深度学习与自然语言处理第三次作业

SY2106318 孙旭东

代码链接

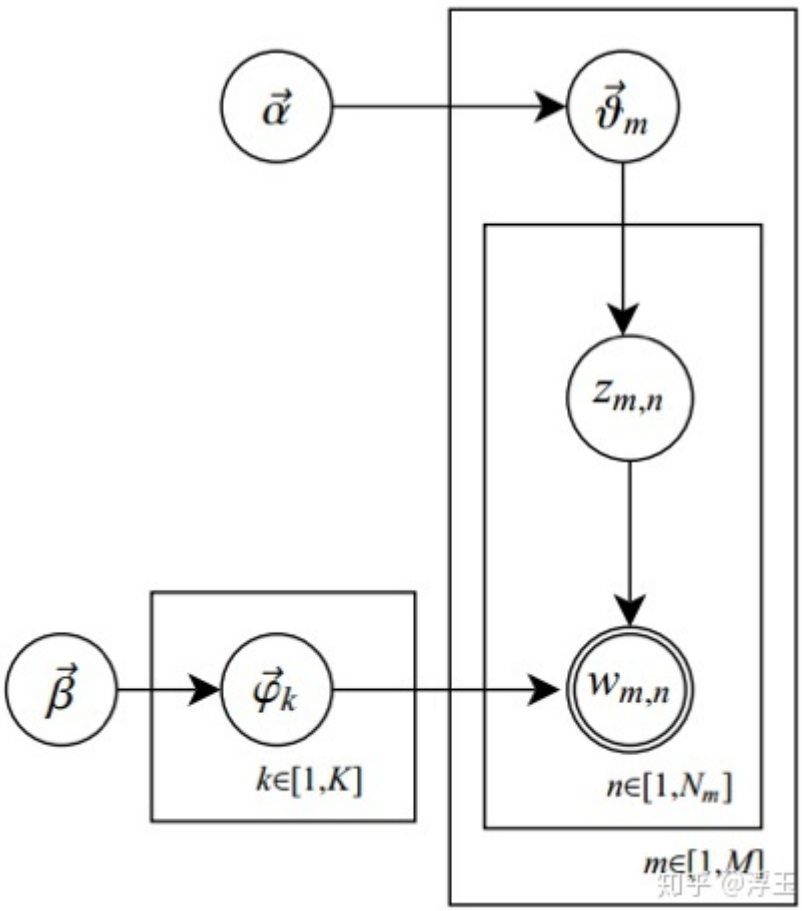
1.作业内容

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。验证与分析分类结果。

2.相关知识

2.1 LDA模型

LDA（Latent Dirichlet Allocation），是一款基于Dirichlet分布的概率主题模型，运用了概率论和词袋模型等知识。从作业内容分析推导这一模型，数据集为金庸的小说，我们从这些小说中均匀抽取M个段落，即得到M篇文档，对应第m个文档中有n_m个词，假定每篇文档中的词都是从一系列主题中选择出来的，每个主题下有对应的词，文档中一个词w_{mn}的产生要经历两个步骤，首先从该文档的主题分布中采样一个主题，然后在这个主题对应的词分布中采样一个词。不断重复完成这个随机生成过程，直到这m篇文档全部完成上述过程。



随机生成过程的两个步骤都符合Dirichlet分布：

- 首先，抽主题：α是某文档主题的Dirichlet分布的超参数，采样得到文档m的主题分布θ_m：$(p_1, p_2, \ldots, p_k)$，k是主题个数，采样出某一主题，得到文档m第n个词的主题编号Z_{mn}；
- 然后，抽主题下的词：β是某主题的词语的Dirichlet分布的超参数，采样得到所有主题词语的分布ϕ，得到主题1$(p_{11}, p_{12}, \ldots, p_{1v})$，...，主题k$(p_{k1}, p_{k2}, \ldots, p_{kv})$，从第一步主题编号$Z_{mn}$对应的主题分布中抽样得到词语$w_{mn}$。

这一过程中，w_{mn}是可以观察到的已知变量，α, β是跟据经验给定的先验参数，其他变量Z_{mn}, θ, ϕ都是未知变量，需要根据观察到的变量学习，这里使用**Gibbs Sampling**算法，其运行方式为，每次抽取概率向量的一个维度，未定其他维度的变量，采样当前维度的值，不断迭代，直到收敛。

初始时，随机给文本中的每个词分配主题$z^{(0)}$，然后统计每个主题z下词出现的概率，及每个文档m下出现主题z的数量，以及每个主题下词的总量，据此估计排除当前词的主题分布，然后根据这个分布为该词采样新主题，最终每个文档的主题分布θ_m，每个主题的词分布ϕ_i收敛，算法停止。

2.2 支持向量机(SVM)

支持向量机（support vector machines, SVM）是一种二分类模型，它的基本模型是定义在特征空间上的**间隔最大的线性分类器**，间隔最大使它有别于感知机；SVM还包括**核技巧**，这使它成为实质上的非线性分类器。SVM的的学习策略就是间隔最大化，可形式化为一个求解凸二次规划的问题，也等价于正则化的合页损失函数的最小化问题。SVM的的学习算法就是求解凸二次规划的最优化算法。

使用**one-against-all**方式可以实现多分类，及对于每一类，将其作为+1类，其余M-1个类作为-1类，构造binary SVM。本实验中即使用了这一方法对段落进行分类。

3.实验过程

3.1数据准备

使用到的数据为金庸的16本武侠小说，对数据进行预处理的代码如下：

```
def get_single_corpus(file_path):
    """
    获取file_path文件对应的内容
    :return: file_path文件处理结果
    """
    corpus = ''
    # unuseful items filter
    r1 = u'[a-zA-Z0-9’!"#$%&\'()*+,-./:: ;<=>?@, 。?★、…【】《》？“”‘’！[\\]^_`{|}~「」『』
 () ]+'
    with open('../stopwords.txt', 'r', encoding='utf8') as f:
        stop_words = [word.strip('\n') for word in f.readlines()]
        f.close()
    # print(stop_words)
    with open(file_path, 'r', encoding='ANSI') as f:
        corpus = f.read()
        corpus = re.sub(r1, '', corpus)
        corpus = corpus.replace('\n', '')
        corpus = corpus.replace('\u3000', '')
        corpus = corpus.replace('本书来自免费小说下载站更多更新免费电子书请关注', '')
        f.close()
    words = list(jieba.cut(corpus))
    return [word for word in words if word not in stop_words]
```

在以utf8编码格式读取文件内容后，删除文章内的所有非中文字符，以及和小说内容无关的片段，得到字符串形式的语料库，然后使用jieba分词进行分词，并使用**百度停用词表**进行停用词的过滤，最终返回小说的分词列表。

本实验中，随机在小说列表中选取200个长度为1000词的段落，为了最后得到主题词能更容易观察到其中的联系，选取了个人比较熟悉的6本小说，最终选取的小说及其标签如下：

```
label_dic = {'鹿鼎记': 0, '射雕英雄传': 1, '神雕侠侣': 2, '天龙八部': 3, '笑傲江湖': 4, '倚天屠龙记': 5}
```

随机生成代码如下：

```
def get_segment(index_file, length):
    """
    随机获得长度为length的选段
    :param index_file: 供选取的小说列表
    :param length: 选段的长度
    :return: 保存的选段路径，选段内容
    """
    segment = []
    with open(index_file, 'r') as f:
        txt_list = f.readline().split(',')
        file_path = txt_list[random.randint(0, len(txt_list)-1)] + '.txt'
        split_words = get_single_corpus(DATA_PATH + file_path)
        start = random.randint(0, len(split_words)-length-1)
        segment.extend(split_words[start: start+length])
    return file_path, segment
```

3.2 LDA模型

首先定义模型需要使用的一些变量及参数：

```
word2id = {}  # word和id对应map
id2word = {}  # id和word对应map
word2topic = []  # 每个word所属主题
topic_num = 20  # 主题数
file_num = len(segments)  # 文档数
word_num = len(word2id)  # 总词数
alpha = len(segments[0])/topic_num/10  # 初始alpha
beta = 1/topic_num  # 初始beta
file2topic = np.zeros([file_num, topic_num]) + alpha  # 文档各主题分布
topic2word = np.zeros([topic_num, word_num]) + beta  # 主题各词分布
topic_count = np.zeros([topic_num]) + word_num * beta  # 每个主题总词数
```

LDA模型的核心部分包括模型初始化和Gibbs Sampling两部分，在模型初始化时，随机为文档中的每个词分配一个主题，之后统计每个主题z下词出现的概率，及每个文档m下出现主题z的数量，以及每个主题下词的总量，初始化代码如下：

```
def initialize():
    for f_idx, segment in enumerate(segments):
        temp_word2topic = []
        for w_idx, word in enumerate(segment):
            init_topic = random.randint(0, topic_num-1)
            file2topic[f_idx, init_topic] += 1
            topic2word[init_topic, word] += 1
            topic_count[init_topic] += 1
            temp_word2topic.append(init_topic)
        word2topic.append(temp_word2topic)
```

经过初始化之后，每个词都随机分到了一个主题，为了避免出现某一词出现极少等现象导致的除零异常，设置了α和$beta$参数。

```
def gibbs_sample():
    global file2topic
    global topic2word
    global topic_count
    new_file2topic = np.zeros([file_num, topic_num])
    new_topic2word = np.zeros([topic_num, word_num])
    new_topic_count = np.zeros([topic_num])
    for f_idx, segment in enumerate(segments):
        for w_idx, word in enumerate(segment):
            old_topic = word2topic[f_idx][w_idx]
            p = np.divide(np.multiply(file2topic[f_idx, :], topic2word[:, word]),
topic_count)
            new_topic = np.random.multinomial(1, p/p.sum()).argmax()
            word2topic[f_idx][w_idx] = new_topic
            new_file2topic[f_idx, new_topic] += 1
            new_topic2word[new_topic, word] += 1
            new_topic_count[new_topic] += 1
    file2topic = new_file2topic
    topic2word = new_topic2word
    topic_count = new_topic_count
```

最后则迭代运行Gibbs Sampling。

3.3 SVM分类

在经过Gibbs Sampling后，可计算得到各个文档对应主题的分布，同时每个文档的标签为对应的小说，因此可以使用其主题分布作为特征训练分类器，判断随机的文档属于哪个小说。

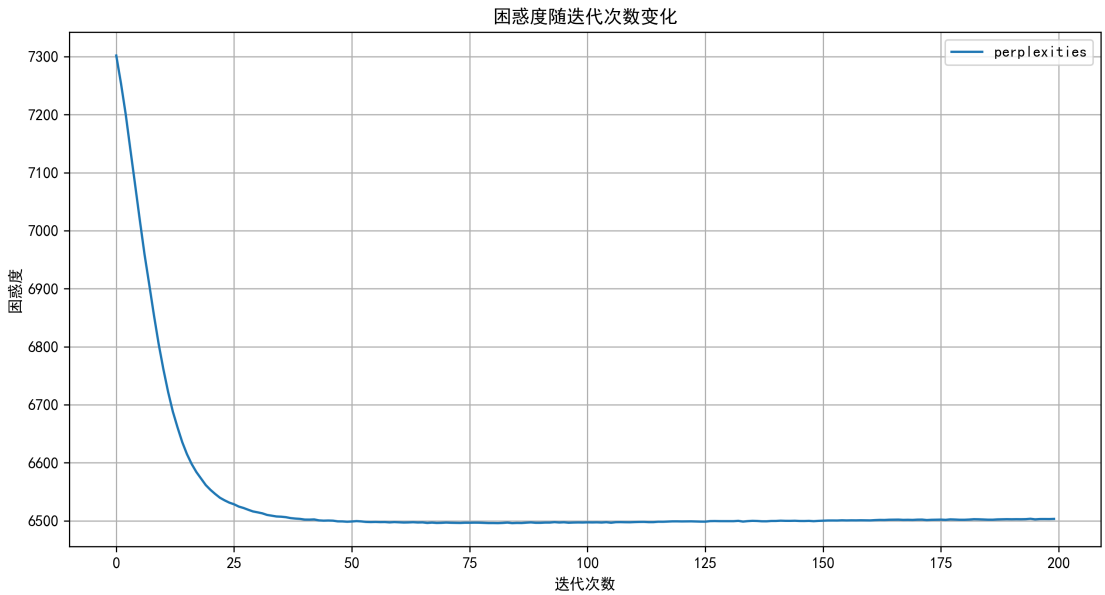
这里使用SVM中的OneVsRestClassifier作为多分类模型，首先随机将200个段落分为训练集和测试集，训练集和测试集比例为4：1，然后使用训练集进行训练，训练集上评估效果，并将模型保存，之后在测试集上进行测试。

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=.2, random_state=3)
model = OneVsRestClassifier(svm.SVC(kernel='linear', probability=True))
classifier = model.fit(x_train, y_train)
joblib.dump(classifier, "classifier.pkl")
y_predict = classifier.predict(x_train)
accuracy = metrics.accuracy_score(y_predict, y_train)
print('overall accuracy:{:.6f}'.format(accuracy))
print('----------------------')
single_accuracy = metrics.precision_score(y_train, y_predict, average=None)
print('accuracy for each class:', single_accuracy)
print('----------------------')
avg_acc = np.mean(single_accuracy)
print('average accuracy:{:.6f}'.format(avg_acc))
```

分别计算了整体预测的准确率，对于每类预测的准确率和对每类预测的算数平均准确率。

4.实验结果

从6本小说中随机选取200个段落，每个段落1000词，设置20个主题，在LDA训练过程中计算困惑度，如图所示，可见经过50次迭代模型已基本稳定。



对每个主题下的词根据概率进行降序排列，部分结果展示如下：

```
主题0的高频词为：
派:0.001880 道:0.001559 弟子:0.000532  五岳:0.000528  少林:0.000516  傲:0.000478 恒山:0.000444  左冷禅:0.000423 嵩山:0.000414  长剑:0.000406
------------------------
主题1的高频词为：
武功:0.001567  却:0.001441 黄蓉:0.001314  功夫:0.000917  已:0.000676 左手:0.000562  使:0.000528 洪七公:0.000528 道:0.000490 听:0.000478
------------------------
主题2的高频词为：
张无忌:0.001369 便:0.000680 已:0.000566 之间:0.000520  今日:0.000456  大:0.000440 上:0.000427 便是:0.000423  性命:0.000410  赵志敬:0.000406
------------------------
主题3的高频词为：
教主:0.001115  人:0.000896 道:0.000731 都:0.000456 婆婆:0.000397  大人:0.000397  使:0.000385 请:0.000334 说道:0.000309  命:0.000296
------------------------
主题4的高频词为：
出:0.000930 一个:0.000727  却:0.000697 双手:0.000537  二人:0.000528  姑娘:0.000486  地下:0.000406  找:0.000402 萧峰:0.000397  心想:0.000385
------------------------
主题5的高频词为：
便:0.001479 说道:0.001039  周伯通:0.000697 却:0.000558 虚竹:0.000528  道:0.000499 还:0.000478 坐:0.000440 和尚:0.000427  女儿:0.000393
------------------------
主题6的高频词为：
去:0.000824 不:0.000647 瞧:0.000545 看:0.000532 才:0.000456 之中:0.000452  无:0.000435 便:0.000347 更:0.000342 伸手:0.000300
------------------------
主题7的高频词为：
道:0.001407 人:0.001335 说:0.001284 便:0.001065 说道:0.000892  下:0.000634 上:0.000604 脸上:0.000583  请:0.000549 去:0.000524
------------------------
主题8的高频词为：
郭靖:0.001965  黄药师:0.000866 再:0.000731 心中:0.000562  两人:0.000549  爹爹:0.000537  梅超风:0.000385 去:0.000351 救:0.000338 见:0.000321
------------------------
主题9的高频词为：
道:0.002624 走:0.000672 快:0.000575 喝道:0.000554  上:0.000549 罢:0.000516 再:0.000507 好:0.000469 下:0.000406 笑:0.000385
------------------------
主题10的高频词为：
道:0.000799 林平之:0.000537 之中:0.000507  一声:0.000503  林震南:0.000503 却:0.000482 说道:0.000469  便:0.000423 不知:0.000418  没:0.000406
------------------------
主题11的高频词为：
杨过:0.001415  甚:0.001411 麽:0.001344 □:0.000925 後:0.000723 小龙女:0.000718 不:0.000680 却:0.000676 著:0.000668 一个:0.000571
------------------------
主题12的高频词为：
已:0.001432 师父:0.000828  中:0.000693 只:0.000604 见:0.000592 身子:0.000549  不知:0.000478  眼见:0.000478  内力:0.000456  倒:0.000414
------------------------
主题13的高频词为：
韦小宝:0.004047 道:0.003321 说:0.001255 不:0.001221 . :0.001183 笑:0.000854 傲:0.000731 小:0.000693 杀:0.000680 好:0.000596
------------------------
主题14的高频词为：
令狐冲:0.001644 都:0.000913 好:0.000685 道:0.000587 剑:0.000507 丐帮:0.000478  没:0.000452 盈盈:0.000435  岳灵珊:0.000431 大:0.000397
------------------------
```

可以看出模型训练效果较好，主题0中出现了派、五岳、少林、恒山、嵩山等词，这与金庸小说中的派系息息相关，而左冷禅作为笑傲江湖中的人物，其本身也是嵩山派的掌门并且使用长剑；主题1中的黄蓉和洪七公；主题5中的虚竹、和尚；主题8中的郭靖、黄药师、梅超风等；主题11中的杨过、小龙女；主题14中的令狐冲、任盈盈、岳灵珊；都是有千丝万缕联系的角色，并且大多属于同一作品。

但也有些不尽人意的情况，其中一些词似乎没有明显的特点，是一些比较常用的词，且意义不大。

接下来使用提取到的特征对文档进行分类，其中{'鹿鼎记': 0, '射雕英雄传': 1, '神雕侠侣': 2, '天龙八部': 3, '笑傲江湖': 4, '倚天屠龙记': 5}为文档和label的对应关系，训练集和测试集的分类效果如下：

```
start training!
finish training
overall accuracy:0.756250
----------------------
accuracy for each class: [0.60377358 0.83333333 0.80952381 1.         0.76       0.83333333]
----------------------
average accuracy:0.806661
```

```
overall accuracy:0.750000
----------------------
accuracy for each class: [0.61538462 0.83333333 0.75       1.         0.88888889 0.5       ]
----------------------
average accuracy:0.764601
```

可以看出模型在训练集和测试集的准确率都接近75%，对于有6类的情况已经达到了不错的效果。

结论

LDA模型能够较好地解决一词多义和多词一意的问题，实验说明了LDA的有效性，并通过SVM进行了验证，针对金庸小说，有效的分类通常是一些有密切关系的人物，或者有关联的动作等。

参考文档

文本主题模型 LDA

LDA主题模型的原理和建模