Ziwen Zhong 92258649

Topic

Visualize data analyze of which twitch users(streamers) have the higher number of viewers. Get a visualized bar chart of the top 20 streamers who are English speakers in real time. (use twitch's API)

(Topic 4)Explore a social media API of your choice to gather actual data from specific user accounts to interpret learning information (e.g. identify topics students have trouble with, identify the major complaint topics, identify major and positive contributers to discussions, identify types of contributions made in discussions). Create a visual dashboard for an administrator to see these analyses.

Statement of interest

With the rise of live broadcasting industry, it has become a daily entertainment for many people to watch live broadcasting. I want to make a visual data table in order to see which English speaking streamer have the higher viewer counts.

Output of the project

To visually display data of the distribution of different users(streamers) that audiences are interested in and the change of each number of viewers over time. Get a visualized bar chart of the top 20 streamers who are English speakers in real time.

Topic scope of the course

Data collection, data analysis, diagrams, API, data visualize

Steps

1. Understand the goal of analyze and framework (*done week 10*)
   What data to collect?
   What is the goal?
   How will the final output be?
2. Get API access from twitch.tv (*done week 10*)

The new Twitch API provides tools for developing integrations with Twitch. Here, as a quick example to get you started, we make a basic request to get the top streams for a specific game, using the Get Streams endpoint.

Step 1: Setup

To make API calls, you need a client ID. To receive one, log into the Twitch developer dashboard (aka console), select the **Apps** tab, and click **Register Your Application**. Enter an app name and your OAuth redirect URI (where your users are redirected after being authorized), and select an app category. Click **Create**, and the app is created and listed on the dashboard as one of your registered apps. Click **Manage** to see the client ID.

Note: App names must not include "Twitch" (as an exact or fuzzy match), or registration will fail.

Step 2: Sample Code

This gets information about the most active streams for game ID 33214.

```
curl -H 'Client-ID: pOgch4mp101fy451do9uod1s1x9i4a' \
-X GET 'https://api.twitch.tv/helix/streams?game_id=33214'
```

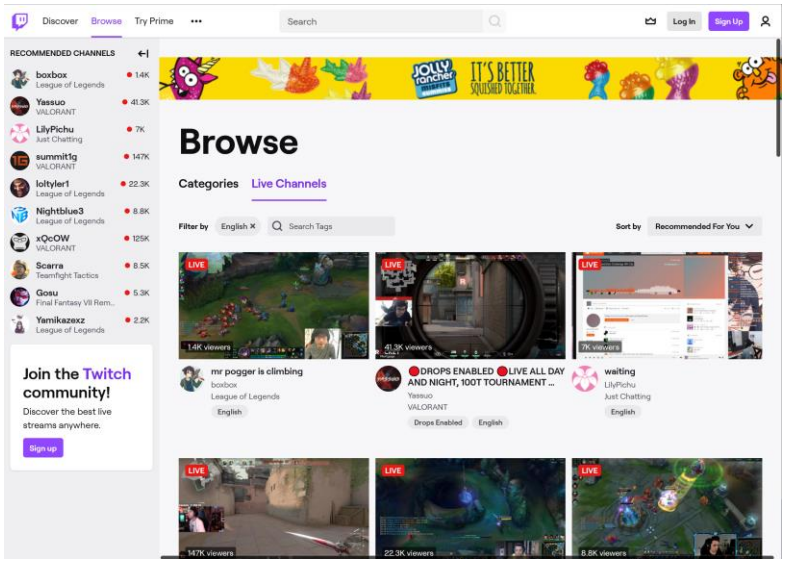
The example uses a dummy client ID. To try it out, copy and paste the example into your terminal, replacing the dummy client ID with your client ID.

Step 3: Results

The API returns an array of stream objects, along with a cursor for pagination. By default, this endpoint returns the 20 most active streams. For brevity, we truncate the long response here:

```
{
  "data": [
```

3. Data collection (use API to gather json data) (*done week 11*)
   3.1. Define top 20 users(streamers) as sample



   3.2. Get the sample's data information

```python
headers = {
    'Client-ID': 'juzpxb4ut6862idxbelm9frlxbp4ry',
}
params = (
    ('broadcaster_language', 'en'),
)

class TwitchSpider(object):
    url = 'https://api.twitch.tv/helix/streams'

    def get_html(self):
        req = requests.get('https://api.twitch.tv/helix/streams', headers=headers, params=params).json()
        return req
```

4. Data processing using python (*due week 11*)
   4.1. Generate the clear and specific data

```
    def get_data(self, req):
        names = jsonpath.jsonpath(req, '$..user_name')
        views = jsonpath.jsonpath(req, '$..viewer_count')
        #print(names,views)
        items = []
        for name, view in zip(names, views):
            item = {'name':name, 'view':view}
            items.append(item)
        return items

    def refine(self, items):
        items = sorted(items, key=self.sort_seed, reverse=True)
        return items

    def sort_seed(self, item):
        return item['view']
```

4.2. Store data into database

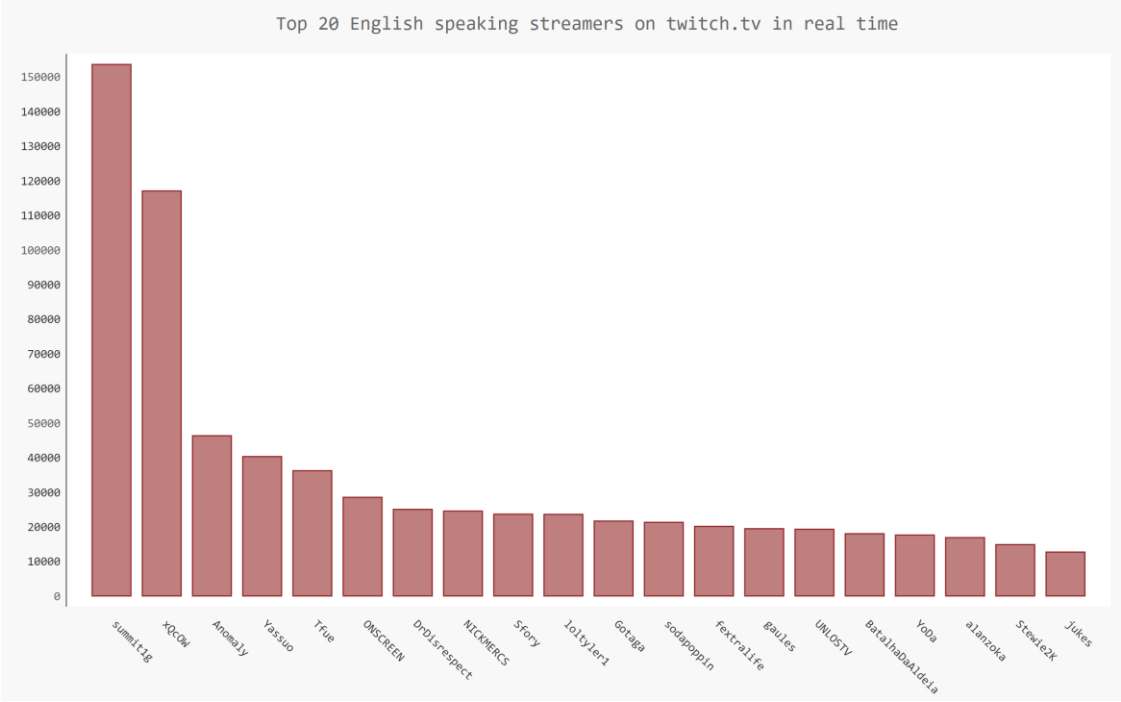
```
with open('repos.json', 'w', encoding='utf-8') as file_write:
    json.dump(items, file_write)
```

5. Data analyze (*done week 12*)

```
[{'name': 'summit1g', 'view': 217319}, {'name': 'ONSCREEN',
'view': 42553}, {'name': 'Anomaly', 'view': 33425}, {'name':
's1mplegjuw', 'view': 25958}, {'name': 'BrookeAB', 'view':
18563}, {'name': 'Greekgodx', 'view': 17797}, {'name': 'brax',
'view': 16996}, {'name': 'nl_Kripp', 'view': 16817}, {'name':
'elded', 'view': 15152}, {'name': 'Lord_Kebun', 'view': 13784},
{'name': 'mrfreshasian', 'view': 13692}, {'name': 'Skadoodle',
'view': 13515}, {'name': 'FEDMYSTER', 'view': 13236}, {'name':
'AlpTV', 'view': 12583}, {'name': 'A_Seagull', 'view': 12240},
{'name': 'Vinesauce', 'view': 11669}, {'name': 'YoDa', 'view':
11433}, {'name': 'NOBRU', 'view': 11153}, {'name': 'aceu',
'view': 10264}, {'name': 'Swagg', 'view': 10242}]
```

6. Visualize data (*done week 12*)

```python
import json
import pygal
from pygal.style import LightColorizedStyle as LCS,LightenStyle as LS

file_path = 'repos.json'
with open(file_path) as f:
    repo_dicts = json.load(f)

names, views = [], []

for repo_dict in repo_dicts:
    names.append(repo_dict['name'])
    views.append(repo_dict['view'])


my_style = LS('#800000', base_style=LCS)

my_config = pygal.Config()
my_config.x_label_rotation = 45
my_config.show_legend = False
my_config.title_font_size = 20
my_config.label_font_size = 16
my_config.major_label_font_size = 16
my_config.truncate_label = 15
my_config.show_y_guides = False
my_config.width = 1000

chart = pygal.Bar(my_config, style=my_style)
chart.title = 'Top 20 English speaking streamers on twitch.tv in real time'
chart.x_labels = names
chart.add('', views)
chart.render_to_file('data_chart.svg')
```



x-aixs shows names of streamers

y-aixs shows number of views of each streamer

7. Test cases which were not working during the project
    1. When requested for the twitch api, console reported

```
{"error":"Unauthorized","status":401,"message":"Must provide a valid Client-ID or OAuth token"}
```

Solution: applied for a twitch development authorization, got a valid client-id.

2. The console reported “cannot resolve module ‘jsonpath’”
   Solution: in cmd, type ‘pip install jsonpath’
3. The console reported “cannot resolve module ‘pygal’”
   Solution: in cmd, type ‘pip install pygal’”

8. Test video, 3 cases which tested on three different date (April 13th, April 19th April 20th). (*done week 14*)
   https://youtu.be/-4ROrPDKxwM