i][topic_no] = freq
    test_label = np.array(test_label)
    print("测试集的精确度为
{:.4f}.".format(sum(classifier.predict(test_features) == test_label) /
len(test_label)))
```

-

参数 topic_num	10	20	50	100	200
训练集准确率	0.1956	0.2311	0.3733	0.4978	0.6889
测试集准确率	0.0800	0.1200	0.3200	0.4400	0.5200

- 部分实验结果截图（topic_num=100）

```
main (1)
 0.07374104), (42, 0.022527844), (43, 0.08857638), (51, 0.09995367), (55, 0.037659723), (60,
 0.046557706), (66, 0.02647018), (74, 0.037923597), (77, 0.030065777), (86, 0.015079992), (87,
 0.07756557), (94, 0.052415214), (95, 0.0593917), (99, 0.014807026)]
[(1, 0.0776313), (2, 0.015801148), (4, 0.0713386), (10, 0.051038057), (14, 0.01857827), (23,
 0.01594266), (24, 0.09735543), (25, 0.08256026), (32, 0.10688525), (35, 0.014200126), (39, 0
 .019034868), (50, 0.05483032), (51, 0.039310057), (52, 0.05425213), (54, 0.04015989), (73, 0
 .017276077), (74, 0.02432717), (83, 0.015018062), (87, 0.07401173), (96, 0.033044096), (97,
 0.020525742), (99, 0.03891134)]
训练SVM分类器
训练集的精确度为:  0.4978.
测试集的精确度为 0.4400.

进程已结束，退出代码 0
```

- 根据实验结论可以得出，topic_num的值越大，准确率越高。这是因为能够表征语料段落的特征更多了，于是在分类环节会有较好的表现。
- stopword的过滤是很有必要的，否则在分词中会有大量的无意义的助词、代词出现，如：你、我、他、在、的、了、啦等等。因此会导致训练出的LDA模型有很大不确定性。
- 按照标准概率模型而言，能够随机分类正确一个语料段落的准确率是 1/16，在topic_num变大时，准确率越来越高，这和LDA的关键词中包含了人名和专有名词有关，提升了分类的准确度。