DL-NLP 第三次实验——LDA

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、实验原理

1. LDA 主题模型

隐含狄利克雷分布（Latent Dirichlet Allocation，LDA），是一种主题模型（topic model），它可以将文档集中每篇文档的主题按照概率分布的形式给出。

LDA 主题模型主要用于推测文档的主题分布，可以将文档集中每篇文档的主题以概率分布的形式给出根据主题进行主题聚类或文本分类。

LDA 主题模型不关心文档中单词的顺序，通常使用词袋特征（bag-of-word feature）来代表文档。

LDA 文本生成模型认为主题可以由一个词汇分布来表示，而文章可以由主题分布来表示。即想要生成一篇文章，可以先按照一定的概率分布选取某个主题，再以某个概率分布选取那个主题下的某个单词，不断重复这两步就可以生成最终文章，具体如下[1]：

1. 从狄利克雷分布α中取样生成文档d_i的主题分布θ_i；

2. 从主题的多项式分布θ_i中取样生成文档d_i的第j个词的主题$z_{i,j}$；

3. 从狄利克雷分布β中取样生成主题$z_{i,j}$对应的词语分布$\phi_{z_{i,j}}$；

4. 从词语的多项式分布$\phi_{z_{i,j}}$中采样最终生成词语$\omega_{i,j}$。

看到文章推断其隐藏的主题分布，就是建模的目的。换言之，人类根据文档生成模型写成了各类文章，然后丢给了计算机，相当于计算机看到的是一篇篇已经写好的文章。现在计算机需要根据一篇篇文章中看到的一系列词归纳出当篇文

章的主题，进而得出各个主题各自不同的出现概率：主题分布。

2. 分类器：单隐藏层的感知器

多层感知器（MLP，Multilayer Perceptron）是一种前馈人工神经网络模型，其将输入的多个数据集映射到单一的输出的数据集上。其在单层神经网络的基础上引入了一到多个隐藏层（hidden layer）。隐藏层位于输入层和输出层之间。

本次实验将 LDA 模型给出的主题概率分布作为特征输入 MLP 中，输出各类概率。考虑到特征的各种因素，考虑使用单隐藏层隐层节点较少的 MLP。

3. 停用词

停用词是指在信息检索中，为节省存储空间和提高搜索效率，在处理自然语言数据(或文本)之前或之后会自动过滤掉某些字或词,这些字或词即被称为 Stop Words（停用词）。通常意义上，停用词大致分为两类。一类是人类语言中包含的功能词，这些功能词极其普遍，与其他词相比，功能词没有什么实际含义，比如'the'、'is'、'at'、'which'、'on'等。但是对于搜索引擎来说，当所要搜索的短语包含功能词，特别是像'The Who'、'The The'或'Take The'等复合名词时，停用词的使用就会导致问题。另一类词包括词汇词，比如'want'等，这些词应用十分广泛，但是对这样的词搜索引擎无法保证能够给出真正相关的搜索结果，难以帮助缩小搜索范围，同时还会降低搜索的效率，所以通常会把这些词从问题中移去，从而提高搜索性能。

三、实验步骤与结果分析

本次实验中，主要分为：文本处理与训练数据准备、LDA 训练、MLP 训练以及模型测试、结果分析四部分。

1. 文本处理与训练数据准备

同实验一，删除压缩包中的无关文档和文件，仅留下 16 个文档文件，将每个文档中的“本书来自 www.cr173.com 免费 txt 小说下载站 更多更新免费电子书请关注 www.cr173.com”删除。本步骤为手工完成。

数据读取与预处理：

```
1.  def load_data(path, ban_stop_words=False, stop_words_path=''):
```

```
    data = []
    names = []
    stop_words = set()
    stop_txt = os.listdir(stop_words_path)
    for file in stop_txt:  #停用词读取
        with open(stop_words_path + '/' + file, 'r', encoding='ANSI') as f:

            for j in f.readlines():
                stop_words.add(j.strip('\n'))
    replace = '[a-zA-Z0-9’!"#$%&\'()（）；：‘“？、》。《，*+,-./:：；
「<=>?@，。?★、…【】《》？“”‘’！[\\]^_`{|}~]+\n\u3000 '
    files = os.listdir(path)
    for file in files:
        with open(path + '/' + file, 'r', encoding='ANSI') as f:
            t = f.read()
            for i in replace:
                t = t.replace(i, '')  #符号清除
            if ban_stop_words:
                for i in stop_words:
                    t = t.replace(i, '')  #停用词清除
            c = jieba.lcut(t)
            data.append(c)
        f.close()
        print("{} loaded".format(file))
        names.append(file.split(".txt")[0])
    return data, names
```

在读取文档时，将其中非中文的部分全部删除掉，其中的 replace 部分同实验 1。停用词参考由百度、哈工大等创造的停用词表综合得到。

数据准备：

```
if __name__ == '__main__':
    words = 1000  #每段的词数
    topics = 100  #主题数
    train_paragraphs = 1000  #训练段落数
    test_paragraphs = math.ceil(0.2*train_paragraphs)  #测试段落数
    ban_stop_words = True  #是否启用停用词过滤
    data, text_names = load_data("./data", ban_stop_words, "./stop")
    text_num = len(data)
  #从每个小说中随机采样
    train_data = []
    train_label = []
    for i in range(text_num):
```

```
        for j in range(math.ceil(train_paragraphs/text_num)):
            start = np.random.randint(0, len(data[i])-words-1)
            train_data.append(data[i][start:start+words])
            train_label.append(i)

    test_data = []
    test_label = []
    for i in range(text_num):
        for j in range(math.ceil(test_paragraphs/text_num)):
            start = np.random.randint(0, len(data[i])-words-1)
            test_data.append(data[i][start:start+words])
            test_label.append(i)
```

2. LDA 训练

```
dictionary = corpora.Dictionary(train_data)  #建立了从 ID 到单词的映射关系的字典
train_corpus = [dictionary.doc2bow(t) for t in train_data]  #转词袋模型
lda = models.LdaModel(corpus=train_corpus, id2word=dictionary, num_topics=to
pics)  #搭建并训练 LDA 模型
```

本次实验采用了 gensim 库进行 LDA 模型建立和训练。

```
train_distribution = lda.get_document_topics(train_corpus)
train_matrix = np.zeros((len(train_label), topics))  #存储训练集主题概率分布
for i in range(len(train_distribution)):
    for j in train_distribution[i]:
        train_matrix[i][j[0]] = j[1]

test_corpus = [dictionary.doc2bow(t) for t in test_data]
test_distribution = lda.get_document_topics(test_corpus)
test_matrix = np.zeros((len(test_label), topics))  #存储测试集主题概率分布
for i in range(len(test_distribution)):
    for j in test_distribution[i]:
        test_matrix[i][j[0]] = j[1]
```

以上为获得基于训练后的 LDA 模型给出的各段文字的主题概率分布。

3. MLP 训练和测试

```
net_x_train = torch.FloatTensor(train_matrix)
net_y_train = torch.LongTensor(train_label)
net_x_test = torch.FloatTensor(test_matrix)
net_y_test = torch.LongTensor(test_label)
```

```
train_loader = DataLoader(TensorDataset(net_x_train, net_y_train), batch_size=96, shuffle=True)
net = torch.nn.Sequential(
    torch.nn.Linear(topics, 16), torch.nn.ReLU(),
    torch.nn.Linear(16, len(text_names))
)
loss_func = torch.nn.CrossEntropyLoss()
optimizer = torch.optim.Adam(lr=4e-2, params=net.parameters())

for epoch in range(500):
    net.train()
    sum_loss = 0
    sum_acc = 0

    for batch_x, batch_y in train_loader:
        batch_y_pred = net(batch_x)
        batch_y_pred_label = torch.argmax(batch_y_pred, dim=1)
        sum_acc += torch.eq(batch_y_pred_label, batch_y).float().sum()
        loss = loss_func(batch_y_pred, batch_y)
        sum_loss += loss

        optimizer.zero_grad()
        loss.backward()
        optimizer.step()
    if (epoch + 1) % 10 == 0:
        print("Epoch: %d/500 || Train || sum loss: %.3f || train_acc: %.2f || %d/%d"
              % (epoch+1, sum_loss, sum_acc/len(train_label), sum_acc, len(train_label)))

net.eval()
pre = net(net_x_test)
predictions = torch.argmax(pre, dim=1)
acc = torch.eq(predictions, net_y_test).float().sum()
loss = loss_func(pre, net_y_test)
print("Test || loss: %.3f || test_acc: %.2f || %d/%d" % (loss, acc/len(test_label), acc, len(test_label)))
```

利用 Pytorch 搭建了单隐藏层含 16 个隐藏节点的 MLP，激活函数选用 ReLU。

4. 结果分析

实验结果如下：

序号	主题数	段落数（±20 以内）	词数	去除停用词	训练集准确率（单次）	测试集准确率（单次）
0	5	200(208)	500	False	25.48%	12.50%
1	20	200(208)	500	False	44.23%	14.58%
2	50	200(208)	500	False	62.50%	16.67%
3	100	200(208)	500	False	78.37%	18.75%
4	100	500(512)	500	False	63.87%	22.32%
5	100	1000(1008)	500	False	54.66%	31.25%
6	100	2000(2000)	500	False	44.30%	34.00%
7	100	1000(1008)	1000	False	61.81%	47.12%
8	100	1000(1008)	2000	False	66.87%	62.02%
9	100	1000(1008)	500	True	77.98%	61.54%
10	100	1000(1008)	1000	True	83.63%	74.04%
11	100	1000(1008)	2000	True	86.11%	76.92%
12	100	2000(2000)	2000	True	95.00%	92.75%

本实验针对主题数、段落数、词数和是否去除停用词对训练效果进行的影响进行了讨论。由 0-3 可知，单纯提高主题数会使训练集准确率显著提升，但对测试集数据影响甚微，考虑可能是段落数和词数决定的信息数量不足以支撑起主题数量，导致 LDA 捕获到了噪声信息进而使得 MLP 发生了过拟合问题。由 3-6 可知，在主题数较高时，单纯提高段落数量相当于整体拥有更多的信息，使得过拟合现象有所好转，并且测试集表现也会变好，这里训练集准确率降低是因为缓解了过拟合问题。由 5、7、8 对比可知，在主题数、段落数量均较高时，提高词数相当于每个样本用于判定的信息变多，此时训练集和测试集的表现均有十分明显的提升。由 5-9,7-10,8-11 对比可知，去除停用词相当于提高了有效信息的占有率，也相当于在几乎不提升计算量的情况下提升了词数或者说有效词数，此时训练集和测试集的表现也均有十分明显的提升。

四、参考文献

1. https://blog.csdn.net/v_JULY_v/article/details/41209515
2. https://blog.csdn.net/weixin_50891266/article/details/116273153