LDA 文本建模

杨洪易 ZY2203119
yhongyi@buaa.edu.cn

摘要:

本文在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，并利用 lda 模型完成文本分类，验证得到了较高的准确度。

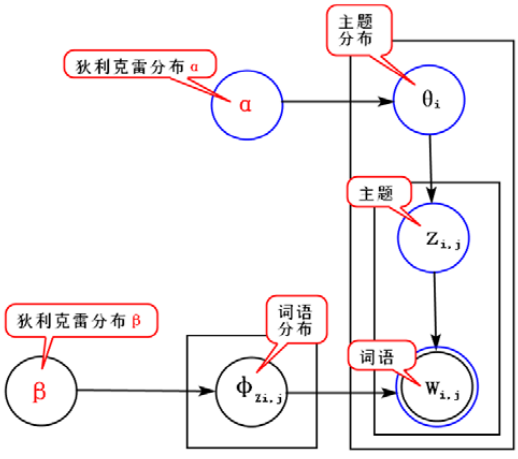
简介:

LDA 算法的生成过程和 PLSA 算法的生成模型的生成过程类似，其最大区别是 LDA 提供了先验分布作为参考，而 PLSA 没有。更具体的，PLSA 的生成过程简述为 d→z→w，而 LDA 的生成过程简述为 θ→z→w←φ。
在 LDA 主题模型中，文章的生成有三个要素，即词语、主题、文章，词语和主题是多对多的关系，每个词语都可能代表着多个主题，每个主题下也有多个代表的词语；主题和文章也是多对多的关系，每个主题都对应着多篇文章，每篇文章也可能有多个主题。
LDA 的具体流程如下:
1.生成文本的话题分布。按照狄利克雷分布 Dir(α) 随机生成一个参数向量 θ_m，循环 M 遍，得到参数向量 $\theta=(\theta_1,...,\theta_M)$

2.生成话题。取第 m 个文本的参数 θ_m，按照多项分布 Mult(θ_m) 随机生成一个话题 z_{mn}，循环 N_m 遍，得到话题向量 $z_m=(z_{m1},...,z_{mN_m})$

3.生成话题的单词分布。按照狄利克雷分布 Dir(α) 随机生成一个参数向量 φ_k，循环 K 遍，得到参数向量 $\varphi_1,...,\varphi_k$

4.生成单词。取第 m 个文本的第 n 个话题的参数 $\varphi_{z_{mn}}$，按照多项分布 Mult($\varphi_{z_{mn}}$) 随机生成一个单词 w_{mn}，循环 N_m 遍，得到单词序列 $w_m=(w_1,...,w_{mN_m})$，其对应话题序列 $z_m=(z_{m1},...,z_{mN_m})$



实现：

1.环境

Anaconda3 python3.9.16

2.实验过程：

读取数据后删去停词与广告，抽取 200 个段落，段字数大于 500。

```
def para_extract(para,label):
    # 段落抽取
    text_ls = []
    text_label = []
    random_indices = random.sample(range(len(para)), 200)
    text_ls.extend([para[i] for i in random_indices])
    text_label.extend([label[i] for i in random_indices])
    return text_ls,text_label
```

```
for sentence in content.split('\n'):
    if len(sentence) < 500:
        continue
    para_list.append(sentence)
    para_label.append(index)
```

使用了 gensim 的 LdaModel 库进行训练，完成文档生成主题、主题生成单词，从而实现随机的文档生成单词，随后通过 gibbs 采样，根据超参数求出文档主题的分布和主题词的分布。

```
# 基于词
lda_word = gensim.models.ldamodel.LdaModel(corpus=corp_word, id2word=dic_word, num_topics=num_topics,
                                           passes=20, alpha='auto', eta='auto')

perplexity_word = -lda_word.log_perplexity(corp_word)
cv_model_word = gensim.models.CoherenceModel(model=lda_word, texts=tokens_word, dictionary=dic_word,
                                             coherence='c_v')
self.pp_word_list.append(perplexity_word)
self.cv_word_list.append(cv_model_word.get_coherence())
# 基于字
lda_char = gensim.models.ldamodel.LdaModel(corpus=corp_char, id2word=dic_char, num_topics=num_topics,
                                           passes=20, alpha='auto', eta='auto')
perplexity_char = -lda_char.log_perplexity(corp_char)
cv_model_char = gensim.models.CoherenceModel(model=lda_char, texts=tokens_char, dictionary=dic_char,
                                             coherence='c_v')
self.pp_char_list.append(perplexity_char)
self.cv_char_list.append(cv_model_char.get_coherence())
```

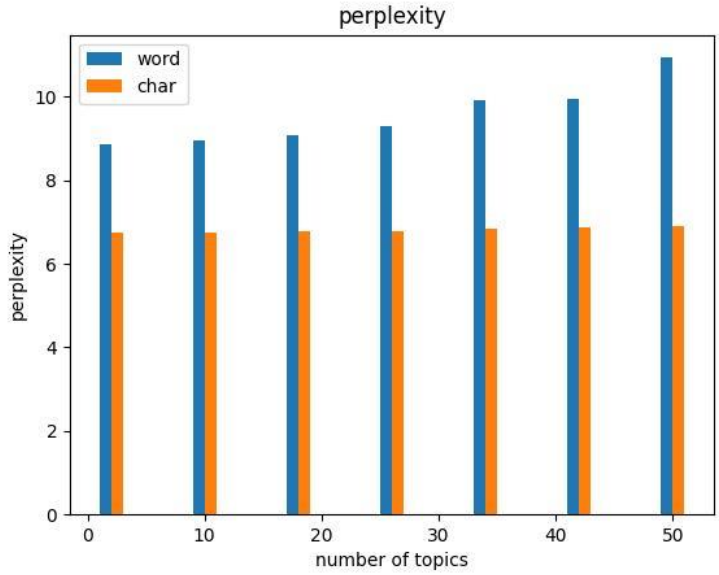
这里使用了困惑度和一致性概念作为评估。其中困惑度可以理解为对于一篇文章 d，所训练出来的模型对文档 d 属于哪个主题有多不确定，这个不确定程度就是困惑度。困惑度越低，说明分类的效果越好；主题的质量是通过一致性来衡量的，这表明人们对它的理解是多么容易。

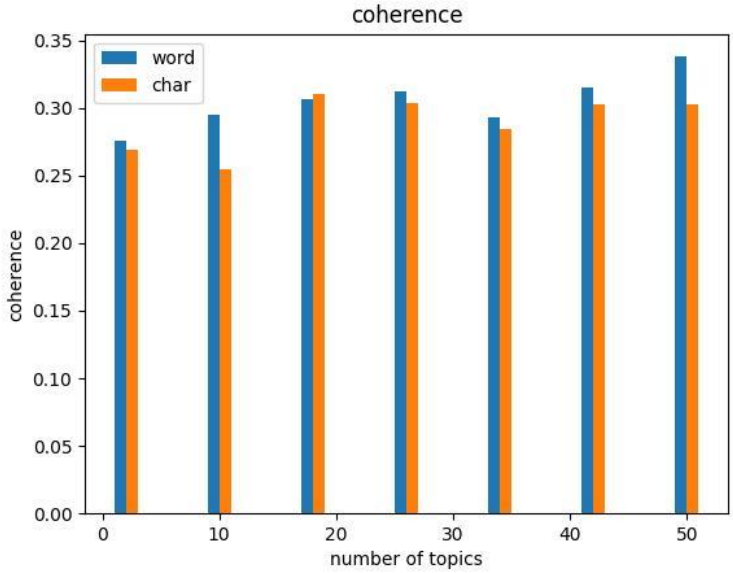
结果：

为节省时间，在 2，10，18，26，34，42，50 个主题分类下分别开展训练并完成了测试，并分别以字和词为单位绘制了他们的困惑度与一致性曲线。

```
Finish! 当前topic数量为: 2
Finish! 当前topic数量为: 10
Finish! 当前topic数量为: 18
Finish! 当前topic数量为: 26
Finish! 当前topic数量为: 34
Finish! 当前topic数量为: 42
Finish! 当前topic数量为: 50
```

从曲线图中信息可以看出，字与词相比，其困惑度与一致性基本均较低；同时无论字词，随主题数增加，困惑度与一致性均呈现上升趋势。可见，在困惑度方面，可能字比词更能准确表达文意；此外，随着主题数量增加，一致性略有波动，呈现增加趋势，主题连贯性更好，但同时过多的主题会导致困惑度上升，不确定程度增加。





结论:

使用词与字进行 LDA 建模分类时，选字为基本单位困惑度更小，主题归类时不确定性更低；同时主题数量增加会增加连贯性，但也会降低在确定性方面的模型性能，应综合考量选择合适主题数量。

参考:

[1]主题模型-LDA - 知乎 (zhihu.com)
[2] LDA 主题模型简介及 Python 实现_阿丢是丢心心的博客-CSDN 博客
[3]Python 文本处理_python 文本处理_noobiee 的博客-CSDN 博客