深度学习与自然语言处理第三次作业

LDA 模型文本建模主题分类

惠仪

SY2202328@buaa.edu.cn

摘要

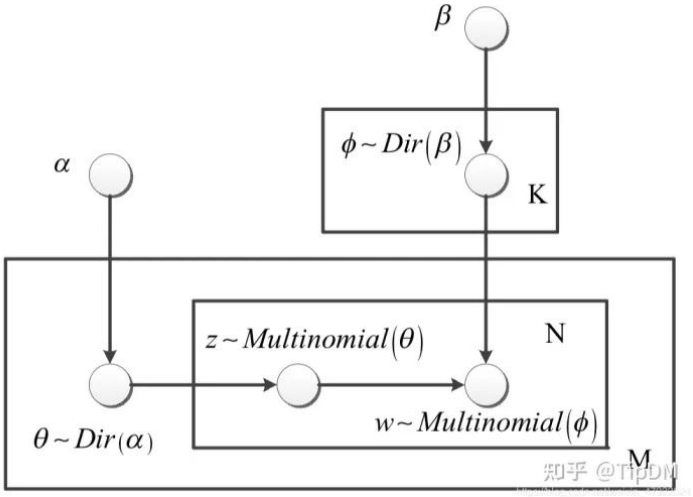
从链接给定的语料库中均匀抽取 200 个段落， 每个段落的标签就是对应段落所属的小说。利用 LDA 模型对于文本建模，并把每个段落表示为主题分布后进行分类。验证与分析分类结果。（1）在不同数量的主题个数下分类性能的变化；（2）以"词"和以"字"为基本单元下分类结果有什么差异

一、理论介绍

LDA 是一种文档主题生成模型，也称为一个三层贝叶斯概率模型，包含词、主题和文档三层结构。我们认为一篇文章的每个词都是通过以一定概率选择了某个主题，并从这个主题中以一定概率选择某个词语。文档到主题服从多项式分布，主题到词服从多项式分布。LDA 是一种主题模型，用于从文本集合中发现隐藏的主题结构。在 LDA 模型中，每个文档被表示为多个主题的混合，每个主题则被表示为多个词的集合。

LDA 模型的基本原理是先假设一个文本集合的生成过程：首先，从主题分布中随机选择一个主题；然后，从该主题的单词分布中随机选择一个单词；重复上述过程，直到生成整个文本。通过对这个过程进行反推，可以得到 LDA 模型的参数估计方法。具体来说，我们需要通过文本数据中观察到的单词来估计每个主题的单词分布以及每个文档的主题分布，然后通过这些参数来推断文本的主题结构。

参数估计：对每篇文档中每个单词的主题进行估计，对每个主题中每个单词的概率分布进行估计。两个参数分别是主题-词语分布参数和文档-主题分布参数。对给定的文档，可以通过先验分布和文档中的单词分布推断出其主题分布，从而得到文档的主题表示。



二、实验步骤

1、定义数据处理，其中包括去除标点符号、无意义词和句子等

```
# 去除符号及中文无意义stopwords
def filter(text):
    a = re.sub(xx, '', text)
    b = re.sub(xx2, '', a)
    pattern = '|'.join(stopwords)
    c = re.sub(pattern, '', b)
    d = re.sub(r'\*', '', c)
    e = re.sub(r'\.', '', d)
    f = re.sub(r'\n', '', e)
    g = re.sub(r'\u3000', '', f)
    h = re.sub('\s+', '', g).strip()
    return h
```

2、根据要求均匀抽取 200 个段落（每个段落大于 500 个词）
在不同的位置提取 500 个词，组成 16 个文档的文本，构建训练集和测试集

```
pos = int(len(con)//13) ####16篇文章，分词后，每篇均匀选取13个500词段落进行建模
for i in range(13):
    if j == 0:
        con_temp = con_temp + con_list[i*pos:i*pos+500]
    else:
        con_temp = con_temp + con_list[i*pos+501:i*pos+1000]
```

3、建立 LDA 模型
创建字典和预料库，训练 LDA 模型，iteration 和 passes 分别表示 LDA 模型的迭代次数和遍历次数。通常情况下，增加迭代次数可以提高模型的表现，但训练时间也会相应增加。

```
# 创建词袋表示法
dictionary = Dictionary(train_docs)
#构建语料库
train_corpus = [dictionary.doc2bow(doc) for doc in train_docs]
test_corpus = [dictionary.doc2bow(doc) for doc in test_docs]
# 训练LDA模型
acc = []
# for num_topics in range(1, 50):
num_topics =16
lda_model = LdaModel(corpus=train_corpus, id2word=dictionary,
                     num_topics=num_topics, iterations=600, passes=3, random_state=42)
```

4、获取预测文档和训练文档的主题分布

```
train_topic_distributions = []
for doc in train_corpus:
    doc_topics = lda_model.get_document_topics(doc)
    doc_topic_distribution = np.zeros(num_topics)
    for topic_id, prob in doc_topics:
        doc_topic_distribution[topic_id] = prob
    train_topic_distributions.append(doc_topic_distribution)

# 使用得到的主题分布对测试数据进行分类
test_topic_distributions = []
for doc in test_corpus:
    doc_topics = lda_model.get_document_topics(doc)
    doc_topic_distribution = np.zeros(num_topics)
    for topic_id, prob in doc_topics:
        doc_topic_distribution[topic_id] = prob
    test_topic_distributions.append(doc_topic_distribution)
```

5、使用 SVM 对主题分布进行分类，先用训练数据构建模型，再用测试的分布来输出预测标签

```
# 使用SVM对主题分布进行分类
svm = SVC(kernel='linear', C=1.0)
svm.fit(train_topic_distributions, labels)
test_predicted_labels = svm.predict(test_topic_distributions)
```

6、在测试集结果进行评估

```
accuracy = accuracy_score(testlabels, test_predicted_labels)
print("Number of LDA topics: ", num_topics, ", Accuracy: ", accuracy)
print(classification_report(testlabels, test_predicted_labels))
train_predicted_labels = svm.predict(train_topic_distributions)
print(classification_report(labels, train_predicted_labels))
```

三、实验设置及结果

1、模型参数：num_topics=16，iteration=200/400/600，passes=3，random_state=42

2、结果输出

以分词为单位

		训练数据			测试数据		
iteration		precision	recall	f1-score	precision	recall	f1-score
200	accuracy	0.50			0.5		
	macro avg	0.34	0.5	0.38	0.34	0.5	0.38
400	accuracy	0.62			0.50		
	macro avg	0.51	0.62	0.54	0.37	0.50	0.40
600	accuracy	**0.75**			0.69		
	macro avg	0.65	0.75	0.68	0.57	0.69	0.60
800	accuracy	**0.75**			**0.75**		
	macro avg	0.65	0.75	0.68	0.65	0.75	0.68

以字为单位

		训练数据	测试数据

iteration		precision	recall	f1-score	precision	recall	f1-score
200	accuracy	0.81			**0.69**		
	macro avg	0.74	0.81	0.76	0.58	0.69	0.61
400	accuracy	0.88			0.56		
	macro avg	0.83	0.88	0.84	0.41	0.56	0.45
600	accuracy	0.94			0.56		
	macro avg	0.91	0.94	0.92	0.44	0.56	0.47
800	accuracy	**0.94**			0.62		
	macro avg	0.91	0.94	0.92	0.52	0.62	0.55

通过调整模型参数能使模型的预测结果产生变化。

根据实验结果可知，当 iteration 不断增大时，以词为单位和以字为单位求解模型的准确率普遍是提高的。当 num_topics=16 时，对于训练数据来说，以字为单位的分类准确率高于以词为单位的准确率，对于测试数据来说，当 iteration 迭代次数增大时，以词为单位的准确率高于以字为单位的准确率。

四、分析

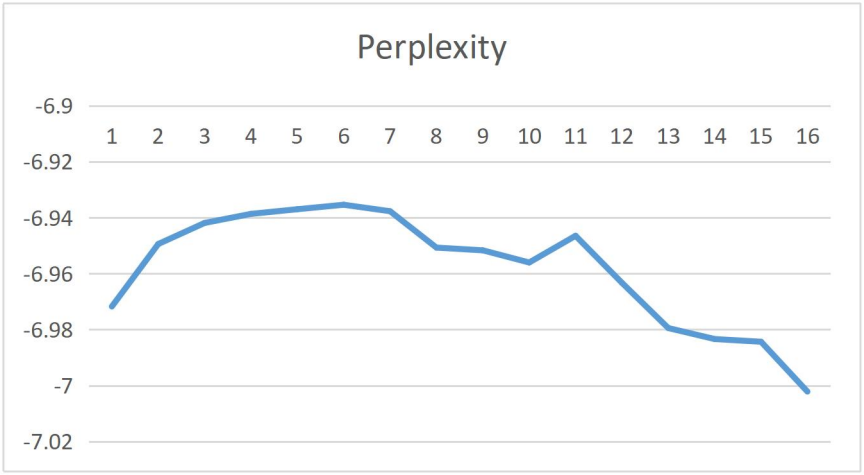
1、在不同数量的主题个数下性能的变化；

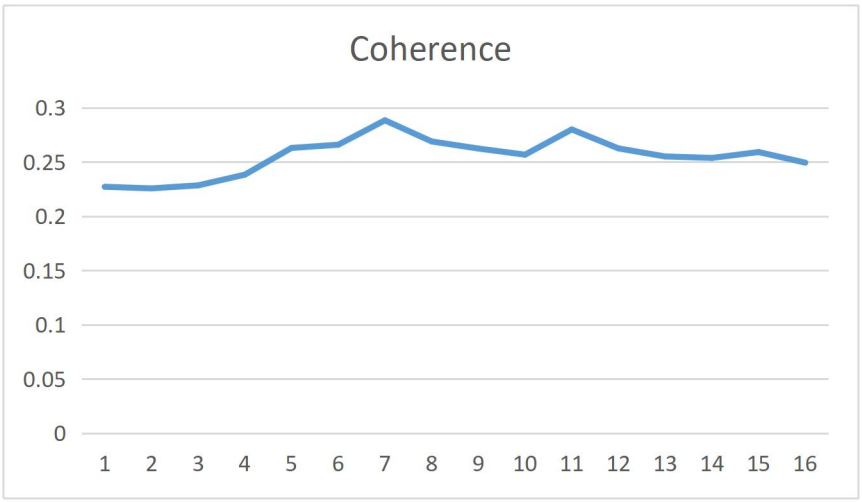
- Perplexity（困惑度）：是一种内部评估指标，用于测量模型在训练数据上的拟合程度。它可以用来评估模型是否过拟合（overfitting）或欠拟合（underfitting），困惑度越小，表示模型的表现越好。在 Gensim 中，计算困惑度时，可以使用 log_perplexity()方法来计算。该方法返回对数似然值的相反数，因此要用负数表示，所以困惑度的值越小表示模型越好。
- Coherence（一致性）：是一种外部评估指标，用于测量 LDA 模型提取主题的准确性。它通常与领域专家的判断或关键词相比较。Coherence 可以衡量相邻单词在话题上的相关性和连贯性，因此可以评估模型是否提取出了有意义的话题。通常来说，Coherence 值越高，表示模型的表现越好。在 Gensim 中，可以使用 CoherenceModel 来计算 Coherence。可以使用不同的评估方法，如 u_mass、c_v 等。

以字为单位

	Perplexity	Coherence
1	-6.9717605421497275	0.2270699060922813
2	-6.949444313083704	0.22560287665645562
3	-6.941912453816487	0.22836145301639724
4	-6.93868051320429	0.23823005058880897
5	-6.9370111082012835	0.26286448174878757
6	-6.935415938516075	0.2658139066321625
7	-6.937729257804843	**0.28832584840914693**
8	-6.950727558055176	0.2688386540443952
9	-6.951696782657733	0.2623150464030986
10	-6.956046583180244	0.25666239607653585
11	-6.946504150792956	0.2798909775341552
12	-6.963300081322973	0.2624220740164738
13	-6.979471859340484	0.2550266274699163

14	-6.98340042913132	0.2536508310873605
15	-6.984357854454563	0.2590663658525384
16	**-7.002166360613819**	0.24930663137743708





当主题数在 1-16 间时，主题数为 16 时困惑度最小模型较好，Coherence 值相差不大。

五、参考文献

[1] https://blog.csdn.net/AZRRR/article/details/90271414

[2] https://blog.csdn.net/qq_41667743/article/details/129378418?utm_medium=distribute.pc_relevant.none-task-blog-2~default~baidujs_baidulandingword~default-0-129378418-blog-80528540.235^v32^pc_relevant_default_base3&spm=1001.2101.3001.4242.1&utm_relevant_index=3