西南交通大學

互联网搜索引擎课程

设计报告 II

学号： 2015114716

班级： 计算机 2015-03 班

学生： 史晨阳

指导老师： 吴晓

目录

项目二. 搜索引擎文本相似度

1. 项目要求

（i）建立并实现文本搜索功能

1. 利用/调用开源搜索引擎 Lucene 或者 Lemur 实现文本搜索引擎。查阅相关资料安装软件。
2. 对经过预处理后的 500 个英文和中文文档/网页建立搜索并实现搜索功能。
3. 通过上述软件对文档建立索引（Indexing），然后通过前台界面或者已提供的界面，输入关键字，展示搜索功能。
4. 前台可通过网页形式、应用程序形式、或者利用已有的界面工具显示。
5. 必须实现英文搜索功能。中文搜索功能作为可选项。

（ii）比较文档之间的相似度

通过余弦距离(Cosine Distance)计算任意两个文档之间的相似度。列出文档原文，并且给出相似度值。

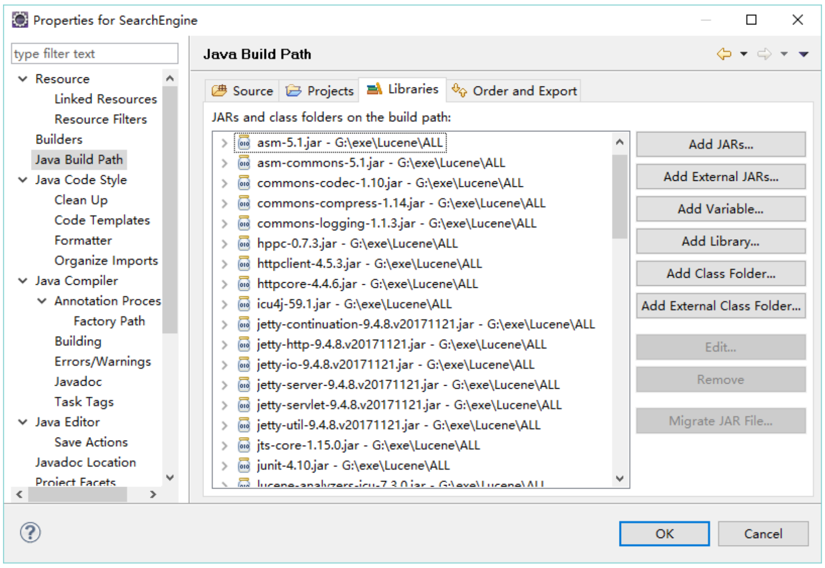
（iii）对下载的文档，利用 K-Means 聚类算法进行聚类

1. 将下载的 500 个中/英文文档聚为 20 个类，并显示聚类之后所形成的三个最大的类，及每个类中代表性的文档（即，离类中心最近的五个文档）。
2. 距离计算公式，可采用余弦距离或者欧式距离（Euclidean metric）。

2. 项目设计与结果可视化

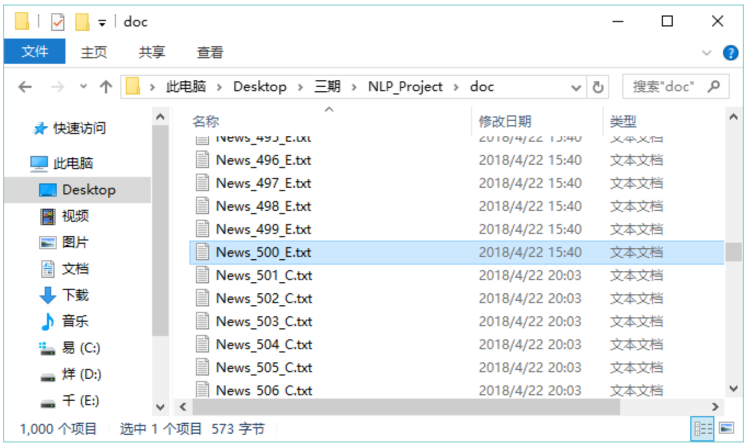
2.1 lucene 安装与导入

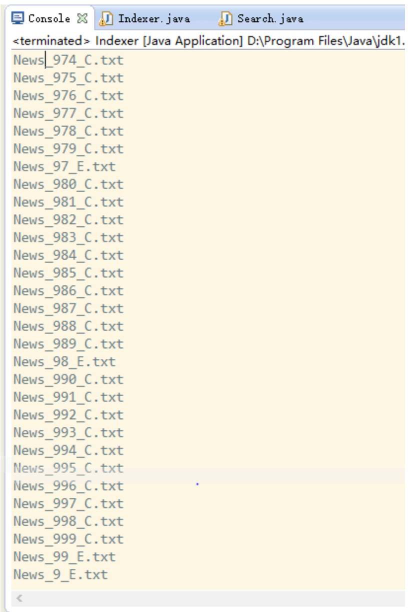
在 eclipse 中，导入外部的 lucene 包。



2.2 文本建立索引

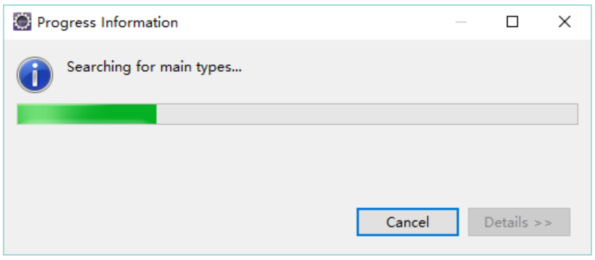
对 C:\Users\Administrator\Desktop\三期\NLP_Project\doc 目录下的文本文件建立索引，代码在本文档最后。

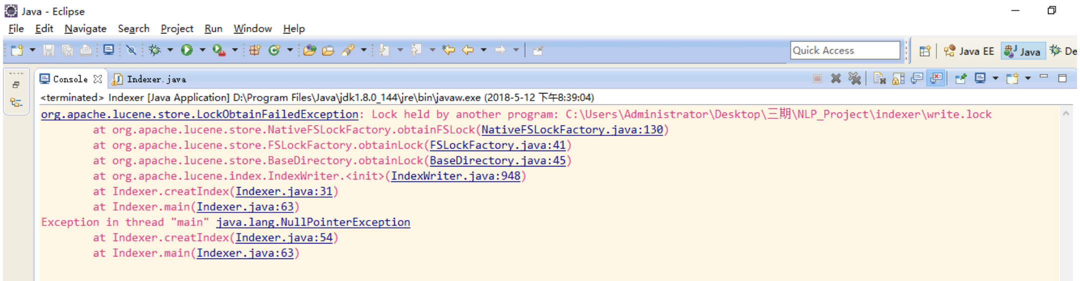




2.3 文本搜索功能

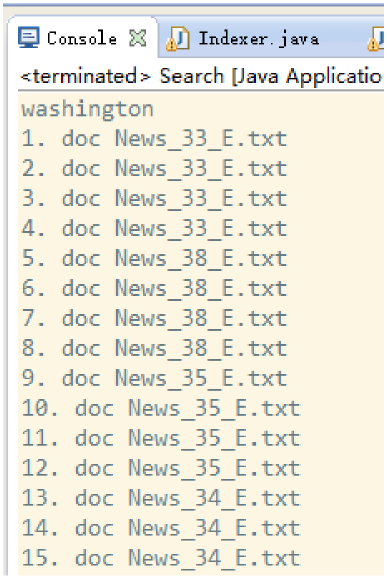
对 1000 个文档实现搜索功能，代码在最后。



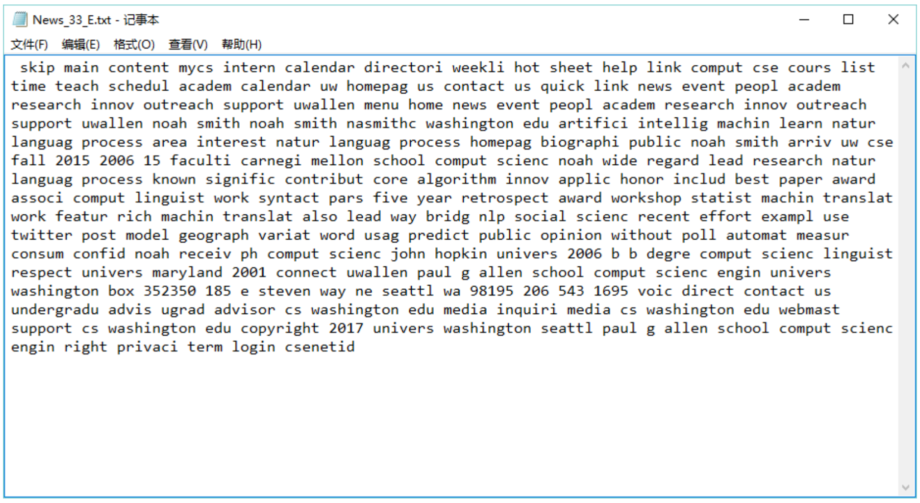


```
<terminated> Indexer [Java Application] D:\Program Files\Java\jdk1.8.0_144\jre\bin\javaw.exe (2018-5-12 下午8:39:04)
org.apache.lucene.store.LockObtainFailedException: Lock held by another program: C:\Users\Administrator\Desktop\三期\NLP_Project\indexer\write.lock
	at org.apache.lucene.store.NativeFSLockFactory.obtainFSLock(NativeFSLockFactory.java:130)
	at org.apache.lucene.store.FSLockFactory.obtainLock(FSLockFactory.java:41)
	at org.apache.lucene.store.BaseDirectory.obtainLock(BaseDirectory.java:45)
	at org.apache.lucene.index.IndexWriter.<init>(IndexWriter.java:948)
	at Indexer.creatIndex(Indexer.java:31)
	at Indexer.main(Indexer.java:63)
Exception in thread "main" java.lang.NullPointerException
	at Indexer.creatIndex(Indexer.java:54)
	at Indexer.main(Indexer.java:63)
```

搜索 washington 之后显示的文档排列



```
<terminated> Search [Java Applicatio
washington
1. doc News_33_E.txt
2. doc News_33_E.txt
3. doc News_33_E.txt
4. doc News_33_E.txt
5. doc News_38_E.txt
6. doc News_38_E.txt
7. doc News_38_E.txt
8. doc News_38_E.txt
9. doc News_35_E.txt
10. doc News_35_E.txt
11. doc News_35_E.txt
12. doc News_35_E.txt
13. doc News_34_E.txt
14. doc News_34_E.txt
15. doc News_34_E.txt
```



```
News_33_E.txt - 记事本
 skip main content mycs intern calendar directori weekli hot sheet help link comput cse cours list
time teach schedul academ calendar uw homepag us contact us quick link news event peopl academ
research innov outreach support uwallen menu home news event peopl academ research innov outreach
support uwallen noah smith noah smith nasmithc washington edu artifici intellig machin learn natur
languag process area interest natur languag process homepag biographi public noah smith arriv uw cse
fall 2015 2006 15 faculti carnegi mellon school comput scienc noah wide regard lead research natur
languag process known signific contribut core algorithm innov applic honor includ best paper award
associ comput linguist work syntact pars five year retrospect award workshop statist machin translat
work featur rich machin translat also lead way bridg nlp social scienc recent effort exampl use
twitter post model geograph variat word usag predict public opinion without poll automat measur
consum confid noah receiv ph comput scienc john hopkin univers 2006 b b degre comput scienc linguist
respect univers maryland 2001 connect uwallen paul g allen school comput scienc engin univers
washington box 352350 185 e steven way ne seattl wa 98195 206 543 1695 voic direct contact us
undergradu advis ugrad advisor cs washington edu media inquiri media cs washington edu webmast
support cs washington edu copyright 2017 univers washington seattl paul g allen school comput scienc
engin right privaci term login csenetid
```

3. 求解余弦相似度

3.1 分析

余弦相似度分析求解参考[2]。为了展示效果，我先采用文件 1, 2 来分步展示。整体和最终的效果在 3.2

首先从键盘输入两个数字，eg: 1, 2, 即想要打开的两个文件的数字。
然后把要打开的两个文件放入 filenames[ ]的 List 中。
示例读取显示 file1，代码和结果如下：

```
In  [9]:
str1 = raw_input("Input file number: ")
str2 = raw_input("Input file number: ")

file1 = "News_" + str1 + "_E.txt"
file2 = "News_" + str2 + "_E.txt"

Input file number: 1
Input file number: 2
```

```
In  [12]:
filenames = ['./doc/' + file1, './doc/' + file2]#relative filepath
f1 = open(filenames[0], 'r').read()
f1

Out[12]:
'\xef\xbb\xbf christoph man thoma siebel professor machin learn professor linguist com
put scienc stanford univers bio christoph man inaugr thoma siebel professor machin learn depart comput scienc linguist stanford univers r
esearch goal comput intellig process understand gener human languag materi man leader appli deep learn natur languag process well known r
esearch tree recurs neural network sentiment analysi neural network depend pars glove model word vector neural machin translat deep langu
ag understand also focus comput linguist approach pars robust textual infer multilingu languag process includ princip develop stanford de
pend univers depend man coauthor lead textbook statist approach natur languag process nlp man sch tze 1999 inform retriev man raghavan sc
h tze 2008 well linguist monograph erg complex predic acm fellow aaai fellow acl fellow past presid acl research acl cole emnlp chi best
paper award b hon australian nation univers ph stanford 1994 held faculti posit carnegi mellon univers univers sydney return stanford fou
nder stanford nlp group stanfordnlp manag develop stanford corenlp softwar contact dept comput scienc gate build 2a 353 serra mall stanfo
rd ca 94305 9020 usa e man cs stanford edu chrman w 1 650 723 7683 f 1 650 725 1449 r gate 248 appoint grayc ujihara gate 215 gujihara st
anford edu brief cv australian come land wide open space ba hon australian nation univers 1989 major mathemat comput scienc linguist phd
stanford linguist 1994 asst professor carnegi mellon univers comput linguist program 1994 96 lectur univers sydney dept linguist 1996 99
asst professor stanford univers dept comput scienc linguist 1999 2006 assoc professor stanford univers dept linguist comput scienc 2006 2
012 professor stanford univers dept linguist comput scienc 2012 presid associ comput linguist 2015 paper older paper avail public list be
com lazi recent stuff often better nlp group public page thing might date googl scholar semant scholar microsoft academ search book intro
duct inform retriev hinrich sch tze prabhakar raghavan cambridg univers press 2008 man sch tze foundat statist natur languag process mit
press 1999 complex predic inform spread lfg 1999 erg argument structur grammat relat 1996 confer talk talk avail onlin 2013 one program c
o chair new intern confer learn represent see iclr 2013 2013 edit realli good workshop scale event sinc grow size exponenti 2013 help org
an first cvsc workshop realli live workshop also help organ second workshop continu vector space model composition eacl 2014 help organ w
```

由于上述文本存在 byte 字符，所以需要进一步过滤。

引入 NLTK 中的分词函数，把 file1 的内容转化成 text1，一个单词是 text1[ ]中的一个元素。

```
In  [20]:
#partition terms and calculate Term-Frequency
from nltk.tokenize import word_tokenize
text1 = word_tokenize(f1)
print text1

['\xef', '\xbb', '\xbf', 'christoph', 'man', 'thoma', 'siebel', 'professor', 'machin', 'learn', 'professor', 'linguist', 'comput', 'scien
c', 'natur', 'languag', 'process', 'group', 'linguist', 'comput', 'scienc', 'stanford', 'univers', 'bio', 'christoph', 'man', 'inaugr',
'thoma', 'siebel', 'professor', 'machin', 'learn', 'depart', 'comput', 'scienc', 'linguist', 'stanford', 'univers', 'research', 'goal',
'comput', 'intellig', 'process', 'understand', 'gener', 'human', 'languag', 'materi', 'man', 'leader', 'appli', 'deep', 'learn', 'natur',
'languag', 'process', 'well', 'known', 'research', 'tree', 'recurs', 'neural', 'network', 'sentiment', 'analysi', 'neural', 'network', 'd
epend', 'pars', 'glove', 'model', 'word', 'vector', 'neural', 'machin', 'translat', 'deep', 'languag', 'understand', 'also', 'focus', 'co
mput', 'linguist', 'approach', 'pars', 'robust', 'textual', 'infer', 'multilingu', 'languag', 'process', 'includ', 'princip', 'develop',
'stanford', 'depend', 'univers', 'depend', 'man', 'coauthor', 'lead', 'textbook', 'statist', 'approach', 'natur', 'languag', 'process',
'nlp', 'man', 'sch', 'tze', '1999', 'inform', 'retriev', 'man', 'raghavan', 'sch', 'tze', '2008', 'well', 'linguist', 'monograph', 'erg',
'complex', 'predic', 'acm', 'fellow', 'aaai', 'fellow', 'acl', 'fellow', 'past', 'presid', 'acl', 'research', 'acl', 'cole', 'emnlp', 'ch
i', 'best', 'paper', 'award', 'b', 'hon', 'australian', 'nation', 'univers', 'ph', 'stanford', '1994', 'held', 'faculti', 'posit', 'carne
```

以下要计算两个文本(file1, file2)中的每个单词的词频。
文本特征提取的方法参考[4]
首先统计两个文本中所有的单词，放在 all_words[ ]中，打印出 list 长度为 1354。
然后用 set( )去掉里面重复出现的单词，即可得到两个文本中的所有单词。
现在长度为 660。

```
In  [49]:
all_words = []
for i in text1:
    all_words.append(i)

f2 = open(filenames[1], 'r').read()
text2 = word_tokenize(f2)
for i in text2:
    all_words.append(i)
len(all_words)

Out[49]: 1354
```

```
In  [54]:
#列出所有的词 去掉重复
all_ = list(set(all_words))
len(all_)

Out[54]: 660
```

对于 all_[ ]中的每一个单词，分别计算它在 text1 和 text2 中出现的次数，即为词频。[1]

打印展示出 text1 的词频向量 vector1:

```
In [56]: #构造词向量
vector1 = []
vector2 = []
for elm in all_:
    num = text1.count(elm) #in text1, count the number of every words in all_
    vector1.append(num)

for elm in all_:
    num = text2.count(elm) #in text1, count the number of every words in all_
    vector2.append(num)

print vector1
```

```
[1, 0, 5, 1, 1, 1, 0, 0, 6, 1, 7, 0, 5, 0, 0, 0, 0, 5, 1, 3, 2, 0, 0, 0, 1, 4, 0, 4, 2, 2, 2, 0, 0, 1, 2, 2, 1, 2, 0, 1, 1, 2, 1, 1, 10, 0,
1, 1, 1, 1, 2, 1, 1, 0, 2, 0, 1, 2, 3, 1, 0, 2, 1, 1, 2, 0, 0, 2, 1, 0, 1, 1, 2, 0, 2, 0, 0, 0, 1, 0, 1, 0, 0, 0, 2, 1, 2, 0, 1, 0, 1, 0, 4,
1, 1, 1, 14, 3, 2, 1, 1, 1, 0, 3, 1, 0, 2, 0, 2, 1, 1, 1, 2, 0, 1, 1, 1, 2, 1, 1, 0, 0, 1, 3, 1, 6, 1, 1, 1, 2, 9, 0, 12, 1, 1, 0, 1, 0, 1,
1, 1, 1, 3, 1, 5, 7, 1, 2, 0, 2, 1, 3, 1, 0, 1, 1, 0, 1, 1, 1, 1, 11, 0, 2, 7, 0, 0, 0, 2, 0, 4, 1, 0, 2, 4, 0, 1, 0, 1, 1, 0, 7, 0, 1, 2,
1, 0, 1, 2, 0, 2, 2, 5, 4, 2, 1, 1, 1, 0, 2, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 2, 1, 1, 0, 1, 3, 1, 1, 3, 1, 1, 2, 9, 3, 1, 1, 3, 1, 2, 0,
0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 4, 0, 1, 1, 3, 0, 0, 0, 1, 0, 1, 2, 0, 1, 2, 1, 1, 13, 2, 3, 0, 1, 1, 1, 1, 2, 0, 1, 1, 0, 0,
1, 1, 2, 1, 1, 1, 2, 1, 1, 1, 0, 2, 1, 1, 1, 0, 5, 1, 0, 2, 1, 0, 0, 0, 1, 1, 3, 1, 1, 2, 0, 1, 1, 0, 0, 15, 1, 2, 1, 1, 0, 1, 1, 1, 1, 0,
0, 0, 0, 3, 0, 2, 0, 2, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 7, 0, 1, 0, 0, 1, 0, 12, 1, 0, 3, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 1, 1, 2, 0,
2, 4, 2, 0, 0, 2, 1, 4, 1, 0, 25, 1, 0, 1, 0, 1, 3, 0, 0, 0, 1, 2, 1, 0, 2, 1, 3, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 2, 1, 1, 0, 1,
4, 1, 0, 1, 9, 1, 1, 1, 0, 1, 0, 5, 0, 2, 1, 2, 0, 0, 0, 1, 0, 4, 4, 0, 3, 1, 0, 1, 1, 2, 1, 1, 1, 5, 1, 0, 0, 1, 0, 1, 0, 5, 1, 1, 10, 3,
0, 4, 0, 2, 1, 1, 1, 0, 0, 3, 1, 0, 0, 0, 2, 0, 0, 1, 0, 2, 0, 0, 0, 1, 0, 0, 0, 1, 1, 3, 0, 3, 1, 3, 1, 2, 1, 1, 1, 1, 0, 0, 0, 0, 0, 4, 0,
1, 15, 1, 1, 2, 0, 0, 1, 1, 1, 2, 1, 1, 1, 0, 3, 1, 2, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 2, 2, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0,
1, 1, 3, 1, 1, 0, 2, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 4, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 2, 0, 1, 2, 2, 1, 1, 1, 6, 1, 0, 1, 3, 0, 0, 1, 1, 1,
1, 1, 5, 0, 1, 1, 2, 23, 0, 1, 1, 2, 1, 0, 0, 1, 0, 0, 0, 0, 1, 2, 0, 1, 0, 2, 1, 0, 0, 1, 1, 1, 2, 3, 0, 1, 0, 0, 0, 1, 1, 1, 1, 2, 1, 1,
1, 3, 1, 0, 2, 1, 2, 2, 2, 0, 5, 2]
```

构造词频向量进行矩阵运算需要引入 numpy，此时 vector1 的元素类型是 int。用余弦距离公式计算相似度值：[3]

```
In [52]: type(vector1[0])
Out[52]: int

In [53]: import numpy as np

vectorA = np.array(vector1)
vectorB = np.array(vector2)

op = np.dot(vectorA, vectorB)/(np.linalg.norm(vectorA)*(np.linalg.norm(vectorB)))
print(op)

0.374238023171
```

3.2 整体效果

输入 1，5，计算文档 1 和 5 的相似度值为 0.325586992076，并显示文本内容如下，与 txt 文件做对比。

```
#引入numpy 计算余弦距离
import numpy as np

vectorA = np.array(vector1)
vectorB = np.array(vector2)

op = np.dot(vectorA, vectorB)/(np.linalg.norm(vectorA)*(np.linalg.norm(vectorB)))
print("Cosine Distance: ")
print(op)
print('\n')
print("News_" + str1 + "_E.txt")
print(f1)
print('\n')
print("News_" + str2 + "_E.txt")
print(f2)

Input file number: 1

Input file number: 5
```

```python
#引入numpy 计算余弦距离
import numpy as np

vectorA = np.array(vector1)
vectorB = np.array(vector2)

op = np.dot(vectorA, vectorB)/(np.linalg.norm(vectorA)*(np.linalg.norm(vectorB)))
print("Cosine Distance: ")
print(op)
print('\n')
print("News_" + str1 + "_E.txt")
print(f1)
print('\n')
print("News_" + str2 + "_E.txt")
print(f2)
```

```
Input file number: 1
Input file number: 5
Cosine Distance:
0.325586992076


News_1_E.txt
 christoph man thoma siebel professor machin learn professor linguist comput scienc natur languag process group linguist comput scienc st
anford univers bio christoph man inaugr thoma siebel professor machin learn depart comput scienc linguist stanford univers research goal
comput intellig process understand gener human languag materi man leader appli deep learn natur languag process well known research tree
recurs neural network sentiment analysi neural network depend pars glove model word vector neural machin translat deep languag understand
also focus comput linguist approach pars robust textual infer multilingu languag process includ princip develop stanford depend univers d
epend man coauthor lead textbook statist approach natur languag process nlp man sch tze 1999 inform retriev man raghavan sch tze 2008 wel
```

```python
#引入numpy 计算余弦距离
import numpy as np

vectorA = np.array(vector1)
vectorB = np.array(vector2)

op = np.dot(vectorA, vectorB)/(np.linalg.norm(vectorA)*(np.linalg.norm(vectorB)))
print("Cosine Distance: ")
print(op)
print('\n')
print("News_" + str1 + "_E.txt")
print(f1)
print('\n')
print("News_" + str2 + "_E.txt")
print(f2)
```

```
taught ling 289 quantit probabilist explan linguist mw 2 15 3 45 160 318 previous taught winter 2002 n e ling 236 winter 2005 ling 235 su
mmer 2007 taught lsa linguist institut statist pars comput linguist industri fall 1999 winter 2001 taught cs 121 artifici intellig text b
ook russel p norvig artifici intellig modern approach ran nlp read group 1999 2002 nlp read group student organ stuff latex use time e gr
ad student use spend write la tex macro actual lie still spend time got two son joel casey opinion book kid http www stanford edu man chr
istoph man man cs stanford edu last modifi nov 19 2016


News_5_E.txt
 dan jurafski professor chair linguist professor comput scienc stanford univers studi comput linguist natur languag process applic behavi
or social scienc past macarthur fellow also write teach languag food jurafski stanford edu margaret jack 117 stanford ca 94305 2150 twitt
er jurafski spring offic hour tue 4 00 5 00 gone may 1 jun 5 bio cv student nlp group dan languag food blog class articl press teach nbsp
winter 2018 cs124 ling180 languag inform tu thu 3 00 4 20 hewlett 200 ling197a undergrad capston seminar fri 12 30pm 2 20pm ivan sag room
autumn 2017 ling294 grad proseminar fri 10 30 11 20am ivan sag room previou cours materi materi nlp mooc chri man book 3rd edn draft chap
```

字体

Desktop › 三期 › NLP_Project › doc

名称	修改日期
News_1_E.txt	2018/5/13
News_2_E.txt	2018/4/22
News_3_E.txt	2018/5/8
News_4_E.txt	2018/4/22
News_5_E.txt	2018/4/22
News_6_E.txt	2018/4/22
News_7_E.txt	2018/4/22
News_8_E.txt	2018/4/22
News_9_E.txt	2018/4/22
News_10_E.txt	2018/4/22
News_11_E.txt	2018/4/22
News_12_E.txt	2018/4/22
News_13_E.txt	2018/4/22
News_14_E.txt	2018/4/22
News_15_E.txt	2018/4/22

1.58 KB

News_1_E.txt - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
christoph man thoma siebel professor machin learn professor linguist comput
scienc natur languag process group linguist comput scienc stanford univers bio
christoph man inaugr thoma siebel professor machin learn depart comput scienc
linguist stanford univers research goal comput intellig process understand gener
human languag materi man leader appli deep learn natur languag process well
known research tree recurs neural network sentiment analysi neural network
depend pars glove model word vector neural machin translat deep languag
understand also focus comput linguist approach pars robust textual infer
multilingu languag process includ princip develop stanford depend univers depend
man coauthor lead textbook statist approach natur languag process nlp man sch
tze 1999 inform retriev man raghavan sch tze 2008 well linguist monograph erg
complex predic acm fellow aaai fellow acl fellow past presid acl research acl
cole emnlp chi best paper award b hon australian nation univers ph stanford 1994
```

News_5_E.txt - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
 dan jurafski professor chair linguist professor comput scienc stanford univers
studi comput linguist natur languag process applic behavior social scienc past
macarthur fellow also write teach languag food jurafski stanford edu margaret
jack 117 stanford ca 94305 2150 twitter jurafski spring offic hour tue 4 00 5 00
gone may 1 jun 5 bio cv student nlp group dan languag food blog class articl
press teach nbsp winter 2018 cs124 ling180 languag inform tu thu 3 00 4 20
hewlett 200 ling197a undergrad capston seminar fri 12 30pm 2 20pm ivan sag room
autumn 2017 ling294 grad proseminar fri 10 30 11 20am ivan sag room previou
cours materi materi nlp mooc chri man book 3rd edn draft chapter speech languag
process dan jurafski jame h martin jame beard award nomine 2018 articl pub googl
```

在 1，350，360 文档之间两两比较。

```
print("News_"  + str1 + "_E.txt")
print(f1)
print('\n')
print("News_"  + str2 + "_E.txt")
print(f2)
```

```
Input file number: 1
Input file number: 350
Cosine Distance:
0.0680498208154


News_1_E.txt
christoph man thoma siebel professor machin learn professor linguist comput scienc natur languag process group linguist comput scienc st
anford univers bio christoph man inaugr thoma siebel professor machin learn depart comput scienc linguist stanford univers research goal
 comput intellig process understand gener human languag materi man leader appli deep learn natur languag process well known research tre
e recurs neural network sentiment analysi neural network depend pars glove model word vector neural machin translat deep languag underst
and also focus comput linguist approach pars robust textual infer multilingu languag process includ princip develop stanford depend univ
ers depend man coauthor lead textbook statist approach natur languag process nlp man sch tze 1999 inform retriev man raghavan sch tze 20
08 well linguist monograph erg complex predic acm fellow aaai fellow acl fellow past presid acl research acl cole emnlp chi best paper a
ward b hon australian nation univers ph stanford 1994 held faculti posit carnegi mellon univers univers sydney return stanford founder s
tanford nlp group stanfordnlp manag develop stanford corenlp softwar contact dept comput scienc gate build 2a 353 serra mall stanford ca
 94305 9020 usa e man cs stanford edu chrman w 1 650 723 7683 f 1 650 725 1449 r gate 248 appoint grayc ujihara gate 215 gujihara stanfo
rd edu brief cv australian come land wide open space ba hon australian nation univers 1989 major mathemat comput scienc linguist phd sta
nford linguist 1994 asst professor carnegi mellon univers comput linguist program 1994 96 lectur univers sydney dept linguist 1996 99 as
```

```
print('\n')
print("News_"  + str1 + "_E.txt")
print(f1)
print('\n')
print("News_"  + str2 + "_E.txt")
print(f2)
```

```
Input file number: 1
Input file number: 360
Cosine Distance:
0.0555615685497


News_1_E.txt
christoph man thoma siebel professor machin learn professor linguist comput scienc natur languag process group linguist comput scienc st
anford univers bio christoph man inaugr thoma siebel professor machin learn depart comput scienc linguist stanford univers research goal
 comput intellig process understand gener human languag materi man leader appli deep learn natur languag process well known research tre
e recurs neural network sentiment analysi neural network depend pars glove model word vector neural machin translat deep languag underst
and also focus comput linguist approach pars robust textual infer multilingu languag process includ princip develop stanford depend univ
ers depend man coauthor lead textbook statist approach natur languag process nlp man sch tze 1999 inform retriev man raghavan sch tze 20
08 well linguist monograph erg complex predic acm fellow aaai fellow acl fellow past presid acl research acl cole emnlp chi best paper a
ward b hon australian nation univers ph stanford 1994 held faculti posit carnegi mellon univers univers sydney return stanford founder s
tanford nlp group stanfordnlp manag develop stanford corenlp softwar contact dept comput scienc gate build 2a 353 serra mall stanford ca
 94305 9020 usa e man cs stanford edu chrman w 1 650 723 7683 f 1 650 725 1449 r gate 248 appoint grayc ujihara gate 215 gujihara stanfo
rd edu brief cv australian come land wide open space ba hon australian nation univers 1989 major mathemat comput scienc linguist phd sta
nford linguist 1994 asst professor carnegi mellon univers comput linguist program 1994 96 lectur univers sydney dept linguist 1996 99 as
```

```
op = np.dot(vectorA,vectorB)/(np.linalg.norm(vectorA)*(np.linalg.norm(vectorB)))
print("Cosine Distance: ")
print(op)
print('\n')
print("News_"  + str1 + "_E.txt")
print(f1)
print('\n')
print("News_"  + str2 + "_E.txt")
print(f2)
```

```
Input file number: 350
Input file number: 360
Cosine Distance:
0.840012584377


News_350_E.txt
 jame livo studi busi manag market kingston upon thame 2006 answer apr 17 2018 author 294 answer 34 6k answer viewswhil take admiss coll
eg essay requir reflect candid part applic essay provid scope opportun understand know candid write therefor admiss essay good opportun
 tell candid could fit part applic similar section curriculum vita write abil candid write provid time space also known applic essay the
refor essay let colleg understand reflect well applic side aspir candid candid often tend glorifi colleg write amount dedic skill requir
 becom doctor engin howev approach colleg look assign help besid also motiv behind admiss essay submit good admiss essay prompt experi s
tori inner reflect candid capitalis writer experi view thought idea toward stream studi therefor first rule write admiss essay self next
 rule expert start earli reflect candid captur success admiss essay choic topic utmost import admiss essay choic must base best topic wo
uld reflect candid admiss essay definit goe beyond mere reflect essay essay write meet candid characterist person stori thought thing ma
tter admiss essay best idea utilis creativ approach focu area cannot known read form meet candid person essay goe beyond current record
 certif reli reflect present candid pen paper reason behind essay judg much candid express utilis one creativ content essay focus narrow
```

4. 文本聚类

4.1 scikit-learn 的 k-means 介绍

文本聚类参考[5]，常用文本聚类工具参考[6]。

K-Means 类实例化方式为：实例=KMeans(n_clusters=8, init='k-means++', n_init=10, max_iter=300, tol=0.0001, precompute_distances='auto', verbose=0, random_state=None, copy_x=True, n_jobs=1, algorithm='auto')

参数说明：

n_clusters：聚类数，也是需要初始化的类中心的个数，默认取值为 8

max_iter：一次聚类算法所执行的最大迭代次数，默认取值为 300

n_init：使用不同的初始化类中心进行聚类的次数，最终输出结果为几次聚

类中效果最好的，以组内距离衡量

4.2 词频矩阵标准化处理

```
In [1]: '''''
sklearn里面的TF-IDF主要用到了两个函数：CountVectorizer()和TfidfTransformer()。
    CountVectorizer是通过fit_transform函数将文本中的词语转换为词频矩阵。
    矩阵元素weight[i][j] 表示j词在第i个文本下的词频，即各个词语出现的次数。
    通过get_feature_names()可看到所有文本的关键字，通过toarray()可看到词频矩阵的结果。
    TfidfTransformer也有个fit_transform函数，它的作用是计算tf-idf值。
'''

import time
import re
import os
import sys
import codecs
import shutil
import numpy as np
from sklearn import feature_extraction
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfTransformer

#将文本中的词语转换为词频矩阵 矩阵元素a[i][j] 表示j词在i类文本下的词频
#vectorizer = CountVectorizer()
#该类会统计每个词语的tf-idf权值
vectorizer = CountVectorizer()
#vectorizer  = TfidfVectorizer()
transformer = TfidfTransformer()
#直接用正则表达式切分，若用tokenize会出现byte字符，无法用utf8 decode
#from nltk.tokenize import word_tokenize

A = [] #一个elm为一个文档的list
```

首先打开所有 500 个英文文档，把每一个文档中的内容先进行分词，然后再都放进 A[ ]，这样就构成一个二维矩阵，row 为文档的个数，1~500，column 为每一个文档中的每一个单词。最后打印 A[499]就显示的是最后一个文档的内容。

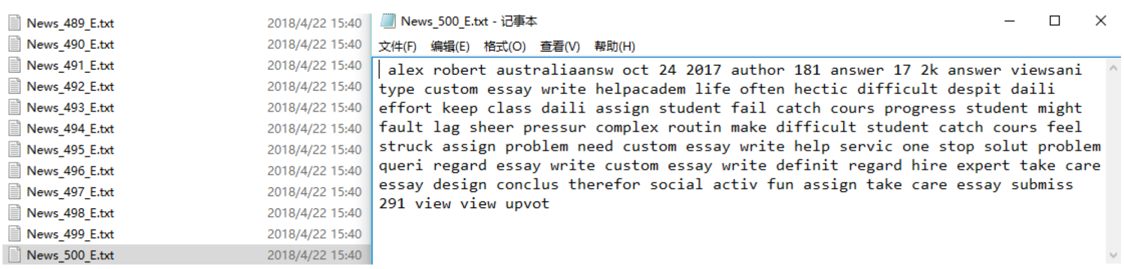
```
num = 1
while(num<501):
    f1 = open('./doc/' + "News_"  + str(num) + "_E.txt", 'r').read()
    reg = re.compile('\\W*')#除了单词外的所有特殊符号包括空格
    text = reg.split(f1)
    B = " ".join(text)
    #type(B)
    A.append(B)
    num += 1

print A[499]
```

```
 alex robert australiaansw oct 24 2017 author 181 answer 17 2k answer viewsani type custom essay write helpacadem life often hectic difficul
t despit daili effort keep class daili assign student fail catch cours progress student might fault lag sheer pressur complex routin make di
fficult student catch cours feel struck assign problem need custom essay write help servic one stop solut problem queri regard essay write c
ustom essay write definit regard hire expert take care essay design conclus therefor social activ fun assign take care essay submiss 291 vie
w view upvot
```

```
In [2]: print len(A)
```

```
500
```

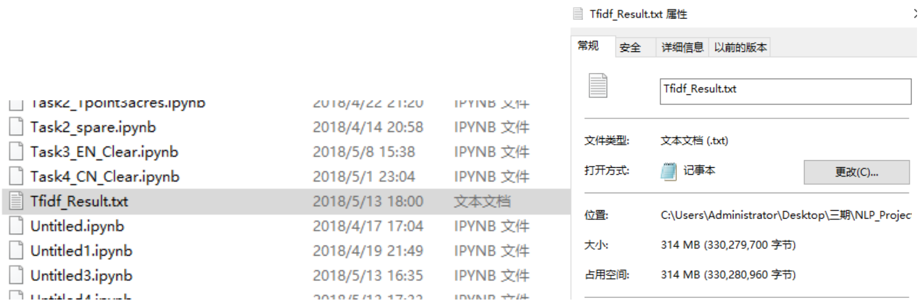


用 Tfidf 函数将 A[ ]转化成词频矩阵，这时打印显示出来大多数是 0.00，在这里显示不完全。

```
In [3]: #第一个fit_transform是计算tf-idf 第二个fit_transform是将文本转为词频矩阵
        tfidf = transformer.fit_transform(vectorizer.fit_transform(A)).toarray()
        tfidf
        #将tf-idf矩阵抽取出来，元素w[i][j]表示j词在i类文本中的tf-idf权重
        #weight = tfidf.toarray()
        #print weight

Out[3]: array([[ 0.,  0.,  0., ...,  0.,  0.,  0.],
               [ 0.,  0.,  0., ...,  0.,  0.,  0.],
               [ 0.,  0.,  0., ...,  0.,  0.,  0.],
               ...,
               [ 0.,  0.,  0., ...,  0.,  0.,  0.],
               [ 0.,  0.,  0., ...,  0.,  0.,  0.],
               [ 0.,  0.,  0., ...,  0.,  0.,  0.]])
```

我把词频矩阵保存在了文本中，但文件有 314MB，我的电脑只有 4GB 内存，根本无法打开这个大文件。




```
In [7]: resName = "Tfidf_Result.txt"
        result = codecs.open(resName, 'w', 'utf-8')
        #打印每类文本的tf-idf词语权重，第一个for遍历所有文本，第二个for便利某一类文本下的词语权重
        for i in range(len(tfidf)):
            result.write(u"-------这里输出第" + str(i) + u"类文本的词语tf-idf权重------")
            for j in range(len(tfidf[i])):
                result.write(str(tfidf[i][j]) + ' ')
            result.write('\r\n\r\n')

        result.close()
```

对数据进行标准化处理之后，这时我们可以看到，矩阵并不都是 0.00。

```
In [4]: # rescale the data: mean 0, std:1
        from sklearn.preprocessing import StandardScaler
        scaler = StandardScaler()
        scaler.fit(tfidf)
        scaled_df = scaler.transform(tfidf)
        scaled_df

Out[4]: array([[-0.08725465, -0.11853788, -0.05542822, ..., -0.04476615,
                -0.04476615, -0.04476615],
               [-0.08725465, -0.11853788, -0.05542822, ..., -0.04476615,
                -0.04476615, -0.04476615],
               [-0.08725465, -0.11853788, -0.05542822, ..., -0.04476615,
                -0.04476615, -0.04476615],
               ...,
               [-0.08725465, -0.11853788, -0.05542822, ..., -0.04476615,
                -0.04476615, -0.04476615],
               [-0.08725465, -0.11853788, -0.05542822, ..., -0.04476615,
                -0.04476615, -0.04476615],
               [-0.08725465, -0.11853788, -0.05542822, ..., -0.04476615,
                -0.04476615, -0.04476615]])
```

4.3 聚成 20 类

```
In [55]: from sklearn.cluster import KMeans
```

```
In [56]: kmeans = KMeans()
         kmeans.set_params(n_clusters = 20)
         kmeans.fit(scaled_df)

Out[56]: KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
             n_clusters=20, n_init=10, n_jobs=1, precompute_distances='auto',
             random_state=None, tol=0.0001, verbose=0)
```

```
In [59]: label = kmeans.labels_
         print label

[ 4  4  4  4  4  4  4 15  4  4  4  4  4  4  4  8  7  4  4  4 19  4  4  4  4
  4  4  4  4  4  4  4  4  4  4  4  4  4  2  4  4  4  4  4  4  4  4  4  4  4
  4  4  4  4  4  4  4  1  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4
  4 14  4  3  4  4 18 17  4  4 18 17  4  4  4  4  4  4  4  4  4  4  4  4  4
  5  4  4  4  4  4  4 16  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  9
  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4 10
  4  4  4  4  4 12  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4
  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4
  4  4  4  4  4  4  4  4 11  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4
  4  4  4  4  4  4 13  4  4  4  6  4  4  4  4  4  4  4  4  4  4  4  4  4  4
  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  0  4  4  4  4  4  4  4  4  4
  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4
  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4
  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4
  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4
  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4
  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4
  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4
  4  4  4  4  4  4  4  4  4  4  4  4  0  4  4  4  4  4  4  4  4  4  4  4  4
  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4]
```

```
In [60]: mylist1 = list(label)
         myset1  = set(mylist1)
         dict1 = {}
         for item in myset1:
             dict1.update({item : mylist1.count(item)})
         dict1

Out[60]: {0: 2,
          1: 1,
          2: 1,
          3: 1,
          4: 478,
          5: 1,
          6: 1,
          7: 1,
          8: 1,
          9: 1,
          10: 1,
          11: 1,
          12: 1,
          13: 1,
          14: 1,
          15: 1,
          16: 1,
          17: 2,
          18: 2,
          19: 1}
```

最后统计每一类中包含的文本个数，478 的文本归为一大类别。

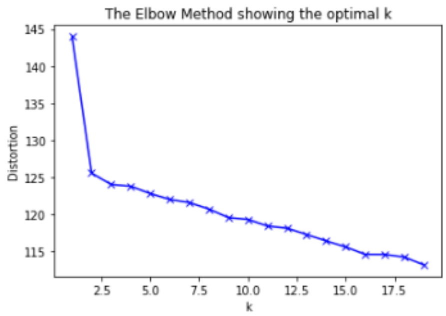
4.4 肘部法则寻找最优 K 值

肘部法则参考 sklearn 的电子书。由于我电脑配置低，而词频矩阵维度高，下面这段代码运行了近 1.5 个小时。

下图曲线类似于人的手肘，“肘关节”部分对应的 K 值就是最恰当的 K 值，但是并不是所有代价函数曲线都存在明显的“肘关节”。下图中很明显 K = 2，应该聚

为 2 类。

In [6]:
```
#Determine optimal k
from sklearn import metrics
from scipy.spatial.distance import cdist
import matplotlib.pyplot as plt
# k means determine k
distortions = []
K = range(1,20)
for k in K:
    kmeanModel = KMeans(n_clusters = k).fit(scaled_df)
    kmeanModel.fit(scaled_df)
    distortions.append(sum(np.min(cdist(scaled_df, kmeanModel.cluster_centers_, 'euclidean'), axis=1)) / scaled_df.shape[0])
# plot the elbow graph
plt.plot(K, distortions, 'bx-')
plt.xlabel('k')
plt.ylabel('Distortion')
plt.title('The Elbow Method showing the optimal k')
plt.show()
```



In [7]:
```
len(distortions)
```

Out[7]:
```
19
```

In [19]:
```
import numpy as np
np.set_printoptions(suppress=True) #不用科学计数法

L1=distortions[0:len(distortions)-1]
L2=distortions[1:len(distortions)]
ret = map(lambda x, y: (x-y)/x  , L1, L2)
ret
```

Out[19]:
```
[0.1277202963026785,
 0.012149603983026 79,
 0.0019203939844369808,
 0.007983892389397383 5,
 0.006474542072765454 9,
 0.0034902068436427243,
 0.0070668645507791271,
 0.0096578494186362728,
 0.0020076727061749148,
 0.0074998226591994786,
 0.0023994967461628026,
 0.0072417936428425386,
 0.0072935816169752881,
 0.0070651118469353142,
 0.0086059457907152 47,
 6.1556549070836383e-05,
 0.0031488206174422606,
 0.0091272611481743107]
```

4.5 聚为 16 类

但我还是计算了一下其余部分的斜率值。除了 2 之外，下降最明显的是 15~16，所以试着把 K 取值为 16。聚类效果如下：

```
In [18]: kmeans2 = KMeans()
         kmeans2.set_params(n_clusters = 16)
         kmeans2.fit(scaled_df)
         label2 = kmeans2.labels_
         print label2
```

```
[ 0  0  0  0  0  0  0 12  0  0  0  0  0  0  0  4  5  0  0  0  0  0  2  0  0
  0  0  0  0  0  0  0  0  0  0 13  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  7  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0 15  0  0
  0  0  8  0  0  0  0  0  3  0  0  0  3  0  0  0  0  0  0  0  0  0  0  0  0
 11  0  0  0  0  0  0 14  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  9  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 10  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  6  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
```

统计每一个类别中元素的个数，可以看出：属于第 0 类的文本共有 484 个

```
In [21]: mylist = list(label2)
         myset  = set(mylist)
         dict = {}
         for item in myset:
             dict.update({item : mylist.count(item)})
         dict

Out[21]: {0: 484,
          1: 1,
          2: 1,
          3: 2,
          4: 1,
          5: 1,
          6: 1,
          7: 1,
          8: 1,
          9: 1,
          10: 1,
          11: 1,
          12: 1,
          13: 1,
          14: 1,
          15: 1}
```

打印出每一个类别的中心点

```
In [20]: center = kmeans2.cluster_centers_
         print center
```

```
[[ -4.28654070e-03   1.95259706e-03  -1.09614474e-02 ...,  -4.47661481e-02
   -4.47661481e-02  -4.47661481e-02]
 [ -6.50609888e-02  -7.09899589e-03   6.13676381e+00 ...,   2.23383079e+01
    2.23383079e+01   2.23383079e+01]
 [ -8.72546481e-02  -1.18537877e-01  -5.54282176e-02 ...,  -4.47661481e-02
   -4.47661481e-02  -4.47661481e-02]
 ...,
 [ -8.72546481e-02  -1.18537877e-01  -5.54282176e-02 ...,  -4.47661481e-02
   -4.47661481e-02  -4.47661481e-02]
 [ -8.72546481e-02  -1.18537877e-01  -5.54282176e-02 ...,  -4.47661481e-02
   -4.47661481e-02  -4.47661481e-02]
 [  6.58844729e-01  -1.18537877e-01  -5.54282176e-02 ...,  -4.47661481e-02
   -4.47661481e-02  -4.47661481e-02]]
```

打印聚类效果的量化值。

```
In [22]: print kmeans2.inertia_

8936256.67816
```

belong_0 保存了所有第 0 类文本的索引，13 就是 News_13_E.txt

```
In [23]: #找属于第类的的文本索引（即label2的下标）
         belong_0 = np.where(label2 == 0)
         belong_0

Out[23]: (array([  0,   1,   2,   3,   4,   5,   6,   8,   9,  10,  11,  12,  13,
                  14,  17,  18,  19,  20,  21,  23,  24,  25,  26,  27,  28,  29,
                  30,  31,  32,  33,  34,  36,  37,  38,  39,  40,  41,  42,  43,
                  44,  45,  46,  47,  48,  49,  50,  52,  53,  54,  55,  56,  58,
                  59,  60,  61,  62,  63,  64,  65,  66,  67,  68,  69,  70,  71,
                  73,  74,  75,  76,  78,  79,  80,  81,  82,  84,  85,  86,  88,
                  89,  90,  91,  92,  93,  94,  95,  96,  97,  98,  99, 101, 102,
                 103, 104, 105, 106, 108, 109, 110, 111, 112, 113, 114, 115, 116,
                 117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129,
                 130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142,
                 143, 144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154, 155,
                 157, 158, 159, 160, 161, 162, 163, 164, 165, 166, 167, 168, 169,
                 170, 171, 172, 173, 174, 175, 176, 177, 178, 179, 180, 181, 182,
                 183, 184, 185, 186, 187, 188, 189, 190, 191, 192, 193, 194, 195,
                 196, 197, 198, 199, 200, 201, 202, 203, 204, 205, 206, 207, 208,
                 209, 210, 211, 212, 213, 214, 215, 216, 217, 218, 220, 221, 222,
                 223, 224, 225, 226, 227, 228, 229, 230, 231, 232, 233, 234, 236,
                 237, 238, 239, 240, 241, 242, 243, 244, 245, 246, 247, 248, 249,
                 250, 251, 252, 253, 254, 255, 256, 257, 258, 259, 260, 261, 262,
                 263, 264, 265, 266, 267, 268, 269, 270, 271, 272, 273, 274, 275,
```

打印第 0 类的中心点。scaled_df 是标准化之后的点的向量数组，计算所有点到中心点的距离，这里用欧氏距离计算。保存到 Dis[ ]

```
In [24]: center[0]

Out[24]: array([-0.00428654,  0.0019526 , -0.01096145, ..., -0.04476615,
                -0.04476615, -0.04476615])

In [25]: scaled_df[0]

Out[25]: array([-0.08725465, -0.11853788, -0.05542822, ..., -0.04476615,
                -0.04476615, -0.04476615])

In [26]: Dis = []
         for ind in range(0,500):#belong_0[0]:
             op = np.sqrt(np.sum(np.square(center[0]-scaled_df[ind])))#第0类中所有点距离中心点的距离
             Dis.append(op)
         Dis

Out[26]: [163.01150535163464,
          180.09412475725074,
          121.57644580163132,
          138.16418408129007,
          166.29335731634293,
          264.45937522612701,
          145.43227666012336,
          444.67510751437885,
          215.80393960521511,
```

一共 500 个点，共计算了 500 次。为了便于后面统计，再把 Dis[ ]从 list 转化成 numpy.array

```
In [27]: len(Dis)

Out[27]: 500

In [28]: Dis_np = np.array(Dis)
         Dis_np

Out[28]: array([ 163.01150535,  180.09412476,  121.5764458 ,  138.16418408,
                 166.29335732,  264.45937523,  145.43227666,  444.67510751,
                 215.80393961,  257.814128  ,  204.71593351,  203.40742881,
                  78.96900842,  210.04381581,  197.41991526,  395.45277845,
                 757.10281557,  249.73621173,  168.12772779,  118.08126722,
                 321.58878252,  359.70132568,  304.25184115,  182.88330596,
                 163.68603044,  177.98028557,  260.81474175,  150.40005329,
                 149.62586135,  265.31131055,  132.5350114 ,  155.86382284,
                  93.0512059 ,   57.83481452,  150.47313662,  231.07376213,
                 226.89663805,  129.97824569,  278.1663021 ,   59.27561904,
                 163.92892649,  111.99631909,  328.29171805,   83.53272945,
                 103.57026474,  224.06568112,  141.25920097,  133.32421022,
                 183.09388371,  223.16846305,  118.84876809,  396.86430144,
                 115.73486799,  258.10466046,  257.12521623,  249.10825053,
                 157.55126272, 8404.35612346,  115.633171  ,  106.95349824,
                 106.95349824,  167.42816817,  168.62916527,  233.8911418 ,
                 216.95379335,  150.40032356,  271.43983596,  227.01773592,
                 136.97197775,  111.9358896 ,  184.35631529,   26.18503264,
                 301.21495448,  192.56946982,  163.21523733,  114.08544326,
                 328.68570507,  282.94274147,  317.56829918,  130.58661061,
```

用一个升序函数 argsort()把距离从小到大排列，然后返回前五个数即距离中心点距离最近的文本的索引值

```
In [29]: Dis_np.argsort()[:5][::-1]#顺序自始至终没有变过，返回前五个数即距离最小的文本的索引
Out[29]: array([102, 387, 146, 494, 437], dtype=int64)
```

上面显示最近的五个文本依次是:102, 387, 146, 494, 437。下一步将这些文本显示出来。

4.6 显示代表性文档

```
In [33]: str1 = raw_input("Input file number: ")
         str2 = raw_input("Input file number: ")
         str3 = raw_input("Input file number: ")
         str4 = raw_input("Input file number: ")
         str5 = raw_input("Input file number: ")

         file1 = "News_"  + str1 + "_E.txt"
         file2 = "News_"  + str2 + "_E.txt"
         file3 = "News_"  + str3 + "_E.txt"
         file4 = "News_"  + str4 + "_E.txt"
         file5 = "News_"  + str5 + "_E.txt"

         filenames = ['./doc/' + file1, './doc/' + file2, './doc/' + file3, './doc/' + file4, './doc/' + file5]#relative filepath
         for i in range(0,5):
             f1 = open(filenames[i], 'r').read()
             print ("\n")
             print (filenames[i])
             print f1
```

```
Input file number: 102
Input file number: 387
Input file number: 146
Input file number: 494
Input file number: 437


./doc/News_102_E.txt
 sergey levin assist professor uc berkeley eec address 754 sutardja dai hall uc berkeley berkeley ca 94720 1758 email prospect stu
dent pleas read contact thank interest lab howev ask contact directli regard undergradu ms phd admiss abl repli new student join l
ab everi year encourag submit applic uc berkeley eec phd program applic review thoroughli need contact directli alreadi student uc
berkeley encourag get touch uc berkeley undergradu student interest particip research pleas also includ transcript cv assist profe
ssor depart electr engin comput scienc uc berkeley research focu intersect control machin learn aim develop algorithm techniqu end
ow machin abil autonom acquir skill execut complex task particular interest learn use acquir complex behavior skill order endow ma
chin greater autonomi intellig see formal biographi click biographi sergey levin receiv bs ms comput scienc stanford univers 2009
 system adapt goal task essenti perform goal driven percept experiment result pr2 robot show method achiev substanti improv accurac
 i final polici 2009 2016 sergey levin


 ./doc/News_387_E.txt
  mack jganswer oct 20 2016if want page us first read pre written sampl sampl would tell add point point ad also take help onlin es
 say write profession best known find write colleg essay need research best essay write firm 134 view


 ./doc/News_146_E.txt
  cristina conati professor comput scienc depart univers british columbia homepag research teach public contact cristina conati pro
 fessor depart comput scienc univers british columbia goal integr research artifici intellig human comput interact cognit scienc cr
 eat intellig user interfac effect reliabl adapt need user particularli interest extend rang user state trait reliabl captur comput
 user model leverag adapt pure cognit featur knowledg skill goal affect state emot mood attitud meta cognit skill e g capabl effect
 explor larg inform space person trait 100 peer review public field intellig user interfac user model affect comput intellig tutor
 system intellig virtual agent research receiv award varieti venu includ umuai journal user model user adapt interact 2002 intern c
 onfer intellig user interfac iui 2007 intern confer user model adapt person umap 2013 2014 tii acm transact intellig interact syst
 em 2014 intern confer intellig virtual agent iva 2016 associ editor umuai tii ieee transact affect comput journal artifici intelli
 g educ serv presid aaac associ advanc affect comput well program confer chair sever intern confer includ umap iui ai educ like joi
 n research group graduat student pleas appli news perk check new paper accept ijcai 2017 iui 2017 aie 2017 edm 2017 best paper awa
 rd 2016 iva 2016 16th intern confer intellig virtual agent nserc discoveri grant acceler 2016 2019 best paper award 2014 acm trans
 act intellig interact system tii best paper award umap 2014 runner best paper award 2014 runner best paper award iui 2014 best pap
 er award umap 2013 killam research fellowship 2013 depart comput scienc ubc


 ./doc/News_494_E.txt
  andrew scharf head koianswer dec 22 2017 author 1 6k answer 398k answer viewsorigin answer write good colleg applic essay write g
 reat essay appli school becom harder ever school demand requir student realli think messag want share admiss committe institut wis
 h compet never level play field applic intens competit top school serv make essay write exercis anxieti ridden advic hire coach se
 e process take advantag expertis sell stori wish share make huge differ candidaci 136 view answer request john kurosaga


 ./doc/News_437_E.txt
  zach kinnaman colleg studentansw aug 17 2013essay take lot time order write excel well thought colleg essay take lot time high sc
 hool student alreadi deal school work sport social life would rather spend time fun thing instead sit organ colleg essay colleg ma
 y may get student option appli similar school requir essay probabl like appli school424 view
```

5. 参考网页

[1] http://www.ruanyifeng.com/blog/2013/03/tf-idf.html
[2] http://www.ruanyifeng.com/blog/2013/03/cosine_similarity.html
[3] https://blog.csdn.net/qq_19707521/article/details/78479532
[4] https://blog.csdn.net/tvetve/article/details/2292111
[5] https://blog.csdn.net/eastmount/article/details/50473675
[6] https://datartisan.gitbooks.io/

6. 附录

```
/**************建立索引*********************/
import java.io.*;
import java.nio.charset.StandardCharsets;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.util.*;
import org.apache.lucene.analysis.standard.StandardAnalyzer;
import org.apache.lucene.document.Document;
import org.apache.lucene.document.Field;
import org.apache.lucene.document.StringField;
import org.apache.lucene.document.TextField;
import org.apache.lucene.index.IndexWriter;
import org.apache.lucene.index.IndexWriterConfig;
import org.apache.lucene.store.Directory;
import org.apache.lucene.store.FSDirectory;
import org.apache.lucene.util.Version;

public class Indexer {
    public static void creatIndex()
    {
        IndexWriter writer = null;
        try{
            //1.create Directory
            Directory directory =
FSDirectory.open(Paths.get("C:\\Users\\Administrator\\Desktop\\三期
\\NLP_Project\\indexer"));
            IndexWriterConfig IWconfig = new IndexWriterConfig(new
StandardAnalyzer());
```

```
            //2.create IndexWriter
            writer = new IndexWriter(directory, IWconfig);
            //3.create Document
            Document document = null;
            //4.add field to document
            File f = new File("C:\\Users\\Administrator\\Desktop\\三期\\NLP_Project\\doc");
            for(File file : f.listFiles()){
                document = new Document();
                document.add(new StringField("path", f.getName(), Field.Store.YES));
                System.out.println(file.getName());
                document.add(new StringField("name", file.getName(), Field.Store.YES));

                InputStream stream = Files.newInputStream(Paths.get(file.toString()));
                //5.partition words
                //UTF-8 encoding
                //document.add(new TextField("content", new BufferedReader(new InputStreamReader(stream, StandardCharsets.UTF_8))));
                document.add(new TextField("content", new FileReader(file)));
                writer.addDocument(document);
            }
        }catch(Exception e){
            e.printStackTrace();
        }finally{
            //6.close writer
            try{
                writer.close();
            }catch(IOException e){
                e.printStackTrace();
            }
        }
    }
    public static void main(String[] args)
    {
        creatIndex();
    }
}

/****************实现搜索*****************/
import java.nio.file.Paths;
```

```
import java.util.Scanner;
import java.io.*;

import org.apache.lucene.analysis.standard.StandardAnalyzer;
import org.apache.lucene.document.Document;
import org.apache.lucene.index.DirectoryReader;
import org.apache.lucene.queryparser.classic.QueryParser;
import org.apache.lucene.search.IndexSearcher;
import org.apache.lucene.search.Query;
import org.apache.lucene.search.ScoreDoc;
import org.apache.lucene.search.TopDocs;
import org.apache.lucene.store.Directory;
import org.apache.lucene.store.FSDirectory;

public class Search {
    public static String indexSearch(String keywords)
    {
        String res = "";
        DirectoryReader reader = null;
        try{
            //1.create Directory
            Directory
directory=FSDirectory.open(Paths.get("C:\\Users\\Administrator\\Desktop
\\三期\\NLP_Project\\indexer"));
            //2.create IndexReader
            reader = DirectoryReader.open(directory);
            //3.create IndexSearcher
            IndexSearcher searcher = new IndexSearcher(reader);
            //4.create Query using to search
            //create parse to identify the content to be searched. the second
parameter indicates searching fields
            QueryParser   parser   =   new   QueryParser("content",   new
StandardAnalyzer());
            Query query = parser.parse(keywords);//searched content
            //5.return TopDocs according to Searcher
            TopDocs tds = searcher.search(query, 20);//20 pieces
            //6.acquire ScoreDocs according to TopDocs
            ScoreDoc[] sds = tds.scoreDocs;
            //7.acquire document according to Searcher and ScoreDoc
            int cou = 0;
            for(ScoreDoc sd : sds)
            {
                cou++;
                Document d = searcher.doc(sd.doc);
```

```
                //8.obtain the searched field value according to document
object
                //in the result, content = null because the index do not save
the content, so we need to obtain it from the ordinary file according to
path and name
                res += cou + ". " + d.get("path") + " " + d.get("name") + "
" + d.get("content") + "\n";
            }
        }catch(Exception e){
            e.printStackTrace();
        }finally{
            //9.close reader
            try{
                reader.close();
            }catch(IOException e){
                e.printStackTrace();
            }
        }
        return res;
    }
    public static void main(String[] args)
    {
        Scanner in = new Scanner(System.in);
        String str = in.next();
        System.out.println(indexSearch(str));
    }
}

/****************任意两个文本之间相似度*********************/
str1 = raw_input("Input file number: ")
str2 = raw_input("Input file number: ")

file1 = "News_"  + str1 + "_E.txt"
file2 = "News_"  + str2 + "_E.txt"
filenames = ['./doc/' + file1, './doc/' + file2]#relative filepath
f1 = open(filenames[0], 'r').read()
f2 = open(filenames[1], 'r').read()

#partition terms and calculate Term-Frequency
from nltk.tokenize import word_tokenize
text1 = word_tokenize(f1)
text2 = word_tokenize(f2)
#分词后的结果放在 list 里面
all_words = []
```

```
for i in text1:
    all_words.append(i)
for i in text2:
    all_words.append(i)

#列出所有的词 去掉重复
all_ = list(set(all_words))

#构造词向量
vector1 = []
vector2 = []
for elm in all_:
    num = text1.count(elm)#in text1,count the number of every words in all_
    vector1.append(num)
for elm in all_:
    num = text2.count(elm)#in text2,count the number of every words in all_
    vector2.append(num)

#引入 numpy 计算余弦距离
import numpy as np
vectorA = np.array(vector1)
vectorB = np.array(vector2)

op = np.dot(vectorA,vectorB)/
(np.linalg.norm(vectorA)*(np.linalg.norm(vectorB)))
print("Cosine Distance: ")
print(op)
print('\n')
print("News_"  + str1 + "_E.txt")
print(f1)
print('\n')
print("News_"  + str2 + "_E.txt")
print(f2)

/****************文本聚类*******************/
import time
import re
import os
import sys
import codecs
import shutil
import numpy as np
from sklearn import feature_extraction
from sklearn.feature_extraction.text import TfidfVectorizer
```

```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfTransformer

#将文本中的词语转换为词频矩阵 矩阵元素 a[i][j] 表示 j 词在 i 类文本下的词频
vectorizer = CountVectorizer()

#该类会统计每个词语的 tf-idf 权值
transformer = TfidfTransformer()
#直接用正则表达式切分，若用 tokenize 会出现 byte 字符，无法用 utf8 decode
#from nltk.tokenize import word_tokenize
A = [] #一个 elm 为一个文档的 list
num = 1
while(num<501):
    f1 = open('./doc/' + "News_"  + str(num) + "_E.txt", 'r').read()
    reg = re.compile('\\W*')#除了单词外的所有特殊符号包括空格
    text = reg.split(f1)
    B = " ".join(text)
    #type(B)
    A.append(B)
    num += 1

#第一个 fit_transform 是计算 tf-idf 第二个 fit_transform 是将文本转为词频矩阵
tfidf = transformer.fit_transform(vectorizer.fit_transform(A)).toarray()
tfidf
#将 tf-idf 矩阵抽取出来，元素 w[i][j]表示 j 词在 i 类文本中的 tf-idf 权重
resName = "Tfidf_Result.txt"
result = codecs.open(resName, 'w', 'utf-8')
#打印每类文本的 tf-idf 词语权重，第一个 for 遍历所有文本，第二个 for 便利某一类文本下的词语权重
for i in range(len(tfidf)):
    result.write(u"-------这里输出第" + str(i) + u"类文本的词语 tf-idf 权重------")
    for j in range(len(tfidf[i])):
        result.write(str(tfidf[i][j]) + ' ')
    result.write('\r\n\r\n')
result.close()

# rescale the data: mean 0, std:1
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit(tfidf)
scaled_df = scaler.transform(tfidf)
scaled_df
```

```
from sklearn.cluster import KMeans
kmeans = KMeans()
kmeans.set_params(n_clusters = 20)
kmeans.fit(scaled_df)

label = kmeans.labels_
print label

mylist1 = list(label)
myset1  = set(mylist1)
dict1 = {}
for item in myset1:
    dict1.update({item : mylist1.count(item)})
dict1

#Determine optimal k
from sklearn.cluster import KMeans
from sklearn import metrics
from scipy.spatial.distance import cdist
import matplotlib.pyplot as plt
# k means determine k
distortions = []
K = range(1,20)
for k in K:
    kmeanModel = KMeans(n_clusters = k).fit(scaled_df)
    kmeanModel.fit(scaled_df)
    distortions.append(sum(np.min(cdist(scaled_df,
kmeanModel.cluster_centers_, 'euclidean'), axis=1)) / scaled_df.shape[0])
# plot the elbow graph
plt.plot(K, distortions, 'bx-')
plt.xlabel('k')
plt.ylabel('Distortion')
plt.title('The Elbow Method showing the optimal k')
plt.show()

import numpy as np
np.set_printoptions(suppress=True) #不用科学计数法

L1=distortions[0:len(distortions)-1]
L2=distortions[1:len(distortions)]
ret = map(lambda x, y: (x-y)/x  , L1, L2)

kmeans2 = KMeans()
kmeans2.set_params(n_clusters = 16)
```

```
kmeans2.fit(scaled_df)
label2 = kmeans2.labels_
print label2

center = kmeans2.cluster_centers_
print center

mylist = list(label2)
myset  = set(mylist)
dict = {}
for item in myset:
    dict.update({item : mylist.count(item)})
dic

#找属于第类的的文本索引（即 label2 的下标）
belong_0 = np.where(label2 == 0)
Dis = []
for ind in range(0,500):#belong_0[0]:
    op = np.sqrt(np.sum(np.square(center[0]-scaled_df[ind])))#第 0 类中所有
点距离中心点的距离
    Dis.append(op)

Dis_np = np.array(Dis)
Dis_np.argsort()[:5][::-1]#顺序自始至终没有变过，返回前五个数即距离最小的文本
的索引
str1 = raw_input("Input file number: ")
str2 = raw_input("Input file number: ")
str3 = raw_input("Input file number: ")
str4 = raw_input("Input file number: ")
str5 = raw_input("Input file number: ")

file1 = "News_"  + str1 + "_E.txt"
file2 = "News_"  + str2 + "_E.txt"
file3 = "News_"  + str3 + "_E.txt"
file4 = "News_"  + str4 + "_E.txt"
file5 = "News_"  + str5 + "_E.txt"
filenames = ['./doc/' + file1, './doc/' + file2, './doc/' + file3, './doc/'
+ file4, './doc/' + file5]#relative filepath
for i in range(0,5):
    f1 = open(filenames[i], 'r').read()
    print ("\n")
    print (filenames[i])
    print f1
```