

Psychoinformatics - Week 3 (Exercises)

by boyonglin (r10945002@ntu.edu.tw)

1 Analyze what videos go viral? (8 points)

Please use YouTube APIs to carry out a data-driven or hypothesis-driven microstudy about the characteristics of viral videos.

You need to present, here in this notebook, AT LEAST two **statistical** figures or tables as supporting evidence for your arguments. Each of these figures/tables deserves 4 points.

```
In [ ]: !pip install oauth2 google-api-python-client
!pip install --upgrade oauth2client
!pip install jieba
!pip install wordcloud
```

```
In [ ]: DEVELOPER_KEY = "AIzaSyCqN9RPUByoYtYRcd7L6CAveYUI7uWUCQc"

from apiclient.discovery import build
youtube = build('youtube', 'v3', developerKey=DEVELOPER_KEY)

def youtube_search(q, max_results=50, order="relevance", token=None, location=None,
                  locationRadius=None):

    youtube = build('youtube', 'v3', developerKey=DEVELOPER_KEY)

    search_response = youtube.search().list(
        q=q,
        type="video",
        pageToken=token,
        order = order,
        part="id,snippet", # Part signifies the different types of data you want
        maxResults=max_results,
        location=location,
        locationRadius=locationRadius).execute()

    title = []
    channelId = []
    channelTitle = []
    categoryId = []
    videoId = []
    viewCount = []
    likeCount = []
    commentCount = []
    favoriteCount = []
    category = []
    tags = []
    videos = []

    for search_result in search_response.get("items", []):
        if search_result["id"]["kind"] == "youtube#video":
```

```

        title.append(search_result['snippet']['title'])

        videoId.append(search_result['id']['videoId'])

        response = youtube.videos().list(
            part='statistics, snippet',
            id=search_result['id']['videoId']).execute()

        channelId.append(response['items'][0]['snippet']['channelId'])
        channelTitle.append(response['items'][0]['snippet']['channelTitle'])
        categoryId.append(response['items'][0]['snippet']['categoryId'])
        favoriteCount.append(response['items'][0]['statistics']['favoriteCount'])
        viewCount.append(response['items'][0]['statistics']['viewCount'])
        likeCount.append(response['items'][0]['statistics']['likeCount'])

        if 'commentCount' in response['items'][0]['statistics'].keys():
            commentCount.append(response['items'][0]['statistics']['commentCount'])
        else:
            commentCount.append([])
        if 'tags' in response['items'][0]['snippet'].keys():
            tags.append(response['items'][0]['snippet']['tags'])
        else:
            tags.append([])

    youtube_dict = {'tags':tags,'channelId': channelId,'channelTitle': channelTitle}

    return youtube_dict

```

```

In [ ]: import pandas as pd

def spider(col):
    # 抓取原始資料
    test = youtube_search("爛台大")
    df = pd.DataFrame(data=test)

    df[col] = df[col].astype(int)
    df = df.sort_values(by=col, ascending=False)
    df['rankGroup'] = pd.qcut(df[col], q=5, labels=False)

    df_tags_grouped = []
    df_title_grouped = []
    df_mean = []

    for i in range(5):
        df_tags_grouped.append(df[df['rankGroup'] == i]['tags'].tolist())
        df_tags_grouped[i] = [item for sublist in df_tags_grouped[i] for item in sublist]
        df_mean.append(df[df['rankGroup'] == i][col].mean())
        df_title_grouped.append(df[df['rankGroup'] == i]['channelTitle'].tolist())

    data = {'Rank': range(5), 'Mean': df_mean, 'Tags': df_tags_grouped}
    df_table = pd.DataFrame(data)
    df_table = df_table.set_index('Rank')
    print(df_table)

```

```

In [ ]: from collections import Counter
import jieba
import matplotlib.pyplot as plt
import wordcloud

```

```

def wordcloud_draw(do_remove_common=True):
    WORDS_PATH = 'dict.txt.big.txt' # 中文詞庫
    TC_FONT_PATH = 'NotoSansTC-Regular.ttf' # 中文字型
    jieba.set_dictionary(WORDS_PATH)

    STOP_WORDS = [' ', '!', '!', '(', ')', '...', '°', '「', '」', '[', ']']

    def segment_text(text):
        seg_list = jieba.lcut(text)
        return [word for word in seg_list if word not in STOP_WORDS]

    wc = wordcloud.WordCloud(background_color='white',
                             margin=4,
                             font_path=TC_FONT_PATH,
                             max_words=100,
                             width=720, height=720)

    word_counts = {}
    dictionary = {}
    common_words = ['新聞', '直播']
    channel_words = []

    fig, axes = plt.subplots(1, 5, figsize=(15, 3))

    for i in range(5):
        tags_array = df_tags_grouped[i]
        seg_list = ' '.join(tags_array)

        seg_list = segment_text(seg_list)

        dictionary[i] = Counter(seg_list)

        for word, count in dictionary[i].items():
            if word in word_counts:
                word_counts[word] += 1
            else:
                word_counts[word] = 1

    if do_remove_common:
        common_words = [word for word, count in word_counts.items() if count >=
        channel_words = ' '.join(df_title_grouped[0])
        channel_words = segment_text(channel_words)
        channel_words = channel_words + [word.lower() for word in channel_words]

    for i in range(5):
        for word in common_words:
            dictionary[i].pop(word, None)

        for word in channel_words:
            dictionary[i].pop(word, None)

    for i in range(5):
        wc.generate_from_frequencies(dictionary[i])

        axes[i].imshow(wc)
        axes[i].axis("off")
        axes[i].set_title(f"Rank {i}")

    plt.show()

```

```
Building prefix dict from d:\NTU BEBI\112-1 Course\PSY&NI\week3\dict.txt.big.txt
...
Loading model from cache C:\Users\poere\AppData\Local\Temp\jieba.u024aa3f7c6bf972
362f3dacb0b11bb62.cache
```

Rank	Mean	Tags
0	2999.1	[55, 55台, 55頻道, tvbs, tvbs新聞, 新聞直播, 新聞, 直播, NE...
1	21584.3	[公視新聞網, pnn, 公視新聞, 公視, 公視晚間新聞, pts news, pts, ...
2	87229.0	[台視新聞, TTV News, 侯友宜, 2024總統大選, 柯文哲, 台大, 兩岸, 外...
3	366398.6	[中天新聞直播, 中天新聞, 新聞, 政治, 娛樂, 生活, 消費, 國際, 兩岸, 社會, ...
4	2527290.0	[FullVersionNews, 一刀未剪看新聞, 一刀未剪, 新聞大白話, 政論, 新聞...

```
Loading model cost 1.164 seconds.  
Prefix dict has been built successfully.
```



```
Building prefix dict from d:\NTU BEBI\112-1 Course\PSY&NI\week3\dict.txt.big.txt
...
Loading model from cache C:\Users\poere\AppData\Local\Temp\jieba.u024aa3f7c6bf972
362f3dach0b11bb62.cache
```

	Mean	Tags
Rank		
0	2999.3	[55, 55台, 55頻道, tvbs, tvbs新聞, 新聞直播, 新聞, 直播, NE...
1	21584.3	[公視新聞網, pnn, 公視新聞, 公視, 公視晚間新聞, pts news, pts, ...
2	87229.1	[台視新聞, TTV News, 侯友宜, 2024總統大選, 柯文哲, 台大, 兩岸, 外...
3	366398.6	[中天新聞直播, 中天新聞, 新聞, 政治, 娛樂, 生活, 消費, 國際, 兩岸, 社會, ...
4	2527291.7	[FullVersionNews, 一刀未剪看新聞, 一刀未剪, 新聞大白話, 政論, 新聞...

```
Loading model cost 1.184 seconds.  
Prefix dict has been built successfully.
```



Results

Rank 0 to rank 4 are sorted from least to most views in "爛台大".

It can be seen from the word cloud that COVID-19 (疫苗), cross-strait issues (兩岸) and Programming language self-study are of concern to everyone. Gossip (婚姻), Ukraine and lab receive relatively little attention.

And we can see that in the second set of experiments, if the channel name is not removed, 中天, 中時, and 東森 TV stations account for the most network traffic.

Please submit the PDF version of your notebook to NTU COOL before 10/6 (Friday).