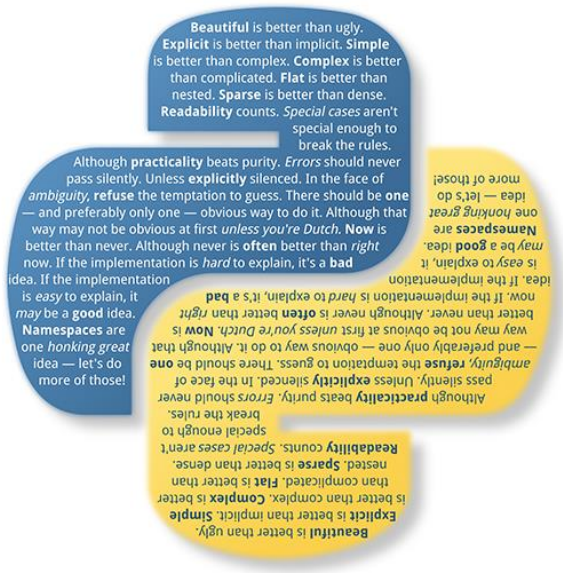
CDA数据分析师
CERTIFIED DATA ANALYST

CDA数据分析师就业班
之
文本分析快速入门



python™

韩要宾

CDA 数据分析师 CERTIFIED DATA ANALYST

目录
CONTENTS

自然语言处理的发展历程

01

自然语言处理的发展历程

从规则到统计

1950年

图灵发表了《计算的机器和智能》，提出让机器和人交流，如果人无法判断与其交流的是人类还是机器，就说明机器具有了智能，可以认为是NLP的源头

20世纪50-70年代

基于语法规则、词性、构词法等方面的自然语言处理，这是一段NLP里程中的“弯路”.

20世纪70年代后

从“规则”到“统计”，基于统计和模型的NLP产生了。.

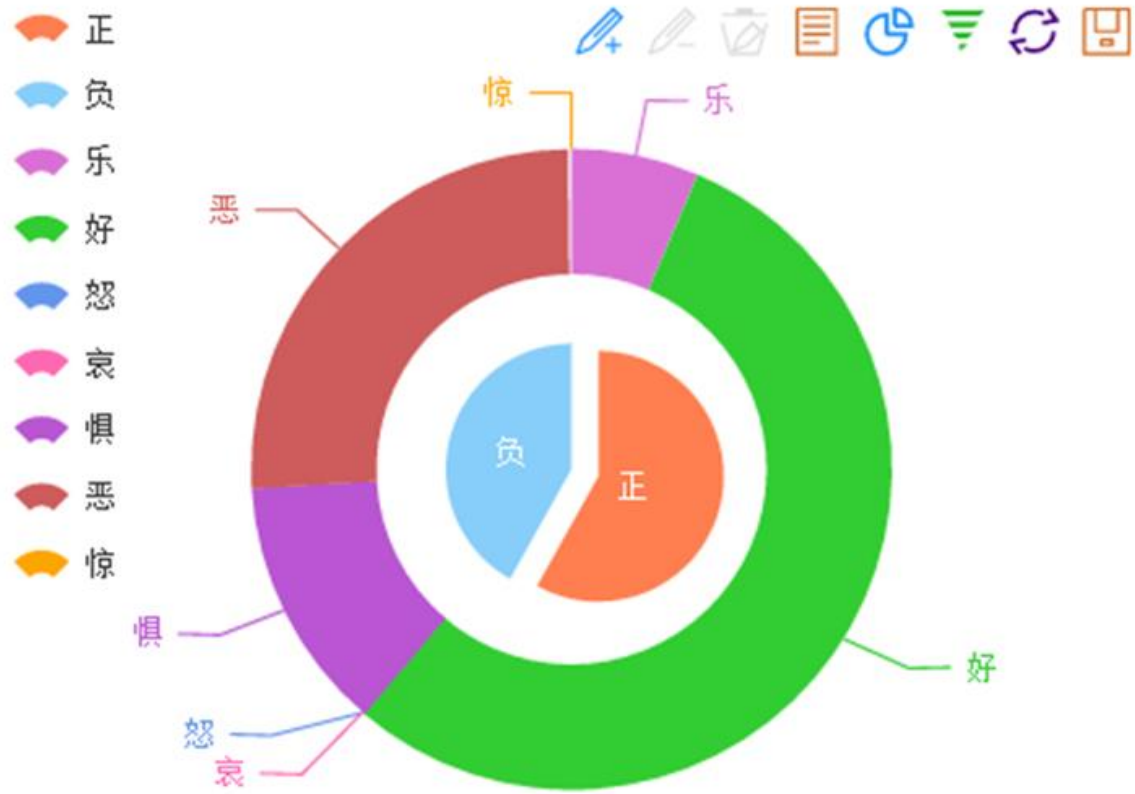
如今

基于神经网络、深度学习等，进行NLP.

02

文本分析的应用场景

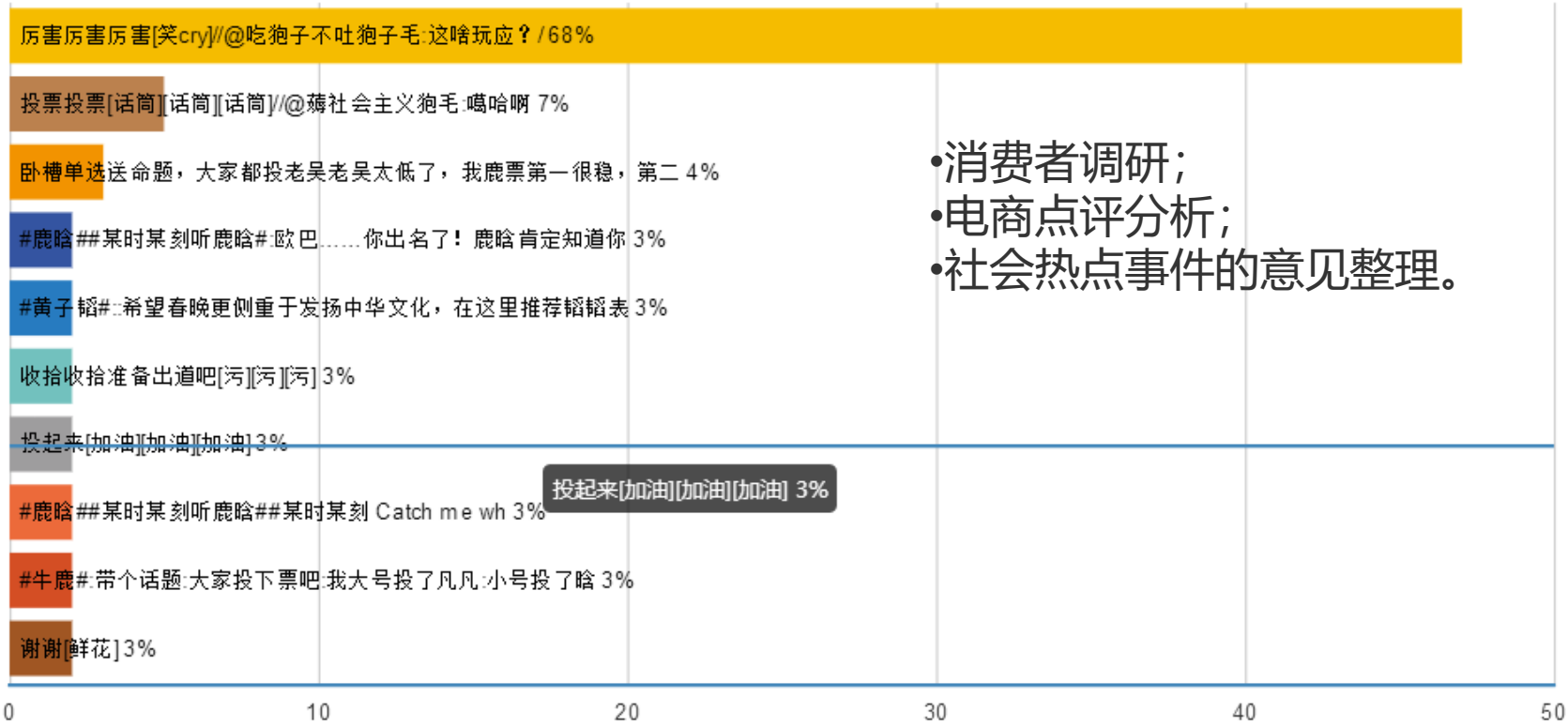
文本分析应用场景



文本分析应用场景

文本分析应用场景

网友观点分析

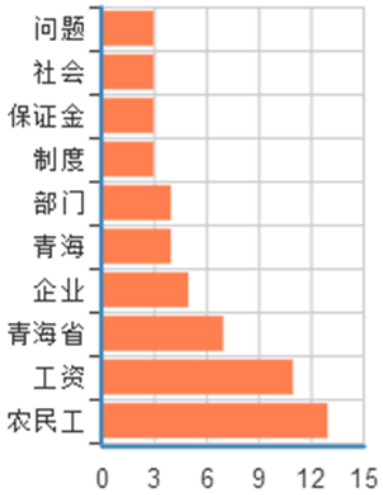


- 消费者调研；
- 电商点评分析；
- 社会热点事件的意见整理。

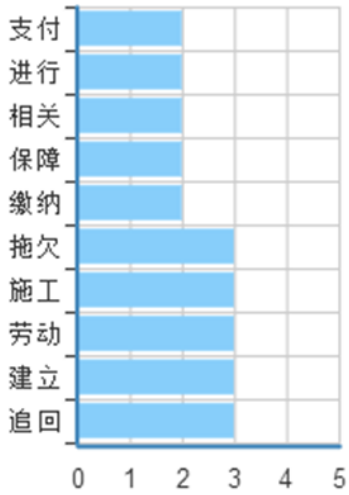
文本分析应用场景

文章关键词抽取

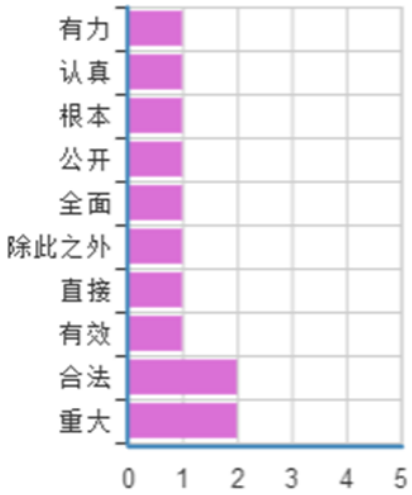
名词



动词



形容词



文本分析应用场景

用户评论关键词抽取

排名	热词	提及量	排名	热词	提及量
1	赢了	777	11	评论	180
2	美国	721	12	美国总统	168
3	希拉里	579	13	图片	156
4	总统	541	14	巴马	143
5	大选	432	15	揭晓	132
6	局座	238	16	初步	132
7	恭喜	229	17	疯子	120
8	当选	216	18	选举	120
9	害了	215	19	历史	119
10	中国	191	20	感觉	90

微博昵称	转发内容	转发数
新浪新闻	【美国大选结果初步揭晓 川普赢了！】美国总统选举初步结果揭晓，共和党总统候选人唐纳德·特朗普战胜民主党候选人、前国务卿希拉里·克林顿赢得总统选举，将成为美国第四十五任总统。美国大选决战日主角们都在干啥？美国人咋看大选？退休后奥巴马每年能挣多少钱？来看解读↓↓http://t.cn/RfwQ2rC	991
帅哥哪里走	#川普赢了#美国变天了，首位无从政经验的地产商人当选总统~~~	2
我是小可0	还是我们川普赢了😂😂😂😂	0
杜马列毛	川普赢了！以后美国官方语言就是四川话了，这是我大四川人民的胜利！	0
经旅青协小分队	#青协观天下#川普赢了希拉里😛 @江西财经职业学院青协	0

03

自然语言处理的基础

—分词

中文分词

分词属于自然语言处理的基础，无论进行文本分类，情感分析等，都需要将文字转换为数值，因为计算机无法识别文字，而将文字转换为数值的基础就是分词，将句子切分成为一个一个的词，然后使用word2vec等方法将语句转换为向量，实现**从语句到词语再到数值**的过程。所以说分词是NLP的一个基础任务。

中文分词

01

英文句子中的词语间有天然分割

英文语句中每个词之间有天然的空格分开，而中文没有

02

英文词语的构词法更加明显

英文很多次可以通过后缀看出词性，而中文不行

03

中文语义更加复杂

比如“前门到了请从后门下车”

中文分词的方法

最大匹配算法（正向/反向）

基于词典的分词算法是应用最广泛、分词速度最快的。很长一段时间内研究者都在对基于字符串匹配方法进行优化，比如最大长度设定、字符串存储和查找方式以及对于词表的组织结构，比如采用TRIE索引树、哈希索引等。

HMM、CRF

基本思路是对汉字进行标注训练，不仅考虑了词语出现的频率，还考虑上下文，具备较好的学习能力，因此其对歧义词和未登录词（没有被收录在分词词表中但必须切分出来的词，包括各类专有名词（人名、地名、企业名等）、缩写词、新增词汇等）的识别都具有良好的效果

双向匹配算法

神经网络、深度学习

常见的分词器都是使用机器学习算法和词典相结合，一方面能够提高分词准确率，另一方面能够改善领域适应性。

正向/逆向分词

正向最大匹配

我一个人吃饭（最大长度设置为5）

我一个人吃

我一个人

我一个

我一

我 ====》得到一个词-我

一个人吃饭

一个人吃

一个人

一个 ====》得到一个词-一个

人吃饭

人吃

人 ====》得到一个词-人

吃饭 ====》得到一个词-吃饭

结果是：**/我/一个/人/吃饭/**

逆向最大匹配

我一个人吃饭（最大长度设置为5）

一个人吃饭

个人吃饭

人吃饭

吃饭 ====》得到一个词-吃饭

我一个人

一个人

个人 ====》得到一个词-个人

我====》得到一个词-一 我

结果是：**/我/一/个人/吃饭/**

HMM分词

隐马尔可夫模型（HMM）：

隐马尔可夫模型（Hidden Markov Model，HMM）是统计模型，它用来描述一个含有隐含未知参数的马尔可夫过程。它的状态不能直接观察到，但能通过观测向量序列观察到，每个观测向量都是通过某些概率密度分布表现为各种状态，每一个观测向量是由一个具有相应概率密度分布的状态序列产生。

两个基本假设：齐次马尔可夫性假设（当前隐状态只依赖前一状态）、观测独立性假设（观测只依赖当前状态）。

HMM分词

隐马尔可夫模型有三个基本问题:

（记HMM模型为$\boldsymbol{\lambda=(A,B,\pi)}$，观测为$\boldsymbol{O}=(o_1,o_2,\dots,o_T)$，状态集合为$\boldsymbol{Q}$）

1、概率计算问题：给定模型$\boldsymbol{\lambda}$和观测$\boldsymbol{O}$,计算模型已知的情况下出现该观测序列$\boldsymbol{O}$的概率$\boldsymbol{P(O|\lambda)}$;

2、学习问题：已知观测序列$\boldsymbol{O}$，估计模型$\boldsymbol{\lambda=(A,B,\pi)}$的参数使得在该模型下出现观测序列$\boldsymbol{O}$的概率$\boldsymbol{P(O|\lambda)}$最大;

3、预测问题：也称解码问题，给定模型$\boldsymbol{\lambda}$和观测$\boldsymbol{O}$，求对给定观测需i额条件概率$\boldsymbol{P(I|\lambda)}$最大的状态序列$\boldsymbol{I}=(i_1,i_2,\dots,i_T)$，即对给定序列，求最可能的状态序列。

HMM分词

隐马尔可夫模型有三个基本问题:

（记HMM模型为$\boldsymbol{\lambda=(A,B,\pi)}$，观测为$\boldsymbol{O}=(o_1,o_2,\dots,o_T)$，状态集合为$\boldsymbol{Q}$）

1、概率计算问题：给定模型$\boldsymbol{\lambda}$和观测$\boldsymbol{O}$,计算模型已知的情况下出现该观测序列$\boldsymbol{O}$的概率$\boldsymbol{P(O|\lambda)}$;

2、学习问题：已知观测序列$\boldsymbol{O}$，估计模型$\boldsymbol{\lambda=(A,B,\pi)}$的参数使得在该模型下出现观测序列$\boldsymbol{O}$的概率$\boldsymbol{P(O|\lambda)}$最大;

3、预测问题：也称解码问题，给定模型$\boldsymbol{\lambda}$和观测$\boldsymbol{O}$，求对给定观测需i额条件概率$\boldsymbol{P(I|\lambda)}$最大的状态序列$\boldsymbol{I}=(i_1,i_2,\dots,i_T)$，即对给定序列，求最可能的状态序列。

本质上隐马尔可夫模型进行分词属于预测问题（解码问题）， 比如大家熟知的语音识别就是该模型的应用。

HMM分词

隐马尔科夫模型进行分词：

隐马尔可夫模型模型进行分词属于预测问题（解码问题），它的思想就是将句子中的每一个字看做是一个观测，而这个观测都来自与一个状态，状态共有4种：

B（词语的开头字）、E（词语的结束字）、M（词语的中间字）、S（单字成词）

状态转移其实只有8中：

(B to M),(B to E),(M to M) ,(M to E) ,(E to B) ,(E to S) ,(S to S) ,(S to B)

HMM分词

这里不对具体的计算过程做讲解，只了解思想，对于一个句子，当我们可以找到每一个字对应的状态，就得到了状态序列，比如一句话的状态序列是BEBEBMEBEBMEBES，分词只能在E或S处切割，所以分词后得到BE/BE/BME/BE/BME/BE/S，对应会原始句子相同位置进行切分即可。

HMM的训练过程（计算过程）需要一些概率论基础，有概率基础的同学可以参考如下资料：

http://blog.csdn.net/weixin_36604953/article/details/78653744

Python分词包——jieba

Jieba分词包的安装：

全自动安装：

easy_install jieba

或者 pip install jieba / pip3 install jieba

Python分词包——jieba

目前的分词技术已经非常成熟，有着优越的准确率和速度，对于python而言，有一个名为jieba的第三方库，可以很好的完成分词任务。以我们前面的文本为例，利用jieba进行分词。

```
349021 龟毛兔角 3 n
349022 龟毛兔角 3 n
349023 龟溪 3 ns
349024 龟玉 3 nz
349025 龟王 3 nz
349026 龟甲 92 ns
349027 龟甲胶 3 nz
349028 龟筮 3 n
349029 龟纹 3 n
349030 龟缩 29 v
349031 龟肉 3 n
349032 龟背 21 n
349033 龟背竹 3 n
349034 龟苓膏 3 n
349035 龟苗 3 n
349036 龟裂 34 v
349037 龟足 5 v
349038 龟鉴 2 n
349039 龟镜 3 nz
349040 龟鳖 3 n
349041 龟鹤遐寿 3 l
349042 龟龄鹤算 3 n
349043 龟龙片甲 3 nz
349044 龟龙麟凤 3 ns
349045 龠 5 g
349046 龢 732 zg
```

左图为jieba的词典，共349046（jieba-0.38版本）个词语，可以说是一个比较完整的词典了。

Python分词包——jieba

- jieba分词算法使用了基于前缀词典实现高效的词图扫描，生成句子中汉字所有可能生成词情况所构成的有向无环图(DAG)，再采用了动态规划查找最大概率路径，找出基于词频的最大切分组合，对于未登录词，采用了基于汉字成词能力的HMM模型

Python分词包——jieba

- 支持三种分词模式：
  - 精确模式，试图将句子最精确地切开，适合文本分析；
  - 全模式，把句子中所有的可以成词的词语都扫描出来，速度非常快，但是不能解决歧义；
  - 搜索引擎模式，在精确模式的基础上，对长词再次切分，提高召回率，适合用于搜索引擎分词。
- 支持繁体分词
- 支持自定义词典

Python分词包——jieba

- jieba.cut 方法接受三个输入参数: 需要分词的字符串；cut_all 参数用来控制是否采用全模式；HMM 参数用来控制是否使用 HMM 模型
- jieba.cut_for_search 方法接受两个参数：需要分词的字符串；是否使用 HMM 模型。该方法适合用于搜索引擎构建倒排索引的分词，粒度比较细
- 待分词的字符串可以是 unicode 或 UTF-8 字符串、GBK 字符串。注意：不建议直接输入 GBK 字符串，可能无法预料地错误解码成 UTF-8
- jieba.cut 以及 jieba.cut_for_search 返回的结构都是一个可迭代的 generator，可以使用 for 循环来获得分词后得到的每一个词语(unicode)，或者用
- jieba.lcut 以及 jieba.lcut_for_search 直接返回 list
- jieba.Tokenizer(dictionary=DEFAULT_DICT) 新建自定义分词器

Python分词包——jieba

Signature: jieba.cut(sentence, cut_all=False, HMM=True)

Docstring:

The main function that segments an entire sentence that contains Chinese characters into seperated words.

主要功能:将包含汉字的整个句子分割成分离的单词。

Parameter:

- sentence: The str(unicode) to be segmented.
- cut_all: Model type. True for full pattern, False for accurate pattern.
- HMM: Whether to use the Hidden Markov Model.

添加自定义词典

- 可以指定自己自定义的词典，以便包含jieba词库里没有的词。虽然jieba有新词识别能力，但是自行添加新词可以保证更高的正确率

- 用法： jieba.load_userdict(自定义词典的路径)

- 词典格式：一个词占一行；每一行分三部分，第一部分为词语，第二部分为词频（可省略），最后为词性（可省略），用空格隔开。

- 注意：自定义词典的文件为文本文件，且编码方式为utf-8。

04

词频分析

词云分析

词云图是一个很好的文本可视化途径，可以简单轻松的看到文本中高频词汇，从而推断出文本的主旨。Python可以轻松实现词云图绘制功能。一个好的词云图不应该直接进行绘制，应当对文本进行清洗，去除停用词、符号等无用元素。（Python的相关库中有内置的停用词，使用者也可以自己开发停词表）

词云分析

基本步骤：

1、对语料进行预处理

（1）分词

（2）统计词频

2、绘制词云

（1）设置词云参数（基本参数以及任意形状）

（2）绘制并展现词云

词云分析-WordCloud

安装词云包 wordcloud：

```
pip install wordcloud
```

测试代码：

```
from wordcloud import WordCloud
```

WordCloud参数属性

WordCloud类的属性

- font_path：string //字体路径
- width：int (default=400) //输出的画布宽度，默认为400像素
- height：int (default=200) //输出的画布高度，默认为200像素
- prefer_horizontal：float (default=0.90) //词语水平方向排版出现的频率，默认 0.9 （所以词语垂直方向排版出现频率为 0.1 ）
- mask：nd-array or None (default=None) //如果参数为空，则使用二维遮罩绘制词云。如果 mask 非空，设置的宽高值将被忽略，遮罩形状被 mask 取代。除全白（#FFFFFF）的部分将不会绘制，其余部分会用于绘制词云。
- scale：float (default=1) //按照比例进行放大画布，如设置为1.5，则长和宽都是原来画布的1.5倍。
- min_font_size：int (default=4) //显示的最小的字体大小
- font_step：int (default=1) //字体步长，如果步长大于1，会加快运算但是可能导致结果出现较大的误差。

WordCloud参数属性

- max_words：number (default=200) //要显示的词的最大个数
- stopwords：set of strings or None //设置需要屏蔽的词，如果为空，则使用内置的STOPWORDS
- background_color：color value (default=”black”) //背景颜色，如background_color='white',背景颜色为白色。
- max_font_size：int or None (default=None) //显示的最大的字体大小
- mode：string (default=”RGB”) //当参数为 “RGBA”并且background_color不为空时，背景为透明。
- relative_scaling：float (default=.5) //词频和字体大小的关联性
- color_func：callable, default=None //生成新颜色的函数，如果为空，则使用self.color_func regexp：string or None (optional) //使用正则表达式分隔输入的文本 collocations：bool, default=True //是否包括两个词的搭配
- colormap：string or matplotlib colormap, default=”viridis” //给每个单词随机分配颜色，若指定color_func，则忽略该方法。

05

文本主题提取

文本主题提取

文本主题提取即提取主题关键词。

能够体现文本内容主题的关键词就称之为主题关键词

文本主题提取的方法：TF-IDF模型

TF-IDF模型

核心思想：

如果某个词或短语在一篇文章中出现的频率TF高，并且在其他文章中很少出现，则认为此词或者短语具有很好的类别区分能力，适合用来分类。

TF-IDF模型

应用场景

- 1、计算出文章中每个词的TF-IDF值之后，进行排序，选取其中值最高的几个作为关键字。
- 2、计算出每篇文章的关键词，从中各选取相同个数的关键词，合并成一个集合，计算每篇文章对于这个集合中的词的词频，生成两篇文章各自的词频向量，进而通过欧氏距离或余弦距离求出两个向量的余弦相似度，值越大就表示越相似。

TF-IDF模型

实施步骤

1、文本语料的预处理

jieba分词中含有analyse模块，在进行关键词提取时可以使用下列代码

方法：jieba.analyse.extract_tags(sentence, topK,withWeight=True)

其中sentence为待提取的文本，topK为返回几个TF/IDF权重最大的关键词，默认值为20。

需要先import jieba.analyse，

2、使用TF-IDF权重来进行关键词获取，首先需要对文本构建词频矩阵，其次才能使用向量求TF-IDF值。