实验内容与要求

实验一 文本相似度计算

1.1 初话 python

一、实验目的

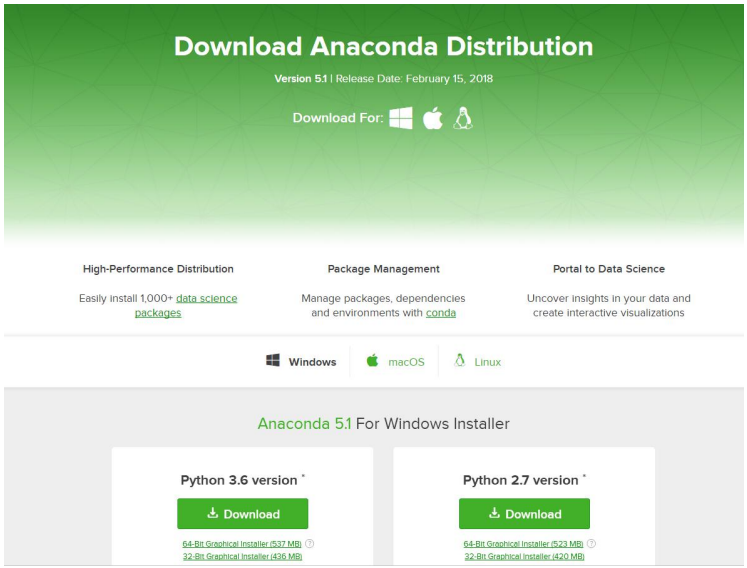
- 熟悉 Python 环境搭建
- 掌握 Python 扩展包安装
- 掌握 Python 文件读写模式

二、实验原理、方法和手段

- Python 环境搭建

以 win10 系统为例：

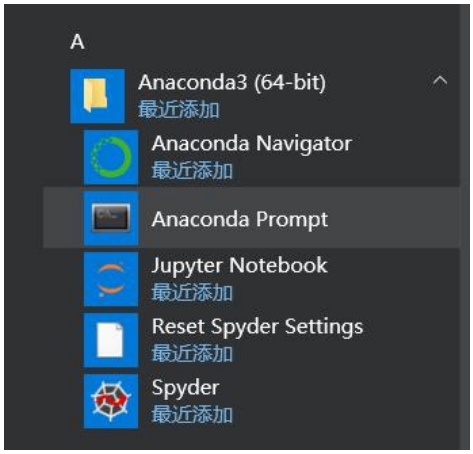
1、安装 anaconda，官网地址：https://www.anaconda.com/download/



2、根据需要选择 python2.7 或者 python3.6 下载，这里选择 python3.6。

3、安装好后，如下图所示，打开 Anaconda Prompt，输入 python，查

看 python 版本号。



4、输入以下命令即可联网升级 pip 工具

python -m pip install -U pip

- Python 扩展包安装

由于 Anaconda 自带很多扩展包，省去很多安装的麻烦。但在实际中，需要更多的扩展包支持，通常有三种方法。

1、在线直接安装：pip install +所需的扩展包名

eg. pip install numpy

2、离线安装：在 https://www.lfd.uci.edu/~gohlke/pythonlibs/下载所需的安装包，然后 pip install +安装包名

eg. pip install scikit_learn-0.19.1-cp36-cp36m-win_amd64.whl

3、源码安装：解压之后：

Python setup.py build

Python setup.py install

- Python 文件读写模式

不同模式打开文件的完全列表：

模式	描述
r	以只读方式打开文件。文件的指针将会放在文件的开头。这是默认模式。
rb	以二进制格式打开一个文件用于只读。文件指针将会放在文件的开头。这是默认模式。一般用于非文本文件如图片等。
r+	打开一个文件用于读写。文件指针将会放在文件的开头。
rb+	以二进制格式打开一个文件用于读写。文件指针将放在文件的开头。一般用于非文本文件如图片等。
w	打开一个文件只用于写入。如果该文件已存在则将其覆盖。如果该文件不存在，创建新文件。
wb	以二进制格式打开一个文件只用于写入。如果该文件已存在则将其覆盖。如果该文件不存在，创建新文件。一般用于非文本文件如图片等。
w+	打开一个文件用于读写。如果该文件已存在则将其覆盖。如果该文件不存在，创建新文件。
wb+	以二进制格式打开一个文件用于读写。如果该文件已存在则将其覆盖。如果该文件不存在，创建新文件。一般用于非文本文件如图片等。
a	打开一个文件用于追加。如果该文件已存在，文件指针将会放在文件的结尾。也就是说，新的内容将会被写入到已有内容之后。如果该文件不存在，创建新文件进行写入。
ab	以二进制格式打开一个文件用于追加。如果该文件已存在，文件指针将会放在文件的结尾。也就是说，新的内容将会被写入到已有内容之后。如果该文件不存在，创建新文件进行写入。
a+	打开一个文件用于读写。如果该文件已存在，文件指针将会放在文件的结尾。文件打开时会是追加模式。如果该文件不存在，创建新文件用于读写。
ab+	以二进制格式打开一个文件用于追加。如果该文件已存在，文件指针将会放在文件的结尾。如果该文件不存在，创建新文件用于读写。

示例

test.txt 的原始文件如下：

```
早上好
您好
how are you?
```

如果我们在 open 文件后，没有进行任何读写，则从末尾加入

```
>>> with open('test.txt','r+') as f:
...         f.writelines("长沙")
```

这时文件变成:

```
长沙好
您好
how are you?
```

>>> with open('test.txt','w+') as f:

... f.writelines("长沙")

这时文件变成:

长沙

1.2 文本相似度计算

一、实验目的

1、能够编程实现最小哈希 Jaccard 估计值，并和 Jaccard 实际值对比

2、熟悉局部敏感哈希原理

二、实验内容

1、根据案例，了解 Jaccard 相似度计算方法

2、根据案例，掌握最小哈希 Jaccard 估计值计算，并和 Jaccard 相似度实际值对比。

3、根据案例，掌握局部敏感哈希原理。

4、计算所给数据集每一项 ID 的相似项。

三、实验原理、方法和手段

1、安装 datasketch.

解压缩 datasketch-master.zip，命令行安装：

Python setup.py build

Python setup.py install

```
(base) D:\pip\datasketch-master>python setup.py build
running build
running build_py
creating build
creating build\lib
creating build\lib\datasketch
copying datasketch\b_bit_minhash.py -> build\lib\datasketch
```

```
(base) D:\pip\datasketch-master>python setup.py install
running install
running bdist_egg
running egg_info
creating datasketch.egg-info
writing datasketch.egg-info\PKG-INFO
writing dependency_links to datasketch.egg-info\dependency_links.txt
```

2、最小哈希 Jaccard 计算

■ 导入安装好的 datasketch，输入两组待测数据。

具体 minhash 实现可参考 datasketch-master\datasketch\minhash.py，这里直接调用该模块。

In [1]:
```
from datasketch import MinHash
```

In [2]:
```
data1 = ['minhash', 'is', 'a', 'probabilistic', 'data', 'structure', 'for',
         'estimating', 'the', 'similarity', 'between', 'datasets']
data2 = ['minhash', 'is', 'a', 'probability', 'data', 'structure', 'for',
         'estimating', 'the', 'similarity', 'between', 'documents']
```

■ 计算 Jaccard 估值

In [3]:
```
m1, m2 = MinHash(), MinHash()
for d in data1:
    m1.update(d.encode('utf8'))
for d in data2:
    m2.update(d.encode('utf8'))
print("Estimated Jaccard for data1 and data2 is", m1.jaccard(m2))
```

```
Estimated Jaccard for data1 and data2 is 0.7109375
```

■ Jaccard 实际值

In [4]:
```
s1 = set(data1)
s2 = set(data2)
actual_jaccard = float(len(s1.intersection(s2)))/float(len(s1.union(s2)))
print("Actual Jaccard for data1 and data2 is", actual_jaccard)
```

```
Actual Jaccard for data1 and data2 is 0.7142857142857143
```

我们发现，估值近似等于实际值。

3、我们对源码进行抽取，进行编译，见<源码 1 文件夹>。

```
(base) C:\Users\gsz\Desktop\新建文件夹 (3)>python minhash_examples.py
Estimated Jaccard for data1 and data2 is 0.7109375
Actual Jaccard for data1 and data2 is 0.7142857142857143
```

我们对源码进一步修改。通过自定义 MinHash 中使用的置换函数的数量来调整精度。

```
m = MinHash(num_perm=256)
```

然而，更多的置换函数意味着：更多的哈希值需要被存储。

MinHash 的速度和内存使用率都与使用的置换函数的数量成线性比例。

4、最小哈希 局部敏感哈希

假设你有一个非常大的集合。提供一个查询（也是一个集合），希望查找集合中具有超出特定阈值的 Jaccard 相似度的集合，并且希望通过其他许多查询来完成。为了有效地执行此操作，可以为每个集合创建一个 MinHash，并且在查询到达时，计算查询 MinHash 和集合的所有 MinHash 之间的 Jaccard 相似度，并返回满足阈值的集合。

所述方法仍然是 O(n)算法，这意味着查询成本相对于集合的数量线性增加。一个流行的选择是使用局部敏感散列（LSH）索引。

```
In [1]: from datasketch import MinHash, MinHashLSH

In [2]: set1 = set(['minhash', 'is', 'a', 'probabilistic', 'data', 'structure', 'for',
                    'estimating', 'the', 'similarity', 'between', 'datasets'])
        set2 = set(['minhash', 'is', 'a', 'probability', 'data', 'structure', 'for',
                    'estimating', 'the', 'similarity', 'between', 'documents'])
        set3 = set(['minhash', 'is', 'probability', 'data', 'structure', 'for',
                    'estimating', 'the', 'similarity', 'between', 'documents'])
```

In [4]:
```
m1 = MinHash(num_perm=128)
m2 = MinHash(num_perm=128)
m3 = MinHash(num_perm=128)
for d in set1:
    m1.update(d.encode('utf8'))
for d in set2:
    m2.update(d.encode('utf8'))
for d in set3:
    m3.update(d.encode('utf8'))
```

In [10]:
```
# Create LSH index
lsh = MinHashLSH(threshold=0.5, num_perm=128)
lsh.insert("m2", m2)
lsh.insert("m3", m3)
result = lsh.query(m1)
print("Approximate neighbours with Jaccard similarity > 0.5", result)
```

```
Approximate neighbours with Jaccard similarity > 0.5 ['m3', 'm2']
```

四、实验条件

运行带 python 编译环境的计算机 1 台。并根据实验需要安装必要的 python 库。

五、实验步骤

1、在源码 1 的基础上实现实验手段步骤 3 中置换函数的数量与精度关系图示。

2、读取文件夹下的 products.txt，调用 datasketch 计算每一个 ID 的相似项或者其他方法皆可。

六、思考题

思考领域：命名实体识别/重名消歧

在文件夹中有一个数据集 CNKI-636603734021580000.txt，是同一姓名的作者近年来所发表论文的信息，但不是同一个作者所发表论文的信息，只是姓名相同。

该数据集包括了论文题名、合作者名、作者单位、文献来源、关键词。思考字段如何组合构建出最能识别出同一姓名作者之间的差

异，以便于区分不同出相同姓名的不同作者。