

A L G O R I T H M

P R A C T I C E

深度学习算法与实践

Steven Tang



Python

AZ

Examples

Python

第6章 组合数据类型

组合数据类型概述

组合数据类型



序列类型

Python语言中有很多数据类型都是序列类型，其中比较重要的是：`str`（字符串）、`tuple`（元组）和`list`（列表）

元组是包含0个或多个数据项的不可变序列类型。

列表则是一个可以修改数据项的序列类型，使用也最灵活

```
In [1]: a=[1,2,3]
```

```
In [2]: a[1]=4
```

```
In [3]: a
```

```
Out[3]: [1, 4, 3]
```

```
In [4]: b=(1,2,3)
```

```
In [5]: b[1]=2
```

```
Traceback (most recent call last):
```

```
File "<ipython-input-5-021cbee3eefa>", line 1, in <module>
    b[1]=2
```

```
TypeError: 'tuple' object does not support item assignment
```

"PYTHON"	2222.2	666	(1,2,3)	["你好",9]
----------	--------	-----	---------	----------

-5 -4 -3 -2 -1

序列类型

序列类型有12个通用
的操作符和函数

操作符	描述
$x \text{ in } s$	如果 x 是 s 的元素，返回True，否则返回False
$x \text{ not in } s$	如果 x 不是 s 的元素，返回True，否则返回False
$s + t$	连接 s 和 t (跟numpy区别)
$s * n$ 或 $n * s$	将序列 s 复制 n 次
$s[i]$	索引，返回序列的第 i 个元素
$s[i:j]$	分片，返回包含序列 s 第 i 到 j 个元素的子序列（不包含第 j 个元素）
$s[i:j:k]$	步骤分片，返回包含序列 s 第 i 到 j 个元素以 k 为步数的子序列
$\text{len}(s)$	序列 s 的元素个数（长度）
$\text{min}(s)$	序列 s 中的最小元素
$\text{max}(s)$	序列 s 中的最大元素
$s.\text{index}(x, i[, j])$	序列 s 中从 i 开始到 j 位置中第一次出现元素 x 的位置
$s.\text{count}(x)$	序列 s 中出现 x 的总次数

序列类型

```
In [1]: [1,2,3]+[2,3]
Out[1]: [1, 2, 3, 2, 3]
```

```
In [2]: "123"+"23"
Out[2]: '12323'
```

```
In [3]: (1,2)+(2,3)
Out[3]: (1, 2, 2, 3)
```

```
In [4]: (1,2)*3
Out[4]: (1, 2, 1, 2, 1, 2)
```

```
In [5]: a=[1,2,2,3,3,3,]
```

```
In [6]: a
Out[6]: [1, 2, 2, 3, 3, 3]
```

```
In [7]: a.count(3)
Out[7]: 3
```

```
In [15]: a
Out[15]: [1, 1, 2, 2, 3, 3, 3]
```

```
In [16]: a.index(2,4)
Traceback (most recent call last):
```

```
File "<ipython-input-16-2e929511023d>", line 1, in <module>
    a.index(2,4)
```

```
ValueError: 2 is not in list
```

```
In [17]:
```

```
In [17]: a.index(2,3)
Out[17]: 3
```

```
In [19]: s='abacd'
          |
```

```
In [20]: s.find('e')
Out[20]: -1
```

```
In [21]: s.find('a')
Out[21]: 0
```

序列类型

元组类型在表达固定数据项、函数多返回值、多变量同步赋值、循环遍历等情况下十分有用。Python中元组采用逗号和圆括号（可选）来表示。

```
In [15]: def switch(x,y):
...:     return y,x
...:
```

```
In [16]: switch(3,4)
Out[16]: (4, 3)
```

```
In [17]: type(switch(3,4))
Out[17]: tuple
```

```
In [21]: list(enumerate(range(10)))
Out[21]:
[(0, 0),
 (1, 1),
 (2, 2),
 (3, 3),
 (4, 4),
 (5, 5),
 (6, 6),
 (7, 7),
 (8, 8),
 (9, 9)]
```

```
In [6]: animals='cat','dog','chick'
```

```
In [7]: type(animals)
Out[7]: tuple
```

```
In [8]: creatures='flower','human',animals
```

```
In [9]: creatures
Out[9]: ('flower', 'human', ('cat', 'dog', 'chick'))
```

```
In [10]: creatures[0]
Out[10]: 'flower'
```

```
In [11]: creatures[0][1]
Out[11]: 'l'
```

```
In [12]: creatures[2][1]
Out[12]: 'dog'
```

```
In [13]: creatures[2][1][2]
Out[13]: 'g'
```


集合类型

集合类型与数学中集合的概念一致，即包含0个或多个数据项的无序组合。集合中元素不可重复，元素类型只能是固定数据类型，例如：整数、浮点数、字符串、元组等，列表、字典和集合类型本身都是可变数据类型，不能作为集合的元素出现。集合用大括号（{}）表示，可以用赋值语句生成一个集合。

```
In [25]: s={1,2,3}
```

```
In [26]: s={1,[1,2]}
```

```
Traceback (most recent call last):
```

```
File "<ipython-input-26-6f43a9442f53>", line 1, in <module>
    s={1,[1,2]}
```

```
TypeError: unhashable type: 'list'
```

集合类型

[]

$S = e$
[]

由于集合是无序组合，它没有索引和位置的概念，不能分片，集合中元素可以动态增加或删除。集合用大括号（{ }）表示，可以用赋值语句生成一个集合。

```
In [29]: S = {425, (10, "CS"), 425}
```

```
In [30]: S
```

```
Out[30]: {(10, 'CS'), 425}
```

```
In [31]: T = {425, "BIT", (10, "CS"), 424, 425, "BIT"}
```

```
In [32]: T
```

```
Out[32]: {(10, 'CS'), 424, 425, 'BIT'}
```

去重 → 集合
Python

使用集合去除重复元素

由于集合元素是无序的，集合的打印效果与定义顺序可以不一致。由于集合元素独一无二，使用集合类型能够过滤掉重复元素。set(x)函数可以用于生成集合。

1. 初始化

```
In [33]: s="Hello"

In [34]: set(s)
Out[34]: {'H', 'e', 'l', 'o'}
```

去重

```
In [35]: l=[1,3,3,2,2,4,9]

In [36]: l_n=list(set(l))

In [37]: l_n
Out[37]: [1, 2, 3, 4, 9]

In [38]: set(l)
Out[38]: {1, 2, 3, 4, 9}
```

集合操作符

集合类型有10个操作符

这 10 个

不仅各元素

操作符	描述
$S - T$ 或 <code>S.difference(T)</code>	返回一个新集合，包括在集合S中但不在集合T中的元素
$S -= T$ 或 <code>S.difference_update(T)</code>	更新集合S，包括在集合S中但不在集合T中的元素
$S \& T$ 或 <code>S.intersection(T)</code>	返回一个新集合，包括同时在集合S和T中的元素
$S \&= T$ 或 <code>S.intersection_update(T)</code>	更新集合S，包括同时在集合S和T中的元素。
$S \Delta T$ 或 <code>s.symmetric_difference(T)</code>	返回一个新集合，包括集合S和T中元素，但不包括同时在其中的元素
$S \Delta= T$ 或 <code>s.symmetric_difference_update(T)</code>	更新集合S，包括集合S和T中元素，但不包括同时在其中的元素
$S T$ 或 <code>S.union(T)</code>	返回一个新集合，包括集合S和T中所有元素
$S = T$ 或 <code>S.update(T)</code>	更新集合S，包括集合S和T中所有元素
$S \leq T$ 或 <code>S.issubset(T)</code>	如果S与T相同或S是T的子集，返回True，否则返回False，可以用 $S < T$ 判断S是否是T的真子集
$S \geq T$ 或 <code>S.issuperset(T)</code>	如果S与T相同或S是T的超集，返回True，否则返回False，可以用 $S > T$ 判断S是否是T的真超集

```
In [42]: S={1,2,3,4}
```

```
In [43]: T={2,3,4,5}
```

```
In [44]: S-T  
Out[44]: {1}
```

```
In [45]: T-S  
Out[45]: {5}
```

```
In [46]: S.difference(T)  
Out[46]: {1}
```


集合类型

集合类型有10个操作函数或方法

函数或方法	描述
<code>S.add(x)</code>	如果数据项x不在集合S中，将x增加到s
<code>S.clear()</code>	移除S中所有数据项
<code>S.copy()</code>	返回集合S的一个拷贝
<code>S.pop()</code>	随机返回集合S中的一个元素，如果S为空，产生KeyError异常
<code>S.discard(x)</code>	如果x在集合S中，移除该元素；如果x不在，不报错
<code>S.remove(x)</code>	如果x在集合S中，移除该元素；不在产生KeyError异常
<code>S.isdisjoint(T)</code>	如果集合S与T没有相同元素，返回True
<code>len(S)</code>	返回集合S元素个数
<code>x in S</code>	如果x是S的元素，返回True，否则返回False
<code>x not in S</code>	如果x不是S的元素，返回True，否则返回False

集合类型

集合类型主要用于三个场景：成员关系测试、元素去重和删除数据项。

集合类型不能下标，集合不能包含组合数据类型。

```
In [49]: S[0]
Traceback (most recent call last):

  File "<ipython-input-49-9ff278f294d3>", line 1,
in <module>
    S[0]
TypeError: 'set' object does not support indexing
```

```
In [50]: S={1,2,2,3}
Traceback (most recent call last):

  File "<ipython-input-50-c1d07a585626>", line 1,
in <module>
    S={{1,2},2,3}
TypeError: unhashable type: 'set'
```

列表类型和操作

列表类型的概念

列表（list）是包含0个或多个对象引用的有序序列，属于序列类型。与元组不同，列表的长度和内容都是可变的，可自由对列表中数据项进行增加、删除或替换。列表没有长度限制，元素类型可以不同，使用非常灵活。

列表类型的概念

列表

由于列表属于序列类型，所以列表也支持成员关系操作符 (`in`)、长度计算函数 (`len()`)、分片 (`[]`)。列表可以同时使用正向递增序号和反向递减序号，可以采用标准的比较操作符 (`<`、`<=`、`=`、`!=`、`>=`、`>`) 进行比较，列表的比较实际上是单个数据项的逐个比较。

enumerated

列表的append和list函数

列表用中括号（[]）表示，也可以通过list()函数将其他类型转化成列表。直接使用list()函数会返回一个空列表。

```
In [36]: ls=[]  
In [37]: ls.append(0)  
In [38]: ls.append('ab')  
In [39]: ls  
Out[39]: [0, 'ab']  
In [40]: list({'a', 'b', })  
Out[40]: ['b', 'a']  
In [41]: list(('a', 'b', ))  
Out[41]: ['a', 'b']  
In [42]: list("PYTHON")  
Out[42]: ['P', 'Y', 'T', 'H', 'O', 'N']
```

这样写是不对的

列表类型的概念

与整数和字符串不同，列表要处理一组数据，因此，列表必须通过显式的数据赋值才能生成，简单将一个列表赋值给另一个列表不会生成新的列表对象。

a → [1, 2, 3, 4]
b → [1, 2, 2, 4]

```
In [43]: a=[1,2,3,4]
In [44]: b=a.copy()
In [45]: a.append(-1)

In [46]: b
Out[46]: [1, 2, 3, 4, -1]

In [47]: s='abc'
In [48]: ss=s
In [49]: s='abcd'

In [50]: ss
Out[50]: 'abc'
```

```
In [51]: s={'a','b','c'}
In [52]: ss=s
In [53]: s.add('e')

In [54]: ss
Out[54]: {'a', 'b', 'c', 'e'}
```

```
In [55]: t=1,2,3
In [56]: tt=t
In [57]: t=1,2,3,4

In [58]: tt
Out[58]: (1, 2, 3)
```

列表类型的操作

$ls = [1, 2, 3, 4]$

$del\ ls[2:4]$

$ls = [1, 4]$

$ls[1:3] = []$

函数或方法	描述
$ls[i] = x$	替换列表ls第i数据项为x
$ls[i:j] = lt$	用列表lt替换列表ls中第i到j项数据（不含第j项，下同）
$ls[i:j:k] = lt$	用列表lt替换列表ls中第i到j以k为步的数据
$del\ ls[i:j]$	删除列表ls第i到j项数据，等价于 $ls[i:j] = []$
$del\ ls[i:j:k]$	删除列表ls第i到j以k为步的数据
$ls += lt$ 或 $ls.extend(lt)$	将列表lt元素增加到列表ls中
$ls *= n$	更新列表ls，其元素重复n次
$ls.append(x)$	在列表ls最后增加一个元素x
$ls.clear()$	删除ls中所有元素
$ls.copy()$	生成一个新列表，复制ls中所有元素
$ls.insert(i, x)$	在列表ls第i位置增加元素x
$ls.pop(i)$	将列表ls中第i项元素取出并删除该元素
$ls.remove(x)$	将列表中出现的第一元素x删除
$ls.reverse(x)$	列表ls中元素反转

$ls.sort$

列表类型的操作

$$[random.randint(0, 100) for i in range(10)]$$

In [63]: ls=list(range(5))

$$[0, 1, 2, 3, 4]$$

In [64]: ls

Out[64]: [0, 1, 2, 3, 4]

In [65]: ls[1:3]=[3,4,5]

In [66]: ls

Out[66]: [0, 3, 4, 5, 3, 4]

In [67]: ls[4]='good'

In [68]: ls

Out[68]: [0, 3, 4, 5, 'good', 4]

In [69]: ls[1:4]=[]

In [70]: ls

Out[70]: [0, 'good', 4]

列表和lambda混合操作

```
In [81]: ls=list(range(5))
```

```
In [82]: nls=map(lambda x:x*x,ls)
```

```
In [83]: nls
```

```
Out[83]: <map at 0x2956ee89358>
```

```
In [84]: list(nls)
```

```
Out[84]: [0, 1, 4, 9, 16]
```

```
In [85]: for i in map(lambda x:x*x,ls):
```

```
....:     print(i)
```

```
....:
```

```
0
```

```
1
```

```
4
```

```
9
```

```
16
```

$ls = [0, 1, 4, 9, 16]$

$nls = \text{map}(\text{lambda } x: x+1, ls)$

$nls = [1, 2, 5, 10, 17]$

列表和lambda混合操作 (reduce)

map \rightarrow 遍历
 \rightarrow 遍历

$1 \times 2 \times 3 \times 4 \times 5$

```
In [104]: ls
Out[104]: [1, 2, 3, 4, 5]

In [105]: print(reduce(lambda x,y: x*y+1, ls))
206

In [106]: (((1*2+1)*3+1)*4+1)*5+1
Out[106]: 206

In [107]: a = reduce(lambda x,y: x*y, range(1,6))

In [108]: a
Out[108]: 120

In [109]: b = map(lambda i: reduce(lambda x,y: x*y,
range(1,i+1),1), [1,2,3,4,5])

In [110]: b
Out[110]: <map at 0x2956ee94ef0>

In [111]: list(b)
Out[111]: [1, 2, 6, 24, 120]
```

$f(f(f(x_0, x_1), x_2), x_3), x_4)$

120

列表和lambda混合操作

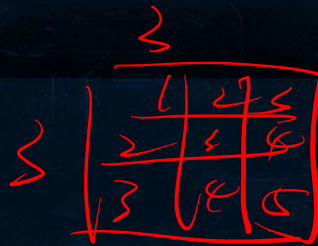
(filter)

values

filter

```
In [112]: ls
Out[112]: [1, 2, 3, 4, 5]
In [113]: ls=filter(lambda x: x % 2 == 0, ls)
In [114]: ls
Out[114]: <filter at 0x2956ee94da0>
In [115]: ls=list(ls)
In [116]: ls
Out[116]: [2, 4]
```


二维列表



A hand-drawn 3x3 grid with numbers 1 through 9. The grid is enclosed in a red border. The numbers are arranged as follows: Row 1: 1, 2, 3; Row 2: 4, 5, 6; Row 3: 7, 8, 9. There are red curly braces on the left and top of the grid.

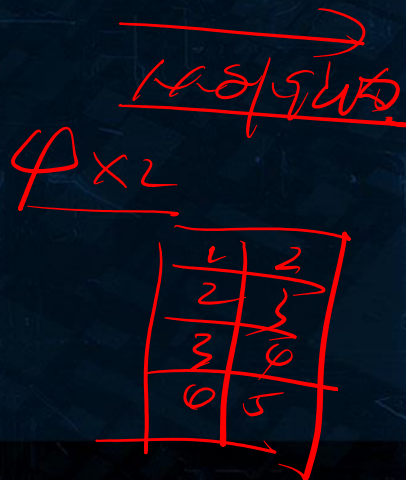
1	2	3
4	5	6
7	8	9

思考：如何创建一个~~3*3~~的列表

```
multi_row_list=[[1,2,3],[2,3,4],[3,4,5]]
```

思考：下面这个列表是什么维度的？

```
multi_row_list=[[1,2],[2,3],[3,4],[4,5]]
```



A hand-drawn 4x2 grid with numbers 1 through 8. The grid is enclosed in a red border. The numbers are arranged as follows: Row 1: 1, 2; Row 2: 3, 4; Row 3: 5, 6; Row 4: 7, 8. There are red curly braces on the left and top of the grid. Above the grid, the text '1-8/9-10' is written with an arrow pointing to the right. To the left of the grid, the text '4x2' is written with an arrow pointing to the grid.

1	2
3	4
5	6
7	8

高维列表

这是一个二维列表 `multi_row_list=[[1,2,3],[2,3,4],[3,4,5]]`，怎么构建一个三维列表？

$[[[1, 2, 3], [2, 3, 4], [3, 4, 5]]]$ \rightarrow $[1, 2, 3, \dots]$
 $2 \times 3 \times 3$

format()支持列表下标

```
In [76]: boy = ['Sam', 22]

In [77]: girl = ["Kate", 19]

In [78]: print ("{0[0]} is {0[1]} years old, {1[0]} is
{1[1]} years old".format(boy, girl))
Sam is 22 years old, Kate is 19 years old
```

列表是一个十分灵活的数据结构，它具有处理任意长度、混合类型的能力，并提供了丰富的基础操作符和方法。当程序需要使用组合数据类型管理批量数据时，请尽量使用列表类型。

字典类型的计算

字典类型的基本概念

Key Value
() [] {}
↓ ↓ ↓
f t 3

通过任意键信息查找一组数据中值信息的过程叫映射，Python语言中通过字典实现映射。

Python语言中的字典可以通过大括号({})建立，建立模式如下：

{<键1>:<值1>,<键2>:<值2>...,<键n>:<值n>}

其中，键和值通过冒号连接，不同键值对通过逗号隔开。字典中键是唯一的。

dictionary
dict

```
In [117]: st={'a'}  
In [118]: dic={'a':1}  
In [119]: type(st)  
Out[119]: set  
In [120]: type(dic)  
Out[120]: dict
```

```
In [150]: dic={'a':1,'a':2}  
In [151]: dic  
Out[151]: {'a': 2}
```

```
In [152]: dic={'a':1,'a':2,'b':2}  
In [153]: dic  
Out[153]: {'a': 2, 'b': 2}
```

字典类型的基本概念

```
In [126]: dic
Out[126]: {'T': '腾讯', 'B': '百度', 'A': '阿里'}

In [127]: dic={'B': '百度', 'A': '阿里', 'T': '腾讯'}

In [128]: dic
Out[128]: {'B': '百度', 'A': '阿里', 'T': '腾讯'}

In [129]: dic={"中国": "北京", "美国": "华盛顿", "法国": "巴黎"}

In [130]: dic
Out[130]: {'中国': '北京', '美国': '华盛顿', '法国': '巴黎'}
```

字典类型的基本概念

字典最主要的用法是查找与特定键相对应的值，这通过索引符号来实现。通过中括号可以增加新的元素和改变原来的元素。

```
In [131]: dic={'B':'百度','A':'阿里','T':'腾讯'}

In [132]: dic['B']
Out[132]: '百度'

In [133]: dic['Z']='字节跳动'

In [134]: dic
Out[134]: {'B': '百度', 'A': '阿里', 'T': '腾讯', 'Z': '字节跳动'}

In [135]: dic['B']='百度中国'

In [136]: dic
Out[136]: {'B': '百度中国', 'A': '阿里', 'T': '腾讯', 'Z': '字节跳动'}
```

字典类型的操作

函数和方法	描述
<u><d>.keys()</u>	返回所有的键信息
<u><d>.values()</u>	返回所有的值信息
<u><d>.items()</u>	返回所有的键值对
<d>.get(<key>,<default>)	键存在则返回相应值，否则返回默认值
<u><d>.pop(<key>,<default>)</u>	键存在则返回相应值，同时删除键值对，否则返回默认值
<d>.popitem()	随机从字典中取出一个键值对，以元组(key, value)形式返回
<d>.clear()	删除所有的键值对
del <d>[<key>]	删除字典中某一个键值对
<key> in <d>	如果键在字典中返回True，否则返回False

```
In [138]: dic.keys()
Out[138]: dict_keys(['B', 'A', 'T', 'Z'])

In [139]: dic.values()
Out[139]: dict_values(['百度中国', '阿里', '腾讯', '字节跳动'])

In [140]: dic.items()
Out[140]: dict_items([('B', '百度中国'), ('A', '阿里'), ('T', '腾讯'), ('Z', '字节跳动')])
```

use `dict.keys()`

字典类型的操作

```
In [141]: 'A' in dic
```

```
Out[141]: True
```

```
In [142]: '阿里' in dic
```

```
Out[142]: False
```

```
In [143]: dic.get('A')
```

```
Out[143]: '阿里'
```

```
In [144]: dic.get('c')
```

```
In [145]: dic.get('c', '查找不到')
```

```
Out[145]: '查找不到'
```

```
In [146]: dic.get('B', '查找不到')
```

```
Out[146]: '百度中国'
```

字典类型的操作

与其他组合类型一样，字典可以通过for...in语句对其元素进行遍历，默认是遍历keys 基本语法结构如下：

for <变量名> in <字典名>:

语句块

```
In [147]: for key in dic:
...:     print(key)
...:
```

B
A
T
Z

```
In [148]: for key,value in dic.items():
...:     print(key)
...:     print(value)
...:
```

B
百度中国
A
阿里
T
腾讯
Z
字节跳动

(key, value)
dic.value

lambda函数与sorted的结合

In [157]: ls=[3,2,5,6,3,2]

In [158]: sorted(ls)

Out[158]: [2, 2, 3, 3, 5, 6]

In [159]: dic={'a':3, 'e':2, 'f':9, 'k':1}

In [160]: sorted(dic, key=lambda item:item[0])

Out[160]: ['a', 'e', 'f', 'k']

In [161]: sorted(dic.items(), key=lambda item:item[0])

Out[161]: [('a', 3), ('e', 2), ('f', 9), ('k', 1)]

In [162]: sorted(dic.items(), key=lambda item:item[1])

Out[162]: [('k', 1), ('e', 2), ('a', 3), ('f', 9)]

In [163]: sorted(dic.items(), key=lambda

item:item[1], reverse=True)

Out[163]: [('f', 9), ('a', 3), ('e', 2), ('k', 1)]

自定义函数与sorted的结合

给定一个整型数组, 将数组中所有的0移动到末尾, 非0项保持不变; 在原始数组上进行移动操作, 勿创建新的数组,

```
num=''.join(input("请输入数字").split())
li=[int(i) for i in num]
def move_zero(x):
    if x == 0:
        return 1
    else:
        return 0
print(sorted(li, key=move_zero))
```

Handwritten examples of the input and output:

Input: "70602"

Output: [2, 0, 6, 0, 2]

Output: [4, 2, 0, 0]

总结

List类型:

变量 = [值,值,值...]

Tuple类型

变量 = (值,值,值...)

Set类型

变量 = {值,值,值...}

Dict类型

变量 = {键:值,键:值,键:值...}

课后作业

随机产生10个三位正整数的列表，然后按照它们个位数的大小进行排序。

用至少两种方法实现将一个字符串倒序。

课后练习

1. 给定一个整型数组, 将数组中所有的0移动到末尾,非0项保持不变;在原始数组上进行移动操作,勿创建新的数组;

情感分析项目介绍

英文词频统计

```
import string

reviewtxt=open('reviews.txt','r').read()

reviewtxt=reviewtxt.lower()

for c in string.punctuation:
    if c in reviewtxt:
        print("True:"+c)
        reviewtxt=reviewtxt.replace(c,"")
    .....
words=reviewtxt.split()

counts={}

for word in words:
    counts[word]=counts.get(word,0)+1#

counts=sorted(counts.items(),key=lambda item: item[1],reverse=True)

for i in range(10):
    word, count=counts[i]
    print("{0:<10}-{1:>5}".format(word,count))
```

英文词频统计

```
In [1]: runfile('C:/Users/Dr. Tang.DESKTOP-DQDB847/Desktop/渡一/  
codes/codes5/wordcount.py')
```

the	336713
.	327192
and	164107
a	163009
of	145864
to	135720
is	107328
br	101872
it	96352
in	93968