一、 赛题说明及结果

a) 赛题

基于主题的文本情感分析

竞赛概述

大赛介绍：

在数据已经成为战略资源及经济资产的今天，通过数据挖掘和机器学习方法来分析海量数据，鼓励学科交叉跨界合作，探索以大数据为基础，涉及政府治理、产业升级等的计算方法及解决方案已经成为时代的发展的迫切需求。

CCF 大数据与计算智能大赛（BDCI）由中国计算机学会主办，已成功举办四届（BDCI 2016回顾：http://www.datafountain.cn/projects/ccfbdci-review/），是目前国内最权威的大数据类赛事之一。

BDCI 2017 共开放12道赛题，期待天才的你在众多珍贵的数据中倘徉探索，豪取百万奖金。

【赛题十二】泰一指尚-基于主题的文本情感分析

赛题背景

近年来，文本情感分析技术在网络营销、企业舆情监控、政府舆论监控等扮演越来越重要的角色。鉴于主题模型在文本挖掘领域的优势，基于主题的文本情感分析技术也成为人们关注的热点，其主要任务是通过挖掘用户评论所蕴含的主题、以及对这些主题的情感偏好，来提高文本情感分析的性能。

以网上电商购物评论为例，原始的主题模型主要针对篇幅较大的文档或者评论句子的集合，学习到的主题主要针对整个产品品牌；而现实情形是，用户评论大多针围绕产品的某些特征或内容主题展开（如口味、服务、环境、性价比、交通、快递、内存、电池续航能力、原料、保质期等等，这说明相比于对产品的整体评分， 用户往往更关心产品特征），而且评论文本往往较短。

任务描述

本次大赛提供脱敏后的电商评论数据。参赛队伍需要通过数据挖掘的技术和机器学习的算法，根据语句中的主题特征和情感信息来分析用户对这些主题的偏好，并以<主题，情感词>序对作为输出。

b) 初赛结果，排名（截图）

67		ooooo	0.69167	1	2017-11-17 00:00:00
68	▼ 9	441的小伙伴	0.69117	3	2017-11-19 19:22:44
69		黑马	0.68708	1	2017-11-19 23:48:57

c) 复赛结果（如有），排名（截图）

54	凑个热闹	0.45833	1	2017-12-07 23:49:53
55	441的小伙伴	0.45779	1	2017-12-05 00:00:00
56	机器爱学习	0.4509	1	2017-12-05 12:53:49

二、 组员及分工

a) 任嘉宁 201530612651 数据预处理，编码，算法验证和优化

b) 张子伟 201530613719 算法选择，算法验证和优化

三、 算法说明

在此次比赛中，我们采用的是关键字最大匹配算法。

算法步骤：

1. 提取测试集评论。
2. 建立情感关键字词典 dic， 经我们观察， 一个情感词的情感值基本是固定的，因此我们建立了一个情感关键字字典，即情感词作为字典的索引，情感值作为值。
3. 建立主题集合 set_themebig， 我们把所有在训练集出现过的主题词都收集在一个集合里面。
4. 主题-情感关键字关联， 我们把训练集中同一条评论的主题和情感词关联起来，即这个主题可能可以用这个情感词来形容。
5. 建立否定词集合， 情感词前面有否定词会改变情感值。
6. 主题 正向最大匹配算法，找到测试集中每一个评论出现过的主题词。
7. 情感词 正向最大匹配算法， 找到测试集中每一个评论出现过的情感词。

四、 核心代码

```
# 找到 句子对应情感的 主题

def theme(sent, emotion):

    set_theme = list2[list1.index(emotion)]

    cutsent=sent[:sent.index(emotion)]+' '+sent[sent.index(emotion)+len(emotion):]

    s1 = cutsent

    s2 = cutsent

    s3 = sent

    temp='NULL'

    dist=100

    while s1 != '':

        lens = 6

        if len(s1) < lens:

            lens = len(s1)

        word = s1[:lens]

        if lens == 1:

            if word in set_theme:

                dis=abs(s3.index(word)-s3.index(emotion))

                if dis< dist:
```

```
            dist=dis
            temp=word
        if dist==100:
            while s2 != '':
                lens = 6
                if len(s2) < lens:
                    lens = len(s2)
                word = s2[:lens]
                if lens == 1:
                    if word in set_themebig:
                        dis=abs(s3.index(word)-s3.index(emotion))
                        if dis< dist:
                            dist=dis
                            temp=word
                    return temp
                while word not in set_themebig:
                    word = word[:len(word)-1]
                    if len(word) == 1:
                        break;
                if word in set_themebig:
                    dis=abs(s3.index(word)-s3.index(emotion))
                    if dis< dist:
                        dist=dis
                        temp=word
                s2 = s2[len(word):]
        return temp
    while word not in set_theme:
        word = word[:len(word)-1]
        if len(word) == 1:
            break;
```

```
        if word in set_theme:
            dis=abs(s3.index(word)-s3.index(emotion))
            if dis< dist:
                dist=dis
                temp=word
        s1 = s1[len(word):]
    return temp

# 主函数
def fmm(in_file, out_file):
    input = open(in_file, 'r',encoding='utf-8')
    output = open('temp.txt', 'w',encoding='utf-8')
    for line in input:
        s1 = decomposition(line)
        num_sent = len(s1)
        t = []
        e = []
        v = []
        for x in range(num_sent):
            ss1 = s1[x]
            while s1[x] != '':
                lens = 6
                if len(s1[x]) < lens:
                    lens = len(s1[x])
                word = s1[x][:lens]
                if lens == 1:
                    if word in dic:
                        ss2=list(jieba.cut(ss1))
                        lenword=len(ss2)
                        for y in range(lenword):
```

```
            if word==ss2[y]:
                num_nega=0
                for z in negative:
                    if z in ss1[max(0,ss1.index(word)-4):ss1.index(word)]:
                        num_nega=num_nega+1
                        word_negative=z+word
                if num_nega>0:
                    if word_negative in e and theme(ss1,word)==t[e.index(word_negative)]:
                        pass
                    else:
                        e.append(word_negative)
                        if theme(ss1,word) in t:
                            t.append('NULL')
                        else:
                            t.append(theme(ss1, word))
                        v.append(str((-1)*int(dic[word])))
                        break
                else:
                    if word in e and theme(ss1,word)==t[e.index(word)]:
                        pass
                    else:
                        e.append(word)
                        if theme(ss1,word) in t:
                            t.append('NULL')
                        else:
                            t.append(theme(ss1, word))
                        v.append(dic[word])
                        break
        break
    while word not in dic:
```

```
    word = word[:len(word)-1]
    if len(word) == 1:
        break;
if word in dic:
    if len(word) == 1:
        ss2=list(jieba.cut(ss1))
        lenword=len(ss2)
        for y in range(lenword):
            if word==ss2[y]:
                num_nega=0
                for z in negative:
                    if z in ss1[max(0,ss1.index(word)-4):ss1.index(word)]:
                        num_nega=num_nega+1
                        word_negative=z+word
                if num_nega>0:
                    if word_negative in e and theme(ss1,word)==t[e.index(word_negative)]:
                        pass
                    else:
                        e.append(word_negative)
                        if theme(ss1,word) in t:
                            t.append('NULL')
                        else:
                            t.append(theme(ss1, word))
                        v.append(str((-1)*int(dic[word])))
                        break
                else:
                    if word in e and theme(ss1,word)==t[e.index(word)]:
                        pass
                    else:
                        e.append(word)
```

```
                if theme(ss1,word) in t:
                    t.append('NULL')
                else:
                    t.append(theme(ss1, word))
                v.append(dic[word])
                break
else:
    num_nega=0
    for z in negative:
        if z in ss1[max(0,ss1.index(word)-4):ss1.index(word)]:
            num_nega=num_nega+1
            word_negative=z+word
    if num_nega>0:
        if word_negative in e and theme(ss1,word)==t[e.index(word_negative)]:
            pass
        else:
            e.append(word_negative)
            if theme(ss1,word) in t:
                t.append('NULL')
            else:
                t.append(theme(ss1, word))
            v.append(str((-1)*int(dic[word])))
    else:
        if word in e and theme(ss1,word)==t[e.index(word)]:
            pass
        else:
            e.append(word)
            if theme(ss1,word) in t:
                t.append('NULL')
            else:
```

```
                        t.append(theme(ss1, word))
                    v.append(dic[word])
            s1[x] = s1[x][len(word):]
    for x in t:
        output.write(x)
        output.write(';')
    output.write(',')
    for x in e:
        output.write(x)
        output.write(';')
    output.write(',')
    for x in v:
        output.write(x)
        output.write(';')
    output.write('\n')
input.close()
output.close()
data = pd.read_csv('temp.txt',header=None,encoding='utf-8')
data.to_csv(out_file, index = False,header=None, encoding = 'utf-8')
```