**文本关键词提取算法总结和Python实现**

Part I：简介

Part II：TF-IDF关键词提取

Part III：TextRank关键词提取

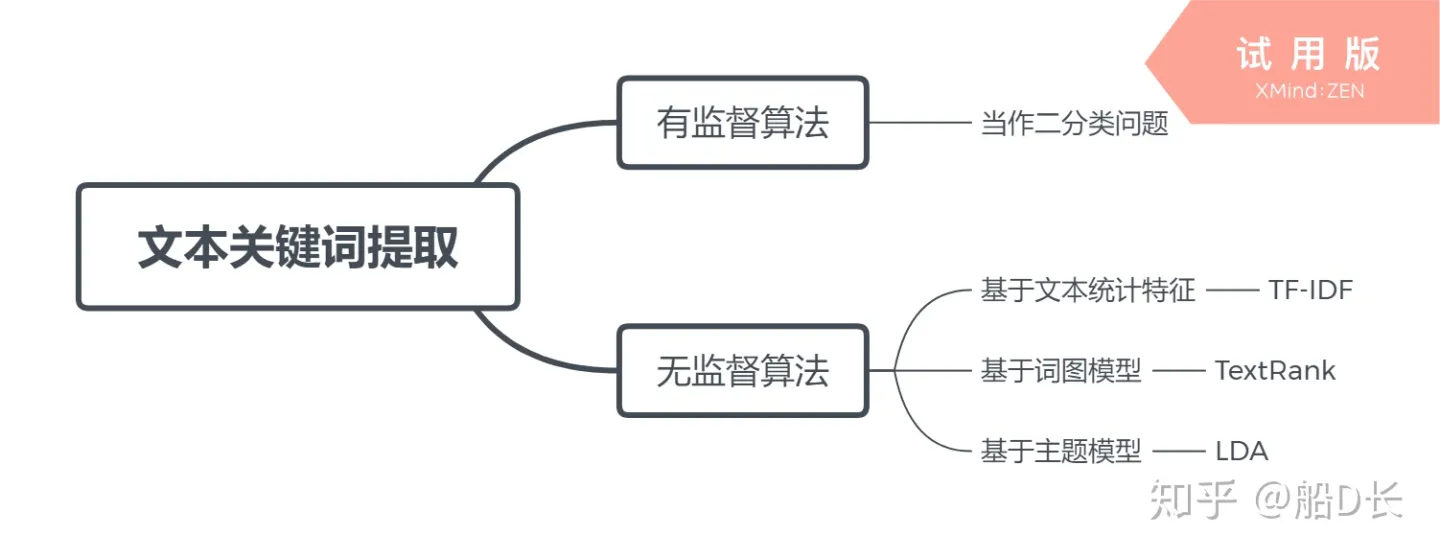
Part IV：算法实现

Part V：总结

背景：最近正在做的实验室项目需要对文本进行关键词提取，于是对关键词提取算法做了一定的调研，在这里总结一下。这篇文章首先会对关键词提取算法进行概括，介绍常用的TF-IDF算法和TextRank算法，最后结合Python jieba库的源码讲解算法的实现。

**Part I：简介**

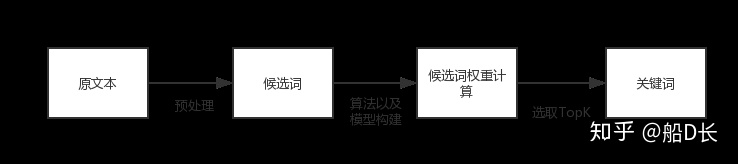
关键词提取是文本挖掘领域一个很重要的部分，通过对文本提取的关键词可以窥探整个文本的主题思想，进一步应用于文本的推荐或文本的搜索。



文本关键词提取算法大致分为有监督和无监督两种：

**有监督算法** 将关键词抽取问题转换为判断每个候选关键词是否为关键词的 **二分类问题** ，它需要一个已经标注关键词的文档集合训练分类模型。然而标注训练集非常费时费力，所以无监督算法更为常用。

**无监督算法** 不需要人工标注的训练集，利用某些方法发现文本中比较重要的词作为关键词，进行关键词抽取。词重要性的衡量有多种方式： **基于文本统计特征、基于词图模型和基于主题模型** ，TF-IDF、TextRank和LDA分别是这几种不同方式的代表。无监督的文本关键词抽取流程如下：



**Part II：TF-IDF关键词提取**

TF-IDF是关键词提取最基本、最简单易懂的方法。判断一个词再一篇文章中是否重要，一个容易想到的衡量指标就是词频，重要的词往往在文章中出现的频率也非常高；但另一方面，不是出现次数越多的词就一定重要，因为有些词在各种文章中都频繁出现，那它的重要性肯定不如哪些只在某篇文章中频繁出现的词重要性强。从统计学的角度，就是给予那些不常见的词以较大的权重，而减少常见词的权重。IDF（逆文档频率）就是这样一个权重，TF则指的是词频。TF和IDF计算公式如下：

词频（TF）=某个词在文章中出现的次数文章的总词数 词频（TF）=\frac{某个词在文章中出现的次数}{文章的总词数}

逆文档频率（IDF）=log（语料库的文档总数包含该词的文档数+1） 逆文档频率（IDF）=log（\frac{语料库的文档总数}{包含该词的文档数+1}）

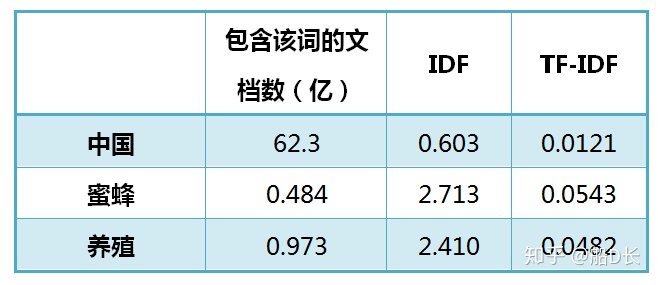
一个词IDF值的计算是根据语料库得出的，如果一个词在语料库中越常见，那么分母就越大，IDF就越小越接近0。分母之所以要加1，是为了避免分母为0（即所有文档都不包含该词）。

最终得到TF-IDF值：

TF−IDF=词频（TF）×逆文档频率（IDF） TF-IDF=词频（TF）\times 逆文档频率（IDF）

可以看出 **TF-IDF与一个词在文档中的出现次数成正比，与该词在整个语料库中出现次数成反比。一个词的TF-IDF值非常高，说明这个词比较少见，但是它在这篇文章中多次出现，那么这个词就非常可能是我们需要的关键词。**

引用阮一峰前辈的文章 [TF-IDF与余弦相似性的应用（一）：自动提取关键词 - 阮一峰的网络日志](https://link.zhihu.com/?target=http%3A//www.ruanyifeng.com/blog/2013/03/tf-idf.html) 中的例子：

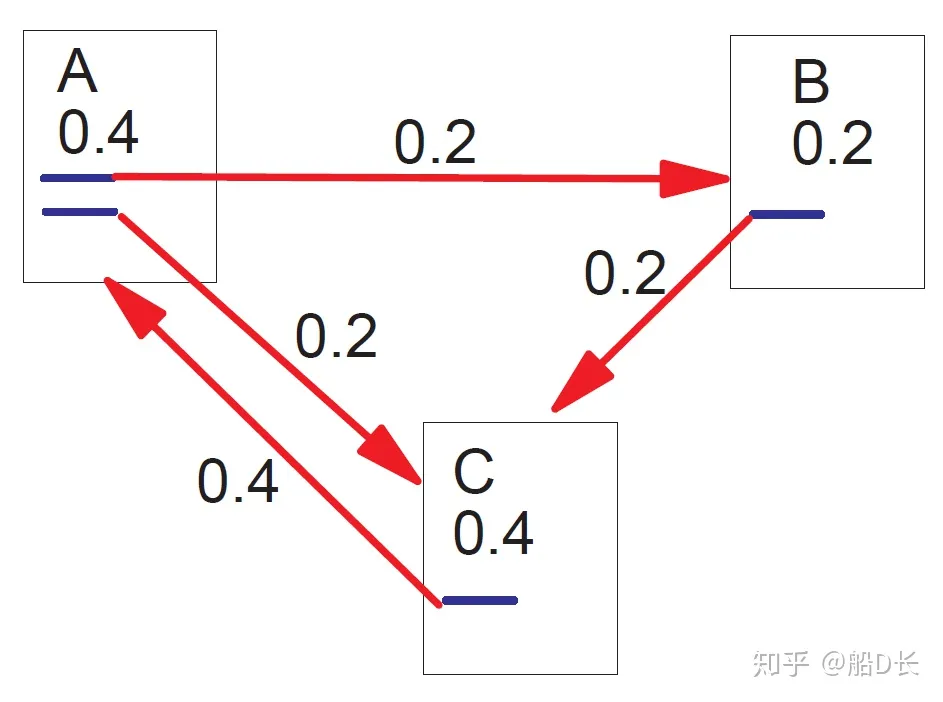


以文章《中国的蜜蜂养殖》为例，“蜜蜂”和“养殖”两个词的TF-IDF值都非常高，作为这篇文章的关键词实际上看也是非常合适的。另外“中国”这个词虽然在文章中的词频并不低“蜜蜂”和“养殖”低，但因为它在整个语料库中经常出现，导致IDF值非常低，所以不会作为文章的关键词。

**Part III：TextRank关键词抽取**

TextRank从词图模型的角度寻找文章的关键词，它的基本思想来源于大名鼎鼎的PageRank算法，PageRank算法是整个Google搜索的核心算法，通过网页之间的链接计算网页的重要性。首先介绍一下PageRank算法：

PageRank算法将整个互联网看作一张有向图，网页是图中的节点，而网页之间的链接就是途中的边。根据重要性传递的思想，如果一个网页A含有一个指向网页B的链接，那么网页B的重要性排名会根据A的重要行来提升。网页重要性传递思想如下图：



PageRank简单描述（来自PageRank论文）

在PageRank算法中，对于网页初始重要值（PR值）的计算非常关键，但是这个值无法预知，于是PageRank论文中给出了一种迭代算法求出这个PR值：为每个网页随机给一个初始值，然后迭代得到收敛值，作为网页重要性的度量。

PageRank求网页i的PR值的计算公式如下：

S(Vi)=(1−d)+d×∑j∈In(Vi)1|Out(Vj)|S(Vj) \(S(V\_i)=(1-d)+d\times \sum\_{j\in In(V\_i) }^{ }\frac{1}{|Out(V\_j)|}S(V\_j)\)

其中，d为阻尼系数，通常为0.85， In(Vi) \(In(V\_i)\) 是指向网页i的网页集合， Out(Vj) \(Out(V\_j)\) 是指网页j中的链接指向的集合， |Out(Vj)| \(|Out(V\_j)|\) 指集合中元素的个数。

**TextRank在构建图的时候将节点由网页改成了词，网页之间的链接改为词之间的共现关系，实际处理时，取一定长度的窗，在窗内的共现关系则视为有效。** 计算公式修改如下：

WS(Vi)=(1−d)+d×∑j∈In(Vi)wji∑Vk∈Out(Vj)wjkWS(Vj) \(WS(V\_i)=(1-d)+d\times \sum\_{j\in In(V\_i) }^{ }\frac{w\_{ji}}{\sum\_{V\_k\in Out(V\_j) }{ }w\_{jk}}WS(V\_j)\)

迭代得到所有词的PR值之后，就可以根据PR值的高低对词进行排序，得到文本的关键词集合。

**Part IV：算法实现**

中文分词中非常常用的Python jieba库 [fxsjy/jieba](https://link.zhihu.com/?target=https%3A//github.com/fxsjy/jieba) 中提供了基于TF-IDF算法和TextRank算法的关键词提取，jieba库的源码明了易懂，所以这一章通过分析jieba库的源码介绍关键词提取算法的实现。

首先来看看jieba库的关键词提取的效果：



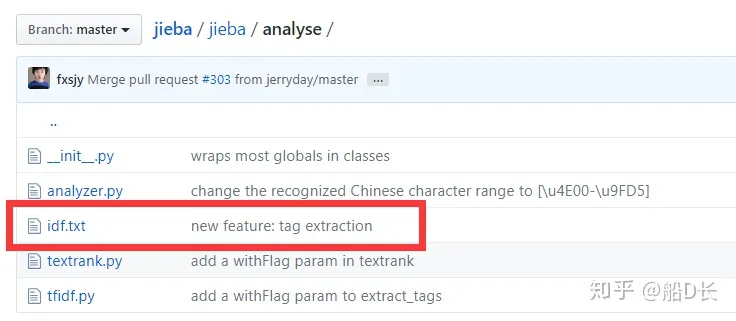
**1、jieba.analyse.extract\_tags(text)**

完整代码位于 [jieba/analyse/tfidf.py](https://link.zhihu.com/?target=https%3A//github.com/fxsjy/jieba/blob/master/jieba/analyse/tfidf.py)

我们看一下关键代码：

|  |
| --- |
| Plaintext def extract\_tags(self, sentence, topK=20, withWeight=False, allowPOS=(), withFlag=False): # （1）中文分词  if allowPOS:  allowPOS = frozenset(allowPOS)  words = self.postokenizer.cut(sentence)  else:  words = self.tokenizer.cut(sentence)  # （2）计算词频TF   freq = {}  for w in words:  if allowPOS:  if w.flag not in allowPOS:  continue  elif not withFlag:  w = w.word  wc = w.word if allowPOS and withFlag else w  if len(wc.strip()) < 2 or wc.lower() in self.stop\_words:  continue  freq[w] = freq.get(w, 0.0) + 1.0  total = sum(freq.values())  # （3）计算IDF  for k in freq:  kw = k.word if allowPOS and withFlag else k  freq[k] \*= self.idf\_freq.get(kw, self.median\_idf) / total  # （4）排序得到关键词集合  if withWeight:  tags = sorted(freq.items(), key=itemgetter(1), reverse=True)  else:  tags = sorted(freq, key=freq.\_\_getitem\_\_, reverse=True)  if topK:  return tags[:topK]  else:  return tags |

extract\_tags()函数将原始文本作为输入，输出文本的关键词集合，代码大致分为四个部分：（1）中文分词 （2）计算词频TF （3）计算IDF （4）将所有词排序得到关键词集合。重点关注一下词频TF和IDF的计算，（2）部分代码简历一个字典freq，记录文本中所有词的出现次数。（3）部分代码计算IDF，前文提到IDF需要通过语料库计算，jieba.analyse中包含一个idf.txt：



idf.txt中记录了所有词的IDF值：



当然你可以使用自己的语料库idf.txt，详见 [fxsjy/jieba](https://link.zhihu.com/?target=https%3A//github.com/fxsjy/jieba) 文档。

**2、jieba.analyse.textrank(text)**

完整代码位于 [jieba/analyse/textrank.py](https://link.zhihu.com/?target=https%3A//github.com/fxsjy/jieba/blob/master/jieba/analyse/textrank.py)

关键代码如下：

|  |
| --- |
| Plaintext def textrank(self, sentence, topK=20, withWeight=False, allowPOS=('ns', 'n', 'vn', 'v'), withFlag=False): # （1）构建词图  g = UndirectWeightedGraph()  words = tuple(self.tokenizer.cut(sentence))  for terms, w in cm.items():  g.addEdge(terms[0], terms[1], w)  # （2）迭代计算所有词的PR值  nodes\_rank = g.rank()  # （3）排序得到关键词集合  if topK:  return tags[:topK]  else:  return tags |

textrank()函数同样将原始文本作为输入，输出文本的关键词集合，代码大致分为三个部分：（1）构建词图：UndirectWeightedGraph()类 （2）调用UndirectWeightedGraph()类的rank()方法迭代计算所有词的PR值（3）排序得到关键词集合

实现的更多细节可以直接阅读jieba库关键词提取部分的源码，代码量很少，清晰易懂~

**Part V：总结**

关键词提取在文本挖掘领域有着非常广泛的应用，因为文本领域的不同，长文本和短文本的文本类型的不同，每种关键词提取方法的效果也不尽相同，实际应用中需要对多种方法进行尝试挑选最合适效果最好的方法。

参考文献：

[TF-IDF与余弦相似性的应用（一）：自动提取关键词 - 阮一峰的网络日志](https://link.zhihu.com/?target=http%3A//www.ruanyifeng.com/blog/2013/03/tf-idf.html)

[如何做好文本关键词提取？从达观数据应用的三种算法说起](https://link.zhihu.com/?target=https%3A//www.infoq.cn/article/text-keyword-extraction)

[“关键词”提取都有哪些方案？](https://www.zhihu.com/question/21104071)

[刘知远 《基于文档主题结构的关键词抽取方法研究》](https://link.zhihu.com/?target=http%3A//nlp.csai.tsinghua.edu.cn/~lzy/publications/phd_thesis.pdf)

[fxsjy/jieba](https://link.zhihu.com/?target=https%3A//github.com/fxsjy/jieba)