TopicSearch设计与实现

> TopicSearch是一个用`python`编写的使用`LDA`算法搜索主题相关的文章的简单程序，使用`LDA`算法有三个好处：第一，搜索结果可以更加多样化，因为一个文章可能具有多个主题，相比用聚类方式对文章预先分类，使用`LDA`可以得到更好的查询效果；第二，尽管两个文章在词袋的级别上共享的关键字不多，但是可能描述的是同一个主题，使用`LDA`可以发现这种用户“心里想要的文章”；第三，当文章抽象成主题之后，就将相似性的匹配由原来几千几万量级的关键字降到了不超过`100`的主题比较，计算量也大大减少

概述

假设我们手上有了一些文章的集合，存储在`.txt`文件中，那么系统将经历以下处理过程：首先对文件中每一个文章都进行分词，分词结束后文章将变成`token`集合，处理掉停用词之后，将其转换为词袋形式，然后在将词频转换为`tf-idf`；至此，文章就转换为`lda`建模所需的语料了，这时的文章就是词条以及其对应的词频，词条之间的顺序不被考虑。接下来就是对语料进行`lda`训练，这个学习过程会得到两个结果：一是从语料中得到一系列的主题，二是每一个文章对应属于每一个主题的概率。之后便可以开始进行查询操作，得到一条查询语句，将查询语句映射到刚才建立好的主题空间，然后与所有文章的主题进行比较，计算与该查询语句欧式距离最小的前几个。

详细设计

- **分词程序：**使用`jieba`中文分词，结巴中文分词有两个主要的方式，一个是将语句切成各不相重合的词段，用于构建词袋，一个是将语句切成可能会重合的词段，也就是他会产生所有可能的词，主要用于搜索引擎查询
  接下来我们将要使用一个`Python`库，专门用于主题建模：`gensim`，这是由一个捷克的机器学习专家开发的
- **语料生成：**调用`gensim`提供的方法：`corpora.Dictionary.doc2bow()`
  他可以将分好词的文档转换为词袋（`bag-of-word`）形式，然后我们需要将词袋中对于的词频转换为`tf-idf`,使用这种转换是说，我们对于能够更好的区分文章的词条要进行加权，而不能很好的区分文章，甚至于几乎在每一篇文章中出现的（非停用词）需要降权，`gensim`也为我们提供了这个工具`models.TfidfModel`完成了这个转换工作
- **`LDA`建模**
  是时候解释一下什么是`LDA(Latent Dirichlet Allocation)`了，`LDA`是一种用于主题建模的非监督式学习方法，基本思想是将人认为概率的奴隶，我们写文章是在概率的支配下选择了一定的主题，而主题有会以一定的概率分布选择出很多单词，比

如“自然语言处理”属于“机器学习”的概率要比他属于“围棋”的概率高，“机器学习”这个主题在产生文章时相比“围棋”主题会更有更大的概率选择“自然语言处理”这个单词。这些单词被选出最终构成了文章。而`LDA`学习的过程就是已知很多文章的样本，从中学习出`k`个主题，类似于聚类，每一个主题事实上就是关于不同单词的概率，以及学习到每个文章属于每个主题的概率，然后就用这个<主题，概率>向量代替原来的文章。使用`models.ldamodel`完成整个建模的工作

- **程序的其他部分**
  这些部分主要与文件的操作有关，涉及到的文件主要有这些：保存语料对象的`corpus`文件，保存`lda`模型对象的`lda`文件，保存`tfidf`模型对象的`tfidf`文件以及保存系统初始化状态的`init`文件。我们直接使用`Python`提供的`pickle`模块将对象序列化并持久化到对应文件中