基于微博用户数据的 K-means 聚类分析

《计算新闻传播学》课程期末个人作业

一、数据来源及描述

本次数据挖掘实验的数据来源自于 2013 年微博数据，数据包含 994 条，内容有用户 id 和用户标签，每个 id 对应 20 个标签，也就是说数据是 20 维的，标签字符长度不定。数据包语言默认是中文字符。

二、kmeans 算法思路及最优 k 的选择方法

2.1 kmeans 及分析算法思路

从 994 个数据对象任意选择 k 个对象作为厨师聚类中心，而对于所剩下其他对象，则根据他们与这些聚类中心的相似度（距离），分别将他们分类给与其最相似的聚类中心所代表的聚类，单后在计算每个所惑新聚类的聚类中心（该剧类所有对象的均值）；不断重复这一过程知道标准测度函数开始收敛。K 个聚类具有以下特点：各个聚类本身尽可能的紧凑，而个聚类之间尽可能的分开。

K-Means 的主要优点有：

1）原理比较简单，实现也是很容易，收敛速度快。

2）聚类效果较优。

3）算法的可解释度比较强。

4）主要需要调参的参数仅仅是簇数 k。

K-Means 的主要缺点有：

1）K 值的选取不好把握。

2）不是突出的数据集比较难收敛。

3）如果各隐含类别的数据不平衡，比如各隐含类别的数据量严重失衡，或者各隐含类别的方差不同，则聚类效果不佳。

4） 采用迭代方法，得到的结果只是局部最优。

5） 对噪音和异常点比较的敏感。

在尝试了 k=10、12、13、14 的情况下的聚类，并最终选定了 K=13。在将 994 个用户聚类后，笔者随机抽取了第十组的用户进行关注分析，抽样结果推算全部样本用户关注情况。

在对抽样类别成员的关注分析中，笔者运用了爬虫、代码、execl 函数以及人工核实的

方法对结果进行验证和分析。

2.2 K 值的评估标准

常见的方法有轮廓系数 Silhouette Coefficient 和 Calinski-Harabasz Index。个人比较喜欢 Calinski-Harabasz Index，得到的 Calinski-Harabasz 分数值 s 越大则聚类效果越好。

1、将样本 x 与簇内的其他点之间的平均距离作为簇内的内聚度 a

2、将样本 x 与最近簇中所有点之间的平均距离看作是与最近簇的分离度 b

3、将簇的分离度与簇内聚度之差除以二者中比较大的数得到轮廓系数，计算公式如下

$$S(i)=\frac{b(i)-a(i)}{max\{a(i),b(i)\}}$$

轮廓系数的取值在-1 到 1 之间。当簇内聚度与分度离相等时，轮廓系数为 0。当 b>>a 时，轮廓系数近似取到 1，此时模型的性能最佳。

或者采用簇内误方差(SSE)进行计算。在对簇的划分中，就使用了 SSE 作为目标函数来划分簇。当 KMeans 算法训练完成后，可以通过使用 inertia 属性来获取簇内的误方差，不需要再次进行计算。可以使用图形工具肘方法，根据簇的数量来可视化簇内误方差。通过图形可以直观的观察到 k 对于簇内误方差的影响。

三、数据挖掘代码

3.1 Kmeans 聚类代码：

```
import pandas as pd
import numpy as np
import os
import sys
from sklearn import feature_extraction
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.feature_extraction.text import CountVectorizer
import os
if __name__ == "__main__":
    datas = open('C:/Users/lenovo/Desktop/kmeans/dataset.txt','r',encoding='utf-8',errors='ignore').readlines()
```

```
    ids = []
    corpus = []
    for tmp in datas:
        ids.append(tmp.split('\t')[0])
        corpus.append(tmp.split('\t')[1])
    vectorizer = CountVectorizer()
    transformer = TfidfTransformer()
    tfidf = transformer.fit_transform(vectorizer.fit_transform(corpus))
    from sklearn.cluster import KMeans
    km = KMeans(n_clusters=13)
    km.fit(tfidf)
    label = km.labels_
    res = pd.DataFrame(list(zip(ids,label)))
    res.head(2)
#当 k=12 时用户聚类结果
    km1 = KMeans(n_clusters=12)
    km1.fit(tfidf)
    label1 = km1.labels_
    res1 = pd.DataFrame(list(zip(ids, label1)))
#当 k=14 时用户聚类结果
    km1 = KMeans(n_clusters=14)
    km1.fit(tfidf)
    label1 = km1.labels_
    res1 = pd.DataFrame(list(zip(ids, label1)))
#当 k=10 时用户聚类结果
    km1 = KMeans(n_clusters=10)
    km1.fit(tfidf)
    label1 = km1.labels_
    res1 = pd.DataFrame(list(zip(ids, label1)))
    #当 k=13 时用户聚类结果
```

```
from sklearn.cluster import KMeans
km = KMeans(n_clusters=13)
km.fit(tfidf)
label = km.labels_
res = pd.DataFrame(list(zip(ids,label)))
res.head(2)
#当 k=12 时，将每组用户打印如下
    c0 = res.loc[res[1]==0,]
    c0
    c1 = res.loc[res[1]==1,]
    c1
    c2 = res.loc[res[1]==2,]
    c2
    c3 = res.loc[res[1]==3,]
    c3
    ……
    c4 = res.loc[res[1]==4,]
    c4
    c10 = res.loc[res[1]==10,]
    c10
    len(c10)#测量数据长度
    c11 = res.loc[res[1]==11,]
    c11
    c12 = res.loc[res[1]==12,]
    c12
```

3.2 用户关注情况分析

随机在 k=13 聚类结果中，抽取第十组的数据结果，利用爬虫技术，获取第十组出现的所有 id 的关注好友姓名和 id。因为微博限制，爬虫只爬下了关注前五页的用户情况（数据文件见 res13）。

3.2.1 爬虫数据分析代码：

#导入第十组的爬虫数据，代码如下：

```
import json
    f = open("C:/Users/lenovo/Desktop/kmeans/res13.json", encoding='utf-8')
    rel_data12 = json.load(f)
    arr1 = []
    for d1 in rel_data12.keys():
        for d2 in rel_data12[d1].keys():
            arr1.append((d1,d2))
    ct = 0
    for m in range(len(arr1)-1):
        for n in range(m+1,len(arr1)):
            if (arr1[m][0]==arr1[n][1] and arr1[m][1]==arr1[n][0]):
                ct+=1
    print('k=13', ct)
```

3.2.2 excel 函数分析：

为进一验证结果的正确性，笔者利用 excel 对用户关注结果进行了进一步的验证。将爬虫数据整理成 excel 表格，左侧栏对应第十组用户 id 信息，右侧栏是每个 id 下用户的前五页关注用户，利用 COUNTIF 函数进行验证，函数如下：

```
=IF(COUNTIF(B:B,E2),1,0)
```

3.2.3 人工检验：

因为爬虫技术只得到了前五页的关注用户信息,因此利用人工关注查询的方式对结果进行再分析，得到更加准确的结果。

四、实验结果

4.1 聚类结果

笔者基于原始数据集,借助 K-means 工具将微博用户按照类别进行聚类,共分为 13 组。

```
import pandas as pd
import numpy as np

import os
import sys
from sklearn import feature_extraction
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.feature_extraction.text import CountVectorizer
import os

if __name__ == "__main__":
datas = open('C:/Users/lenovo/Desktop/kmeans/dataset.txt','r',encoding='utf-8',errors='ignore').readlines()
ids = []
corpus = []
for tmp in datas:
    ids.append(tmp.split('\t')[0])
    corpus.append(tmp.split('\t')[1])

vectorizer = CountVectorizer()
transformer = TfidfTransformer()
tfidf = transformer.fit_transform(vectorizer.fit_transform(corpus))


from sklearn.cluster import KMeans
km = KMeans(n_clusters=13)
km.fit(tfidf)
label = km.labels_

res = pd.DataFrame(list(zip(ids,label)))
res.head(2)
```

```
In [6]: c0 = res.loc[res[1]==0,]
```

```
In [7]: c0
```

Out[7]:

	0	1
2	1678781851	0
15	1029007695	0
19	1062003303	0
29	1104268797	0
30	1105243673	0
35	1151002655	0
54	1240642125	0
63	1275207805	0
64	1276679282	0
65	1281091387	0
66	1286115047	0
70	1306775507	0
88	1408891592	0
89	1409557911	0
91	1410772313	0
107	1481814940	0
108	1491408501	0
111	1505392202	0
122	1579037245	0
125	1590266587	0
129	1611597374	0

运行代码可得用户聚类结果如下：

笔者将微博用户按照类别聚类 13 组。

第一类：

0	1
2	1678781851
15	1029007695
19	1062003303
29	1104268797
30	1105243673
35	1151002655
54	1240642125
63	1275207805
64	1276679282
65	1281091387
66	1286115047
70	1306775507
88	1408891592
89	1409557911
91	1410772313
107	1481814940
108	1491408501
111	1505392202
122	1579037245
125	1590266587
129	1611597374
133	1630436561
139	1649026457
141	1651763583
143	1652719463
145	1653702492
146	1653823773
148	1655426815
152	1661672805
154	1670053880
...	...
799	2563494105
806	2574044605
807	2574757744
809	2579186134
812	2583525605
820	2598187052
832	2608114991
839	2609501315

840	2609774727
859	2626499283
861	2626872312
863	2629627813
872	2639695573
876	2644026961
885	2650984623
897	2662293443
898	2662590691
914	2678706687
916	2679927753
923	2686232102
925	2687969972
930	2691890375
936	2700902864
940	2703757621
948	2709747645
954	2713198045
965	2728658893
969	2732136787
972	2740100795
982	2792977370

156 rows × 2 columns

第二类：

0	1
5	1961796627
28	1103558101
32	1131463913
45	1217062072
69	1305108382
86	1403096670
121	1575364637
134	1631824791
204	1746348203
212	1752881674
247	1816271522
281	1859310782
284	1866639713
309	1897593071
314	1903951291
328	1922132094

330	1926365987
362	1964614173
363	1966417743
368	1970795737
374	1976219922
379	1985076605
396	2007555404
408	2021731903
414	2031575803
416	2032815117
425	2050711850
432	2057365522
434	2058490360
442	2066910733
...	...
495	2127489534
511	2150971217
522	2161516623
537	2174704514
607	2269448535
619	2282474965
631	2300426417
640	2316878537
650	2331501435
699	2407919017
718	2424678722
738	2463720793
740	2465295581
758	2484817703
761	2491592385
762	2492028994
765	2494963093
791	2552672235
810	2580181791
815	2585042240
817	2588619543
889	2653649045
896	2660523751
921	2684971641
922	2685282291
926	2688704007
933	2696048362
945	2707245893

947	2707715014
988	2815276392

62 rows × 2 columns

第三类：

0	1
10	2258319335
34	1147548630
40	1194972061
49	1221214427
50	1223417940
53	1236988792
75	1343110160
79	1395411312
96	1432427651
99	1444300551
100	1445946204
140	1649787232
171	1698812042
198	1734174761
200	1738644283
217	1761173420
222	1766626567
234	1786736317
243	1806375920
278	1855195060
295	1879345744
297	1880066151
298	1883862101
302	1886852804
307	1895903182
311	1900846093
316	1909232225
319	1912459281
344	1948085205
345	1949416832
...	...
858	2624232283
864	2631801824
865	2632334424
867	2632500797
870	2635946463

873	2640159123
878	2645139840
886	2651541771
888	2651818177
907	2669556742
910	2675396913
913	2676353170
917	2680215031
919	2682740683
920	2683544663
928	2690290353
935	2697025541
937	2702534944
943	2706322991
950	2711821102
951	2712027085
952	2712560372
953	2712781533
956	2715112694
958	2716381353
967	2729715607
975	2755974537
976	2761487620
977	2784245190
978	2784564342

131 rows × 2 columns

第四类：

0	**1**
0	1213230260
6	2025032474
7	2066076725
13	1003229242
14	1028598137
16	1032380037
18	1048035162
20	1066288815
22	1075987832
25	1088986640
26	1093636310
27	1100404614
33	1141303333

38	1162275802
44	1215070212
47	1217421981
48	1219078160
58	1251026227
61	1256391421
68	1299967044
72	1319417172
73	1321701473
74	1331493713
76	1351209513
78	1362393140
82	1400657743
102	1453932652
105	1469261212
110	1501081977
115	1544593902
...	...
782	2534268771
786	2549589610
788	2550471637
790	2550975887
792	2553179927
793	2559309810
798	2563218995
801	2566928273
802	2568772100
808	2576671624
813	2583974362
814	2584268124
825	2602280773
829	2605823322
831	2608051713
841	2609960817
849	2616003123
855	2621841313
881	2647285843
891	2655938354
894	2659229263
895	2659323174
905	2667181171
906	2668572771
912	2675711024

949	2711585801
959	2716612242
961	2722848824
964	2724271022
974	2753747173

172 rows × 2 columns

第五类：

0	1
151	1661119943
201	1739090517
237	1794425807
239	1797422282
242	1802688164
254	1827639110
315	1904348123
355	1957854731
359	1963148700
375	1977601365
378	1978630265
386	1995599073
387	1997113962
397	2009585681
409	2024652190
423	2045862445
465	2095923392
466	2096236200
478	2115893792
491	2125957574
496	2127796914
500	2132305674
583	2239300370
593	2259338054
604	2268299214
610	2271692882
617	2280622694
639	2315425182
667	2355028870
687	2379688351
726	2445875204
766	2500847620
783	2538401882

811	2582778794
879	2645891150
915	2679846990
934	2696343904
942	2705860373
946	2707679693
957	2715705081
979	2786622000
981	2792936862
991	2847466262

第六类：

0	1
84	1401077780
193	1731753764
250	1822898550
252	1826733997
299	1884522347
312	1903616191
327	1919328320
384	1993834794
431	2057290195
455	2082252731
467	2097239261
485	2122003031
509	2146514155
517	2158621923
546	2182933434
549	2184515730
555	2191816054
573	2231380923
574	2231905593
601	2265088314
623	2286281545
628	2294015047
673	2366050300
736	2459844647
767	2501956865
768	2504433601
779	2530570601
781	2533425707
803	2571062133

823	2600219791
828	2605148215
844	2610501225
857	2623113551
868	2633826902
869	2635934055
874	2642350167
899	2662869090
904	2667179637
918	2681647002
944	2707043281

第七类：

23	1076057590
36	1155860855
37	1161639692
39	1179924182
42	1196136335
51	1225727654
52	1230005695
56	1241809503
57	1243325445
67	1299308451
81	1398762171
83	1400714603
87	1406646262
93	1412709504
94	1425295451
98	1444217544
101	1449706683
103	1455436067
109	1497464950
112	1505648151
113	1508854385
114	1529224452
116	1558675505
117	1561726373
123	1579070605
138	1648741671
147	1653848945
158	1677838312
164	1693745147

166	1697473487
...	...
540	2175755133
559	2196639831
563	2202391584
596	2260078305
606	2269447713
613	2274929403
655	2338984813
656	2339213997
665	2350800612
669	2357926577
683	2375738797
730	2456231580
731	2457735692
735	2459826153
747	2476157283
751	2478216251
775	2521834565
784	2539261854
827	2604259803
837	2609155162
843	2610482902
847	2614168945
853	2621085071
860	2626705521
866	2632486867
892	2656522750
911	2675699690
938	2702922692
980	2786778690
983	2793025974

106 rows × 2 columns

第八类：

0	1
186	1728219241
260	1840222832
322	1915503933
332	1931852505
333	1932971343
349	1950898630

353	1954054052
356	1958662831
377	1978619767
398	2010064217
399	2010856665
403	2014574175
433	2058376655
437	2063387041
438	2064456785
439	2064745745
453	2075422491
458	2087036623
464	2092064034
473	2105024092
493	2127107460
494	2127241961
501	2132328614
526	2165906222
532	2168928842
533	2169130274
556	2192436712
633	2305328244
638	2313665982
649	2328941622
668	2357516840
701	2410428150
706	2413203711
720	2427808630
743	2471084542
757	2483296907
856	2622146603
882	2647510930

第九类：

0	1
8	2121821804
126	1594279755
206	1748009682
207	1748536590
219	1762475540
231	1775291865
259	1836789274
268	1843131374

286	1869459164
287	1869650211
288	1870777325
292	1873798037
318	1912055912
331	1931484145
343	1947008212
388	1997217234
421	2035488295
428	2051632241
446	2072004285
460	2090867153
484	2121760524
488	2123771615
502	2132663147
530	2168421392
531	2168810722
543	2176802674
547	2183321064
553	2190248344
571	2230139942
590	2255887221
642	2317363344
700	2408158372
722	2429790807
728	2450431382
729	2454306450
755	2482128901
769	2506175474
805	2573750840
850	2616338392
908	2669852842

第十类：

0	1
106	1473996772
131	1621921544
153	1666285261
291	1873428394
370	1971401953
410	2024913745
497	2130688003
516	2157575052

523	2162618644
542	2175925320
566	2210329087
575	2234057891
579	2236550293
597	2260535017
616	2279211563
618	2281704034
653	2337218767
693	2397436251
696	2402417925
723	2430878452
763	2492142671
794	2559708412
800	2564286727
836	2608993431
846	2613986553
851	2617757075
854	2621650922
880	2647272141
887	2651606021
890	2654256313
901	2665328147
927	2689136053
931	2693752002
932	2695970542
968	2731602180
973	2740439390
990	2830422684
993	2854841802

第十一类：

0	1
9	2257307207
43	1202197767
104	1459884350
149	1658780981
150	1659263692
216	1760092601
263	1840450847
272	1851919291
306	1893687663
323	1915567373

350	1952737807
475	2106917402
507	2137431363
538	2175250557
572	2230996084
591	2256825873
622	2285695435
724	2435327933
748	2477294207
871	2637128907
939	2703133237

第十二类：

0	1
1	1320644705
46	1217213193
77	1357052241
157	1677770243
199	1734892303
208	1748744233
213	1757014837
227	1773281917
289	1871541223
373	1975141497
452	2074570445
524	2163155342
562	2201079027
626	2292103632
684	2376276534
962	2723016575

第十三类：

0	1
3	1682789624
4	1863230267
11	2304771990
12	2728054061
17	1034438302
21	1073150225
24	1081573174
31	1107281623
41	1195103993
55	1240724101

59	1256071265
60	1256156924
62	1272604192
71	1310937857
80	1396429921
85	1401354804
90	1410000432
92	1411862024
95	1428439722
97	1437001030
119	1565375643
120	1571086234
124	1579497145
127	1602425131
132	1627835552
155	1670704631
161	1684314860
162	1686914850
174	1700700283
178	1706167745
...	...
760	2488905073
773	2516231520
816	2586539002
830	2607428785
838	2609316861
862	2629015673
875	2643327047
877	2644967854
883	2647741197
884	2650904991
893	2657520314
900	2664386187
902	2665723233
903	2665748071
909	2670712587
924	2687301911
929	2690346100
941	2705542110
955	2714033541
960	2717227731
963	2723686902
966	2728817161

970	2735084683
971	2738753101
984	2793526197
985	2803977362
986	2804021502
987	2805085040
989	2827570032
992	2849387244

131 rows × 2 columns

4.2 用户关注关系结果

4.2.1 基于代码的用户互相关注结果为：

In [24]:
```
import json
f12 = open("C:/Users/lenovo/Desktop/kmeans/res.json", encoding='utf-8')
rel_data12 = json.load(f12)
arr1 = []
for d1 in rel_data12.keys():
    for d2 in rel_data12[d1].keys():
        arr1.append((d1,d2))

ct = 0
for m in range(len(arr1)-1):
    for n in range(m+1,len(arr1)):
        if (arr1[m][0]==arr1[n][1] and arr1[m][1]==arr1[n][0]):
            ct+=1
print('k=12',ct)
```

```
k=12 0
```

In [26]:
```
import json
f13 = open("C:/Users/lenovo/Desktop/kmeans/res.json", encoding='utf-8')
rel_data13 = json.load(f13)
arr1 = []
for d1 in rel_data12.keys():
    for d2 in rel_data12[d1].keys():
        arr1.append((d1,d2))

ct = 0
for m in range(len(arr1)-1):
    for n in range(m+1,len(arr1)):
        if (arr1[m][0]==arr1[n][1] and arr1[m][1]==arr1[n][0]):
            ct+=1
print('k=13',ct)
```

```
k=13 0
```

4.2.2 基于 excel 的用户单项关注结果：

	1473996772	2149200103	0
	1473996772	2238856913	0
	1473996772	2614129897	0
	1473996772	1479768345	0
	1473996772	2647272141	1
	1473996772	2213293225	0
	1473996772	2627078017	0
	1473996772	2716395772	0
	1473996772	2719663233	0

仅有 1 个用户有单项关注。

4.2.3 基于人工检测的用户关注结果：

样本组内用户社交联系整体松散，但具有较为紧密（即双向关注）的社交关系个例。

五、实验分析

5.1 用户分组画像

根据随机抽样调查结果表明，目前已有近半数的微博ID已经注销，其余的微博用户画像如下：

1.微博注册时间长：

2010年-2012年的注册用户居多，说明这群用户距今已有6-8年的注册时间，是新浪微博比较早期的用户，用户年龄基本在30岁以上。

2.微博发送总数多：

大部分用户发送超过4000条微博，但在最近一年微博更新频率较低甚至没有发送微博。说明被抽样的群体在一段时间内非常喜爱互联网社交，曾是微博重度用户。

3.微博关注人数多、粉丝少：

根据随机抽样调查发现，除少数加V账号关注人数较少、粉丝较多外，大部分人关注超过500人，甚至少数人微博关注将近2000人，但大部分账号的粉丝数少于关注人数。因为这些用户关注人数过多，难以从关注的微博发现他们准确的爱好，大多爱好广泛。

4.用户标签维度广：

在未注销的微博中，约有三分之一的用户带有自定义标签。他们的标签是从多个角度进行自我描述，例如一位用户的标签是“天秤座、上网、自由、宅、看新闻、英语、政治经济、房地产”，是从性格、兴趣点、从业方向等角度描述，每个标签的涵盖内容也非常广。说明这群用户拥有较为广泛而又不够鲜明的爱好。

5.2 用户社交紧密程度分析：微博调查与标签相似度结果比对

（1）数据挖掘结果总结：

在数据预处理及聚类分组、群体画像后，笔者最终选取 k=13 时第十组的数据进行聚类分析，基础数据共 38 条。由于原始数据集仅有用户 id，未提供用户微博名，根据微博检索规则，38 条数据中最后手动翻查有拥有对应 id 用户的数量为 19 条，即 19 位可翻查用户。具体数据字段见下（标“/”即为未检索到相关用户）：

用户id	用户微博名
1473996772	红粉泪滴
1621921544	/
1666285261	/
1873428394	/
1971401953	/
2024913745	竹杖芥鞋
2130688003	灰暗区域
2157575052	/
2162618644	/
2175925320	弱弱的小兔
2210329087	臧小沐
2234057891	我想下班回家了
2236550293	班杰明Benjamin
2260535017	/
2279211563	/
2281704034	南京栖霞山人
2337218767	中关村广场购物中心
2397436251	京韵视屏
2402417925	沁河一钓翁
2430878452	/
2492142671	/
2559708412	James笑嘻嘻
2564286727	/
2608993431	/
2613986553	/
2617757075	全谈辅具
2621650922	老谭说金
2647272141	/
2651606021	刘曾Zeng爱严
2654256313	/
2665328147	山东聊大律师事务所
2689136053	幸子童鞋
2693752002	信途网
2695970542	/
2731602180	Micro-mood
2740439390	/
2830422684	/
2854841802	/

最终得出的实验结果如“四.2 用户标签相似度数据结果”所示，用户标签相似度结果为零，即理论值不相关。

（2）手动微博调查结果：

为进一步验证通过数据挖掘呈现的用户标签相似度结果与用户实际社交间的关系，进一步说明微博用户间的标签相似度是否恰当反映了用户间的社交紧密程度，笔者对于可翻查的第十组内 19 位用户进行了微博调查。

- **调查问题：**19 位用户的社交关系。在微博中，社交关系主要表现为关注与被关注，因此，笔者定义了“双向关注”、“单向关注”、“单向被关注”三种关系状态；
- **调查结果：**

	红粉泪滴	竹杖莽鞋	灰暗区域	弱弱的小兔	臧小沐	我想下班	班杰明Be	南京栖露	中关村广	京韵视屏	沁河一钓翁	James笑嘻	全谈辅具	老谭说金	刘曾Zeng	山东聊大律	幸子童鞋	信途网	Micro-mood
红粉泪滴	/																		
竹杖莽鞋		/									双向关注								
灰暗区域			/									双向关注							
弱弱的小兔				/							双向关注								
臧小沐					/														
我想下班回家了						/													
班杰明Benjamin							/				单向被关注						单向被关注		
南京栖露山人								/											
中关村广场购物中心									/										
京韵视屏										/		双向关注							
沁河一钓翁		双向关注		双向关注			单向关注				/								
James笑嘻嘻			双向关注							双向关注		/							
全谈辅具													/						
老谭说金														/					
刘曾Zeng爱严															/				
山东聊大律师事务所																/			
幸子童鞋							单向关注										/		双向关注
信途网																		/	
Micro-mood																	双向关注		/

第十组十九位用户相关关系表格（可点开查看大图）

A. 用户社交紧密度：从 19 位用户来看，10 位用户未参与任何互动社交关系，即脱离于样本组的社交网络之外，另外 9 位用户具有一定的社交联系；社交紧密度一般。

B. 用户社交关系：

a. 双向关注：如图色块所示，在 9 位具有社交联系的用户中，共有五组“双向关注”关系，即社交联系最紧密；五组关系分别为“沁河一钓翁 - 竹杖莽鞋”、“沁河一钓翁 - 弱弱的小兔”、“James 笑嘻嘻 - 灰暗区域”、“James 笑嘻嘻 - 京韵视屏”、“Micro-mood - 幸子童鞋”。可以推断，“沁河一钓翁”及“James 笑嘻嘻”两位用户在第十组的样本中，处于较社交中心的位置；

b. 单向关注/被关注：如图色块所示，在 9 位具有社交联系的用户中，共有两组“双向关注”关系。值得注意的是，两组关系均为其他用户对某一用户的关注，即“沁河一钓翁 - 班杰明 Benjamin”、“山东聊大律师事务所 - 班杰明 Benjamin”。因此，可以推测，“班杰明 Benjamin”用户在第十组的样本中，处于较受欢迎的社交位置。

（3）标签相似度与微博调查结果对比：

通过对比样本组内微博用户间的标签相似度及微博调查中发现的用户社交紧密程度，可以得出以下结论及思考：

A. 结论：标签相似度不能完全反映用户社交紧密程度，但可供参考

综合以上结果，可以发现，根据 K-means 得出的实验对象用户标签相似度结果为零，即理论值不相关。然而，在实际微博调查中，实验对象中的 19 位用户具有一定较为紧密（即双向关注）的社交关系，且具有社交关系不同紧密程度的分层，与用户标签绝对不相关的结果不完全符合。因此，标签相似度不能完全反映用户社交紧密程度。

不过，由于实际调查中发现，实验对象中的 19 位用户虽然具有一定社交关系，但整体紧密度不高，且有近半数用户游离在这一社交网络之外。因此，标签相似度对于用户社交紧

密程度的整体特征可能具有一定参考价值，具体社交关系细节不能完全根据标签相似度推导。

B. 思考：造成标签相似度与社交紧密性不一致的影响因素

（1） 影响因素一：社交关系本身的特点，非绝对化“人以群分”

虽然日常生活中常提及“物以类聚，人以群分”，但具体到人与人的社交关系中，并非如此绝对化。社交关系的紧密程度不仅与用户之间的相似度相关，也与用户实际生活中的身份、所处环境、生活交集，以及性格互补的交友需要相关。因此，也有可能出现某用户与和自己完全不同的用户成为朋友等情况。

综合以上分析，以微博平台为例，用户标签相似度与社交紧密性不一致的情况事实上符合社交关系本身多元化、互补化特质的影响，也可以在现实生活中找到答案。

（2） 影响因素二：原始数据集时间较早，可能面临人际关系变动

社交关系的另一特点是变动大、不够稳定。由于原始微博标签及用户数据集时间创立于2013 年，距今已有五年时间，因此，标签相似度数据基于 2013 年的微博数据，而手动翻查的用户社交关系紧密度数据则基于 2018 年的微博，很可能面临较大范围的人际关系变动，如取消关注、拉黑、重新关注、微博销号等情况，由此可能对最终的实验结果产生影响。

（3） 影响因素三：无明确微博用户名，实际微博调查的样本组用户有限

本次选取的第十组基础数据共 38 条。由于原始数据集仅有用户 id，未提供用户微博名，根据微博检索规则，38 条数据中最后手动翻查有拥有对应 id 用户的数量为 19 条。因此，事实上，标签相似度数据基于 38 位用户，而社交紧密性数据仅基于 19 位用户，面临一半数据缺失的情况。这一样本用户数据前后不一致的情况，也可能对最终两份数据对比的情况产生影响。