**Report of Deep Learning for Natural Langauge Processing**

高志磊

gaozhilei@buaa.edu.cn

**Introduction**

从给定的语料库中均匀抽取200个段落，将从同一小说中抽取的段落组合到一起，形成16篇文档，作为训练集。将200个段落作为测试集。利用LDA模型进行文本建模，利用训练集训练出不同主题的词频分布，然后将200个段落表示为主题分布后进行分类，验证分类结果，并分析主题个数分类性能的变换以及分别以“词”和“字”为基本单元下分类结果的差异。

**Methodology**

**M1: LDA模型概述**

LDA模型是一种用于文本分析的概率模型，它最早由Blei等人在2003年提出，旨在通过对文本数据进行分析，自动发现其隐藏的主题结构。被广泛应用于文本挖掘、信息检索、自然语言处理等领域。

LDA模型的核心思想是将文本表示为一组概率分布，其中每个文档由多个主题混合而成，每个主题又由多个单词组成。LDA模型的基本原理是先假设一个文本集合的生成过程为：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。具体来说，LDA模型的生成过程包括以下三个步骤：

1）对一篇文档的每个位置，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。

**M2: LDA模型生成**

首先定义文档（doc）集合为Doc，文章主题（topic）集合为Topic，Doc中的每个文档doc可以看做一个单词序列，其中表示第个单词，假设一篇文档doc共有个单词。

Doc中的所有不同单词组成一个集合Voc，LDA模型以文档集合Doc作为输入，最终训练出两个结果向量，表示Topic中所有主题的数量，表示Voc中所有词的数量。

对每个文档doc，对应到不同topic的概率，其中表示doc对应Topic中第个topic中词的概率，计算方法如下：



表示doc中对应第个topic的词的数目，表示doc中所有词的总数。

对Topic中的第个topic，生成不同单词的概率，其中，表示第个topic生成Voc中第个单词的概率。计算方法如下：



其中表示对应到第个topic的Voc中第个单词的数目，表示所有对应到第个topic的单词总数。

LDA的核心公式如下：



直观的看这个公式，就是以Topic作为中间层，可以通过当前的和给出了文档doc中出现单词的概率。其中利用计算得到，利用计算得到。实际上，利用当前的和，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，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，就会反过来影响和。

下面给出一种LDA的学习过程：

LDA算法开始时，先随机地给和赋值（对所有的doc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中的第个单词，如果令该单词对应的topic为，可以把上述公式改写为：



2）现在我们可以枚举Topic中的topic，得到所有的，其中。然后可以根据这些概率值结果为中的第个单词选择一个topic。最简单的想法是取令最大的（注意，这个式子里只有是变量），即

3）然后，如果中的第个单词在这里选择了一个与原先不同的topic，就会对和有影响了（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的计算。对Doc中所有的doc中的所有进行一次的计算并重新选择topic看作一次迭代。这样进行次循环迭代之后，就会收敛到LDA所需要的结果了。

**M2: 程序实现**

（一）语料预处理。所用数据库为金庸小说，读取文本后需要进行一些处理包括：

1）删除开头无用信息；

2）删除中文停词及标点符号（利用cn\_stopwords.txt文件）；

3）删除空白符号，比如空格、换行符等。

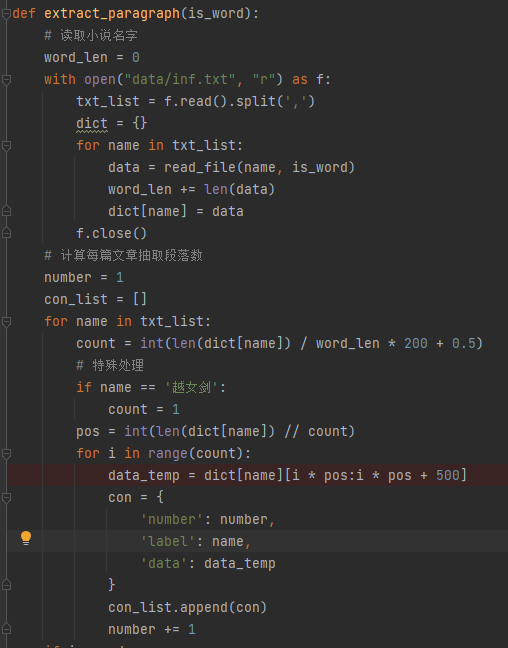
程序如下：



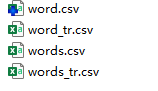
（二）段落抽取。题目要求从语料库中均匀抽取200个段落，每个段落大于500词，利用jiebe分词工具进行分词后，根据每篇小说的词数占总语料库词数的比例确定该篇小说应抽取的段落数，每篇小说抽取的段落数如下：



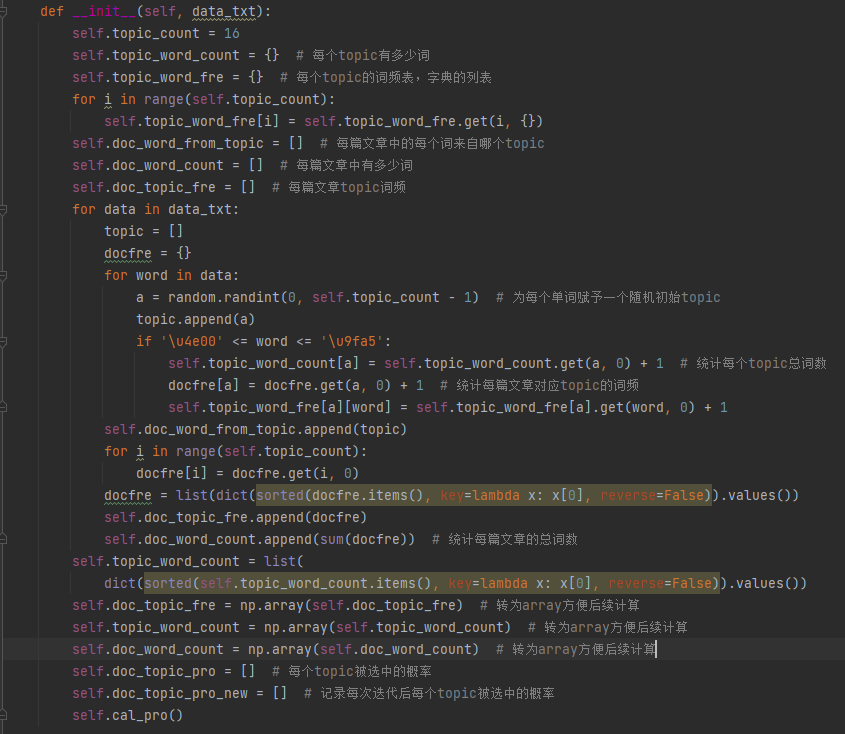
代码如下：



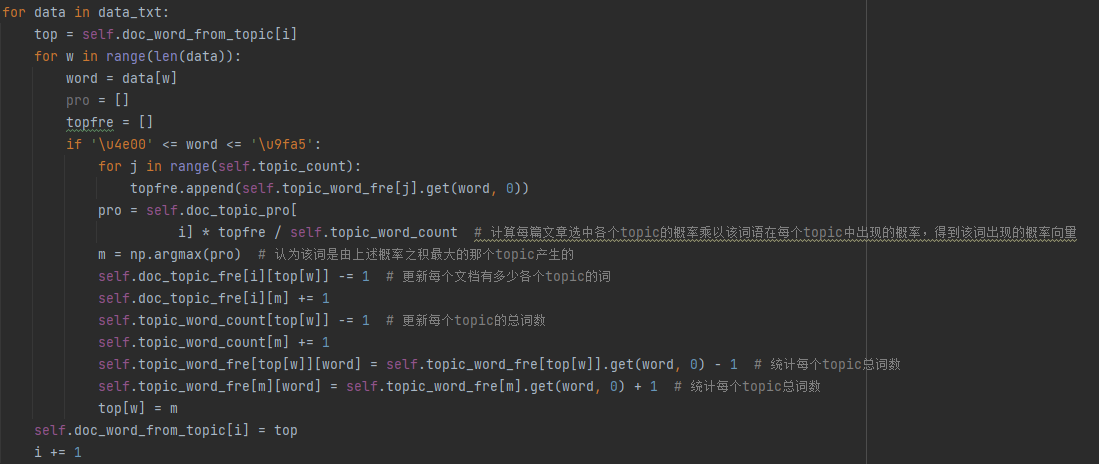
分别以词和字为基本单元进行处理，将训练集和测试集分别保存为如下文件：



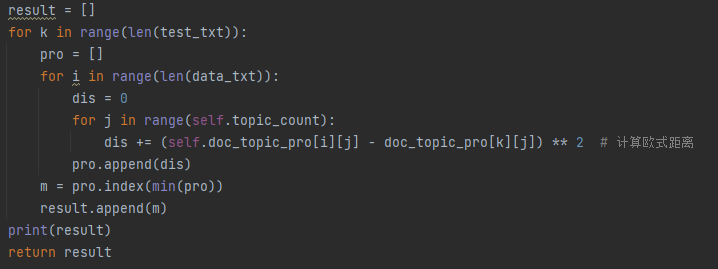
（三）初始化。将从每篇小说中抽取的所有段落组合到一起作为训练集，进行训练。首先为每篇文章中的词随机分配一个topic，然后统计每篇文章的topic频率以及每个topic的词频。



（四）训练。根据式计算每个topic下的词出现在这个位置的概率，然后更改该词的topic为概率最大的topic，随后进行文章topic频率以及topic词频的更新，当频率无变化时，说明算法已经收敛，迭代终止。



（五）测试。将200个段落利用训练好的topic词的分布计算段落的主题分布，并且利用欧氏距离选择该段落与哪一个样例更加接近，便认为该段落属于这一类。

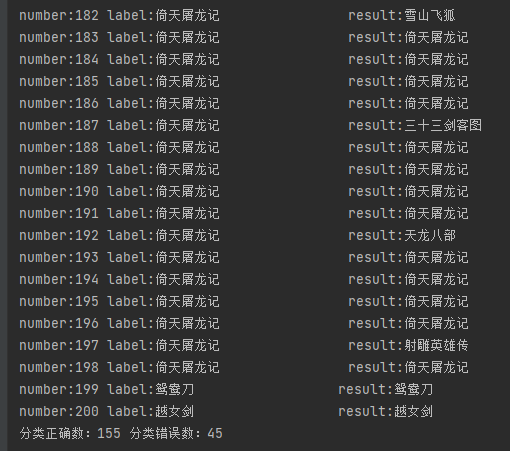


**Experimental Studies**

选择主题数量为16，200个段落从新组合成16篇文档，每篇文档主题分布如下：



计算200个段落的主题分布，随后进行分类，分类结果如下：



改变主题个数验证主题个数对结果的影响，由于初始化是随机初始化，每次运行结果不同，针对每个主题个数运行多次，结果如下：

Table 1：实验结果（以词为基本单元）

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 主题个数 | 正确分类个数 | 错误分类个数 | 正确率 | 平均正确率 |
| 16 | 155 | 45 | 77.5% | 79% |
| 153 | 47 | 76.5% |
| 166 | 34 | 83% |
| 50 | 186 | 14 | 93% | 92.6% |
| 184 | 16 | 92% |
| 186 | 14 | 93% |
| 100 | 191 | 9 | 95.5% | 96.7% |
| 193 | 7 | 96.5% |
| 196 | 4 | 98% |

从表中可以看出，随着主题个数的增多，分类结果正确率不断增加。

下面是以字为基本单元进行训练测试的结果：

Table 2：实验结果（以字为基本单元）

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 主题个数 | 正确分类个数 | 错误分类个数 | 正确率 | 平均正确率 |
| 16 | 106 | 96 | 52.48% | 50.5% |
| 92 | 110 | 45.54% |
| 108 | 94 | 53.47% |
| 50 | 145 | 57 | 71.78% | 73.93% |
| 154 | 48 | 76.24% |
| 149 | 53 | 73.76% |
| 100 | 171 | 31 | 84.65% | 85.97% |
| 178 | 24 | 88.12% |
| 172 | 30 | 85.15% |

**Conclusions**

根据表1可以看出，随着主题数量的增多，分类性能逐渐变好，相当于每篇文档比较的特征数目变多，比较效果更好。

根据表2可以看出，在相同主题数量的情况下，以字为基本单元的分类效果比以词为基本单位的分类效果差，这是因为针对不同主题，以字为基本单元的特异性较小，不同字之间的差异较小，而不同词的差异性较大，所以以词为基本单元分类效果更好。

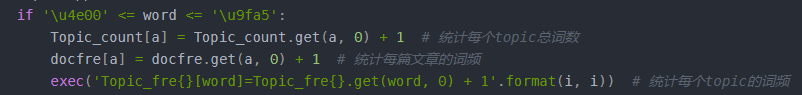
**References**

本文LDA初始化、训练及测试代码参考该博客内容：<https://blog.csdn.net/weixin_44966965/article/details/124556948>。

与原代码相比，本文做了如下改进：

1）采用类的结构进行重构，将训练、测试过程分开。

2）修改原文代码Bug，原代码初始化topic统计词频时统计的是固定topic的词频，即默认第一篇小说的词全部来自topic\_0。如图所示：



3）本文实现了主题数量可调，原代码主题数量恒等于语料库小说数量，存在一定问题。