Report of Homework Two

张海尼
ZY2303215

Abstract

报告的研究问题包括两个部分。第一部分：利用 LDA 模型在指定语料库上进行文本建模；第二部分：使用随机森林分类算法实现文本分类。在研究问题的基础上，报告探讨了主题个数 T、分词的基本单元（“词”和“字”），以及每个段落中 token 的长度 K 对分类性能和主题模型性能的差异。

Introduction

I1：文本建模

报告中使用 LDA 主题模型对指定语料库进行文本建模，主题数量为 T。其中，指定语料库为从金庸小说集中均匀抽取的 1000 个段落（每个段落有 K 个 token），段落的标签为其所属的小说。主题模型是对寻找文本中隐含主题的一种建模方法，每个主题表示为词表中词语的概率分布。主题模型通过学习一系列的文档，根据统计规律发现抽象主题。

I2：文本分类

基于 LDA 主题模型获得每个段落的主题分布，并根据主题分布进行文本分类。文本分类模型使用随机森林模型，分类结果使用 10 次交叉验证（900 做训练，100 做测试循环十次），分类性能使用准确度衡量。

Methodology

M1：主题模型——LDA

LDA（Latent Dirichlet Allocation）是由 Blei 等人于 2003 年提出的文本分析模型，其核心思想是将文本表示为一组主题的概率分布，其中每个主题由多种单词组成。

定义文档为 doc，文档集合为 Doc，主题为 topic，主题集合为 Topic。Doc 中每个文档 doc 看做一个单词序列 $< w_1, w_2, \cdots, w_n >$，其中 w_i 为第 i 个单词，n 为 doc 总词数。Doc 中

所有不同单词组成集合 Voc，Voc 的总词数为 m。

每个 doc 对应于不同主题的概率为 $\theta_d =< p_{t1}, p_{t2}, \cdots, p_{tk} >$，其中 p_{ti} 表示 doc 对应的 Topic 中第 i 个 topic 的概率，k 为 Topic 中总主题数。$p_{ti} = n_{ti} / n$，其中，n_{ti} 为 doc 中属于第 i 个 topic 中词的数目。

对 Topic 中的第 t 个 topic，生成不同单词的概率为 $\varphi_t =< p_{w1}, p_{w2}, \cdots, p_{wm} >$，其中 p_{wi} 为第 t 个 topic 中生成 Voc 中第 i 个单词的概率。$p_{wi} = N_{wi} / N$，其中，N_{wi} 为第 t 个 topic 中对应到 Voc 中第 i 个词的数目，N 为对应到第 t 个 topic 的总词数。

LDA 的核心公式如下：

$$p(w|d) = p(w|t)p(t|d)$$

其中，$p(t|d)$ 使用 θ_d 计算得到，$p(w|t)$ 使用 φ_t 计算得到。以 Topic 为中间层，通过 θ_d 和 φ_t，可以计算一个 doc 中的一个词在不同 topic 下所对应的 $p(w|d)$，并根据结果对该词所属 topic 进行更新，若其 topic 改变，则 θ_d 和 φ_t 也随之改变。训练多个循环后，可以得到相对准确的主题模型，并根据主题模型计算文本对应不同主题的概率，用以后续的分类算法。

M2：分类模型——随机森林

随机森林（Random Forest，RF）是一种基于决策树的 Bagging 类型的集成分类算法，通过组合多个弱分类器的结果进行分类，使得模型整体的结果具有较高的精确度和泛化性能。

Bagging 算法一般使用 boostrap 进行随机采样，每棵树采集相同的样本数量（一般小于原始样本量）。使用该采样方法可以在不同的采样过程中得到不同的采样集，保证样本的随机性。RF 使用 CART 决策树作为弱学习器，与一般的 CART 树不同，在产生每棵树时，每棵树都是随机选取少数特征，保证了特征的随机性。两个随机性的引入，使得 RF 具有较好的抗噪声能力。

随机森林的泛化能力由两方面保证。一是选取特征的随机性，降低了过拟合的风险。二是决策树的改进，相较于普通决策树，RF 会在节点上的所有样本特征中选择一个最优特征划分决策树的左右子树，将随机性的效果进一步扩大，增强模型的泛化能力。

随机森林对数据集的适应力较强，适于处理高维度数据，保证其在不同主题数下均有较好的分类能力。

Experimental Studies

E1：段落抽取

1.1 数据处理与验证的算法流程

1) 所用数据库为 16 部金庸小说；
2) 删除部分文件开头的无用信息；
3) 根据 cn_stopwords.txt 删除中文停词；
4) 删除无意义符号，包括标点符号和空白符号等；
5) 根据每篇小说所含词数占语料库总词数的比例确定每篇小说抽取的段落数；
6) 根据计算的段落数抽取段落，并保存为语料库文件。

1.2 部分代码

```
# 去除无用信息
txt = txt.replace('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', '')

data = []
if cutIsTrue is False:
    dir = "ch_corpus_word/"

    # 去除无意义字符
    for word in txt:
        if (word not in extra_characters) and (not word.isspace()):
            data.append(word)
else:
    dir = "ch_corpus_words/"

    # 分词
    cut_words = jieba.lcut(txt)

    # 去除无意义字符
    for word in cut_words:
        if (word not in extra_characters) and (not word.isspace()):
            data.append(word)
```

```
id = 1
con_list = []
count_sum = 0
for name in names:
    count = math.floor(len(dict[name])/word_len * 1000 + 0.48)
    # print(f"{name}:{count}----{len(dict[name])}")

    # 抽取段落
    pos = int(len(dict[name])//count)
    for i in range(count):
        tmp = dict[name][(i*pos):(i*pos+K)]
        con = {
            'id': id,
            'name': name,
            'data': tmp
        }
        con_list.append(con)
        id += 1
```

1.3 实验结果

Figure 1：以字为基本单元的段落划分结果

Figure 2：以词为基本单元的段落划分结果

E2: LDA 模型

2.1 计算信息熵的算法流程

1) 所用语料库从 16 部金庸小说中提取；
2) 构建词典，将语料向量化表示；
3) 训练 LDA 模型；
4) 评估 LDA 模型性能；
5) 保存模型。

2.2 语言模型部分代码

2.2.1 模型训练

```
# 在所有段落上建模
dataset = pd.DataFrame()
dataset['parag'] = data.iloc[:, -1]
dataset['parag'] = dataset['parag'].apply(lambda x: eval(x))
dataList = dataset['parag'].tolist()

# 构建词典，语料向量化表示
dict = corpora.Dictionary(dataList)

# 删掉只在不超过20个文本中出现过的词，删掉在90%及以上的文本都出现了的词
dict.filter_extremes(no_below=20, no_above=0.9)
dict.compactify()  # 去掉因删除词汇而出现的空白

corpus = [dict.doc2bow(text) for text in dataList]  # 表示为第几个单词出现了几次

# LDA模型
print(f'cutIsTrue: {cutIsTrue}   K:{K}  T:{T}  ----------------')
ldamodel = LdaModel(corpus, num_topics=T, id2word=dict, passes=40, random_state=6)  # 分为T个主题
# 字: passes=40, random_state=6
```

2.2.2 模型评价

```
# 加载模型
savePath = save_dir + "_" + str(K) + "_" + str(T)
# 加载 LDA 模型
lda = LdaModel.load(savePath + "_ldaModel.model")

# 评估模型性能
coherence_model_lda = CoherenceModel(model=lda, texts=dataList, dictionary=dict, coherence='c_v')
coherence_lda = coherence_model_lda.get_coherence()
per = lda.log_perplexity(corpus)
print(f'perplexity:{per}     coherence:{coherence_lda}')

vis = pyLDAvis.gensim_models.prepare(lda, corpus, dict)
pyLDAvis.show(vis, local=False)
```

2.3 实验结果

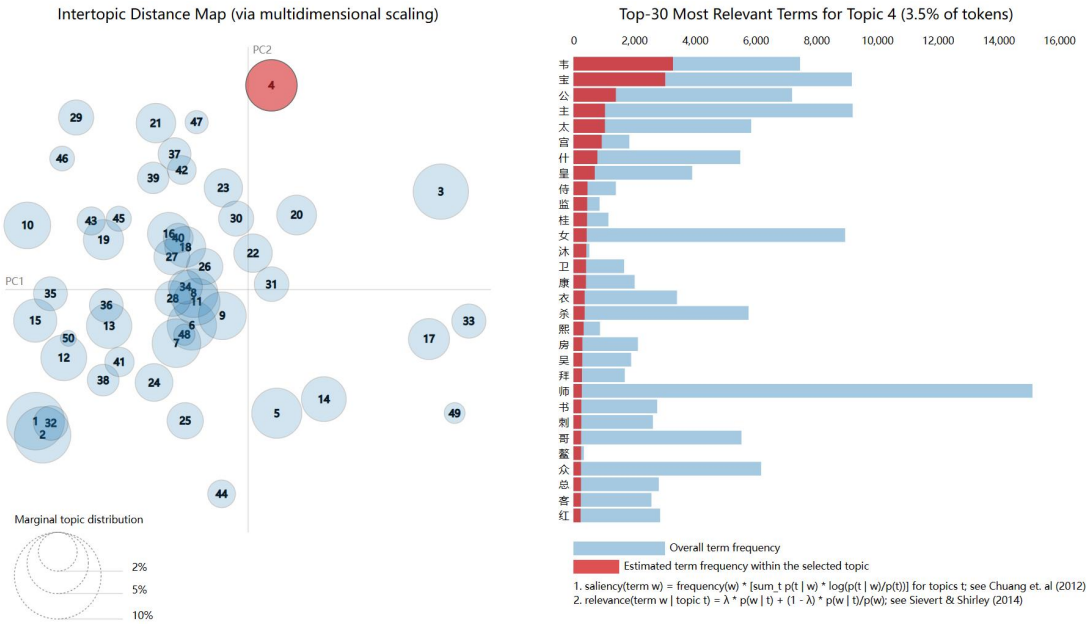

Figure 3：K=3000，T=50 时以字为基本单元的模型评估结果

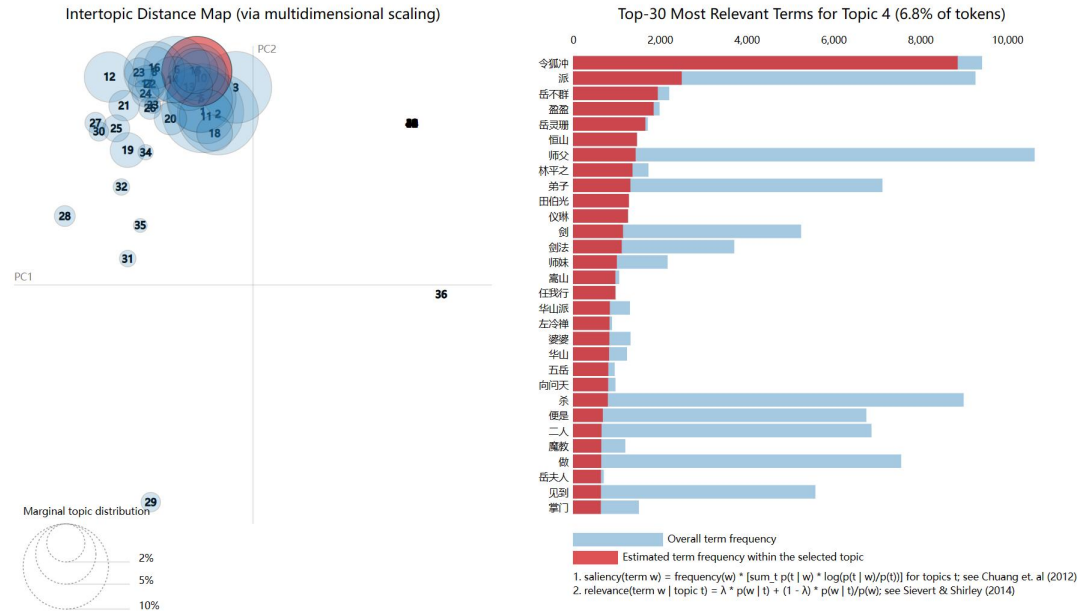

Figure 4：K=3000，T=50 时以词为基本单元的模型评估结果

E3：随机森林分类

3.1 计算信息熵的算法流程

1) 对每个段落计算主题分布；
2) 获取段落对应标签；
3) 按照 9:1 划分训练集和测试集；
4) 训练随机森林模型；
5) 交叉验证 10 次，以准确率均值作为模型评价标准。

3.2 语言模型部分代码

```
# 将每个段落进行做主题分布
topic_matrix = []
for tmp in dataList:
    cor = dict.doc2bow(tmp)

    topic_distribution = lda.get_document_topics(cor,  minimum_probability=0)
    topic_distribution = [prob for topic, prob in topic_distribution]

    topic_matrix.append(topic_distribution)

topic_matrix = np.array(topic_matrix)

# 获取标签
label = data.iloc[:, -2].tolist()
label = np.array(label)

label_encoder = LabelEncoder()
label_num = label_encoder.fit_transform(label)

# 训练集训练分类模型
# 将标签和topic对应，然后划分数据集，进行分类
result = []
for i in range(10):
    X_train, X_test, y_train, y_test = train_test_split(topic_matrix, label_num, test_size=100, random_state=i*20)
    clf = RandomForestClassifier(n_estimators=100, max_depth=12, random_state=0)
    # 字：n_estimators=100, max_depth=12, random_state=0
    clf.fit(X_train, y_train)

    x_pred = clf.predict(X_train)
    x_acc = accuracy_score(y_train, x_pred)

    y_pred = clf.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    precision = precision_score(y_test, y_pred, average='micro')
    recall = recall_score(y_test, y_pred, average='micro')
    F1 = 2*precision*recall/(precision+recall)
    tmp_result = [accuracy, precision, recall, F1]
```

3.2 实验结果

Table 1：LDA 模型评估及分类性能

基本单元	T	K	Perplexity	Coherence	测试集准确率
字	50	20	-5.905	0.221	0.233
	50	100	-6.542	0.314	0.426
	50	500	-6.824	0.336	0.781
	50	1000	-6.856	0.403	0.839
	50	3000	-6.874	0.462	0.884
词	50	20	-4.522	0.290	0.248
	50	100	-6.466	0.297	0.460
	50	500	-7.570	0.336	0.822
	50	1000	-8.018	0.438	0.858
	50	3000	-8.570	0.461	0.941
	16	500	-7.388	0.373	0.767

	30	500	-7.452	0.350	0.819
	50	500	-7.570	0.336	0.822
	70	500	-7.591	0.333	0.800
	90	500	-7.710	0.317	0.790

Conclusions

C1: LDA 主题模型性能

从 Table 1 中可以看出，当主题数 T 和 token 取值 K 相同时，以词为基本单元的主题模型性能优于以字为基本单元的主题模型。以词为基本单元时，其困惑度和一致性的得分的绝对值相较于以字为基本单元时都相对较高，相应的分类模型准确率也更高。Figure 3 和 Figure 4 给出了 K=3000，T=50 时的主题模型的可视化评估，从两图可以看出，以字为基本单元时，主题间较为分散，且每个主题的重要程度较小（气泡面积较小），不利于分类；以词为基本单元时，训练出的主题重要程度较大，即主题更有代表性，利于分类。从两图右侧中主题中关键词打分排序也可看出，以词为基本单元的主题模型中人名的排序较高，相较于单字更能体现不同小说的特点，符合常理认知。

随着 K 值的增加，不同基本单元的主题模型性能变化呈现出相同的趋势，即主题模型困惑度和一致性的得分的绝对值也随之上升，其分类结果的准确度也随之增加，主题模型表现出来更好的性能，说明长文本相较于短文本的独特性更高，使用长文本进行训练能够提取出更为有效的文本主题。

C2：随机森林模型分类性能

根据 Table 1 可知，当 K 固定位置 500 时，随着 T 从 16 上升到 50，测试集准确度稳定上升，分类模型性能逐渐提高，当 T 继续增大时，测试集准确度逐渐下降，分类模型性能下降。在选择主题模型时，应当选择合适的主题数进行训练，若主题数过高，则主题间过于分散，不利于文本分类。

根据 Table 1 可知，以词为基本单元的分类性能优于以字为基本单元的分类性能。根据 Figure 3 和 Figure 4，以词为基本单元的主题模型中人名的排序较高，主题的独特性更高，能够更好地体现不同小说的特点，而以字为单位时独特性较低，因此以词为基本单元的分类性能更高，符合常理认知。

References

[1] https://zhuanlan.zhihu.com/p/134161509