Dcard 成大板之使用者特性及文章分析

Dcard 簡介

Dcard 目前為全台最大的大學生匿名社群交流平台，每個人可隱匿姓名甚至可以隱匿系級、學校，在此平台上自由發言，因此有極大的言論自由，由此平台之文章可真切了解到在線學生們的真實感受。卡友每天都能抽一張卡，透過對方的自我介紹決定是否加入對方好友，若兩方皆同意，便能互相成為卡友，互相聊天。　Dcard 中除了有各種喜好看板，也有校園看板，讓同校的學生能夠互相分享有關於校園的人事物。

為何選擇分析這些資料？

由於個人也算是常在 Dcard 板上遊蕩的卡友，平時較喜歡看熱門看板的貼文，但也總是會點進成大的看板，看最近有沒有發生什麼事。然而，常常看到「成大 116 廢文板」等類似貼文或留言回覆。便不禁讓我好奇，Dcard 成大板的使用者有哪些特性、最常使用那些字詞、還有最令成大學生們的文章有哪些、發文頻率等等。

分析完使用者特性後，我們便可得知發文者大都來自何系，以及知道大部分的文章類型，若有任何需要透過校園傳遞之消息，便可得知此板的最大客群為哪些類型，透過 Dcard 傳遞重要訊息，建立有效溝通之平台。

爬蟲程式：

原本是要直接使用 Dcard 的 API 去作爬蟲，但發現他爬一次只能爬 30 筆文章資料，要作分析的化資料量相當不足。後來在網路上查到了這支「Dcard-spider」爬蟲程式。他透過使用者輸入資料數，將此數除以 30，計算需爬的 page 數，再把資料都爬下來。除了爬文章，他也可以下載圖片、文章，並可以針對特定資料作爬蟲，因此速度相當快。

我的程式會重複使用到以下程式碼：

```
from dcard import Dcard
dcard = Dcard()
article_metas = dcard.forums('ncku').get_metas(num=number, sort='new')
```

第一二行便是去 import 其爬蟲程式，並讓 dcard 為 Dcard 的 instance。
第三行的 forums()是搜尋到我要去的版，get.metas( )可以幫我把我要的文章數內容全部爬下來，其參數 num 是輸入需要的文章數量，sort 可以依使用者需求要按照最新或是熱門排序。

該爬蟲程式 get.metas( )實際程式碼如下

```
    def get_metas(self, name, sort, num, before, timebound=''):

        def filter_metas(metas):
            if num >= 0 and page == pages:
                metas = metas[:num - (pages - 1) * self.metas_per_page]
            if timebound:
                metas = [m for m in metas if m['updatedAt'] > timebound]
            return metas

        def eager_for_metas(bundle):
            page, metas = bundle
            if num >= 0 and page == pages + 1:
                return False
            if len(metas) == 0:
                logger.warning('[%s] 已到最末頁，第%d 頁!', name, page)
            return len(metas) != 0

        def get_single_page_metas():
            while True:
```

```
            yield self.client.get_json(url, params=params)

        url = route.posts_meta(name)
        params = {'popular': 'true' if sort == 'popular' else 'false'}
        if before:
            params['before'] = before

        pages = -(-num // self.metas_per_page)

        paged_metas = zip(count(start=1), get_single_page_metas())

        for page, metas in takewhile(eager_for_metas, paged_metas):
            params['before'] = metas[-1]['id']
            metas = filter_metas(metas)
            if len(metas) == 0:
                return
            yield metas
```

url = route.posts_meta(name)　此函式為組成到該版的網址，name 為傳入版的名稱。

pages = -(-num // self.metas_per_page) 為計算我需要抓的頁數(每頁 30 筆，self.metas_per_page=30)

for page, metas in takewhile(eager_for_metas, paged_metas):
此式為持續往前找尋使用者的資料數，直到尋找完畢或是無資料為止。

若 eager_for_metas()=False，此迭代就會停止，此為 take while(predicate, iterable)的特性。而 eager_for_metas()=False 會發生在當他找完我們需要的資料頁數的資料時，表示其已經全部搜尋完，回傳 False。


```
[
   {
       "id": 229293954,
       "title": "深度心理治療",
       "excerpt": "目前非常需要找固定心理師（希望療程約十年以上），但是又不希望因為更換居住城市而受影響，各位朋友能否告訴肥宅該怎麼做？",
       "anonymousSchool": false,
```

```
        "anonymousDepartment": true,
        "pinned": false,
        "forumId": "abc4fa45-a456-49c6-8948-8981ad0b0f97",
        "replyId": null,
        "createdAt": "2018-07-21T14:40:56.125Z",
        "updatedAt": "2018-07-21T14:40:56.125Z",
        "commentCount": 1,
        "likeCount": 0,
        "withNickname": false,
        "tags": [],
        "topics": [],
        "forumName": "成功大學",
        "forumAlias": "ncku",
        "gender": "M",
        "school": "國立成功大學",
        "replyTitle": null,
        "reportReason": "",
        "hidden": false,
        "withImages": false,
        "withVideos": false,
        "media": []
    },
    {
```


由爬蟲程式爬下來的程式會如上圖，為一個 list 將多個 dict 包在裡面的形式。因此我使用 while 迴圈，尋找 article_metas[i]中我要的 key 值，例如:性別、科系、愛心數等，以此方式將所需資料記錄起來，再進一步作圖分析。接下來將會對各程式作介紹並對各主題作推論分析。

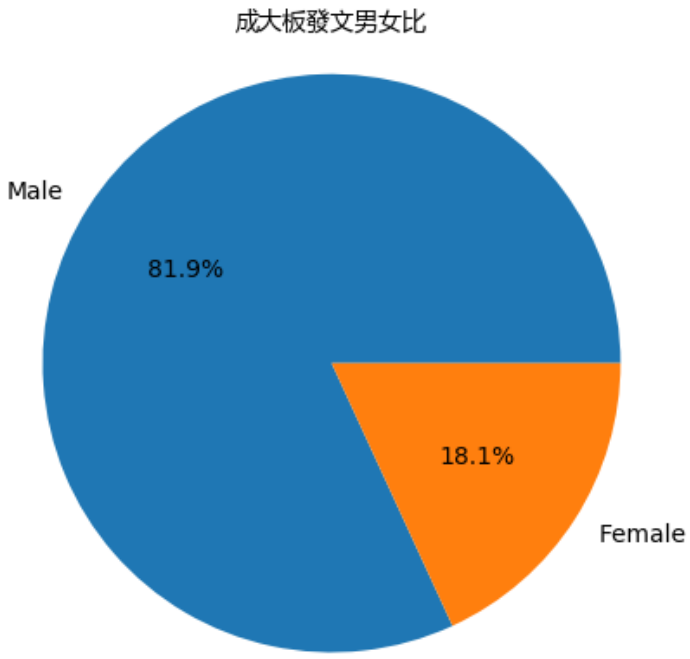
使用者特質

成大板發文男女比例：

```
    def gender(number):
        article_metas = dcard.forums('ncku').get_metas(num=number,
sort='new')
        male=0
        female=0
        with open('gender.txt','w+',encoding='utf-8-sig') as s:
            i=0
            while (i<number):
                if article_metas[i]["gender"]=="M":
                    male+=1
                else:
                    female+=1
                i+=1

        # record into gender.txt
            text="男 : "+str(male)+"\n女 : "+str(female)
            s.write(text)

        # print
            print("男 : ",male)
            print("女 : ",female)
```

在此直接用變數 male 及 female 紀錄男女個數至 gender.txt，最後再輸出並製成圓餅比例圖。以下位製圖之 code。

```
        # make diagram
            labels = ['Male','Female']
            size = [male,female]
            plt.title('成大板發文男女比',fontproperties=myfont)
            plt.pie(size , labels = labels,autopct='%1.1f%%')
            plt.axis('equal')
            plt.show()
```



在往前的 10000 篇的貼文中，有 8186 篇貼文來自於男性，1814 篇貼文來自於女性，男性發文占 81.9%，女性發文占 18.1%。由此可見 Dcard 成大上的男性使用者較為活躍。

成大板發文匿系性 及 最常發文之科系(不匿系)：

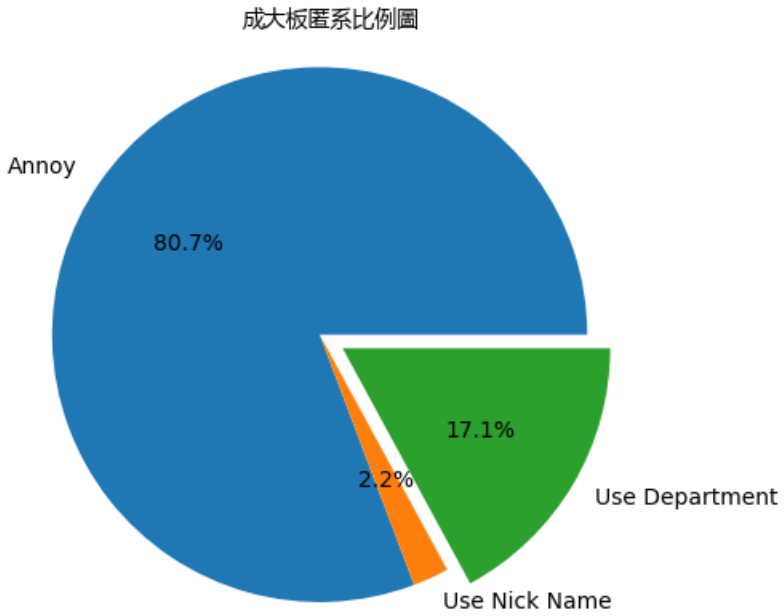
```
    def department(number):
        article_metas = dcard.forums('ncku').get_metas(num=number,
sort='new')
        department={}   #建立科系的字典，key 為科系，value 為數量
        anony=0
        nonanony=0
        use_depart=0
        with open('department.text','w+',encoding='utf-8-sig') as s:
            i=0
            while(i<number):
                if article_metas[i]["anonymousDepartment"]==True:
                    anony+=1
                else:
                    nonanony+=1
                    if article_metas[i]["withNickname"]==False:
                        use_depart+=1

                   depart=article_metas[i]["department"]
                                                #紀錄該科系到變數 depart

                        if  depart in department.keys() :
                                                 #如果已有該科系的 key
                            department[depart]+=1   #該系 value+1
                        else:
                            department[depart]=1  #新增該系的 key，value=1
                i+=1
            department_sorted=sorted(department.items(), key = lambda
item:item[1],reverse=True) #將此字典翻轉
            data=[]
            labels=[]

            for j in range(10):
                data.append(department_sorted[j][1])
                labels.append(department_sorted[j][0])

            print("匿名 : ",anony,"\n 不匿名 : ",nonanony,"\n 使用科
系:",use_depart)
```

```
            total="總共"+str(anony+nonanony)
            text=total+"\n 匿名:"+str(anony)+" 不匿名:"+str(nonanony)+" 使
用科系: \n"+str(use_depart)+" 使用暱稱: \n"+str(nonanony-use_depart)+"\n"
            s.write(text)
            for j in department_sorted:
                s.write(str(j))            #紀錄科系
                s.write("\n")
```

在此用 annoy、nonanony 變數直接紀錄匿系、不匿系文之文章數。另外利用 use_depart 紀錄直接使用系所名稱(非 nick name)的文章數，並依照科系分別紀錄在 department{}的 key(科系)和 value(數量)中，最後利用 sort 便能排出前十最常發文之科系(不匿系)。以下為製圖之 code。

```
      # make diagram
            myfont = FontProperties(fname='C:/Windows/Fonts/MSYH.TTC')
                                                          #引入中文字體
          # make anony diagram - pie chart
            label_annoy = ['Annoy','Use Nick Name','Use Department']
            separated = (0,0,.1)
            size = [anony,nonanony-use_depart,use_depart]
            plt.title('成大板匿系比例圖',fontproperties=myfont)
            plt.pie(size , labels =
label_annoy,autopct='%1.1f%%',explode=separated)
            plt.axis('equal')
            plt.show()
          # make department diagram - bar chart
            df = DataFrame(data, labels)
            ax = df.plot(kind = 'barh', rot = 0,legend=False)
            for label in ax.get_yticklabels() :
                label.set_fontproperties(myfont)
            ax.set_xlabel('Number of Posts')
            ax.set_ylabel('Depaetment')
            ax.invert_yaxis()
            plt.title('前十大高發文數之科系',fontproperties=myfont)
            plt.show()
```

1. **成大板發文匿系性：**

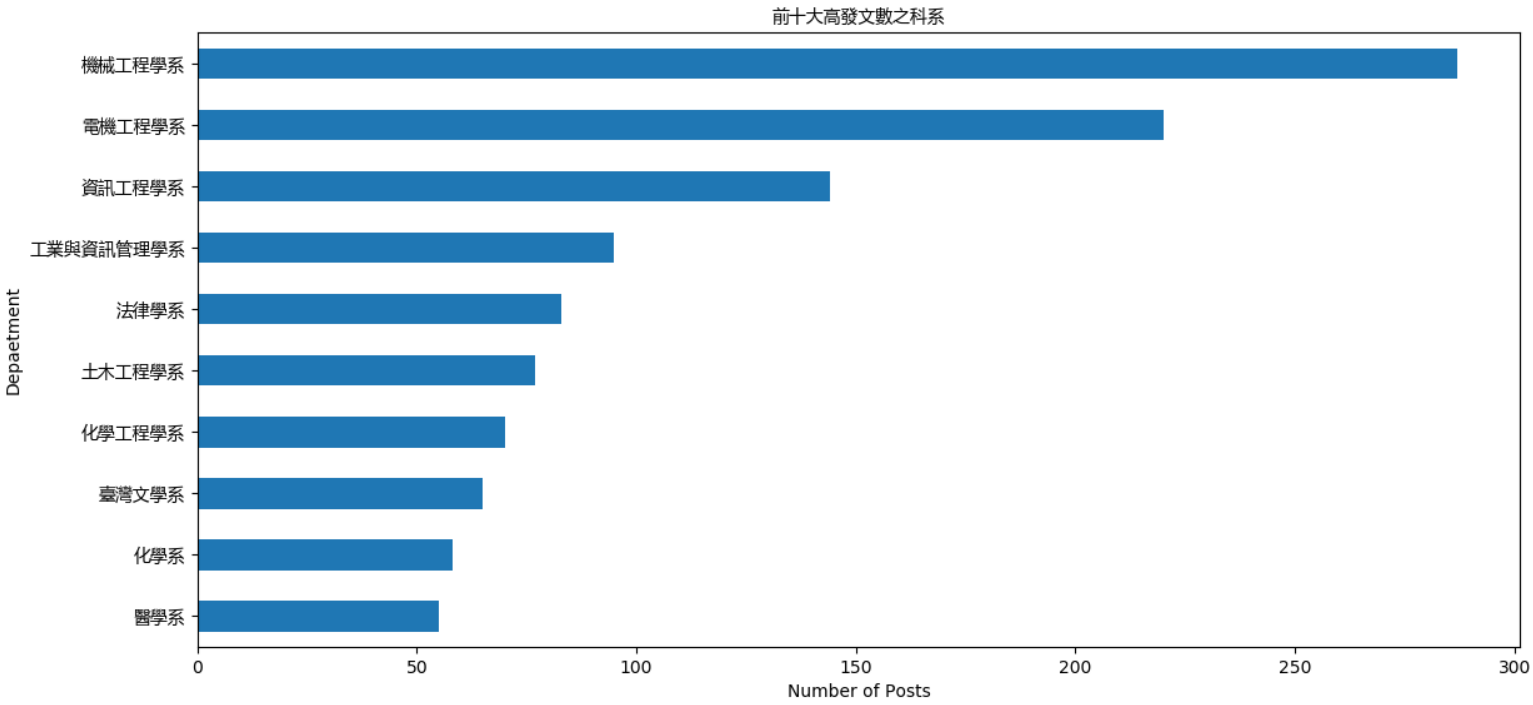


在往前的 10000 篇的貼文中，有 8076 篇匿系，1707 篇無匿系，217 篇使用暱稱。由下圖可見，匿系的使用者比例相當高，高達 80.7%。我認為發文匿系可能有以下幾項原因:

1. 文章內容直接有關系所，因此使用者不敢直接以系所發文
2. 內容範圍狹隘，若被該系上同學看到會直接認出
3. 有提到關於人的特徵且內容有褒貶含意時。
4. 純粹廢文抒發心情時。

而依我平時經驗，大部分情形皆為 4。

2. 成大板最常發文之科系(不匿系)：



從爬蟲 10000 筆資料顯示，裡面有 1707 篇無匿系。因此以此 1707 筆資料作分析，以科系為分類，排名出前十大最常發文之科系。而前五名分別為機械系 287 篇、電機系 219 篇、資訊系 144 篇，皆超過百篇貼文。尤其機械系更是近高達 300 篇。但由此數據並不能直接說明機械系的同學很喜歡發文，頂多只能解釋為機械系同學較光明正大，偏好公開科系發文。

另外觀察此圖表可以發現，前三名就有兩系來自電資學院，而另外八系有四系來自工學院、一系管院、一系醫學院、一系社科院。可見前十大愛發文之科系大都來自二類科系，且與電子硬體設備相處越久科系，發文數越高。

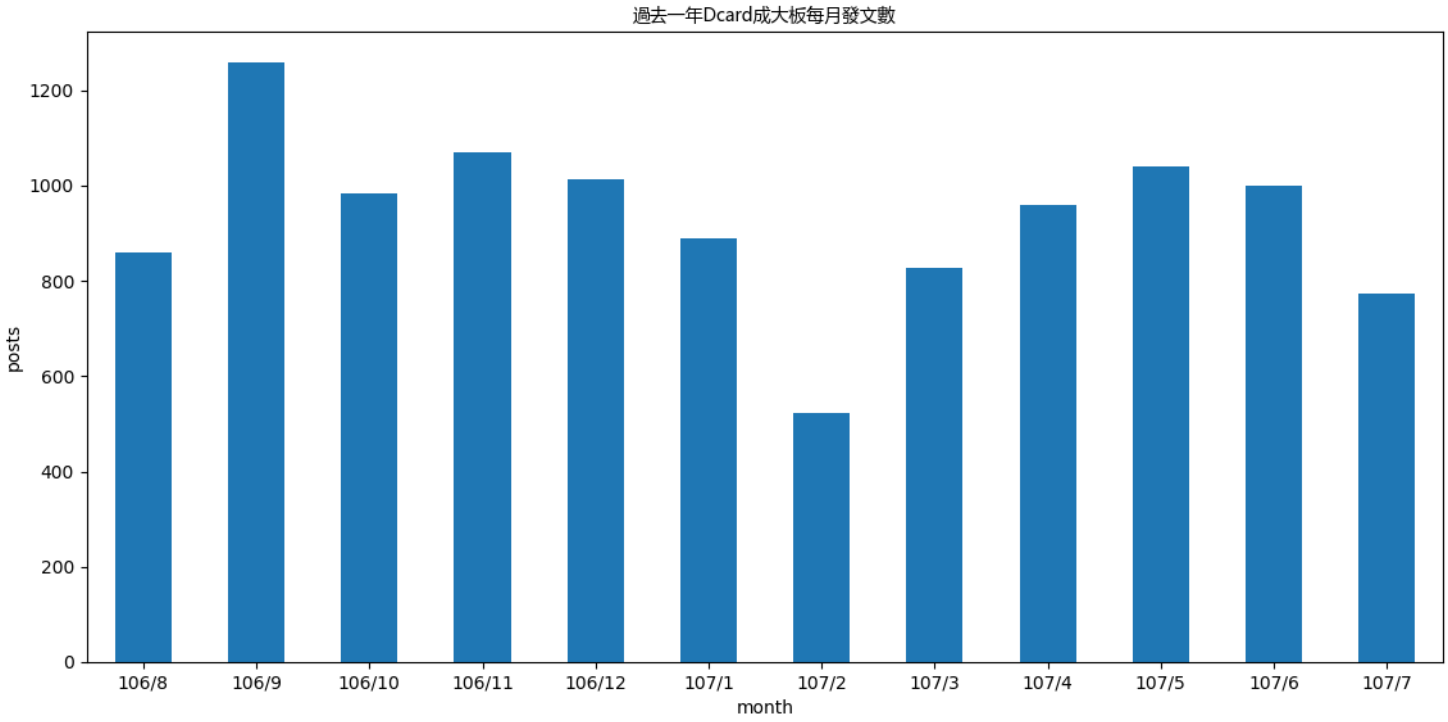
文章分析

成大板每月發文數：

```
    def articles_month():
        with open('month.text','w+',encoding='utf-8-sig') as s:
            article_metas = dcard.forums('ncku').get_metas(num=number,
sort='new')
            i=0
            month_count=[0]*13
            while (i<number):

                date=article_metas[i]["createdAt"]
                index=[5,6]        #日期形式: "2018-07-21T15:41:03.486Z"
                month=''           #預設月份為空字串

                if date[3]=='8' and date[6]=='8':   #不記 2018 年 8 月
                    pass
                else:
                    for j in index:
                        digit=date[j]
                        month+=digit
                    month=int(month)          #記錄該文章月份，並轉為 int

                for k in range(1,13):
                    if month == k:
                        month_count[k]+=1      #該月份文章數+1
                i+=1

            month_count[7]=773                 #107/7 實質為 773 篇
            month_index=0
            for num in month_count:
                if num!=0:
                    result=str(month_index)+"月: "+str(num)+"篇\n"
                    s.write(result)
                month_index+=1
```

為得到過去一年每月發文數，我向前爬了 11500 筆資料，以得到每個月的正確數據。由於 date 形式為字串，其 5.6 位置正好為月份，為不重複記到今年 8 月的月份，另以 date[3]=='8' and date[6]=='8' 判斷若為今年 8 月之文章，則跳過。最後在將記錄好的文章數值記到 month__count 的 list 中，再製成長條圖，以下為製圖之 code。

```
# make diagram
            myfont = FontProperties(fname='C:/Windows/Fonts/MSYH.TTC')
            data=[]
            labels=[]

            for j in range(8,13):                    #106/8 - 106/12
                text="106/"+str(j)
                labels.append(text)                  #x 軸 label
                data.append(month_count[j])          #每月發文數記到 list

            for k in range(1,8):                     #107/1 - 107/7
                text="107/"+str(k)
                labels.append(text)
                data.append(month_count[k])

            df = DataFrame(data, labels)
            ax = df.plot(kind = 'bar', rot = 0,legend=False)  #bar
chart ,legend:不要有圖例
            ax.set_xlabel('month')
            ax.set_ylabel('posts')
            plt.title('過去一年 Dcard 成大板每月發文數
',fontproperties=myfont)
            plt.show()
```



1 月: 890 篇
2 月: 524 篇
3 月: 827 篇
4 月: 960 篇
5 月: 1040 篇
6 月: 999 篇
7 月: 773 篇
8 月: 859 篇
9 月: 1259 篇
10 月: 984 篇
11 月: 1069 篇
12 月: 1012 篇

上圖為從去年(106 年)8 月至今年(107 年)7 月各月發文數之長條圖。由下圖可見，開學時為發文巔峰期，推論為新生潮，在還不認識學長姐的情況下，大多利用此平台向他人詢問問題；2 月時發文數最為低落，個人推論寒假假期較短，大多都正在經歷考試、過年、營隊、實習或是剛開學，此階段處於繁忙時期，季節交替快速，假日匆促而過，較少時間發文。

成大板最熱門的文章：

```
    def like(number):
        article_metas = dcard.forums('ncku').get_metas(num=number,
sort='new')
        with open('like.json','w+',encoding='utf-8-sig') as s:
            i=0
            like_list=[]
            a={"like_max":189,"id":0,"index":0,"title":'0'}  #預設一個較高
的值讓夠高 like 的文章可以存進 list
            for k in range(1):
                like_list.append(a)

            while (i<number):     #開始有 title 功能時
                for d in range(len(like_list)):
                    if article_metas[i]["likeCount"] >
like_list[d]["like_max"]:

b={"like_max":article_metas[i]["likeCount"],"id":article_metas[i]["id"]
,"index":i,"title":article_metas[i]["title"]}
                        like_list.insert(d,b)
                        break
                i+=1
            while len(like_list)>10:    #只留下前十大
                like_list.pop()
            for like in like_list:
                t1=json.dumps(like, indent=4,ensure_ascii=False)
                s.write(t1)

            s.write("\n 文章網址: \n")
            for like in like_list:
                url=like["title"]+" :
"+'https://www.dcard.tw/f/ncku/p/'+str(like["id"])+"\n"
                s.write(url)
```

此 code 利用迴圈將該 list 中所有 dictionary 的 like_count 記錄下來，利用比大小的方式，若比 like_list 第一個值大，則將其 id、index、愛心數、title

值記錄下來，插入至該 list 前頭。最後為留下前十名，用 for 迴圈將 10 名後的文章都 pop 掉。最後將前十名 dict 記錄到 like.json 中，並利用其 id，附上所有文章之網址。以下為用 panda 製作表格的 code。

```
            select_df = pd.DataFrame(like_list)
            out_df = select_df[select_df.loc[:,"like_max"] > 200] # 選出
讚數超過 200 的文章
            print(out_df)
```

```
C:\Users\柯\X\dcard123\dcard-spider>python like.py
          id  index  like_max                                  title
0  227653085   7932       524                          讀了四年的成大
1  227928001   6504       357                              給愛心
2  227785525   7292       353                          南台一日遊
3  227832546   6988       326  新聞：女多於男，恐嫁不掉（內附截圖與殘酷真相）
4  227346590   9362       307              做功德求畢業之成大學生空間整理
5  227329018   9473       295        奧莉薇老闆娘喊破喉嚨，引眾人圍觀不捨
6  229139492   1072       294                  還在奮鬥的夥伴們
7  228621070   3783       281                      關於家境
8  228223843   5243       280            [自薦課程] 人工智慧應用實務
9  229140459   1064       274                  互相幫助的成大
```

文章網址:

讀了四年的成大 ：https://www.dcard.tw/f/ncku/p/227653085
給愛心 ：https://www.dcard.tw/f/ncku/p/227928001
南台一日遊 ：https://www.dcard.tw/f/ncku/p/227785525
新聞：女多於男，恐嫁不掉（內附截圖與殘酷真相） ：
https://www.dcard.tw/f/ncku/p/227832546
做功德求畢業之成大學生空間整理 ：
https://www.dcard.tw/f/ncku/p/227346590
奧莉薇老闆娘喊破喉嚨，引眾人圍觀不捨 ：
https://www.dcard.tw/f/ncku/p/227329018
還在奮鬥的夥伴們 ：https://www.dcard.tw/f/ncku/p/229139492
關於家境 ：https://www.dcard.tw/f/ncku/p/228621070
[自薦課程] 人工智慧應用實務 ：https://www.dcard.tw/f/ncku/p/228223843
互相幫助的成大 ：https://www.dcard.tw/f/ncku/p/229140459

此為往前爬 10000 篇文章後，留下前十高愛心數之文章，並利用程式附上其前十大文章網址。熱名第一名文章高達 524 個愛心，其餘也皆超過 270 個讚。個人有點進去看各文章內容，想藉此分析成大使用者的喜好。但文章性質有點差異過大，大致認為成大學生偏好三類型文章

1. 絕對優質好文，真實寫出作者的觀點及感慨，引起大多人共鳴
2. 長篇創作文，以詼諧方式呈現作者內心感受。俗稱「長篇幹話文」
3. 騙愛心文，該類文章有兩種特性，一是文字絕不超過三句，二是按愛心會有好報。

成大板最常使用之 tag：

Dcard 在 3 月改版，增加了 tag 的新功能，讓 Dcard 使用者們可以在文章下方 tag 任意字串。我找到了第一篇開始使用 tag 的文章，我以該文章為起始點，向後爬到最新的文章(8/6)。

```
    def tag():
        number=4500
        with open('topic.txt','w+',encoding='utf-8-sig') as s:
            article_metas = dcard.forums('ncku').get_metas(num=number,
sort='new')
            i=0
            count=0
            A=set()         #紀錄 tag 的集合
            topic={}        #建立 tag 的 dict，key 為 tag，value 為數量

            while (article_metas[i]["id"]>=228480142):  #首篇有 tag 的文章
                if article_metas[i]["topics"]: # 當 topic 不為空集合時
                    for j in range(len(article_metas[i]["topics"])):
                        tags=article_metas[i]["topics"][j]
                    count+=1
                    A.add(tags)

                    if tags in topic.keys() : # 如果已有該 tag
                        topic[tags]+=1
                    else:
                        topic[tags]=1
                i+=1

            print("不重複 tag 數",len(A))  # 不重複 tag 數為 len(A)個
            print("總 tag 數",count)            # 總 tag 數為 count 個
            key_list=list(topic)    # Dict -> List
            topic_sorted=sorted(topic.items(), key = lambda
item:item[1],reverse=True)

            for j in range(50):             #將前 50 常用 tag 存入 topic.txt
                text=str(j+1)+"\t- "+str(topic_sorted[j][0])+"\t\t:
"+str(topic_sorted[j][1])+"\n"
                s.write(text)
```

```
            for topic in range(20):         #顯示前 20 常用 tag
                print(topic+1,"-
",topic_sorted[topic][0],topic_sorted[topic][1])
```

為紀錄不重複的 tag 數，建立 A 集合將 tag 都記錄在其中。我將成大使用者有打過的 tag 全部記錄起來，計算所有各個 tag 數，藉此分析成大學生最常 tag 的話題。 一共有 2459 個 tag 數，而不重複 tag 數為 640 個。前 20 最常 tag 字詞如下

```
不重複tag數 640
總tag數 2463
```

```
1 -  成大 341
2 -  問卦 206
3 -  廢文 141
4 -  出來面對 140
5 -  QQ 134
6 -  嘻嘻 124
7 -  妹子 102
8 -  問號 59
9 -  好吃嗎 51
10 -  尋人 42
```

```
11 -  姆咪 37
12 -  民航哥 37
13 -  宿舍 35
14 -  不爽 34
15 -  學術 22
16 -  心靈雞湯 20
17 -  蘇慧貞 19
18 -  學妹 18
19 -  轉系 18
20 -  傳說 17
```

第一名 tag 字詞-成大，個人認為無分析價值，暫且跳過。前十大會發現「問卦、廢文、妹子」佔了相當高的比例，可看出成大學生之心靈需求渴望強盛。而其他「出來面對、QQ、嘻嘻、問號、好吃嗎、姆咪」是口頭禪字詞，可視為成大學生經典口語詞。

成大板文章最常見字詞之文字雲：

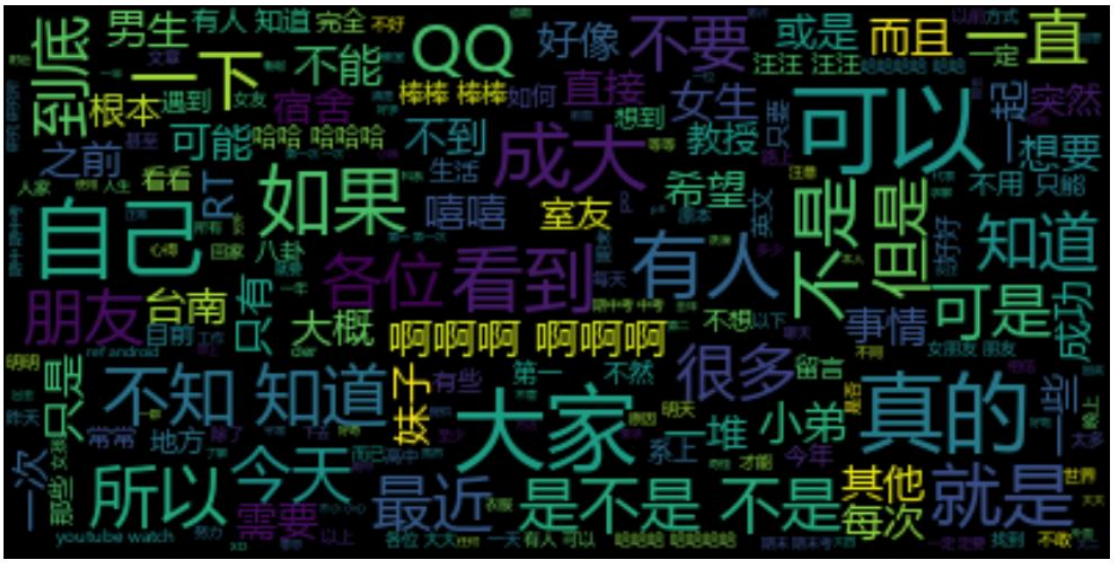
首先同樣向前爬 10000 筆文章，將所有文章內容記錄起來，寫進 content.txt。collect_ids 可以幫助我快速取得所有文章的 id，透過 dcard.post(ids)便能直接取得文章內容。

```
    def collect_ids(metas):
        return [meta['id'] for meta in metas]

    def content(number):
        ids = dcard.forums('ncku').get_metas(num=number,
callback=collect_ids)

        with open('content.txt','w+',encoding='utf-8-sig') as f:
            articles = dcard.posts(ids).get(comments=False, links=False)
            for article in articles.result():
                f.write(article['content'])
```

再利用 jiaba 及文字雲套件，把所有字拆開再組合成常見字詞，分析成大板文章中最常使用之字詞。由於 dcard 支援 imgur 圖片上傳，文章常會出現該圖片網址，因此利用 WordCloud 的 stopwords 功能，輸入該類停用詞，便能不會被計算到文字雲文字頻率中。

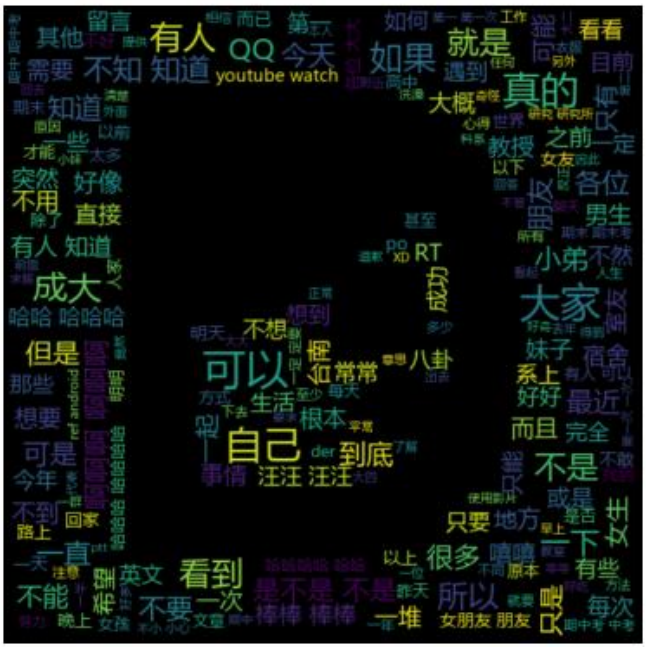
```
import matplotlib.pyplot as plt
from wordcloud import WordCloud
import jieba
from scipy.misc import imread

def make_wordcloud(readtext,imagename):

    text_from_file_with_apath = open(readtext,encoding='utf-8-
sig').read()

    wordlist_after_jieba = jieba.cut(text_from_file_with_apath, cut_all
= True)
    wl_space_split = " ".join(wordlist_after_jieba)    # jieba 中文分词

    stopwords = set()    #停用詞
```

```
stopwords.update(['https:imgur'],['https'],['imgur'],['jpg'],['com'],['dcard'],['tw'],['www'],['http'],['png'])

    my_wordcloud = WordCloud(background_color="white",max_font_size = 35,mask=imagename,stopwords=stopwords,font_path='C:/Windows/Fonts/MSYH.TTC').generate(wl_space_split) #max_font_size:最大的文字大小

    process_word = WordCloud.process_text(my_wordcloud, wl_space_split)
# 查看詞頻，方便重新增加停用词
    sort = sorted(process_word.items(),key=lambda e:e[1],reverse=True) #sort 為 list
    print(sort[:50])

    plt.imshow(my_wordcloud,interpolation='bilinear')
    plt.axis("off")
    plt.show()

while True:
    mask_or_not=input("Do you want to use a Mask? enter Y or N ")
    if mask_or_not=="Y":
        imagename= imread("D.png")   #mask 遮罩圖
        break
    elif mask_or_not =="N":
        imagename=None
        break
    else:
        print("error,please enter the correct character")

readtext="content.txt"
make_wordcloud(readtext,imagename)
```



前十高頻率字詞包含 ：
('可以', 1495), ('大家', 1284), ('真的', 1154), ('自己', 1127), ('成大', 893), ('有人', 829), ('如果', 828), ('不是', 743), ('看到', 730), ('就是', 728)

WordCloud 另外還有以圖片遮罩之功能，得利用這些字詞拼湊成 Dcard 品牌 Lodo 之形狀，如下圖。



結論

根據以上對 Dcard 社群平台成大版的爬蟲分析，可以得知 Dcard 成大版男性發文者偏多，大多來自二類科系，極大多數都匿名發文。熱門的文章類型差異很大，且騙愛心文頗受大家青睞。且成大學生偏好發「問卦、廢文、妹子」類型的話題貼文。

成大板相較於其他校版是相當活躍的校園看板，但發文內容依分析下來篇少有營養貼文。但好貼文仍會受到大家愛戴，得到極高的愛心數。希望成大板發文風氣能夠改善，讓更多好貼文被看到。個人並不排斥廢文，但友善的發文環境應該被建立，而不是留言只見不雅言詞或是戰系。

以上分析及推論大多包含個人主觀意見，可能與真實情況不同，盡情包涵。