单位代码
学　号　SY2303806
分类号

北京航空航天大學
BEIHANG UNIVERSITY

基于金庸小说数据集的 LDA 主题提取与分类
第二次课后作业

院（系）名称　自动化科学与电气工程学院
专　业　名　称　自动化
学　生　姓　名　芦川川

2024 年 05 月

芦川川 自动化科学与电气工程学院 sy2303806@buaa.edu.cn

目录

1 内容介绍

从给定的语料库中均匀抽取 1000 个段落作为数据集（每个段落可以有 K 个 token, K 可以取 20，100，500, 1000, 3000），每个段落的标签就是对应段落所属的小说。利用 LDA 模型在给定的语料库上进行文本建模，主题数量为 T，并把每个段落表示为主题分布后进行分类（分类器自由选择），分类结果使用 10 次交叉验证（i.e. 900 做训练，剩余 100 做测试循环十次）。

实现和讨论如下的方面：

（1）在设定不同的主题个数 T 的情况下讨论其分类性能有无变化；

（2）讨论以"词"和以"字"为基本单元下分类结果的差异

（3）分析不同的取值的 K 的短文本和长文本对主题模型性能的影响

2 实验原理

2.1.1 简介

一篇文档应该有多个主题，每个主题的比例不同，每一个主题下面也应该有很多词语，每个词语的比例也不同。主题模型就是用数学框架来体现出文档的这种特点，主题模型自动分析每篇文档，统计文档内的词语，根据统计的信息来断定当前文档含有哪些主题，以及每个主题所占的比例各为多少。从上面的定义

可以看出，主题模型其实主要在学习两个分布，文档-主题分布（doc-topic）和主题-词分布（topic-word）。既然是分布就要满足两个条件，第一是非负性，第二是积分或者求和为1。也就是doc-topic矩阵或topic-word矩阵中，任意一行元素均为非负数且元素和为1。Topicmodels主要可以分为四大类：

1.无监督无层次结构,主要有：PLSA(Hofmann1999)，LDA，Correlated Topic Model，CTM主要是为了克服标准LDA模型不能建模话题在文档中出现的相关性的缺点，将LDA中文档话题分布服从的Dirichlet分布改为Logistic正态分布。例如CTM论文中举的一个例子是在Science杂志语料中，一篇遗传学文章很可能也跟健康和疾病有关，但是却不大可能跟射线天文学有关。因为Logistic正态分布不再是Multinomial分布的共轭分布，因此模型的解变得更加复杂。对此，作者使用的方法是，在变分推理的过程中，继续使用Taylor展开式以简化似然函数下界的复杂性。

2.无监督有层次结构,主要有：HLDA，HDP：标准 LDA 模型中话题的个数 K 需要已知，然而很多时候确定 K 的大小是一件困难的事情。HDP 能够根据数据自动确定 K 的大小。

3.有监督无层次结构,主要有：S-LDA，Disc-LDA，MM-LDA，Author-Model，LabeledLDA，PLDA 等。

4.有监督有层次结构,主要有：hLLDA，HSLDA。

除上述集中类型的话题模型外，还有一些半监督的话题模型，主要有：Semi-LDA,SSHLDA。

2.1.2 LAD 模型

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。LDA中文翻译为：潜在狄利克雷分布。LDA主题模型是一种文档生成模型，是一种非监督机器学习技术。它认为一篇文档是有多个主题的，而每个主题又对应着不同的词。一篇文档的构造过程，首先是以一定的概率选择某个主题，然后再在这个主题下以一定的概率选出某一个词，这样就生成了这篇文档的第一个词。不断重复这个过程，就生成了整篇文章(当然这里假定词与词之间是没有顺序的，即所有词无序的堆放在一个大袋子中，称之为词袋，这种方式可以使算法相对简化一些)。LDA的使用是上述文档生成过程

的逆过程，即根据一篇得到的文档，去寻找出这篇文档的主题，以及这些主题所对应的词。LDA是NLP领域一个非常重要的非监督算法。

所谓生成模型，就是说，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”这样一个过程得到。文档到主题服从多项式分布，主题到词服从多项式分布。

LDA是一种非监督机器学习技术，可以用来识别大规模文档集或语料库中潜藏的主题信息。它采用了词袋的方法，这种方法将每一篇文档视为一个词频向量，从而将文本信息转化为了易于建模的数字信息。但是词袋方法没有考虑词与词之间的顺序，这简化了问题的复杂性，同时也为模型的改进提供了契机。每一篇文档代表了一些主题所构成的一个概率分布，而每一个主题又代表了很多单词所构成的一个概率分布。

对于语料库中的每篇文档，LDA定义了如下生成过程：

1.对每一篇文档，从主题分布中抽取一个主题；

2.从上述被抽到的主题所对应的单词分布中抽取一个单词；

3.重复上述过程直至遍历文档中的每一个单词。

语料库中的每一篇文档与T（通过反复试验等方法事先给定）个主题的一个多项分布相对应，将该多项分布记为θ。每个主题又与词汇表中的V个单词的一个多项分布相对应，将这个多项分布记为φ。

先定义一些字母的含义：文档集合D，主题（topic)集合T

D中每个文档d看作一个单词序列<$w1,w2,...,wn$>，wi表示第i个单词，设d有n个单词。D中涉及的所有不同单词组成一个大集合，LDA以文档集合D作为输入，希望训练出的两个结果向量：对每个D中的文档d，对应到不同Topic的概率θd<$pt1,...,ptk$>，其中，pti表示d对应T中第i个topic的概率。计算方法是直观的，$pti=nti/n$，其中nti表示d中对应第i个topic的词的数目，n是d中所有词的总数。对每个T中的topict，生成不同单词的概率φt<$pw1,...,pwm$>，其中，pwi表示t生成VOC中第i个单词的概率。计算方法同样很直观，$pwi=Nwi/N$，其中Nwi表示对应到topict的VOC中第i个单词的数

目，N表示所有对应到topict的单词总数。LDA的核心公式如下：

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

直观的看这个公式，就是以Topic作为中间层，可以通过当前的θd和φt给出了文档d中出现单词w的概率。其中$p(t|d)$利用θd计算得到，$p(w|t)$利用φt计算得到。

实际上，利用当前的θd和φt，我们可以为一个文档中的一个单词计算它对应任意一个Topic时的$p(w|d)$，然后根据这些结果来更新这个词应该对应的topic。然后，如果这个更新改变了这个单词所对应的Topic，就会反过来影响θd和φt。

LDA算法开始时，先随机地给θd和φt赋值（对所有的d和t）。然后上述过程不断重复，最终收敛到的结果就是LDA的输出。再详细说一下这个迭代的学习过程：

1.针对一个特定的文档ds中的第i单词wi，如果令该单词对应的topic为tj，可以把上述公式改写为：

$$pj(wi\,|ds)=p(wi\,|tj)*p(tj\,|ds)$$

2.现在我们可以枚举T中的topic，得到所有的$pj(wi\,|ds)$，其中j取值1~k。然后可以根据这些概率值结果为ds中的第i个单词wi选择一个topic。最简单的想法是取令$pj(wi\,|ds)$最大的tj（注意，这个式子里只有j是变量）。

3.然后，如果ds中的第i个单词wi在这里选择了一个与原先不同的topic，就会对θd和φt有影响了（根据前面提到过的这两个向量的计算公式可以很容易知道）。它们的影响又会反过来影响对上面提到的$p(w|d)$的计算。对D中所有的d中的所有w进行一次$p(w|d)$的计算并重新选择topic看作一次迭代。这样进行n次循环迭代之后，就会收敛到LDA所需要的结果了。

3 数据处理

对金庸小说进行处理，从每本小说中均匀抽取段落并分词，使得每段可以有不同的 token，分词方法有按字分和按词分。处理得到 900 个段落作为训练集，100 的段落作为测试集，段落标签为对应的小说名。

4 实验结果

使用随机森林分类器，设置每段的token分别为20，100，500，1000；topic数量分别为 5，20，100，500，分别按字分词和结巴分词，共 32 次实验，每次

实验才用 10 次交叉验证，准确率结果如下表所示，其中表格横向为 topic，纵向为 token：

	5	20	100	500
20	0.1	0.09	0.1	0.1
100	0.11	0.14	0.13	0.16
500	0.24	0.29	0.47	0.42
1000	0.35	0.46	0.68	0.75

图表 1 按词分类结果

	5	20	100	500
20	0.12	0.14	0.14	0.16
100	0.14	0.32	0.37	0.35
500	0.42	0.74	0.82	0.77
1000	0.49	0.82	0.89	0.85

图表 2 按字分类结果

5 总结

可以在两个表格中整体看出，相同的 token 和 topic 的情况下，按字分类比按词分类准确率要高。

当增加 token 数量时，分类的准确率基本是不算增加的，因此每段的 token 数量有助于提高分类的准确率。

当 token 数过少时，改变 topic 数量基本不会对准确率有影响，当 token 数量超过一定数量，本次实验为 100 时，可以看到增加 topic 在 100 左右时准确率最高，当 topic 数量再高时，准确率会有所下滑。

6 参考文献

[1] https://blog.csdn.net/qq_33419476/article/details/126629289

[2] https://blog.csdn.net/weixin_42691585/article/details/113971857