词云生成

<http://www.cnblogs.com/Yiutto/p/5998296.html>

**转载地址：https://zhuanlan.zhihu.com/p/20436581**  
上文末尾提到 Python 下还有一款词云生成器。[amueller/word\_cloud · GitHub](https://link.zhihu.com/?target=https%3A//github.com/amueller/word_cloud)

可以直接使用命令 pip install wordcloud 安装，自行补充完整依赖。

网上搜不到有关的中文文章，自己探索了一下，也算做出了结果。由于没有中文支持的说明和中文文档，在此简单补充并翻译部分文档。翻译可能有出入仅供参考，只是为了加深自己理解所用。

#后面有示例和展示。想了想放到一篇里面太臃肿，所以另起一篇展示。

[API Reference](https://link.zhihu.com/?target=http%3A//amueller.github.io/word_cloud/references.html)

API 分为三部分

生成词云的部分、生成图像颜色的部分和随机颜色函数。

**wordcloud.WordCloud**

|  |  |
| --- | --- |
| 1 | class wordcloud.WordCloud(font\_path=None, width=400, height=200, margin=5, ranks\_only=False, prefer\_horizontal=0.9,mask=None, scale=1, color\_func=<function random\_color\_func at 0x2b8b422a31b8>, max\_words=200, stopwords=None,random\_state=None, background\_color='black', max\_font\_size=None) |

主要部分，用来生成词云并生成图像。

**参数：**

**font\_path : string**

用到的字体文件的路径（支持OTF 或 TTF格式）。默认字体是   
DroidSansMono 。非 Linux 系统需自行调整路径参数。

说明：Windows 下中文支持的方法。

1、准备好一个中文字体，如 simhei.ttf

2、将其复制到 C:\Python27\Lib\site-packages\wordcloud\

3、在 wordcloud.py 中修改 FONT\_PATH 为相应字体。（注意是大写，大写的字体路径是默认路径，小写的字体是后面使用时可以再次指定的，会覆盖默认字体）

FONT\_PATH = os.environ.get("FONT\_PATH", os.path.join(os.path.dirname(\_\_file\_\_),

"simhei.ttf"))

4、注意中文的编码应为 utf-8 ，否则会导致乱码无法显示

#可以参见不知道什么语言的用户提出的 issue [Word cloud unicode (multilanguage) problem · Issue #70 · amueller/word\_cloud · GitHub](https://link.zhihu.com/?target=https%3A//github.com/amueller/word_cloud/issues/70)

**width : int (default=400)**

画布宽度

**height : int (default=200)**

画布高度

**ranks\_only : boolean (default=False)**

是否只用词频排序而不是实际词频统计值，默认 False

**prefer\_horizontal : float (default=0.90)**

词语水平出现的频率，默认 0.9 （即垂直出现频率为 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)**

计算与绘制图像间的比例。对于较大的词云图像，使用比例而非较大的画布会显著提升绘图速度，但是可能会造成词语间的粗糙拟合。

**max\_words : number (default=200)**

词语的最大数量

**stopwords : set of strings**

屏蔽词

**max\_font\_size : int or None (default=None)**

最大词的最大字号。如果不指定，则为图像高度。

说明：

画布越大，运行耗时会显著提升。如果你需要输出一幅较大的词云图像，尝试设置较小的画布并使用比例参数来调节。

基于最大字号和比例参数，算法可能会更倾向于使用词语的排名顺序而非它们的实际频率。

属性：

**words\_: list of tuples (string, float)**

带有频率的词语

**layout\_:list of tuples (string, int, (int, int), int, color))**

编码拟合好的词云。每个词语的字符串、字号、位置、方向和颜色的编码。

方法：

|  |  |
| --- | --- |
| 1 | static \_\_init\_\_(font\_path=None, width=400, height=200, margin=5, ranks\_only=False, prefer\_horizontal=0.9, mask=None,scale=1, color\_func=<function random\_color\_func at 0x2b8b422a31b8>, max\_words=200, stopwords=None,random\_state=None, background\_color='black', max\_font\_size=None) |

**static fit\_words(frequencies) / generate\_from\_frequencies**（这两个一个意思）

依据词语和频率生成词云

参数：frequencies : array of tuples

包含了词语和频率的数组

**static generate(text) / generate\_from\_text**

从文本中生成词云，调用 \_text 和 fit\_words

**static process\_text(text)**

将长文本分词并去除屏蔽词（此处指英语，中文分词还是需要自己用别的库先行实现，使用上面的 fit\_words(frequencies) ）

参数：text : string

需要处理的文本

返回：words : list of tuples (string, float)

带有相应频率的词语

说明：有标记词频的更好方法，但是作者没有把那些全部包括进来。

**static recolor(random\_state=None, color\_func=None)**

对现有输出重新着色。重新上色会比重新生成整个词云快很多。

参数：

random\_state : RandomState, int, or None, default=None

如果非空，会使用固定的随机状态。  
如果给出一个整数，它会被用作 random.Random 状态的一个种子（根源）

**color\_func : function or None, default=None**

生成新颜色的函数，如果为空，则使用 self.color\_func

**static to\_array()**

转化为 numpy array

返回：image : nd-array size (width, height, 3)

numpy 矩阵的词云图像

**static to\_file(filename)**

输出到文件

参数：filename : string

输出的文件路径

**wordcloud.ImageColorGenerator**

|  |  |
| --- | --- |
| 1 | class wordcloud.ImageColorGenerator(image) |

基于 RGB 图像生成颜色。词语将会使用一定矩形区域内的平均颜色上色。这个对象可以被调用作为词云或者重新上色方法的颜色函数。

参数：image : nd-array, shape (height, width, 3)

用来生成词语颜色的图像。 Alpha 通道会被忽略。大小应与画布大小相同。

**static \_\_init\_\_(image)**

**wordcloud.random\_color\_func**

|  |  |
| --- | --- |
| 1 | wordcloud.random\_color\_func(word=None, font\_size=None, position=None, orientation=None, font\_path=None, random\_state=None) |

随机生成色调

默认的上色方法。仅仅会生成一个随机的色调（ 80% 颜色值 50% 亮度）

参数：

word, font\_size, position, orientation : ignored.

random\_state : random.Random object or None, (default=None)

如果一个随机的对象已经给出，这会用作生成随机数字（没看懂）