1. 百度贴吧、微博社区'滴滴出行'网约车评论文本信息展示，其中百度贴吧数据 860 条，微博社区滴滴出行 862 条，微博社区滴滴打车 795 条，共 2517 条评论数据进行分析

1	经常用，一款实用的 APP，希望能再多点优惠
2	并且打车的发票在手机端就可以直接开出，而且我觉得打车的补贴还是比较多的，滴滴打车的优势是方便...
3	滴滴出行是个不错的出行平台，最大限度的提高了出行者的权益，但在司机的准入上要求不严，以致连续...
4	滴滴就是出了事情或者责任就会逃避
5	很想给滴滴好评的，因为它确实方便我出行了，而且滴滴司机给我的印象也不错。无奈最近它的大众形象...

```
df = pd.read_excel('E:\Desktop\semantic analysis\didi.xls')
df1 = pd.read_excel('E:\Desktop\semantic analysis\weibo_滴滴出行.xls')
df2 = pd.read_excel('E:\Desktop\semantic analysis\weibo_滴滴打车.xls')
df.shape[0],df1.shape[0],df2.shape[0]
```

```
(860, 862, 795)
```

2.消费者关于网约车的评论文本数据预处理

（1）文本去重

剔除重复的文本数据，加强数据的可用性，利用 python 的 drop_duplicates（）函数对 dataframe 结构进行去重处理

文本数据去重之前数据为 2517 条

```
data = pd.read_csv('E:\Desktop\semantic analysis\didicomment.csv',encoding='gbk')
data.shape[0]
```

```
2517
```

文本数据去重之后数据为 2482 条

```
data_last = data.drop_duplicates()
data_last.shape[0]
```

```
2482
```

（2）基于语义网络的消费者网约车评论分析分词处理

加载需要处理的文本数据，利用 Python 的 jieba 库进行分词处理，利用 jieba 库的 cut 方法进行分词处理，文本数据分词部分结果展示：

[['一款', '实用', 'APP', '希望', '多点', '优惠'],['打车','发票','手机','开出','打车','补贴','滴滴','打车','优势','方便快捷','一竿子','推翻','滴滴','社会','行业','贡献','网友','历史','出门在外','流量','变得','完善','支付','方式','一会','专车','来接','滴滴','女子','外出','遭受','侵害','例子','确实','出行','价格','明朗','十点','县城']]

```
def seg_to_list(sentence,pos=False):
    if not pos:
        seg_list = jieba.cut(sentence)
    else:
        seg_list = psg.cut(sentence)
    return seg_list
```

（3）停用词剔除处理

要对去重后的文本数据进行停用词剔除处理，加载库中的停用词表，对文本数据进行处理，剔除已经停用的词

停用词表部分词语展示：一时 一来 一样 一次 一片 一番 一直 一致 一般 一起 一转眼 一边 一面 七 万一 三 三天两头 三番两次 三番五次 上 上下 上升 上去 上来 上述 上面 下 下列 下去 下来 下面 不 不一 不久 不了 不亦乐乎 不仅 不仅...而且 不仅仅 不仅仅是 不会 不但 不但...而且 不光 不免 不再 不力 不单

```
def get_stopword_list():
    stop_word_path = './stopword.txt'#
    stopword_list = [sw.replace('\n','')for sw in open(stop_word_path,encoding='utf8').readlines()]
    return stopword_list
```

（4）干扰词过滤处理

经常在数据中出现但是却没有多大用处的词汇过滤，过滤词中词以及长度<2 的词

```
def word_filter(seg_list,pos=True):
    stopword_list = get_stopword_list()
    filter_list = []


    for seg in seg_list:
        if not pos:
            word = seg
            flag = 'n'
        else:
            word = seg.word
            flag = seg.flag
        if not flag.startswith('n'):
            continue

        if not word in stopword_list and len(word)>1:
            filter_list.append(word)
    return filter_list
```

4.利用 Ucient6 软件对文本数据进行语义网络分析

采用语义网络分析对自然语言进行处理，分解输入文本数据中的句法关系，分析句子的深层格结构，记录语义关系，关键词提取及中心度分析是社会网络分析中十分重要的部分，因此对文本进行关键词提取进行中心度分析，而中心度主要三个指标为点度中心性（Degree Centrality）、中间中心性（Between Centrality）、特征向量中心性（Eigenvector Centrality）

（1） Tf-idf 关键词提取：

在一份给定的文件里，词频指的是某一个给定的词语在该文件中出现的次数。这个数字通常会被归一化，以防止它偏向长的文件。逆向文件频率是一个词语普遍重要性的度量。某一特定词语的 IDF，可以由总文件数目除以包含该词语之文件的数目，再将得到的商取对数得到。某一特定文件内的高词语频率，以及该词语在整个文件集合中的低文件频率，可以产生出高权重的 TF-IDF。

TFIDF 的主要思想是：如果某个词或短语在一篇文章中出现的频率 TF 高，并且在其他文章中很少出现，则认为此词或者短语具有很好的类别区分能力，适合用来分类。TFIDF 实际上是：TF * IDF，TF 词频，IDF 反文档频率。TF 表示词条在文档 d 中出现的频率指的是某一个给定的词语在该文件中出现的次数。IDF 的主要思想是：如果包含词条 t 的文档越少，也就是 n 越小，IDF 越大，则说明词条 t 具有很好的类别区分能力。如果某一类文档 C 中包含词条 t 的文档数为 m，而其它类包含 t 的文档总数为 k，显然所有包含 t 的文档数 n=m+k，当 m 大的时候，n 也大，按照 IDF 公式得到的 IDF 的值会小，就说明该词条 t 类别区分能力不强。是指果包含词条的文档越少，IDF 越大，则说明词条具有很好的类别区分能力。

在一份给定的文件里，词频指的是某一个给定的词语在该文件中出现的频率。这个数字是对词数的归一化，以防止它偏向长的文件。对于在某一特定文件里的词语 t_i 来说，它的重要性可表示为：

$$\mathrm{tf_{i,j}} = \frac{n_{i,j}}{\sum_k n_{k,j}}$$

以上式子中 $n_{i,j}$ 是该词在文件d_j中的出现次数，而分母则是在文件d_j中所有字词的出现次数之和。

逆向文件频率是一个词语普遍重要性的度量。某一特定词语的 IDF，可以由总文件数目除以包含该词语之文件的数目，再将得到的商取对数得到：

$$\mathrm{idf_i} = \log \frac{|D|}{|\{j : t_i \in d_j\}|}$$

其中|D|：语料库中的文件总数$|\{j : t_i \in d_j\}|$：包含词语t_i的文件数目（即$n_{i,j} \neq 0$的文件数目）如果该词语不在语料库中，就会导致被除数为零，因此一般情况下使用
$1 + |\{j : t_i \in d_j\}|$ $\mathrm{tfidf_{i,j}} = \mathrm{tf_{i,j}} \times \mathrm{idf_i}$

某一特定文件内的高词语频率，以及该词语在整个文件集合中的低文件频率，可以产生出高权重的 TF-IDF。

文本总体高频词汇（top36）：

words	frequency	words	frequency	words	frequency	words	frequency
DiDi	5415	master	220	online taxi-hailing	169	condition	100
driver	1670	fast ride	216	rubbish	159	Shuttle transfer	98
passenger	584	phone	210	Beijing	148	place	98
customer	530	player	209	video	143	world	94
servicer	509	Following wind	207	hour	140	Liu Qing	87
game	507	cellphone	204	software	124	user	87
company	408	Order form	190	get on	120	function	87
full text	408	time	189	Car owner	118	friend	86
platform	227	taxi	172	system	111	city	76

（2） 中心度分析及网络图生成

将文本数据处理成以分号分隔的格式，如：

滴滴;感觉;出租车;滴滴;价格;方式;滴滴;感觉;力度;感觉;滴滴;专职;司机

滴滴;行程;地点;地点;方案

滴滴;消失;消失;习惯

滴滴;顺风;消费者;滴滴;快车;滴滴;快车;事件;滴滴

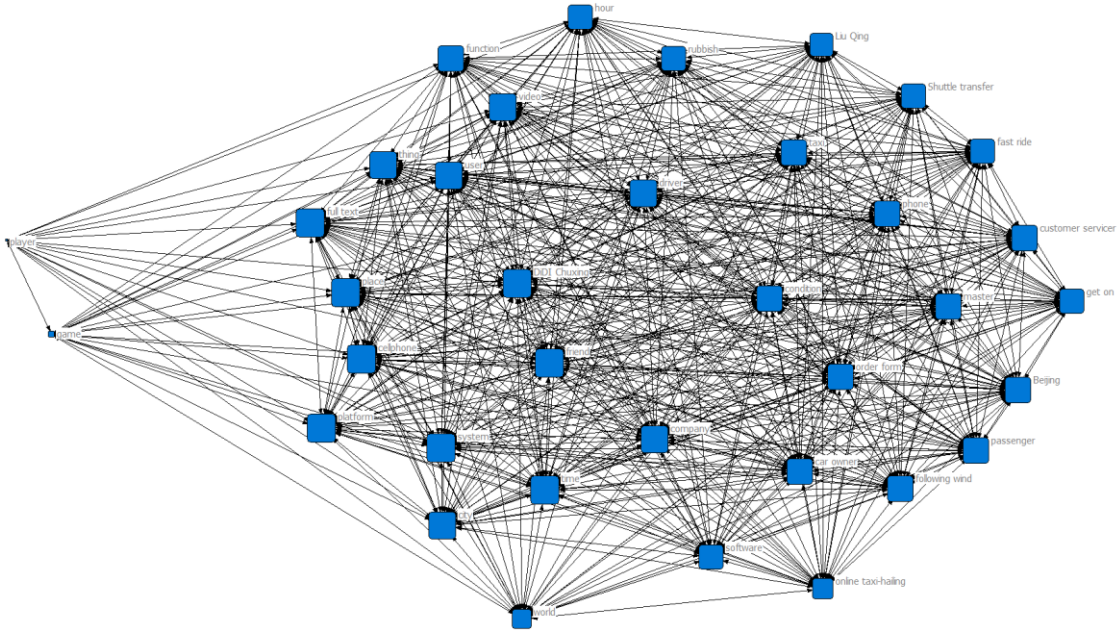
再用 Ucient6 软件导入数据生成词频数，高频词汇选取过多，绘制出来的网络图不利于观察，经过多次实验与尝试，取其中词频数 top36 生成 36x36 共现矩阵，利用共现矩阵绘制网络图

Top36 关键词生成的 36x36 共现矩阵:

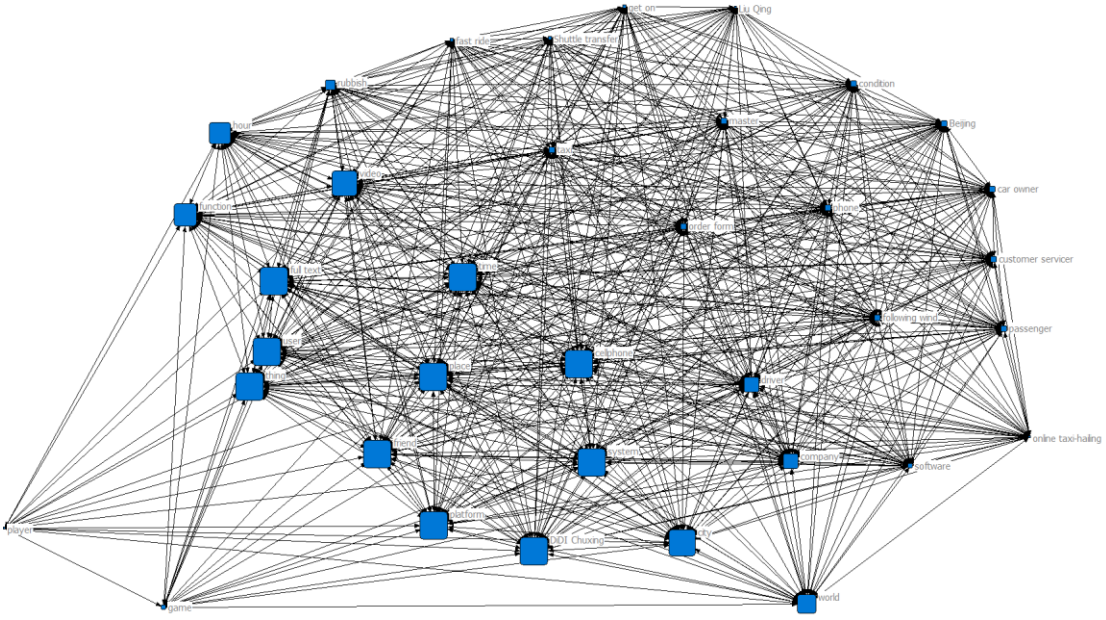
		DiDi Chuxir	driver	passenger	customer s	game	company	full text	platfrom	master	fast ride	phone	player	following v	cellphone	order form	time
2	DiDi Chuxi	0	757	295	288	261	270	232	408	138	171	146	157	101	90	129	133
3	driver	757	0	229	159	1	134	140	208	68	89	103	0	35	57	90	63
4	passenger	295	229	0	56	0	72	74	90	24	39	42	0	23	24	44	33
5	customer s	288	159	56	0	0	54	59	105	21	51	82	0	16	28	46	31
6	game	261	1	0	0	0	1	7	59	0	0	0	133	0	5	0	14
7	company	270	134	72	54	1	0	38	73	10	30	30	0	11	9	21	20
8	full text	232	140	74	59	7	38	0	88	17	34	32	2	22	13	38	20
9	platform	408	208	90	105	59	73	88	0	38	53	71	43	29	38	50	47
10	master	138	68	24	21	0	10	17	38	0	11	17	0	4	14	18	8
11	fast ride	171	89	39	51	0	30	34	53	11	0	25	0	17	12	18	15
12	phone	146	103	42	82	0	30	32	71	17	25	0	0	13	25	32	26
13	player	157	0	0	0	133	0	2	43	0	0	0	0	0	4	0	10
14	following v	101	35	23	16	0	11	22	29	4	17	13	0	0	4	9	18
15	cellphone	90	57	24	28	5	9	13	38	14	12	25	4	4	0	10	15
16	order form	129	90	44	46	0	21	38	50	18	18	32	0	9	10	0	22
17	time	133	63	33	31	14	20	20	47	8	15	26	10	18	15	22	0
18	taxi	119	58	20	9	0	24	18	24	14	24	9	0	12	7	10	11
19	online taxi	90	39	20	4	0	25	26	43	4	8	3	0	12	4	7	7
20	rubbish	94	52	17	31	1	30	24	13	8	4	11	0	7	5	14	3
21	Beijing	98	32	10	9	0	13	14	34	8	8	9	0	12	6	3	11
22	video	123	48	20	9	2	8	8	24	14	8	3	1	7	3	1	2
23	hour	96	58	28	37	1	10	21	38	5	17	22	1	10	9	10	13
24	software	93	44	19	18	0	13	9	16	8	7	12	0	7	6	10	14
25	get on	102	72	31	18	0	7	17	46	20	14	17	0	8	8	16	11
26	car owner	69	23	29	17	0	29	23	17	2	13	15	0	12	6	16	11
27	system	80	36	17	29	11	13	14	29	4	7	13	4	7	4	13	8
28	condition	79	51	27	33	0	21	20	39	6	12	17	0	4	9	14	8
29	Shuttle tra	70	24	14	13	0	11	11	14	4	19	9	0	9	4	6	5
30	place	73	51	22	14	5	10	12	26	10	11	6	2	2	7	18	8
31	world	73	5	3	1	42	4	3	23	2	0	3	27	3	5	3	2
32	Liu Qing	75	34	15	34	0	10	17	38	8	15	12	0	10	3	9	10
33	user	49	23	13	6	4	6	11	19	1	5	4	2	7	5	4	7
34	function	66	39	9	14	6	9	6	18	5	9	13	2	2	13	5	2
35	friend	60	17	12	7	2	4	10	27	3	6	3	1	16	3	5	6
36	city	64	20	7	4	8	11	14	29	5	5	4	5	15	4	4	8
37	thing	58	39	20	22	1	16	17	31	4	8	12	1	4	5	13	9

taxi	online taxi	rubbish	Beijing	video	hour	software	get on	car owner	system	condition	Shuttle transfer	place	world	Liu Qing	user	function	friend	city
119	90	94	98	123	96	93	102	69	80	79	70	73	73	75	49	66	60	64	58
58	39	52	32	48	58	44	72	23	36	51	24	51	5	34	23	39	17	20	39
20	20	17	10	20	28	19	31	29	17	27	14	22	3	15	13	9	12	7	20
9	4	31	9	9	37	18	18	17	29	33	13	14	1	34	6	14	7	4	22
0	0	1	0	2	1	0	0	0	11	0	0	5	42	0	4	6	2	8	1
24	25	30	13	8	10	13	7	29	13	21	11	10	4	10	6	9	4	11	16
18	26	24	14	8	21	9	17	23	14	20	11	12	3	17	11	6	10	14	17
24	43	13	34	24	38	16	46	17	29	39	14	26	23	38	19	18	27	29	31
14	4	8	8	14	5	8	20	2	4	6	4	10	2	8	1	5	3	5	4
24	8	4	8	8	17	7	14	13	7	12	19	11	0	15	5	9	6	5	8
9	3	11	9	3	22	12	17	15	13	17	9	6	3	12	4	13	3	4	12
0	0	0	0	1	1	0	0	0	4	0	0	2	27	0	2	2	1	5	1
12	12	7	12	7	10	7	8	12	7	4	9	2	3	10	7	2	16	15	4
7	4	5	6	3	9	6	8	6	4	9	4	7	5	3	5	13	3	4	5
10	7	14	3	1	10	10	16	16	13	14	6	18	3	9	4	5	5	4	13
11	7	3	11	2	13	14	11	11	8	8	5	8	2	10	7	2	6	8	9
0	11	5	6	8	5	14	12	5	3	11	11	7	1	1	7	3	5	11	3
11	0	0	13	9	0	1	0	4	1	6	6	4	2	7	6	0	11	15	8
5	0	0	3	1	7	19	1	6	6	3	2	5	0	1	1	3	2	1	2
6	13	3	0	8	3	5	4	4	1	2	9	3	2	7	3	3	9	12	3
8	9	1	8	0	3	3	6	3	2	6	4	1	0	6	1	3	6	5	1
5	0	7	3	3	0	6	5	9	7	6	6	7	0	10	4	3	3	0	3
14	1	19	5	3	6	0	6	7	6	8	8	8	1	0	5	7	3	2	2
12	0	1	4	6	5	6	0	5	11	8	6	14	2	2	4	7	3	5	11
5	4	6	4	3	9	7	5	0	6	9	1	2	2	9	3	4	3	3	5
3	1	6	1	2	7	6	11	6	0	5	2	5	3	6	3	1	3	1	3
11	6	3	2	6	6	8	8	9	5	0	4	10	2	7	2	3	6	5	7
11	6	2	9	4	6	8	6	1	2	4	0	5	0	4	1	2	3	3	1
7	4	5	3	1	7	8	14	2	5	10	5	0	3	2	1	3	1	6	5
1	2	0	2	0	0	1	2	2	3	2	0	3	0	0	1	0	1	2	3
1	7	1	7	6	10	0	2	9	6	7	4	2	0	0	2	2	7	6	4
7	6	1	3	1	4	5	4	3	3	2	1	1	1	2	0	1	8	4	0
3	0	3	3	3	3	7	7	4	1	3	2	3	0	2	1	0	2	3	3
5	11	2	9	6	3	3	3	3	3	6	3	1	1	7	8	2	0	6	1
11	15	1	12	5	0	2	5	3	1	5	3	6	2	6	4	3	6	0	2
3	8	2	3	1	3	2	11	5	3	7	1	5	3	4	0	3	1	2	0

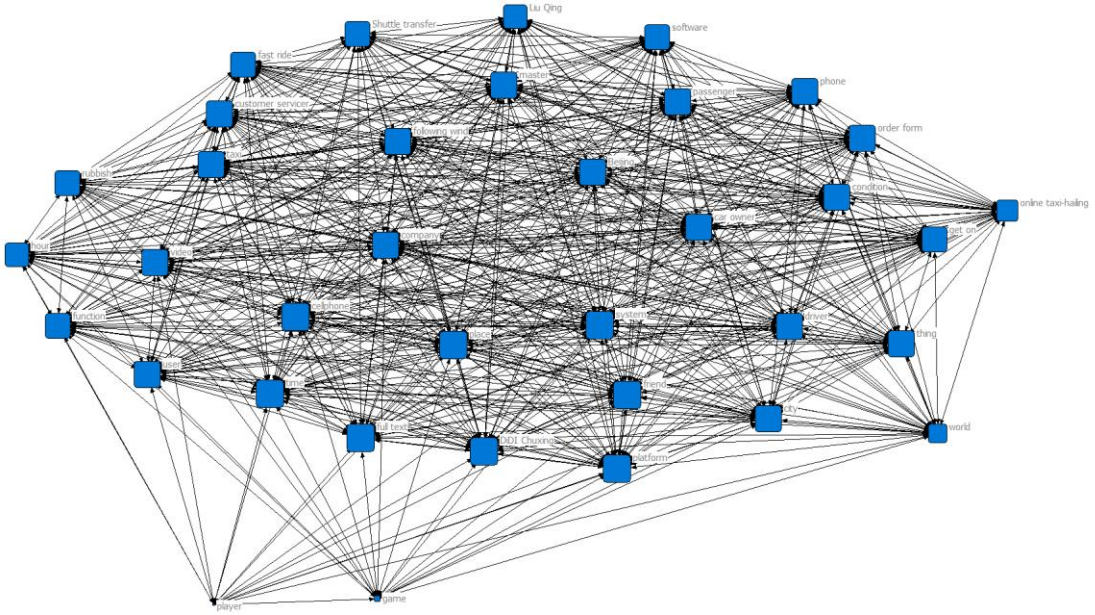
将 Ucient6 生成的 36x36 共现矩阵导入 Netdraw 中进行可视化，形成网约车在线评论语义网络图 degree　centrality 分析图:



betweennesss centrality 分析图如下:

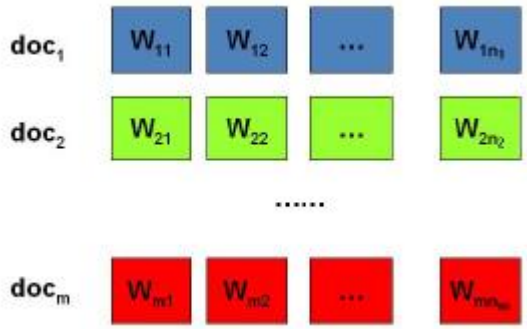


eigenvector centrality 分析图如下:

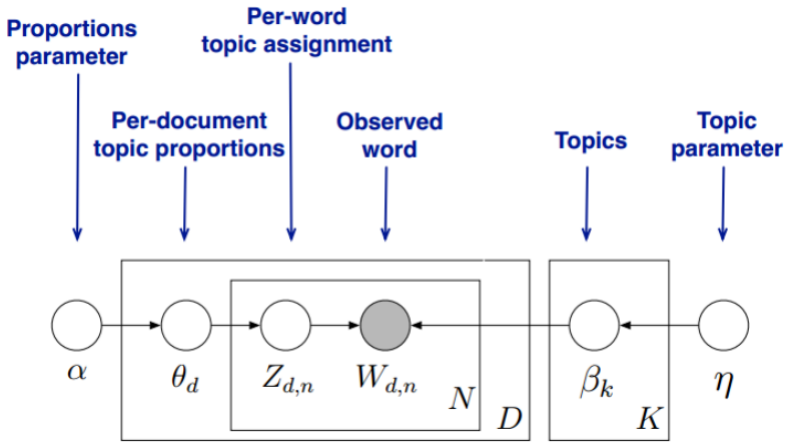


5.基于 LDA 模型的网约车评论分析

展开定义一些字母的含义：文档集合 D，主题（topic)集合 T ， D 中每个文档 d 看作一个单词序列<w1,w2,...,wn>，wi 表示第 i 个单词，设 d 有 n 个单词。



D 中涉及的所有不同单词组成一个大集合 V，LDA 以文档集合 D 作为输入，希望训练出的两个结果向量，对每个 D 中的文档 d，对应到不同 Topic 的概 θd<pt1,...,ptk>，其中，pti 表示 d 对应 T 中第 i 个 topic 的概率。计算方法是直观的，pti=n_{ti}/n，其中 nti 表示 d 中对应第 i 个 topic 的词的数目，n 是 d 中所有词的总数。对每个 T 中的 topict，生成不同单词的概率 φt<pw1,...,pwm>，其中，pwi 表示 t 生成 VOC 中第 i 个单词的概率。计算方法同样很直观，pwi=N_{wi}/N，其中 N_{wi} 表示对应到 topict 的 VOC 中第 i 个单词的数目，N 表示所有对应到 topic 的单词总数。LDA 的公式：p(w|d)=p(w|t)*p(t|d)直观的看这个公式，就是以 Topic 作为中间层，可以通过当前的 θd 和 φt 给出了文档 d 中出现单词 w 的概率。其中 p(t|d)利用 θd 计算得到，p(w|t)利用 φt 计算得到。实际上，利用当前的 θd 和 φt，我们可以为一个文档中的一个单词计算它对应任意一个 Topic 时的 p(w|d)，然后根据这些结果来更新这个词应该对应的 topic。然后，如果这个更新改变了这个单词所对应的 Topic，就会反过来影响 θd 和 φt。



将文本数据导入模型，得到的 topic 结论为 k=3 结论最佳

Topic1	Topic2	Topic3
DiDi Chuxing	driver	Player
platform	Software	game
Passenger	Car owner	place
Customer servicer	Master	friend
Order form	Shuttle transfer	
phone	Taxi	
rubbish	Liu Qing	
cellphone	Video	
time	Function	
Company	Following wind	
user	Online taxi-hailing	
Fast ride	Beijing	
Condition		
Full text		
World		
City		
Get on		
System		
Hour		

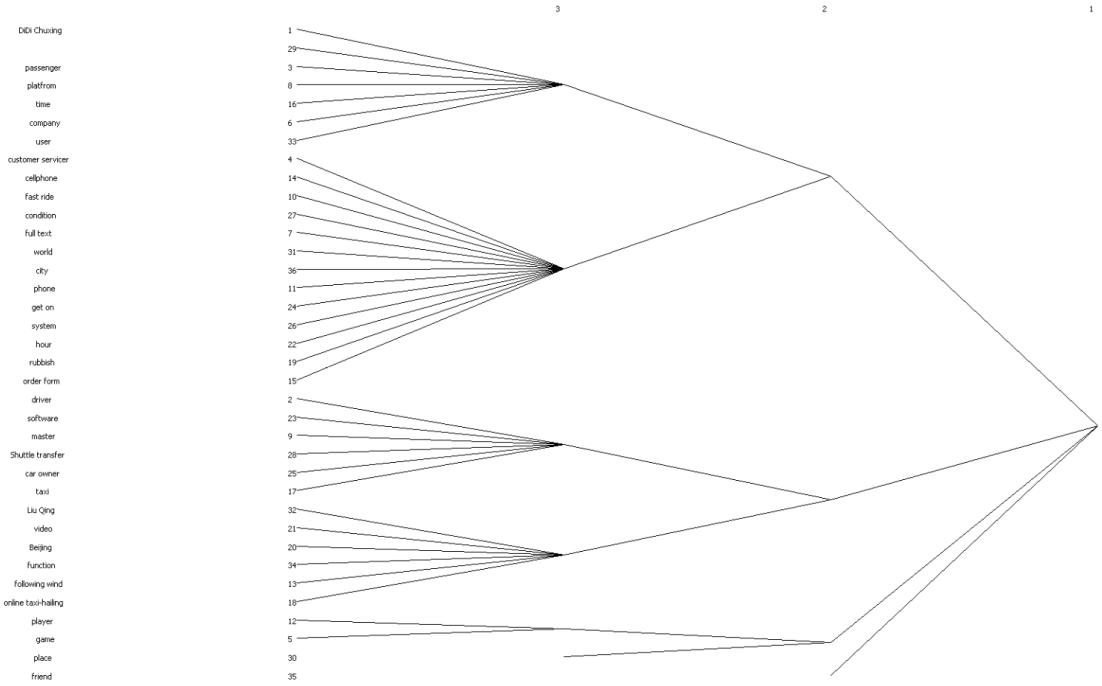
6.基于凝聚子群的滴滴网约车客户评论关键词 CONCOR 分析

CONCOR 分析聚类结果使用 Ucinet 中的 CONCOR 计算可以对待分析的关键词进行聚类计算，同时计算出各大类的密度矩阵，CONCOR 是一种迭代相关收敛法。它基于如下事实：如果对一个矩阵中的各个行之间的相关系数进行重复计算，最终产生的将是一个仅由 1 和-1 组成的相关系数矩阵。进一步说我们可以据此把将要计算的一些项目分为两类：相关系数分别为 1 和-1 的两类

形成的相关系数矩阵如下：

	DiDi	drive	passe	custo	game	compa	full	platf	maste	fast	phone	playe	follo	cellp	order	time	taxi	onlin	rubbi	Beiji	video	hour	softw	get o	car o	syste	condi	Shutt	place	world	Liu Q	user	funct	frien	city
DiDi Chuxing	1.00	0.80	0.92	0.84	0.11	0.90	0.91	0.95	0.88	0.89	0.85	0.21	0.72	0.89	0.87	0.91	0.84	0.74	0.82	0.76	0.89	0.87	0.81	0.87	0.63	0.89	0.88	0.75	0.90	0.32	0.77	0.85	0.90	0.60	0.64	0.88
driver	0.80	1.00	0.99	0.97	0.77	0.99	0.99	0.97	0.98	0.98	0.93	0.64	0.95	0.96	0.97	0.97	0.96	0.91	0.94	0.94	0.96	0.96	0.96	0.96	0.94	0.96	0.96	0.96	0.97	0.72	0.92	0.95	0.96	0.91	0.88	0.95
passenger	0.92	0.99	1.00	0.96	0.62	0.96	0.98	0.95	0.95	0.96	0.93	0.50	0.90	0.95	0.97	0.95	0.96	0.90	0.95	0.90	0.92	0.95	0.93	0.96	0.89	0.93	0.96	0.90	0.97	0.59	0.90	0.93	0.96	0.84	0.84	0.96
customer servicer	0.84	0.97	0.96	1.00	0.68	0.96	0.97	0.96	0.96	0.97	0.98	0.56	0.92	0.97	0.97	0.97	0.94	0.90	0.93	0.92	0.92	0.98	0.94	0.95	0.91	0.97	0.97	0.92	0.95	0.65	0.96	0.93	0.96	0.87	0.85	0.95
game	0.11	0.77	0.62	0.68	1.00	0.72	0.67	0.75	0.73	0.69	0.57	1.00	0.75	0.67	0.60	0.74	0.72	0.71	0.64	0.79	0.78	0.64	0.71	0.64	0.63	0.73	0.60	0.73	0.64	0.98	0.66	0.73	0.70	0.76	0.78	0.60
company	0.90	0.99	0.96	0.96	0.72	1.00	0.99	0.97	0.97	0.98	0.92	0.59	0.96	0.96	0.96	0.97	0.96	0.92	0.95	0.94	0.97	0.96	0.97	0.95	0.93	0.96	0.96	0.95	0.95	0.68	0.93	0.96	0.96	0.91	0.88	0.94
full text	0.91	0.99	0.98	0.97	0.67	0.99	1.00	0.97	0.97	0.98	0.95	0.56	0.94	0.97	0.98	0.98	0.95	0.92	0.93	0.93	0.95	0.98	0.95	0.97	0.91	0.96	0.97	0.93	0.97	0.65	0.95	0.96	0.96	0.90	0.87	0.97
platform	0.95	0.97	0.95	0.96	0.75	0.97	0.97	1.00	0.97	0.97	0.93	0.66	0.94	0.96	0.95	0.98	0.95	0.92	0.95	0.94	0.96	0.96	0.94	0.94	0.90	0.98	0.95	0.94	0.95	0.75	0.95	0.96	0.97	0.91	0.91	0.96
master	0.88	0.98	0.95	0.96	0.73	0.97	0.97	0.97	1.00	0.97	0.91	0.60	0.94	0.96	0.94	0.97	0.96	0.91	0.92	0.95	0.97	0.95	0.96	0.96	0.87	0.95	0.94	0.95	0.97	0.69	0.91	0.95	0.97	0.90	0.89	0.93
fast ride	0.89	0.98	0.96	0.97	0.69	0.98	0.98	0.97	0.97	1.00	0.95	0.56	0.95	0.96	0.97	0.98	0.96	0.92	0.95	0.94	0.95	0.98	0.96	0.95	0.91	0.97	0.97	0.97	0.95	0.65	0.95	0.95	0.96	0.90	0.88	0.95
phone	0.85	0.93	0.93	0.98	0.57	0.92	0.95	0.93	0.91	0.95	1.00	0.46	0.86	0.96	0.97	0.93	0.86	0.83	0.92	0.84	0.86	0.98	0.89	0.93	0.86	0.95	0.97	0.86	0.93	0.55	0.95	0.88	0.93	0.81	0.77	0.97
player	0.21	0.64	0.50	0.56	1.00	0.59	0.56	0.66	0.60	0.56	0.46	1.00	0.61	0.57	0.48	0.65	0.58	0.58	0.52	0.65	0.66	0.53	0.58	0.52	0.51	0.66	0.49	0.59	0.55	0.97	0.55	0.64	0.62	0.65	0.69	0.49
following wind	0.72	0.95	0.90	0.92	0.75	0.96	0.94	0.94	0.94	0.95	0.86	0.61	1.00	0.90	0.90	0.95	0.95	0.94	0.88	0.97	0.96	0.92	0.93	0.89	0.90	0.92	0.90	0.96	0.89	0.68	0.92	0.97	0.90	0.96	0.93	0.88
cellphone	0.89	0.96	0.95	0.97	0.67	0.96	0.97	0.96	0.96	0.96	0.96	0.57	0.90	1.00	0.97	0.96	0.91	0.87	0.91	0.90	0.92	0.98	0.94	0.97	0.85	0.96	0.96	0.90	0.95	0.65	0.94	0.92	0.97	0.86	0.84	0.95
order form	0.87	0.97	0.97	0.97	0.60	0.96	0.98	0.95	0.94	0.97	0.97	0.48	0.90	0.97	1.00	0.95	0.91	0.87	0.93	0.88	0.90	0.98	0.93	0.97	0.89	0.95	0.98	0.89	0.96	0.58	0.93	0.92	0.94	0.83	0.81	0.98
time	0.91	0.97	0.95	0.97	0.74	0.97	0.98	0.98	0.97	0.98	0.93	0.65	0.95	0.96	0.95	1.00	0.94	0.91	0.94	0.94	0.96	0.97	0.94	0.95	0.91	0.98	0.95	0.95	0.95	0.75	0.94	0.97	0.97	0.92	0.90	0.94
taxi	0.84	0.96	0.95	0.94	0.72	0.96	0.95	0.95	0.96	0.96	0.86	0.58	0.95	0.91	0.91	0.94	1.00	0.92	0.93	0.95	0.97	0.91	0.96	0.93	0.88	0.91	0.90	0.97	0.94	0.67	0.88	0.94	0.95	0.90	0.90	0.89
online taxi-hailing	0.74	0.91	0.90	0.90	0.71	0.92	0.92	0.92	0.91	0.92	0.83	0.58	0.94	0.87	0.87	0.91	0.92	1.00	0.86	0.96	0.92	0.87	0.87	0.90	0.87	0.88	0.89	0.90	0.88	0.68	0.89	0.95	0.87	0.94	0.96	0.89
rubbish	0.82	0.94	0.95	0.93	0.64	0.95	0.93	0.95	0.92	0.95	0.92	0.52	0.88	0.91	0.93	0.94	0.93	0.86	1.00	0.87	0.90	0.92	0.95	0.89	0.91	0.94	0.93	0.91	0.91	0.61	0.88	0.89	0.93	0.80	0.80	0.91
Beijing	0.76	0.94	0.90	0.92	0.79	0.94	0.93	0.94	0.95	0.94	0.84	0.65	0.97	0.90	0.88	0.94	0.95	0.96	0.87	1.00	0.97	0.90	0.92	0.90	0.86	0.91	0.89	0.94	0.89	0.74	0.91	0.96	0.92	0.97	0.97	0.87
video	0.89	0.96	0.92	0.92	0.78	0.97	0.95	0.96	0.97	0.95	0.86	0.66	0.96	0.92	0.90	0.96	0.97	0.92	0.90	0.97	1.00	0.91	0.95	0.92	0.86	0.92	0.89	0.96	0.93	0.75	0.89	0.95	0.94	0.93	0.92	0.87
hour	0.87	0.96	0.95	0.98	0.64	0.96	0.98	0.96	0.95	0.98	0.98	0.63	0.92	0.98	0.98	0.97	0.91	0.87	0.92	0.90	0.91	1.00	0.93	0.95	0.89	0.97	0.98	0.91	0.94	0.62	0.96	0.93	0.95	0.87	0.83	0.96
software	0.81	0.96	0.93	0.94	0.71	0.97	0.95	0.94	0.96	0.96	0.89	0.58	0.93	0.94	0.93	0.94	0.96	0.87	0.95	0.92	0.95	0.93	1.00	0.92	0.88	0.93	0.90	0.94	0.93	0.66	0.88	0.92	0.94	0.86	0.85	0.88
get on	0.87	0.96	0.96	0.95	0.64	0.95	0.97	0.94	0.96	0.95	0.93	0.52	0.89	0.97	0.97	0.95	0.93	0.90	0.89	0.90	0.92	0.95	0.92	1.00	0.83	0.92	0.95	0.89	0.98	0.62	0.91	0.93	0.94	0.86	0.84	0.94
car owner	0.63	0.94	0.89	0.91	0.63	0.93	0.91	0.90	0.87	0.91	0.86	0.51	0.90	0.85	0.89	0.91	0.88	0.87	0.91	0.86	0.86	0.89	0.88	0.83	1.00	0.90	0.90	0.91	0.85	0.59	0.87	0.88	0.84	0.84	0.81	0.89
system	0.89	0.96	0.93	0.97	0.73	0.96	0.96	0.98	0.95	0.97	0.95	0.66	0.92	0.96	0.95	0.98	0.91	0.88	0.94	0.91	0.92	0.97	0.93	0.92	0.90	1.00	0.96	0.92	0.94	0.74	0.95	0.93	0.95	0.88	0.86	0.95
condition	0.88	0.96	0.96	0.97	0.60	0.96	0.97	0.95	0.94	0.97	0.97	0.49	0.90	0.96	0.98	0.95	0.90	0.89	0.93	0.89	0.89	0.98	0.90	0.95	0.90	0.96	1.00	0.90	0.95	0.58	0.95	0.92	0.94	0.84	0.83	0.98
Shuttle transfer	0.75	0.96	0.90	0.92	0.73	0.95	0.93	0.94	0.95	0.97	0.86	0.59	0.96	0.90	0.89	0.95	0.97	0.90	0.91	0.94	0.96	0.91	0.94	0.89	0.91	0.92	0.90	1.00	0.90	0.68	0.90	0.94	0.92	0.91	0.90	0.87
place	0.90	0.97	0.97	0.95	0.64	0.95	0.97	0.95	0.97	0.95	0.93	0.55	0.89	0.95	0.96	0.95	0.94	0.88	0.91	0.89	0.93	0.94	0.93	0.98	0.85	0.94	0.95	0.90	1.00	0.64	0.89	0.93	0.95	0.84	0.83	0.95
world	0.32	0.72	0.59	0.65	0.98	0.68	0.65	0.75	0.69	0.65	0.55	0.97	0.68	0.65	0.58	0.75	0.67	0.68	0.61	0.74	0.75	0.62	0.66	0.62	0.59	0.74	0.58	0.68	0.64	1.00	0.63	0.73	0.71	0.73	0.78	0.59
Liu Qing	0.77	0.92	0.90	0.96	0.66	0.93	0.95	0.95	0.91	0.95	0.95	0.55	0.92	0.94	0.93	0.94	0.88	0.89	0.88	0.91	0.89	0.96	0.88	0.91	0.87	0.95	0.95	0.90	0.89	0.63	1.00	0.91	0.90	0.90	0.86	0.94
user	0.85	0.95	0.93	0.93	0.73	0.96	0.96	0.96	0.95	0.95	0.88	0.64	0.97	0.92	0.92	0.97	0.94	0.95	0.89	0.96	0.95	0.93	0.92	0.93	0.88	0.93	0.92	0.94	0.93	0.73	0.91	1.00	0.93	0.95	0.94	0.92
function	0.90	0.96	0.96	0.96	0.70	0.96	0.96	0.97	0.97	0.96	0.93	0.62	0.90	0.97	0.94	0.97	0.95	0.87	0.93	0.92	0.94	0.95	0.94	0.94	0.84	0.95	0.94	0.92	0.95	0.71	0.90	0.93	1.00	0.85	0.85	0.92
friend	0.60	0.91	0.84	0.87	0.76	0.91	0.90	0.91	0.90	0.90	0.81	0.65	0.96	0.86	0.83	0.92	0.90	0.94	0.80	0.97	0.93	0.87	0.86	0.86	0.84	0.88	0.84	0.91	0.84	0.73	0.90	0.95	0.85	1.00	0.97	0.85
city	0.64	0.88	0.84	0.85	0.78	0.88	0.87	0.91	0.89	0.88	0.77	0.69	0.93	0.84	0.81	0.90	0.90	0.96	0.80	0.97	0.92	0.83	0.85	0.84	0.81	0.86	0.83	0.90	0.83	0.78	0.86	0.94	0.85	0.97	1.00	0.84
thing	0.88	0.95	0.96	0.95	0.60	0.94	0.97	0.96	0.93	0.95	0.97	0.49	0.88	0.95	0.98	0.94	0.89	0.89	0.91	0.87	0.87	0.96	0.88	0.94	0.89	0.95	0.98	0.87	0.95	0.59	0.94	0.92	0.92	0.85	0.84	1.00

现将从数据中提取的共现矩阵导入 Ucient6 软件进行 CONCOR 分析，形成聚类图表如下：



7.研究结论

滴滴出行自 2015 年推出，随着中国科技的发展和智能技术的运用，网约车出现在大众视野并得以推广，目前滴滴出行是中国大陆现在最大的网约车线上平台，而消费者对它的出现评论褒贬不一，因此对此爬取评论数据并做了一些研究。

一、对消费者整体文本评论进行关键词提取，选择其中 top36,

DiDi-Chuxing, driver, passenger, customer-servicer, game, company, full-text, platform, master, fast-ride, phone, player, following-wind, cellphone, order-form, time, taxi, online-taxi-hailing, rubbish, Beijing,video,hour, software,get-on, car-owner, system, condition, Shuttle-transfer, place, world, Liu Qing,user,function, friend,city,thing

通过这些关键词以及出现的频数，可以看出消费者在使用滴滴出行时，对司机、客服、公司、平台评论较多，2500 份左右的数据中频数都大于 400，说明用户在使用滴滴出行时关心的因素是服务态度，以及司机配备的安全性和和公司、平台的信任与售后。

二、对文本评论进行 LDA 主题模型处理

在提取文本数据关键词后，继续进行主题模型分析，经过多次训练的数据 topic=3，Topic1 为滴滴出行平台，Topic 为 2 为滴滴出行系统，Topic3 为用户使用场景，在两千多份数据中，用户对滴滴出行的三大块进行评论，说明用户对滴滴出行平台、滴滴出行的系统以及使用过程中都有自己的想法，可以从这三个方面去进行反思。

三、对文本评论语义网络分析-中心度分析、CONCOR 分析

度中心性只是衡量节点中心性的指标之一,指标中介中心性/中间中心性,以经过某个节点的最短路径数目来刻画节点重要性的指标,特征向量中心性,一个节点的重要性既取决于其邻居节点的数量（即该节点的度），也取决于其邻居节点的重要性。在生成的网络图中,可以看到 player 和 game 度较小,而 platform,car owner,driver 较大,频数较大的之间相关性较强，可以发现用户的大多数焦点在于平台、司机、车主。

对文本评论数据进行 CONCOR 分析，将与关键词相关的词汇进行聚类，子类为 7 类，分别为

DiDi-Chuxing, passenger, company, platform,time,user

full-text, master, fast-ride, phone, cellphone, order-form, rubbish, customer-servicer, condition, world, city, get-on,system, hour, thing

driver, software, car-owner, Shuttle-transfer, taxi

Liu Qing, video, Beijing, function, following-wind, online-taxi-hailing

player, game,

friend

place

```
#tf-idf  LDA 主题模型代码
import math
import jieba
import jieba.posseg as psg
from gensim import corpora,models
from jieba import analyse
import functools
#停用词表加载
def get_stopword_list():
    stop_word_path = './stopword.txt'#停用词所在路径
    stopword_list = [sw.replace('\n','')for sw in
open(stop_word_path,encoding='utf8').readlines()]
    return stopword_list
#定义分词方法
#pos 为是否只保留名词的依据
def seg_to_list(sentence,pos=False):
    if not pos:
```

```
            seg_list = jieba.cut(sentence)#不进行词性标注的分析
        else:
            seg_list = psg.cut(sentence)#进行词性标注的分析
        return seg_list
#定义干扰词过滤方法
def word_filter(seg_list,pos=True):
        stopword_list = get_stopword_list()   #获取停用词
        filter_list = []   #存储过滤后的词
        #根据 pos 确定是否执行此行过滤
        #不进行过滤则都标记为 m 并保留
        for seg in seg_list:
            if not pos:
                word = seg
                flag = 'n'
            else:
                word = seg.word
                flag = seg.flag
            if not flag.startswith('n'):
                continue
            #过滤停用词中词以及长度<2 的
            if not word in stopword_list and len(word)>1:
                filter_list.append(word)
        return filter_list
#加载数据集
def load_data(pos=False,corpus_path='./corpus.txt'):
        doc_list = []
        for line in open(corpus_path,'r',encoding='gbk'):
            content=line.strip()
            seg_list = seg_to_list(content,pos)#对句子进行分词
            filter_list = word_filter(seg_list,pos)#对分词结果过滤干扰词
            doc_list.append(filter_list)
        return doc_list
#topk 关键词
def cmp(e1,e2):
        import numpy as np
        res = np.sign(e1[1]-e2[1])
        if res != 0:
            return res
```

```
        else:
            a = e1[0] + e2[0]
            b = e2[0] + e1[0]
            if a>b:
                return 1
            elif a==b:
                return 0
            else:
                return -1
#idf 统计方法
def train_idf(doc_list):
    idf_dic={}
    #文档总数
    tt_count = len(doc_list)
    #每个词出现的文档数
    for doc in doc_list:
        for word in set(doc):
            idf_dic[word]=idf_dic.get(word,0.0)+1.0
    #按工时转化为 idf 值，分母加 1 进行平滑处理
    for k,v in idf_dic.items():
        idf_dic[k] = math.log(tt_count/(1.0+v))
    #对没有出现的次数，默认出现一次
    default_idf = math.log(tt_count/(1.0))
    return idf_dic,default_idf
#YF-IDF 类
class Tfldf(object):
    #参数为：训练好的 idf 字典，默认的 idf 值，处理后的待提取样本，关键词数量
    def __init__(self,idf_dic,default_idf,word_list,keyword_num):
        self.word_list=word_list
        self.idf_dic,self.default_idf=idf_dic,default_idf
        self.tf_dic=self.get_tf_dic()   #tf 数据
        self.keyword_num=keyword_num
    #统计 tf 值
    def get_tf_dic(self):
        tf_dic={}
        for word in self.word_list:
            tf_dic[word]=tf_dic.get(word,0.0)+1.0
        tt_count=len(self.word_list)
```

```
            for k,v in tf_dic.items():
                tf_dic[k]=float(v)/tt_count
            return tf_dic
        #计算 tf-idf 值
        def get_tfidf(self):
            tfidf_dic={}
            for word in self.word_list:
                idf=self.idf_dic.get(word,self.default_idf)
                tf=self.tf_dic.get(word,0)
                tfidf=tf*idf
                tfidf_dic[word]=tfidf
            #根据 tf-idf 的排序，keyword num 个作为关键词
            for k,v in
sorted(tfidf_dic.items(),key=functools.cmp_to_key(cmp),reverse=True)[:self.keyword_num]:
                print(k+'/',end='')
            print()
    #主题模型的类
    class  TopicModel(object):
        #参数:处理后的数据集，关键词数量，具体模型（LSI,LDA），主题数量
        def __init__(self,doc_list,keyword_num,model='LSI',num_topics=4):
            #停用 gensim 的接口将文本转化为向量表示
            #构建词空间
            self.dictionary=corpora.Dictionary(doc_list)
            #使用 BOW 模型向量化
            corpus = [self.dictionary.doc2bow(doc) for doc in doc_list]
            #每个词根据 tf-idf 提取
            self.tfidf_model=models.TfidfModel(corpus)
            self.corpus_tfidf=self.tfidf_model[corpus]
            self.keyword_num=keyword_num
            self.num_topics=num_topics
            #选择加载的模型
            if model=='LSI':
                self.model=self.train_lsi()
            else:
                self.model=self.train_lda()
            word_dic = self.word_dictionary(doc_list)
            self.wordtopic_dic = self.get_wordtopic(word_dic)
        #训练 LSI
```

```
        def train_lsi(self):
            lsi =
models.LsiModel(self.corpus_tfidf,id2word=self.dictionary,num_topics=self.num_topics)
            return lsi
        #训练 lda
        def train_lda(self):
            lda =
models.LdaModel(self.corpus_tfidf,id2word=self.dictionary,num_topics=self.num_topics)
            return lda
            #得到数据集的主题一词分布
            word_dic = self.word_dictionary(doc_list)
            self.wordtopic_dic=self.get_wordtopic(word_dic)
        def get_wordtopic(self,word_dic):
            word_dic = {}
            for word in word_dic:
                single_list = [word]
                wordcorpus=self.tfidf_model[self.dictionary,doc2bow(single_list)]
                wordtopic=self.model[wordcorpus]
                wordtopic_dic[word] = wordtopic
            return wordtopic_dic
        #计算词的分态与文本的相似度，取最高的几个作为关键词
        def get_simword(self,word_list):
            sentcorpus = self.tfidf_model[self.dictionary.doc2bow(word_list)]
            senttopic = self.model[sentcorpus]
            #余弦相似度计算
            def calsim(l1,l2):
                a,b,c =0.0,0.0,0.0
                for t1,t2 in zip(l1,l2):
                    x1 = t1[1]
                    x2 = t2[1]
                    a += x1*x1
                    b += x1*x1
                    c += x2*x2
                sim = a/math.sqrt(b*c) if not (b*c) == 0.0 else 0.0
                #计算输入文本和每个词的主题分布相似度
                sim_dic = {}
                for k,v in self.wordtopic_dic.items():
                    if k not in word_list:
```

```
                    continue
                sim = calsim(v,senttopic)
                sim_dic[k] = sim
            for k,v in
sorted(sim_dic.items(),key=functools.cmp_to_key(cmp),reverse=True)[:self.keyword_num]:
                print(k+'/',end='')
            print()
        #词空间构建方法和向量化方法，在没有 gensim 接口时的一般处理放法
        def word_dictionary(self,doc_list):
            dictionary = {}
            for doc in doc_list:
                dictionary.extend(doc)
            dictionary = list(set(dictionary))
            return dictionary
        def doc2bowvec(self,word_list):
            vec_list = [1 if word in word_list else 0 for word in self.dictionary]
            return vec_list
    if __name__ == '__main__':
        corpus_path='E:\Desktop\semantic analysis\didicomment1.txt'
        pos = True
        for line in open(corpus_path,'r',encoding='gbk'):
            content=line.strip()
            seg_list = seg_to_list(content,pos)#对句子进行分词
        filter_list = word_filter(seg_list,pos)
        print('tf-idf 的结果：')
        #alist = [['行程'],['吴中区'],['权限'],['客服'],['前提'],['事实'],['苏州'],['费用'],['小时'],['司
机'],['滴滴']]
        doc_list    = load_data(pos)
        idf_dic,default_idf = train_idf(doc_list)
        tfidf_model = Tfldf(idf_dic,default_idf,filter_list,keyword_num=11)
        print(tfidf_model.get_tfidf())
        print(tfidf_model.get_tf_dic())
        idic = {}
        adic = {'滴滴': 0.01710739783964967, '行程': 0.7356973209733757, '前提':
0.3789865755429755, '客服': 0.3931205445071967, '司机': 0.07126008072381139, '事实':
0.3470599740576011, '费用': 0.23719385421057773, '权限': 0.4223082743279721, '小时':
0.20069711117339314, '苏州': 0.3260304592687753, '吴中区': 0.44764984358473237}
        for k,v in adic.items():
```

```
        idic[k] = int(v*count)
print(idic)
print('lda 的结果：')
topic_model = TopicModel(doc_list,keyword_num=10,model='LDA')
word_dic = topic_model.word_dictionary(doc_list)
wordtopic_dic = topic_model.get_wordtopic(word_dic)
print(wordtopic_dic)
get_list,get_topic = topic_model.get_simword(filter_list)
print(get_list,get_topic)
a = topic_model.doc2bowvec(filter_list)
```