一、目的：

利用左右信息熵，互信息，改进的互信息，tfidf和规则发现新词，弥补分词工具的不足。

二、步骤：

基于互信息计算两个词的内部凝固程度并排序

基于左右信息熵计算一个词的自由运用程度（如果一个词左右搭配丰富的话更可能独立成

词）并排序

改进互信息计算两个词的内部凝固程度

计算tfidf，辅助过滤高频但不重要的词

综合各项指标，得出一个总的得分并排序。

三、大体思路如下：

基于互信息计算两个词的内部凝固程度并排序。基于左右信息熵计算一个  
词的自由运用程度并排序。计算TF-IDF，基于TF-IDF再算一个指标。基于依存语法规则找  
出新词。把得出的结果归一化到0到1之间，最后综合利用各个指标算出一个总的值。

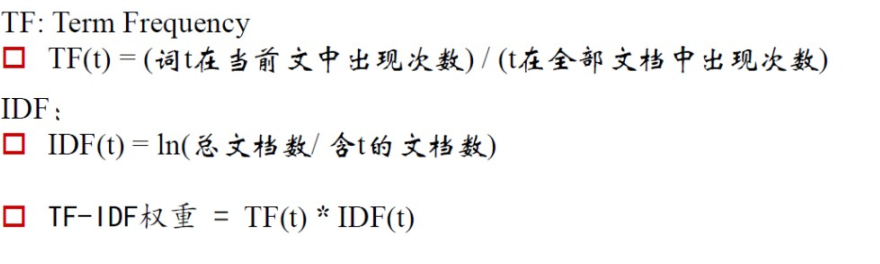
四、原理介绍：

1. 互信息，左右熵



举个例子，在一个2400万字的数据中，“电影”一共出现了2774次，出现的概率约为  
0.000113。“院”字则出现了4797次，出现的概率约为0.0001969。如果两者之间真的毫无  
关系，它们恰好拼在了一起的概率就应该是0.000113×0.0001969，约为2.223乘以10的–8  
次方。但事实上，“电影院”在语料中一共出现了175次，出现概率约为7.183乘以10的–6次  
方，是预测值的300多倍。所以电影和院不完全独立，当“电影”和“院”连续出现时，电影院  
更可能作为一个词，体现在上面的式子中就是互信息大。互信息越大往往一个词结合的就  
越紧密。说往往是因为有特例，如果词频，共现频次很低的情况下互信息会很大，但是不  
能直接反应组成短语的能力。比如“上党镇、小丁巷”只出现一次，“上党镇”只出现一  
次，“小丁巷”也只出现一次。这样算出来的互信息会很大，但这种组合只出现一次，过分  
强调了它的重要性。所以还需要对互信息改进。一个最简单的改进就是给联合概率P(X，  
Y)加一个乘方。或者直接过滤掉低频组合。两种方法我都试了，最后把他们两个的结果综  
合了一下。  
判断是否独立成词的另一个标准是所处语境的丰富程度。这可以用左右熵来衡量。比如  
说“中华人民共和国”中的，“人民共和”左边的搭配特别少，他不太可能独立成词。而“被  
子”左边的搭配特别丰富，如晒被子、盖被子、洗被子、一条被子等等，所以“被子”更可能  
可以作为一个独立的词。左右熵就是用来判断一个词左右的搭配是否丰富。

1. TF\_IDF  
   TF-IDF是一种统计方法，用以评估一个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



简单来讲，如果一个词在很多文档都会出现，比如“的”，那它可能是一个不太重要的词，  
此时IDF的值会变小。而某些专业性的词，如“民诉法”，出现这个词的文档很少，可能在某  
些文档中反复出现，那它对这几篇文章而言很可能是一个比较重要的词。  
以上是TF-IDF的原始的意思，简单来说就是求一个词对一篇文档的重要性。但我这边需要  
得到一个词对所有文档（一个案由）的重要性，这边对TD-IDF的结果做简单处理，有两种  
方案。第一种方案，对每个单词，求他在所有文档的TD-IDF的值的总和。第一种方案，对  
每个单词，求他在所有文档的TD-IDF的值的总和除以包含他的文档数。

五、具体实现：  
互信息左右熵的算法思路很简单，主要问题是当数据量比较大的时候计算速度和占用内存  
的问题。以计算右熵为例，比如有两个连在一起的词“中通快递”，我需要在所有文档中找  
出它右边所有可能的搭配。这样我就要把所有文档都遍历一遍。但是这只是计算一个词的  
右熵，对于所有的组合都这样计算的话相当于有两层大循环，比如有100万词汇量，那计  
算量就是100万平方，也就是1万亿。计算量太大了。所以需要改进。而且如果一次性读太  
多的数据会特别消耗内存。  
我的做法是用两层字典存储。用一个字典Dict存储总的结果。每次读一篇或者几篇文章，  
分词。如果新来的分词结果长度为n。我以第一个词和第二个次为一个整体，看这种组合  
有没有出现过，如果这种组合不在Dict中，那我在Dict中创建一个键值key（第一个词跟第  
二个次的整体），这个键对应的值还是一个字典，这个子字典存这种组合右边所有可能出  
现的情况，接着看第三个词在不在Dict[key]中。如果不在，Dict[key][ word]=1；如果在，  
Dict[key][ word]+=1。接着把第二个词和第三个词的组合看成一个整体，看他在不在Dict  
中。依次往后。这样的好处是每次可以只处理一点数据，不需要一次性读取所有数据，占  
用内存少。而且只用到了一层大的循环（只需要一个长度为3的滑动窗口把所有文档遍历  
一遍），新词来了之后只需要在字典中查找之前有没有存过，而字典查找是非常快的（也  
可以存成列表，但是列表搜索很慢，效率会大大降低）。而且对于重复出现的词只需要在  
字典中次数加1，不会增加额外的存储量。

我是用ltp做分词的，那个准确率高一点，缺点就是没有去停词，甚至连标点都没有去  
掉。但是标点符号也是有用的，我不需要考虑标点符号两侧的词语的组合。这样反而节省  
内存跟计算量。另外我把所有二元组合的情况存完之后，在计算互信息之前，对之前的结  
果做了一个预处理。去掉一些低频的二元组合。实际操作的时候是把频数小于3的二元组  
合全部去掉。不参与后面的运算。  
具体结果如下：  
香樟 嘉苑 12.165332244903269  
中誉 现代城 12.165332244903269  
三塘 南村 12.165332244903269  
妇幼 保健院 12.165332244903269  
“香樟 嘉苑”互信息最高，12.165332244903269分。但他实际上只出现3次。香樟和“嘉  
苑”没有单独出现过，导致他的互信息的值算出来很高。排名靠前的几个结果基本是类似的  
情况。如果不设置过滤词频小于3的组合这种情况会更严重。过滤词频小于3的组合之后，  
再用其他指标来制衡一下，最终得到一个中和的结果。  
存在的问题：  
1) 结果是建立在分词的基础上的，实际效果受到分词结果的影响。  
2) 左右熵受词频影响很大。频数高的组合其左右搭配往往也更丰富，导致左右熵会很大。  
而对于低频组合，他们的左右熵差距非常小。

2.TD-IDF  
我算TD-IDF的时候试了不同的方法，用ltp分词，结巴模块算TD-IDF。用结巴分词，结巴  
算TD-IDF，用ltp分词，自己写的程序算TD-IDF。  
ltp分词准确率高一点，但是它保留所有东西，不去停词，结果还要做一些预处理才能算  
TD-IDF。结巴支持在分词的时候去停用词，然后只要简单调整格式就可以直接算TDIDF。  
结巴自带的程序算TD-IDF确实快，比我自己写的程序快了一倍。  
TD-IDF做的部分结果如下，一是一个词在所有文档中的TD\_IDF值求和：  
被告 721.842400016  
原告 562.753604223  
离婚 201.64607766  
婚后 176.458305257  
夫妻感情 163.598556505

接下来把互信息，左右熵，TF-IDF，依存语法的结果做归一化处理，各自化简到0到1  
之间。  
其中归一化的时候我做了一些调整，有些指标极值很大，比如房屋租赁案由二阶词  
频“房屋 租赁”出现了次数最多，有24902次，而第10的“案件 受理费”只出现了4036次，如  
果直接归一化会导致最后的值集中在0.1以下，区分不开。我的做法是让前1%直接取1，  
后1%直接取0，中间98%的数据归一化，这样让中间的值有效分开。同时前1%本来就比  
较特殊，不用担心把后面的值放大之后影响到前1%词的排序。  
另外，TF-IDF求的是一个词对于一篇文章的重要程度，而我需要的是一个词对一个案由  
的重要程度，需要对TF-IDF的结果做简单处理，有两种方案。第一种方案，对每个单词，  
求他在所有文档的TD-IDF的值的总和。另一种方案，对每个单词，求他在所有文档的TDIDF的值的总和除以包含他的文档数，最后两种方案取个平均（如果只用第二种方案会出  
现很多人名，因为某些人名只在某些文档中出现，不在其它文档出现，这样他的TF-IDF很  
高）。  
然后把所有结果求和，其中一个二阶词频的TF-IDF的值用他的两个组成员的TF-IDF和  
代替，组成员的TF-IDF越大，这个值就越大，说明他越重要。  
最后得出一个总分，部分结果如下：  
夫妻 感情 3.183312543920108  
生育 儿子 2.138898438078408  
生育 女儿 2.099469583649713  
原告 父母 1.937929643152678  
夫妻 关系 1.753075122836605

六、总结：

单一指标效果有限，综合多个指标，效果会有所提升。由于没有标准答案对  
比，无法学习出各项指标前的最优权重。目前只是把各项均一化的值简单相加。另外在做  
的过程会遇到很多问题，比如结果中有乱码，那就要回去在最开始过滤乱码。发现结果中  
有很多人名地名公司名，但是这些都不需要，然后再返回去修改程序。  
下面的结果，Total是用语法过滤过的结果，稍微好点，但是推荐的词相对而言少一  
点，Total\_1另一种方案是不用语法过滤，推荐的词多，但是效果稍微差点。