TF-IDF公式：

**作用：** 评估一字词对于一个文件集或一个语料库中的其中一份文件的重要程度。字词的重要性随着它在文件中出现的次数成正比增加，但同时会随着它在语料库中出现的频率成反比下降。

**概念： 词频 (term frequency, TF)** 指的是某一个给定的词语在该文件中出现的次数。

**逆向文件频率 (inverse document frequency, IDF)** 是一个词语普遍重要性的度量。某一特定词语的IDF，可以由总文件数目除以包含该词语之文件的数目，再将得到的商取对数得到。

**思想：** TFIDF实际上是：TF \* IDF

IDF：如果包含词条t的文档越少，也就是n越小，IDF越大（见后续公式），则说明词条t具有很好的类别区分能力。

**公式：**它的重要性可表示为：

 \mathrm{tf_{i,j}} = \frac{n_{i,j}}{\sum_k n_{k,j}}

      以上式子中 n_{i,j} 是该词 t_{i}  在文件d_{j}中的出现次数，而分母则是在文件d_{j}中所有字词的出现次数之和。

 \mathrm{idf_{i}} =  \log \frac{|D|}{|\{j: t_{i} \in d_{j}\}|}

其中|D|：语料库中的文件总数； |\{ j: t_{i} \in d_{j}\}| ：包含词语 t_{i} 的文件数目（即 n_{i,j} \neq 0的文件数目）如果该词语不在语料库中，就会导致被除数为零，因此一般情况下使用1 + |\{j : t_{i} \in d_{j}\}|

**StringToWordVector**

# 参数设置

|  |  |
| --- | --- |
| IDFTransform | 值为true时，把文档中单词出现的次数val（或者TFTransform转化后的值）转化为val∗Math.log(doc\_total\_num/doc\_num\_contain\_this\_word) |
| TFTransform | 值为true时，把单词出现的次数val转化为  Math.log(val+1) |
| attributeIndices | 设置文档实例的哪些属性被选择 |
| attributeNamePrefix | 属性名的前缀  比如，词“美丽”作为一个属性，该词生成的属性的属性名就是“attributeNamePrefix美丽” |
| doNotOperateOnPerClassBasis | 值为true时，“wordsToKeep”和“minTermFreq”是基于所有文档，而不是基于每个类的文档 |
| invertSelection | 值为true时，反选“attributeIndices”选择的属性 |
| lowerCaseTokens | 值为true时，所有单词被转化为小写。 |
| minTermFreq | 设置（有类别时，基于每一类）最小单词频率 |
| normalizeDocLength | 用于用文档长度对属性值进行归一化公式为：  attribute\_value∗AvgDocLength/docLength |
| outputWordCounts | 值为true时，记录单词在文档中出现的次数，而不是boolean值 |
| periodicPruning | 指定删减词典的频率 |

# input数据，设置数据格式

# batchFinished()，处理数据（Tokenzier，Stemming，Stopwords）

**Tokenzier**的作用就是对于一个长的String，遍历扫描一遍，按那些字符进行切分。 若需要按句子为单位进行切分，可以新建新的Tokenzier，选择- delimiters 为". ! \n ? "或中文句子结束符号“。！？”等

**stemmer:**像是中文的同义词转换，同一个此类的词包。

常用的Stemming算法：

LookUp算法： 最简单的算法，建一个lookup table，如 cats -> cat. 若查询的词是cats则返回cat。特点：简单易用，但table表量比较大。

Affix算法： ①Suffix-stripping 算法： 根据单词的时态规则将是时态词转换为词根。 如将以ed，ing，ly结尾的词去掉其后缀，生成词根。动词的时态大部分有规则，一部分没规则，对于没规则的可以建一个lookup表。二者相结合。

**stopwords:** 设置停用词，停用词包含在一个文件当中，文件中一行一个停用词，以#开头的行被当作注释处理。

useStoplist: 是否使用停用词，值为false时，不使用“stopwords”选项定义的停用词。

wordsToKeep: 设置（有类别时，基于每一类）保留多少个单词（每一类中，出现次数前wordsToKeep名的单词被保留）。

1、首先Tokenizer，根据单词分割符取出word。若默认的 WordTokenizer 采用" \r\n\t.,;:\'\"()?!" 等英文分割符。这一部会把"boy."等后面的‘.'句号或感叹号等都去掉。也可弥补IKAnalyzer中文分词不足。

2、然后Stemming，取出词根。

3、最后在轮到Stopwords。stopwords.is(word)做词根word在stopword的list中，则取出。应该不会对此单词进行map映射和在vector出现。filter.setStopwords(new File(stopwordfile)); 这样设置一下，就可以将stopwordfile中的word（一行一个）当成stopword啦。

# determineDictionary(); 统计计算（TF，IDF）

# 归一化

# output