



Python与金融数据挖掘(5)

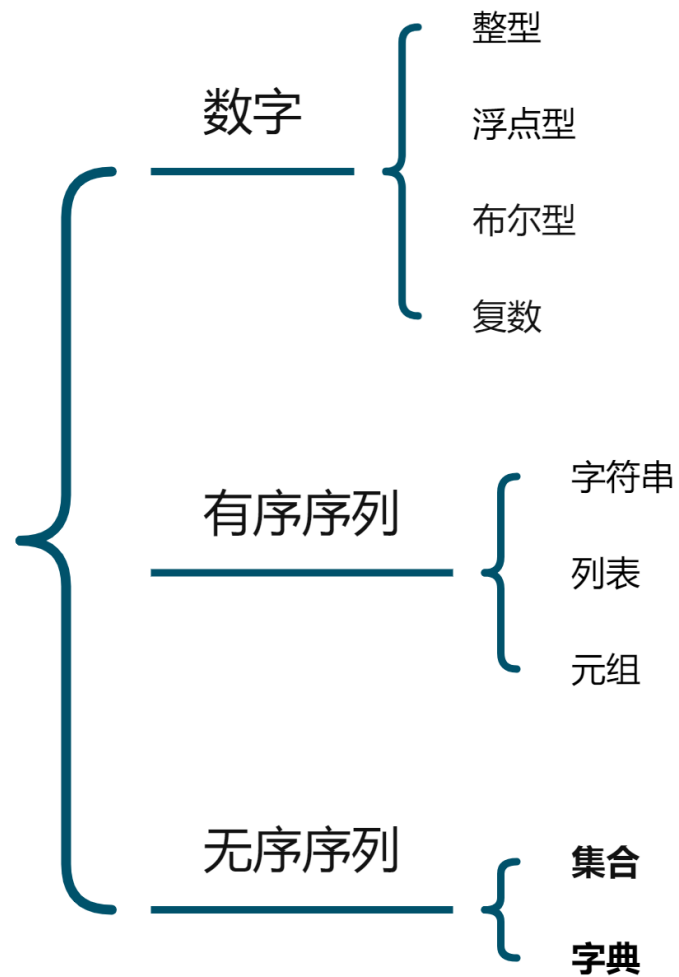
文欣秀

wenxinxiu@ecust.edu.cn



数据类型

Python数据类型



二十大报告词云案例

```
import jieba
import matplotlib.pyplot as plt
from wordcloud import WordCloud
from imageio.v2 import imread
fobj=open("二十大报告.txt","r",encoding="utf-8")
txt=fobj.read()
words=jieba.lcut(txt)
```

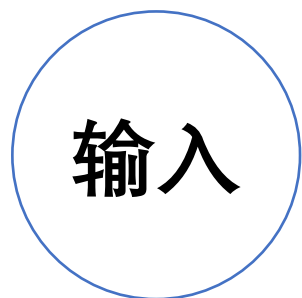

二十大报告词云案例

```
pic = imread('cloud.jpg')
wc=WordCloud(mask=pic,font_path='msyh.ttc', #中文字体
              repeat=False, background_color='white', #设置背景颜色
              max_words=110, max_font_size=120, #设置字体最大值
              min_font_size=10, random_state=50, #设置配色方案
              scale=10)
wc.generate_from_frequencies(counts)
plt.imshow(wc) #将数值以图片形式显示出来
plt.show()
```



[illegible]

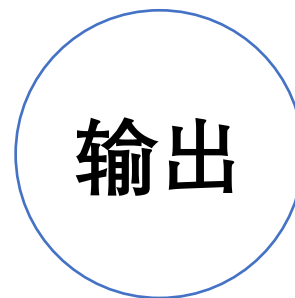
词频统计问题的IPO描述



从文件中读取一
篇待分析的文章



采用字典数据结构
统计词语出现的频率



根据词频进行图形
绘制或统计高频词语

解题步骤

第一步： 转换英文文献为文本文件，保存为AI_in_Finance.txt

读取文件内容存入字符串中

```
fobj = open("AI_in_Finance.txt", "r")  
paper=fobj. read()
```

解题步骤

第二步：分解并提取英文文章的单词

- ◆ 通过`paper.lower()`函数统一字母为小写
- ◆ 使用`paper.replace()`方法将英文单词的分隔符(空格、标点符号或者特殊符号) 统一为空格
- ◆ 使用`paper.split()`方法分解单词

字符串常量 (**string**模块)

常量名称	常量内容
string.punctuation	'!"#\$%&\'()*+,-./:;<=>?@[\\]^_`{ }~'
string.digits	'0123456789'
string.ascii_letters	'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ KLMNOPQRSTUVWXYZ'
string.printable	'0123456789abcdefghijklmnopqrstuvwxyzAB CDEFGHIJKLMNOPQRSTUVWXYZ!"#\$% &\'()*+,-./:;<=>?@[\\]^_`{ }~\t\n\r\x0b\x0c'
string.whitespace	' \t\n\r\x0b\x0c'

代码实现 (一)

```
import matplotlib.pyplot as plt
from wordcloud import WordCloud
from imageio.v2 import imread
import string
fobj = open("AI_in_Finance.txt", "r")
paper=fobj. read()
paper = paper. lower()
for ch in string.punctuation:
    paper = paper. replace(ch, " ") #将特殊字符替换为空格
words = paper.split( )
```

问题分析

第三步： 统计每个单词出现次数： 全部单词保存在列表 words 中， 定义一个字典counts={}, 键值对为：

<单词>： <出现次数>

依次统计每个单词出现次数

单词计数方法

```
if word in counts:  
    counts[word]=counts[word]+1  
else:  
    counts[word]=1
```

可简化为:

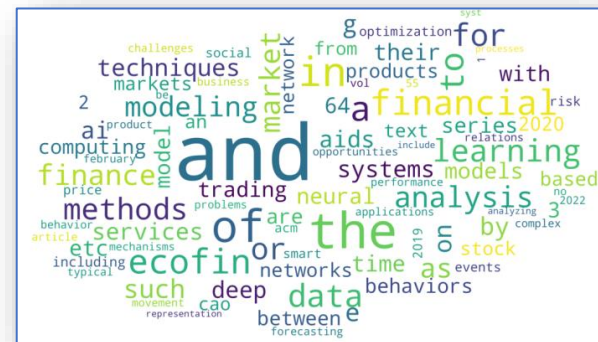
```
counts[word]=counts.get(word,0)+1
```

代码实现 (二)

```
counts = { }  
for word in words:  
    counts[word] = counts.get(word,0) + 1  
for word in words:  
    print(word, counts[word])
```

代码实现 (三)

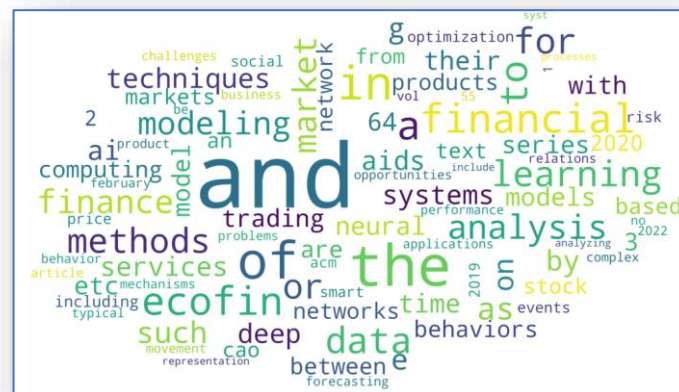
```
pic = imread('cloud.jpg')
wc=WordCloud(mask=pic,font_path='msyh.ttc', #中文字体
              repeat=False, background_color='white', #设置背景颜色
              max_words=110, max_font_size=120, #设置字体最大值
              min_font_size=10, random_state=50, #设置配色方案
              scale=10)
wc.generate_from_frequencies(counts)
plt.imshow(wc) #将数值以图片形式显示出来
plt.show()
```



问题

分析结果表明高频词汇都是**冠词**、**代词**、**连接词**等语法型词汇，并不能代表文章含义。

可以采用**集合**类型构建一个排除词汇库**excludes**，将语法型词汇排除。



拓展练习

```
able  
about  
above  
according  
accordingly  
across  
actually  
after  
afterwards  
again  
against  
ain't  
all  
allow  
allows
```

如何从文件中读取所有停用词并进行判断？

从文件获取排除词

```
excludes=set()
```

```
with open("stop.txt", "r") as handle:
```

```
    for i in handle:
```

```
        i=i. strip()
```

```
        excludes. add(i)
```



stop.txt - 记事本

文件

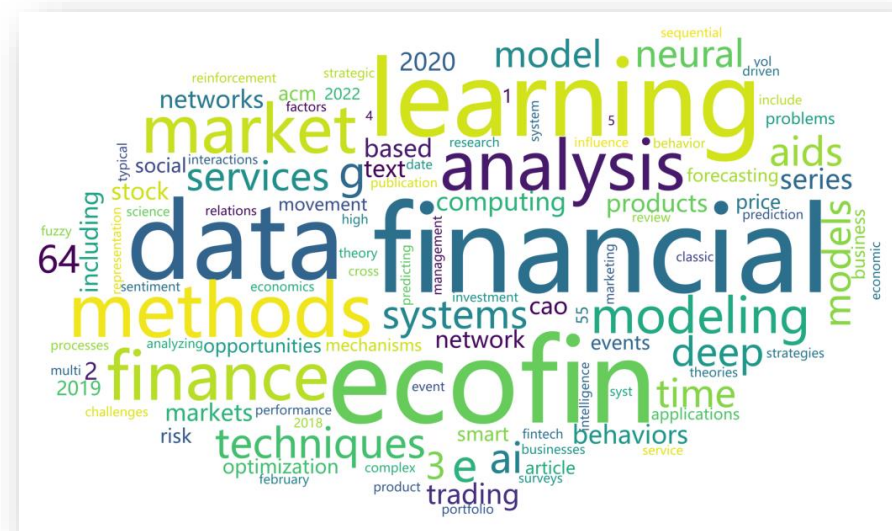
编辑

查看

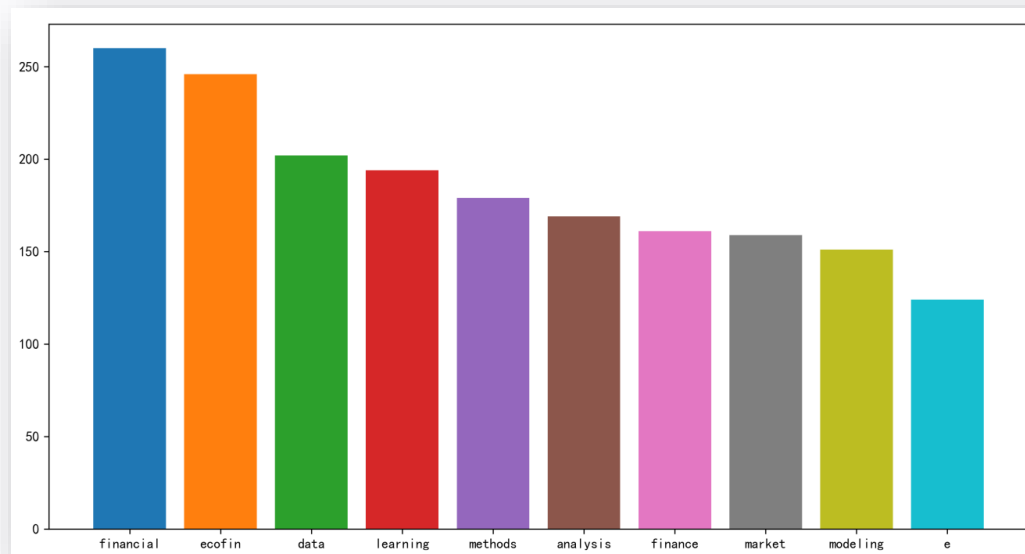
able
about
above
according
accordingly
across
actually
after
afterwards
again

代码实现 (二) 修改

```
excludes=set()
with open("stop.txt", "r") as handle:
    for i in handle:
        i=i. strip()
        excludes. add(i)
counts = { }
for word in words:
    if word in excludes:
        continue
    else:
        counts[word] = counts. get(word,0) + 1
```



将一个大型程序按照其功能分解成若干个相对独立的功能模块分别进行设计，最后把这些功能模块按层次关系进行组装。



函数定义

`def` 函数名 ([形式参数列表]):

执行语句

[return 返回值]

```
def mul(x,y):
```

```
    z=x*y
```

```
    return z
```

```
num1=int(input("请输入第一个数: "))
```

```
num2=int(input("请输入第二个数: "))
```

```
result=mul(num1,num2)
```

```
print("结果是%d" % result)
```


函数定义

```
def 函数名 ([形式参数列表]):
```

执行语句

[return 返回值]

```
def draw():  
    for i in range(100):  
        circle(i)  
        right(90)
```

```
from turtle import *  
speed(0)  
pencolor("red")  
draw()
```

参数传递

- ◆ 位置传递
- ◆ 关键字传递
- ◆ 默认值参数传递
- ◆ 元组传递
- ◆ 字典传递

位置传递

- ◆ 调用函数时，按照函数声明时参数顺序依次进行参数传递

```
def fun1(a, b):  
    if (a>b):  
        return a  
    else:  
        return b  
  
print(fun1(7, 3))
```

关键字传递

- ◆ 调用函数时，明确指定把某个实参值传递给某个形参

```
def fun2(a, b):  
    if (a>b):  
        print("a=",a)  
        return a  
    else:  
        return b  
  
print(fun2(b=2,a=7))
```

默认值参数传递

◆ 在定义函数时直接对形参赋值

```
def fun3(a, b=2):  
    if (a>b):  
        return a  
    else:  
        return b  
  
print(fun3(7))
```

默认值参数传递

- ◆ 函数调用时，可以全部、部分或不用默认值

```
def area(r=1.0, pi=3.14):  
    return r*r*pi
```

```
面积1=3.14  
面积2=81.67  
面积3=81.71
```

```
print("面积1=%.2f" % area())    #全部用默认值
```

```
print("面积2=%.2f" % area(5.1)) #部分用默认值
```

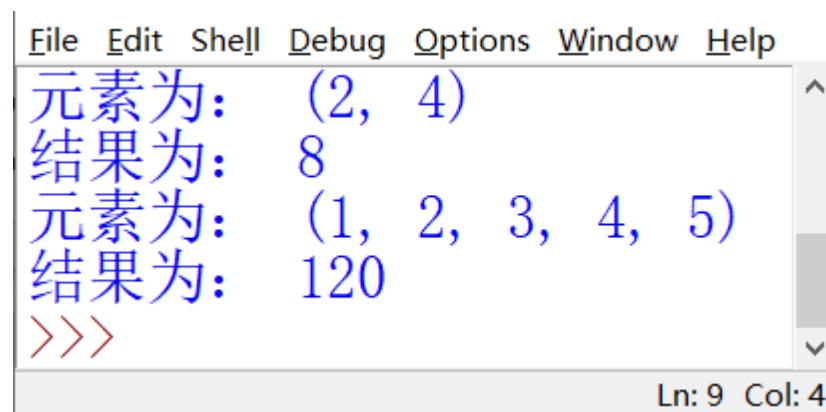
```
print("面积3=%.2f" % area(5.1,3.14159)) #不用默认值
```

元组类型变长参数传递

元组类型变长参数传递： 在函数声明时，若在某个参数名称前面加一个星号“*”，则表示该参数是一个元组类型可变长参数。

```
def mul(*number):  
    print("元素为: ",number)  
    total=1  
    for i in number:  
        total=total*i  
    return total
```

```
print("结果为: ", mul(2,4))  
print("结果为: ", mul(1,2,3,4,5))
```

A screenshot of a Python shell window showing the execution of the provided code. The window has a menu bar with 'File', 'Edit', 'Shell', 'Debug', 'Options', 'Window', and 'Help'. The output is displayed in blue text: '元素为: (2, 4)' followed by '结果为: 8' on the first call, and '元素为: (1, 2, 3, 4, 5)' followed by '结果为: 120' on the second call. The prompt ' >>>' is visible at the bottom. The status bar at the bottom right shows 'Ln: 9 Col: 4'.

```
File Edit Shell Debug Options Window Help  
元素为: (2, 4)  
结果为: 8  
元素为: (1, 2, 3, 4, 5)  
结果为: 120  
>>>  
Ln: 9 Col: 4
```

字典类型变长参数传递

字典类型变长参数传递： 在函数声明时，若在其某个参数名称前面加两个星号“**”，则表示该参数是一个字典类型可变长参数。

```
def func(**dict):  
    print(dict)  
    print(sum(dict.values()))
```

```
func(a=1,b=2,c=3)
```

```
File Edit Shell Debug Options Window Help  
{ 'a': 1, 'b': 2, 'c': 3}  
6  
>>>
```


lambda匿名函数

lambda: Python预留的关键字，可以定义一个匿名函数，函数体是一个简单的表达式而不是语句块，通常用在只使用一次的场景

形 式: `lambda argument_list: expression`

例 子: `lambda x: x**3` 输入x，输出x的3次方

lambda匿名函数

- ◆ 可以把匿名函数赋值给一个变量，再调用该函数

```
def f(x,y):  
    return x+y
```

```
x,y=2,3
```

```
print("x+y=%d"%(f(x,y)))
```

```
x,y=2,3
```

```
z=lambda x,y:x+y
```

```
print("x+y=%d"%(z(x,y)))
```

map() 函数

- ◆ 接收一个函数 `f` 和一个或多个 `list`, 并通过把函数 `f` 依次作用在 `list` 的每个元素上, 得到一个新的 `list` 并返回。

```
>>> price=[35.8, 24.9, 23.5]
```

```
>>> list(map(lambda x: x*2, price))
```

```
>>> price1=[35.8, 24.9, 23.5]
```

```
>>> price2=[88.5, 32.0, 12.9]
```

```
>>> list(map(lambda x,y: x+y, price1, price2))
```

案例分析

斐波那契数列
黄金分割为什么美丽



越来越接近0.618

斐波那契数列
(1/1)

函数的递归调用

- ◆ 函数在自身的定义中又调用自身
- ◆ 分析递归问题的关键

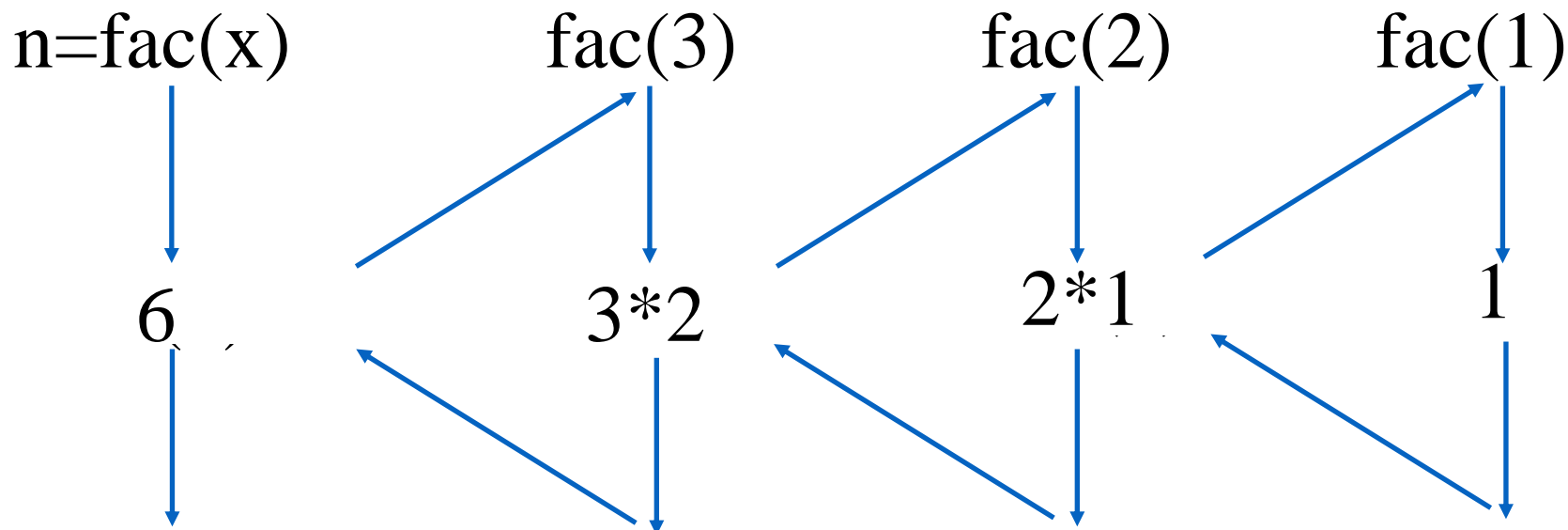
递推公式： 每次调用自身时参数的变化规律

结束条件： 递归调用结束的条件

递归算法n的阶乘

$$n! = \begin{cases} 1 & (n=1) \quad \# \text{结束条件, 基本解} \\ n * (n-1)! & (n \geq 2) \quad \# \text{递推公式} \end{cases}$$

函数的递归调用



```
def fac(n):  
    if n==1:  
        f=1  
    else:  
        f= n*fac(n-1)  
    return (f)
```

```
x=int(input("请输入: "))  
n=fac(x)  
print(n)
```

变量的作用域

变量作用域：指变量能被有效使用的范围

全局变量：在函数之外定义的变量，在整个程序范围内起作用

局部变量：指在某个函数内部定义的变量，在该函数范围内起作用

变量的作用域

说明： 全局变量与局部变量同名时， 函数内部局部变量起作用， 函数外部全局变量起作用

```
a, b=5, 6
def fun():
    a, b=10, 15
    print("函数内： a=%d, b=%d" % (a, b))
fun()
print("函数外： a=%d, b=%d" % (a, b))
```

全局变量

```
C=0 #在文件开头声明全局变量
def modifyConstant():
    global C
    print(C)
    C+=1
    return

if __name__ == '__main__':
    modifyConstant()
    print(C)
```

全局变量

#b.py 把全局变量定义在一个单独的模块中

```
gl_1 = 'hello'
```

```
gl_2 = 'world'
```

#a.py 在其它模块中使用

```
import b
```

```
def hello_world():
```

```
    print(b.gl_1,b.gl_2)
```

```
if __name__ == '__main__':
```

```
    hello_world()
```

课堂练习

以下程序的执行结果是 ()

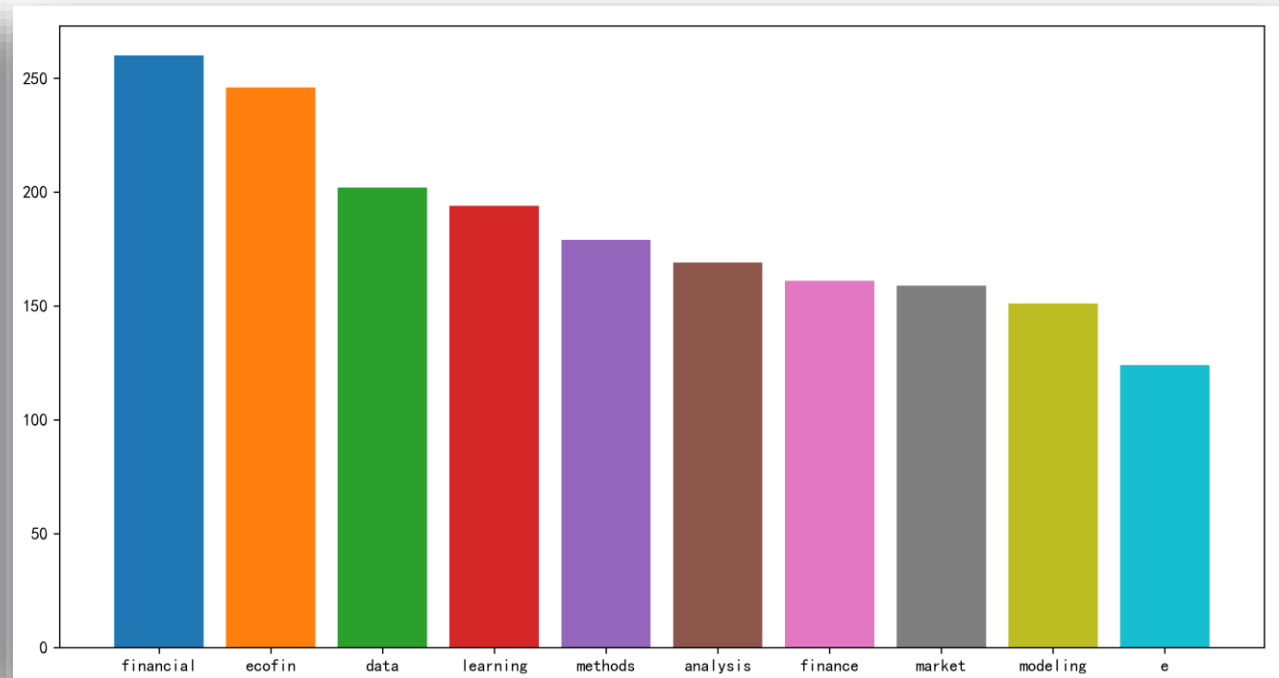
```
result = 2
def test(result, m):
    result= pow(result, m)
    print(result, end=" ")
test(result,3)
print(result)
```

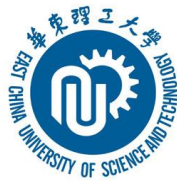
A、 8 2

B、 2 2

C、 2 8

D、 8 8





谢 谢