华为codeTopics说明文档

#### 基本使用

* maven项目，直接通过pom.xml文件作为入口点导入项目。
* F：\ \target\_projects\ \ ... 目录下存放主题建模的目标对象，比如5gCore，项目中每个微服务对应其中一个文件目录，根据目录判定某个微服务下有哪些文件。
* 首先把test目录下的example文件删掉，这个文件里的内容就是主题模型在数据预处理、模型训练过程中得到的中间结果文档。要重新一步一步运行项目，需要把已存在的结果文档删除，才能重新运行。
* 运行FinalTestForMicroTopic.java文件，得到微服务关注点数据。

#### 执行流程及核心逻辑

注：可直接运行FinalTestForMicroTopic.java文件一次性完成分词预处理、模型训练和关注点提取与输出，运行该文件只需要等待程序执行完毕即可。下面介绍的是支撑FinalTestForMicroTopic.java执行的三个核心方法。其中preProcess()方法负责数据预处理。ldaProcess()方法负责主题模型训练，耗时较大。postProcess()方法负责建立微服务与关注点的映射关系，进而输出各个微服务关注点信息。在实际运行过程中，若仅修改postProcess()方法，则不需要执行前两个方法，节省时间。FinalTestForMicroTopic.java执行逻辑如下图所示。

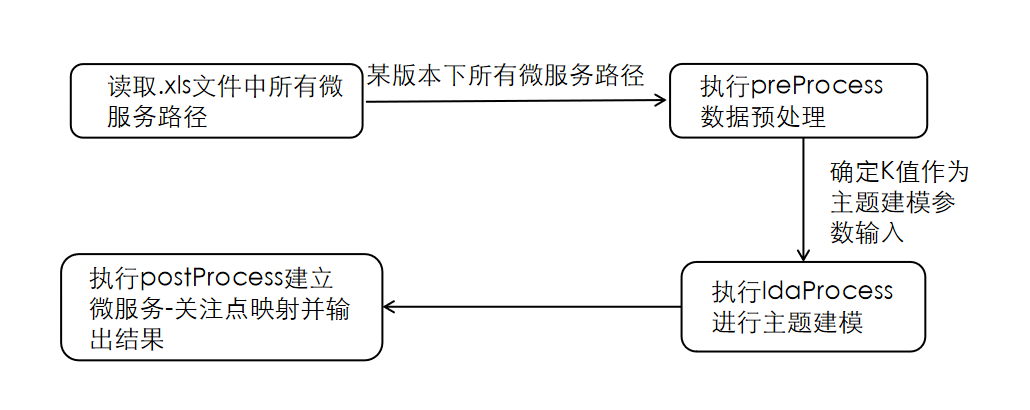


图1 FinalTestForMicroTopic.java执行逻辑

* 首先运行preProcess()方法。该方法主要是对每个java文件进行分词、统一转小写、去停用词、去除噪音、对词性进行统一等操作，得到的是一个一个数组，每个数组存放一个文件里面各个清洗好的分词。
  + 输入：项目源代码文件，扫描得到所有.java .go等结尾的文件，案例项目扫描得到117个文件
  + corpus 传入源代码路径（taget\_projects目录下）及过滤器得到项目内所有文件，
  + 
  + 输出：得到.dat文件是预处理后的分词结果（lda阶段要用到），.flist是对应的文件路径（postprocess阶段要用到），.dat中每一行对应.flist中每一行路径的文件
* 然后运行ldaProcess()方法,该方法输入是preprocess()方法清洗好了的分词，输出是一个model，这个model有多个属性，可自行选择model中哪些需要的属性进行输出。model是业内统一定义的一个LDA模型的标准输出的数据结构，对于model具体有哪些属性内容可自行百度。
  + 输入：第一步预处理后得到的分词集合，主要通过引入sockeshop.dat文件（这是test1()的输入）test2()的输入主要是test1()训练完成的model-final模型
  + 输出：主要输出内容看三个文档（只看-final的文档，带数字的都是中间结果，-final的文档才是所有跑完了的结果）
    - model-final.theta
    - 每行代表一个文档的所有主题的权重，目前给每个文档预设20个主题，故每行都有20列。每个主题在该文档内的权重都不一样，可选取权重大的主题作为当前文档的核心主题。如第i列权重最大，则第i个主题是当前文档的核心主题。其中i对应model-final.twords文档中各个主题的标号
    - model-final.twords
    - 有20个主题，以及每个主题下各个分词的权重，选取了每个topic下前50个权重最大的词
    - model-final.phi
    - 文件中所有词经过预处理后得到的不重复且有用词的集合，这些词在每个topic下的权重组成一个矩阵，行数为topic数，列数为所有词集合的长度。.twords为.phi文档中有用信息提取的结果。
* 然后运行PostprocessTest.java中的test()方法，主要对主题进行筛选以及对每个主题下相关联的微服务/文件列表。其中TC值用于区分当前主题是一个功能性主题还是一个噪声主题。
  + 输入：①根据sockshop.flist文件得到的模型训练的**文件路径顺序列表**；②各个微服务下的文件集合的Map集合serviceFiles;③和test2()一样拿到某一次训练的模型model。
  + 注意点：在输入数据中，必须要保证sockshop.flist中的文件路径字符串与relations.json中文件路径字符串**保持一致**。由于relations.json是在mac电脑下爬取的路径，故均为左斜杠/,而在运行LDATest得到的sockshop.flist文件里路径均按windows命名习惯，也就是右斜杠\。后续由于都在windows下执行，故两个文件里都是右斜杠，但原来代码里的filename.split("/")[6];要改成filename.split(" \ \ \ \ ")[6];注意是四个右斜杠。后续自己运行的时候只需要把这句代码改一下就行了。
  + 输出：
  + 最终将所有功能性主题，该主题下的各关键词及词的权重，及该主题对应的TC值，以及与该主题相关联的微服务名/文件名。其中文件名是将与该主题关联的所有微服务下的所有文件名输出。第三步的输出结果封装在FunctionalTopic类中。