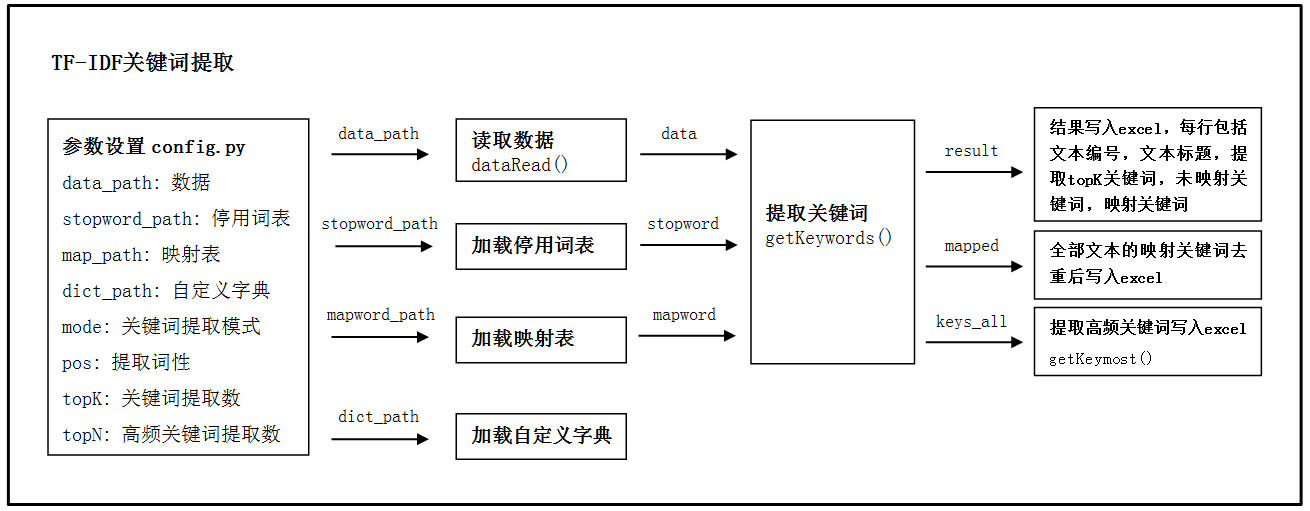
本文档主要针对基于TF-IDF算法的文本关键词提取流程进行说明。

TF-IDF关键词提取流程如图1所示。



**图1. TF-IDF关键词提取流程**

1、利用config.py对参数进行配置。

1）data\_path为文本数据所在文件夹路径；

2）stopword\_path为停用词表路径；

3）map\_path为映射表路径；

4）dict\_path为自定义字典路径；

5）mode为关键词提取模式，'tf' 或 'tfidf'；

6）pos为提取关键词词性，具体词性缩写见《jieba分词词性缩写》 ；

7）topK为每个文本提取的关键词个数；

8）topN为所有文本提取的高频关键词个数。

2、读取数据，加载停用词表、映射表、自定义字典。

1） 利用dataRead()读取数据。输入为data\_path，输出data，包含id、title、abstract三项内容；

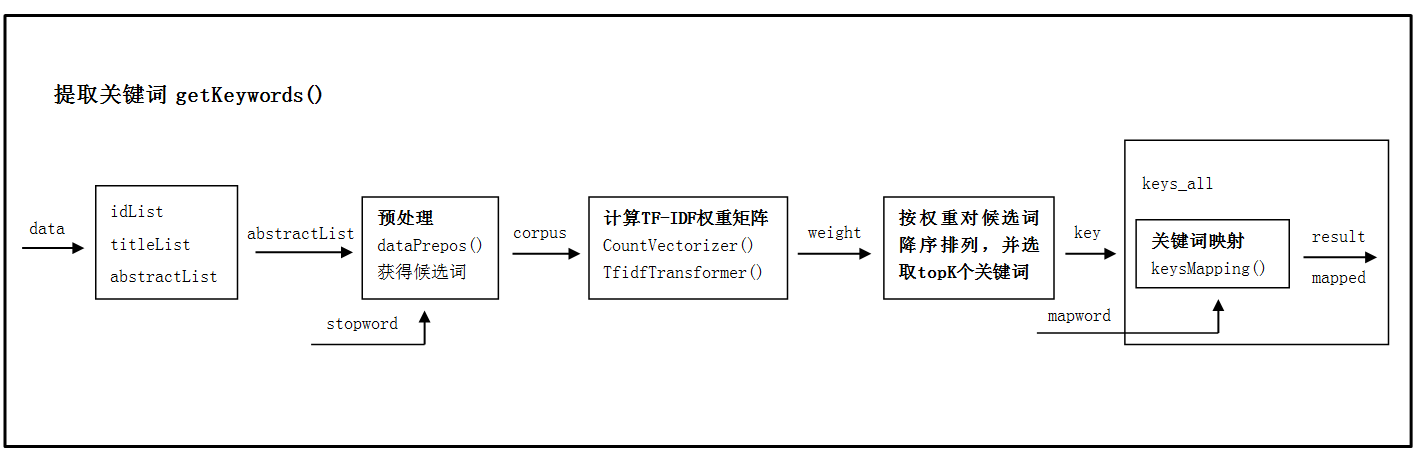
2） 加载停用词表，输出stopword；

3） 加载映射表，输出mapword；

4） 加载自定义字典，用于jieba分词。若使用jieba分词中自带字典，可将jieba.load\_userdict(dict\_path)注释掉。

3、提取关键词。

关键词提取模块如图2所示。



**图2.关键词提取模块**

1）若mode = 'tfidf'，则多个文本作为整体调用getKeywords()函数一次；若mode = 'tf'，则每个文本单独调用getKeywords()，需输入文本id；

2）data输入后，拆分为3个list，分别保存文本编号idList、文本标题titleList及文本内容abstractList；

3）利用dataPrepos()函数对每个文本进行预处理得到候选词。预处理包括jieba分词、去停用词、词性筛选。输入为每个文本的内容text及停用词表stopword，输出为所有文本的候选词corpus；

4）基于scikit-learn库中的CountVectorizer()和TfidfTransformer()函数，计算文本的TF-IDF矩阵weight；

5）将候选词按照weight降序排列，提取出每个文本的topK个词作为关键词key，并将所有文本的key合并至keys\_all，用于后续高频关键词提取；

6）利用keysMapping()函数对每个文本进行关键词映射。输入为关键词key以及映射表mapword，输出key\_left\_mapped包含映射得到的关键词以及未被映射的关键词。将所有文本映射得到的关键词合并至mapped (有重复)；

7）将所有文本的id、title、提取的topK关键词、映射得到的关键词以及未被映射的关键词以DataFrame形式写入result返回。

4、映射关键词去重，高频关键词提取及结果写入。

1）对getKeywords()输出的映射关键词mapped去重；

2）利用getKeymost()函数提取高频关键词。输入为keys\_all，输出为topN个高频关键词key\_most；

3）结果(result、mapped、key\_most)写入excel文档。