Automatic Sampling and Analysis of YouTube Data

The YouTube API

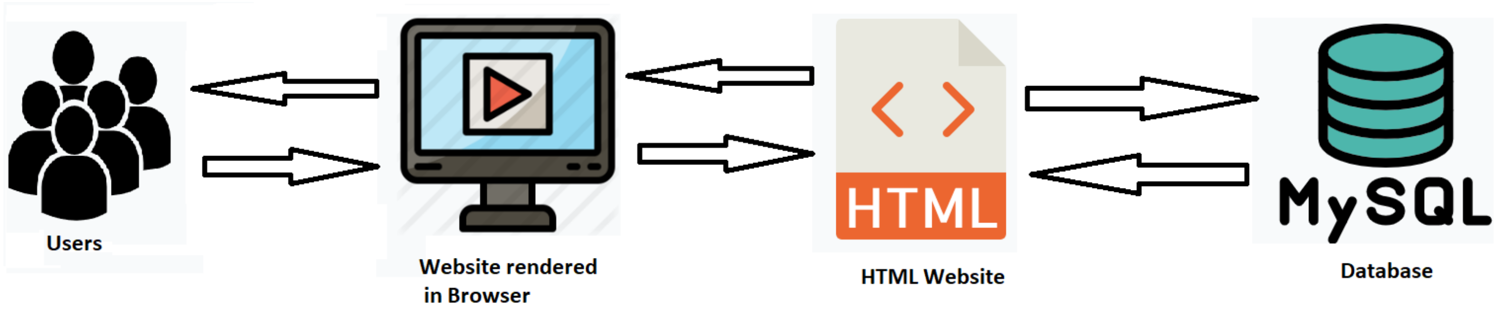
Julian Kohne
Johannes Breuer
M. Rohangis Mohseni

2021-02-24

The YouTube API

Overview

- All data on *YouTube* is stored in a MySQL database
- The website itself is an HTML page, which loads content from this database
- The HTML is rendered by a web browser so the user can interact with it
- Through interacting with the rendered website, we can either retrieve content from the database or send information to the database
- The YouTube website is
    - built in HTML,
    - uses CSS for the "styling"
    - dynamically loads content using Ajax from the Database



How do we get Data From Websites?

- Theoretically, we could gather all the information manually by clicking on the things that are interesting to us and copy/pasting them. However, this is tedious and time-consuming. **We want a way of automatizing this task**. The solution to our problem is...

- Webscraping

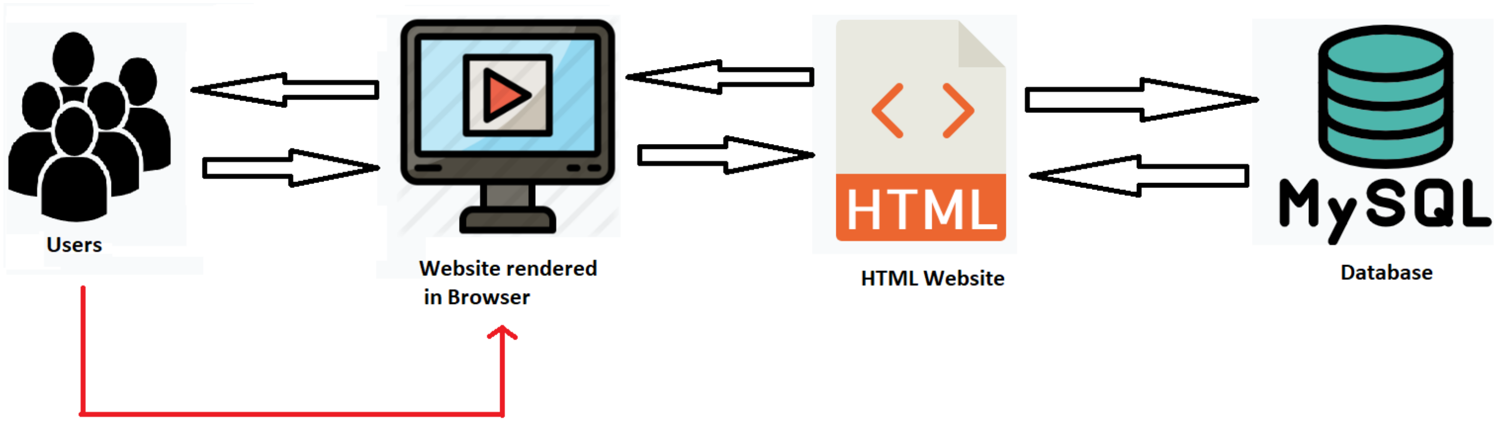
  1) **Screenscraping:** Getting the HTML-code out of your browser, parsing & formatting it, then analyzing the data

  2) **API harvesting:** Sending requests directly to the database and only getting back the information that you want and need

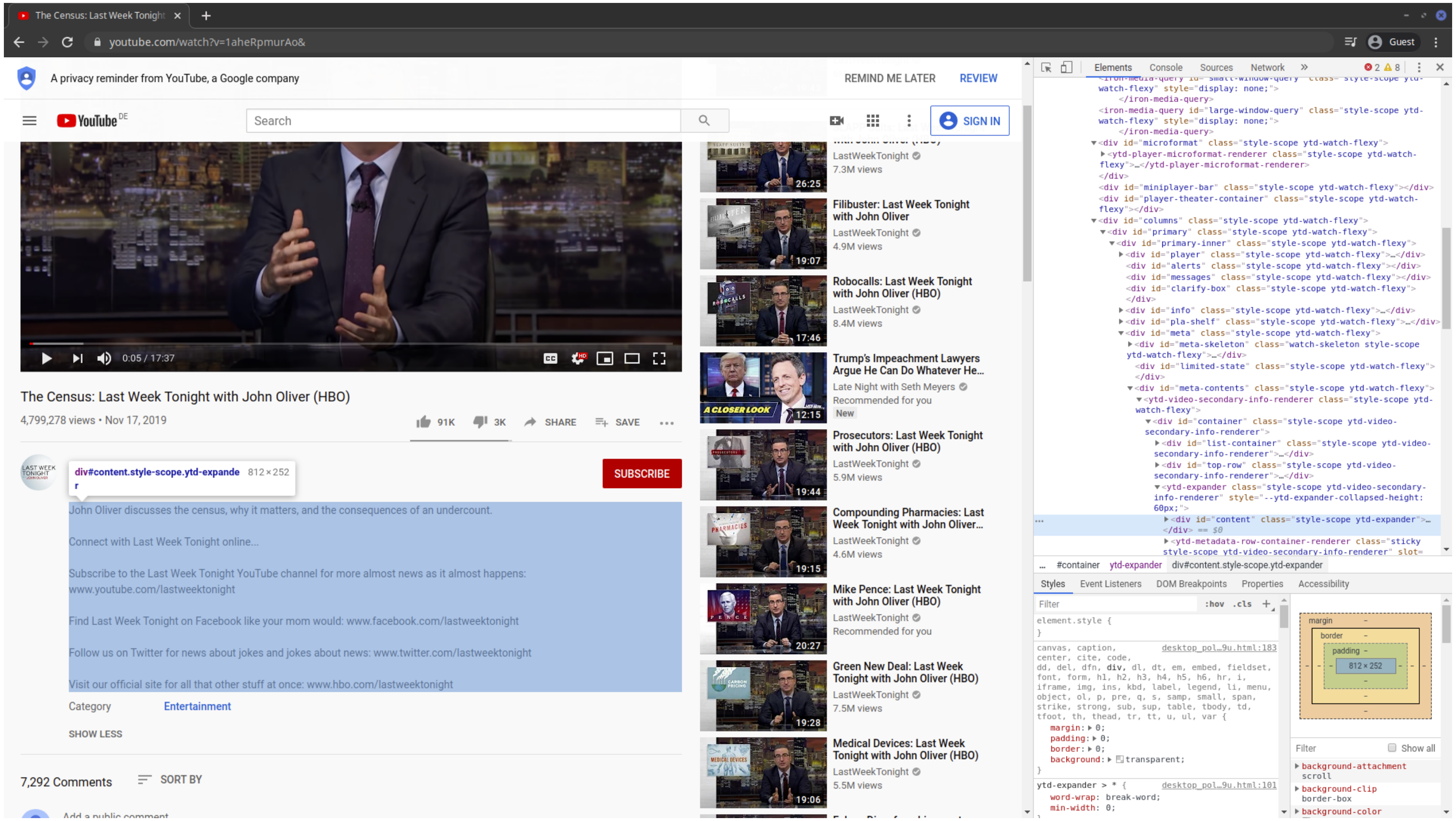
Screenscraping

Screenscraping

- Screenscraping means that we are downloading the HTML text file, which contains the content we are interested in but also a lot of unnecessary clutter that describes how the website should be rendered by the browser



Screenscraping



youtube.com/watch?v=1aheRpmurAo&

A privacy reminder from YouTube, a Google company — REMIND ME LATER — REVIEW

The Census: Last Week Tonight with John Oliver (HBO)

4,799,278 views • Nov 17, 2019 — 91K — 3K — SHARE — SAVE

div#content.style-scope.ytd-expander 812×252

John Oliver discusses the census, why it matters, and the consequences of an undercount.

Connect with Last Week Tonight online...

Subscribe to the Last Week Tonight YouTube channel for more almost news as it almost happens: www.youtube.com/lastweektonight

Find Last Week Tonight on Facebook like your mom would: www.facebook.com/lastweektonight

Follow us on Twitter for news about jokes and jokes about news: www.twitter.com/lastweektonight

Visit our official site for all that other stuff at once: www.hbo.com/lastweektonight

Category — Entertainment

SHOW LESS

7,292 Comments — SORT BY

Filibuster: Last Week Tonight with John Oliver — LastWeekTonight — 4.9M views — 19:07

Robocalls: Last Week Tonight with John Oliver (HBO) — LastWeekTonight — 8.4M views — 17:46

Trump's Impeachment Lawyers Argue He Can Do Whatever He... — Late Night with Seth Meyers — Recommended for you — New — 12:15

Prosecutors: Last Week Tonight with John Oliver (HBO) — LastWeekTonight — 5.9M views — 19:44

Compounding Pharmacies: Last Week Tonight with John Oliver... — LastWeekTonight — 4.6M views — 19:15

Mike Pence: Last Week Tonight with John Oliver (HBO) — LastWeekTonight — Recommended for you — 20:27

Green New Deal: Last Week Tonight with John Oliver (HBO) — LastWeekTonight — 7.5M views — 19:28

Medical Devices: Last Week Tonight with John Oliver (HBO) — LastWeekTonight — 5.5M views — 19:06

```
    watch-flexy" style="display: none;">
        </iron-media-query>
    <iron-media-query id="large-window-query" class="style-scope ytd-
    watch-flexy" style="display: none;">
        </iron-media-query>
  ▼<div id="microformat" class="style-scope ytd-watch-flexy">
    ▶<ytd-player-microformat-renderer class="style-scope ytd-watch-
    flexy">…</ytd-player-microformat-renderer>
    </div>
    <div id="miniplayer-bar" class="style-scope ytd-watch-flexy"></div>
    <div id="player-theater-container" class="style-scope ytd-watch-
    flexy"></div>
  ▼<div id="columns" class="style-scope ytd-watch-flexy">
    ▼<div id="primary" class="style-scope ytd-watch-flexy">
      ▼<div id="primary-inner" class="style-scope ytd-watch-flexy">
        ▶<div id="player" class="style-scope ytd-watch-flexy">…</div>
         <div id="alerts" class="style-scope ytd-watch-flexy"></div>
         <div id="messages" class="style-scope ytd-watch-flexy"></div>
         <div id="clarify-box" class="style-scope ytd-watch-flexy">
         </div>
        ▶<div id="info" class="style-scope ytd-watch-flexy">…</div>
        ▶<div id="pla-shelf" class="style-scope ytd-watch-flexy">…</div>
        ▼<div id="meta" class="style-scope ytd-watch-flexy">
          ▶<div id="meta-skeleton" class="watch-skeleton style-scope
          ytd-watch-flexy">…</div>
           <div id="limited-state" class="style-scope ytd-watch-flexy">
           </div>
          ▼<div id="meta-contents" class="style-scope ytd-watch-flexy">
            ▼<ytd-video-secondary-info-renderer class="style-scope ytd-
            watch-flexy">
              ▼<div id="container" class="style-scope ytd-video-
              secondary-info-renderer">
                ▶<div id="list-container" class="style-scope ytd-video-
                secondary-info-renderer">…</div>
                ▶<div id="top-row" class="style-scope ytd-video-
                secondary-info-renderer">…</div>
                ▼<ytd-expander class="style-scope ytd-video-secondary-
                info-renderer" style="--ytd-expander-collapsed-height:
                60px;">
…                 ▶<div id="content" class="style-scope ytd-expander">…
                  </div> == $0
                  ▶<ytd-metadata-row-container-renderer class="sticky
                  style-scope ytd-video-secondary-info-renderer" slot=
```

… #container ytd-expander div#content.style-scope.ytd-expander

```
element.style {
}
canvas, caption,                    desktop_pol…9u.html:183
center, cite, code,
dd, del, dfn, div, dl, dt, em, embed, fieldset,
font, form, h1, h2, h3, h4, h5, h6, hr, i,
iframe, img, ins, kbd, label, legend, li, menu,
object, ol, p, pre, q, s, samp, small, span,
strike, strong, sub, sup, table, tbody, td,
tfoot, th, thead, tr, tt, u, ul, var {
  margin: ▶ 0;
  padding: ▶ 0;
  border: ▶ 0;
  background: ▶ transparent;
}
ytd-expander > * {                  desktop_pol…9u.html:101
  word-wrap: break-word;
  min-width: 0;
```

```
▶ background-attachment
  scroll
▶ background-clip
  border-box
▶ background-color
```

Screenscraping

- To automatically obtain data, we can use a so-called GET request
- A GET request is an HTTP method for asking a server to send a specific resource (usually an HTML page) back to your local machine
- You can try it out in your console
- This is the basic principle that all the scraping packages build-on
- We will not use this directly and will let the higher-level applications handle this under the hood

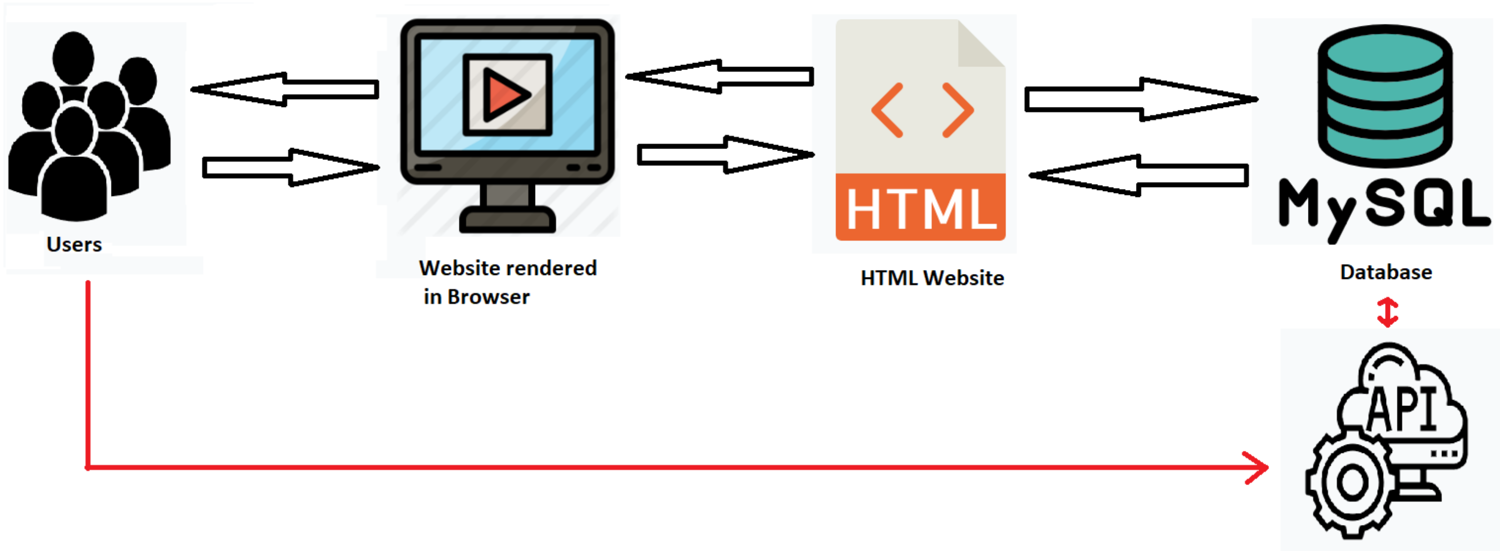
Screenscraping

- Advantages of Screenscraping:
    - You can access everything that you are able to access from your browser
    - You are (theoretically) not restricted in how much data you can get
    - (Theoretically) Independent from API-restrictions
- Disadvantages of Screenscraping:
    - Extremely tedious to get information out of HTML-pages
    - You have to manually look up the Xpaths/CSS/HTML containers to get specific information
    - Reproducibility: The website might be tailored to stuff in your Cache, Cookies, Accounts etc.
    - There is no guarantee that even pages that look the same have the same underlying HTML structure
    - You have to manually check the website and your data to make sure that you get what you want
    - If the website changes anything in their styling, your scripts won't work anymore
    - Legality depends on country

API-Harvesting

API Harvesting

- An Application Programming Interface…
  - is a system built for developers
  - directly communicates with the underlying database(s)
  - is a voluntary service of the website
  - controls what information is accessible, to whom, how, and in which quantities



API-Harvesting

- APIs can be used to:
  - embed content in other applications
  - create bots that do something automatically
  - scheduling/moderation for content creators
  - collect data for (market) research purposes
- Not every website has their own API. However, most large social media Websites, e.g.:
  - Facebook
  - Twitter
  - Instagram
  - Wikipedia
  - Google Maps

API Harvesting

- Advantages of API Harvesting:
    - No need to interact with HTML files, you only get the information you asked for
    - The data you get is already nicely formatted (usually JSON files)
    - You can be confident that what you do is legal (if you adhere to the Terms of Service and respect data privacy and copyright regulations)
- Disadvantages of API Harvesting:
    - Not every website has an API
    - You can only get what the API allows you to get
    - There are often restricting quotas (e.g., daily limits)
    - Terms of Service can restrict how you may use the data (e.g., with regard to sharing or publishing it)
    - There is no standard language to make queries, you have to check the documentation
    - Not every API has a (good) documentation

Screenscraping vs. API-Harvesting

If you can, use an API, if you must, use screenscraping instead

The YouTube API

Summary

- Fortunately, *YouTube* has its own, well-documented API that developers can use to interact with their database (most *Google* services do)
- To find an API for a given website, Programmable Web is a good starting point
- We will use the YouTube API today

Let's Check Out the API!

- Google provides a sandbox for their API that we can use to get a grasp of how it operates
- We can, for example, use our credentials to get search for videos with the keyword "Brexit"
- Example
- Keep in mind: We have to log in with the *Google* account we used to create the app to use the API
- What we get back is a JSON-formatted response with the formats and information we requested in the API sandbox

What is JSON?

- Java Script Object Notation
- Language-independent data format (like .csv)
- Like a nested List of Key:Value pairs
- Standard data format for many APIs and web applications
- Better than tabular formats (.csv / .tsv) for storing large quantities of data by not declaring missing data
- Represented in `R` as a list of lists that typically needs to be transformed into a regular dataframe (this can be tedious)

What is JSON?

```
'{
  "first name": "John",
  "last name": "Smith",
  "age": 25,
  "address": {
    "street address": "21 2nd Street",
    "city": "New York",
    "postal code": "10021"
  },
  "phone numbers": [
    {
      "type": "home",
      "number": "212 555-1234"
    },
    {
      "type": "mobile",
      "number": "646 555-4567"
    }
  ],
  "sex": "male"
}'
```

Most Important Parameters

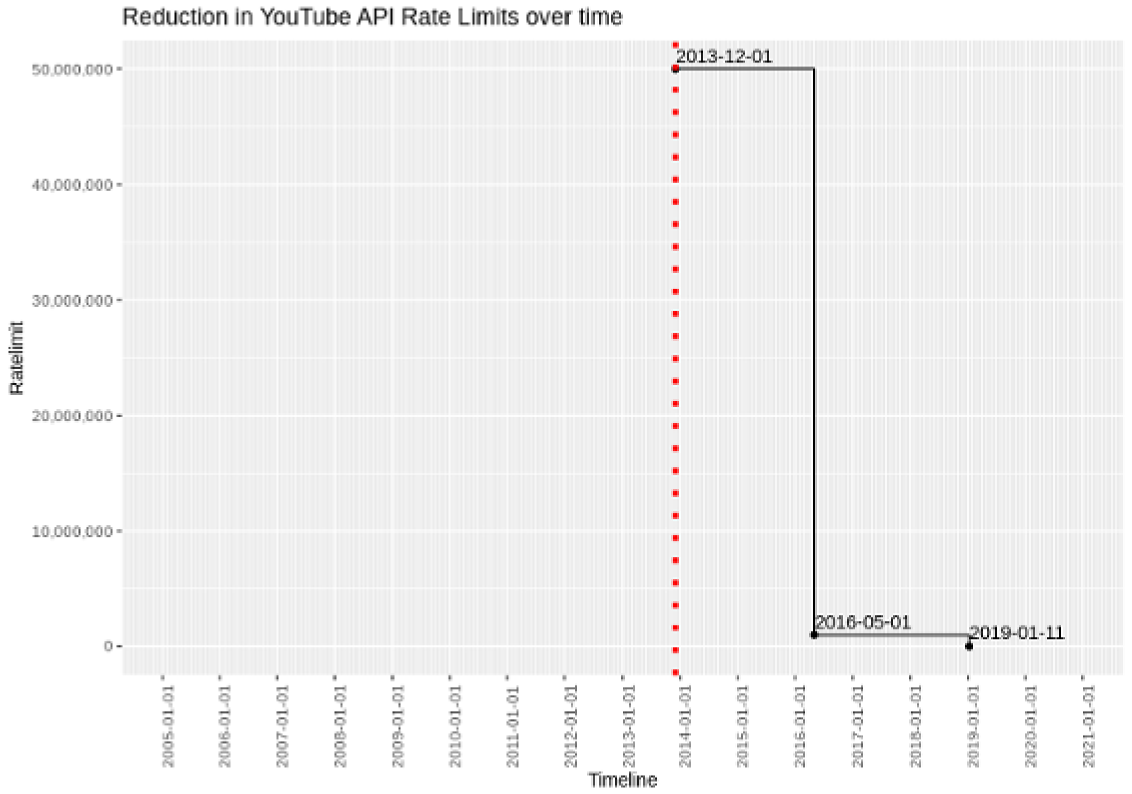
- All possible parameters for the *YouTube* API are listed here
- Keep in mind that some information is only visible to owners of a channel or author of a video
- Keep in mind that not all information is necessarily available for all videos (e.g., live videos)

Using it from R

- We can simplify the process of interacting with the YouTube API by using a dedicated R package
- The package handles the authentication with our credentials and translates R commands into API calls
- It also simplifies the JSON response to a standard dataframe automatically for many requests
- In essence, we can run R commands and get nicely formatted API results back
- For this workshop, we will thus use the tubeR package

Rate Limits

- With the API, you have a limit of how much data you can get
- This limit has constantly decreased over the last decade



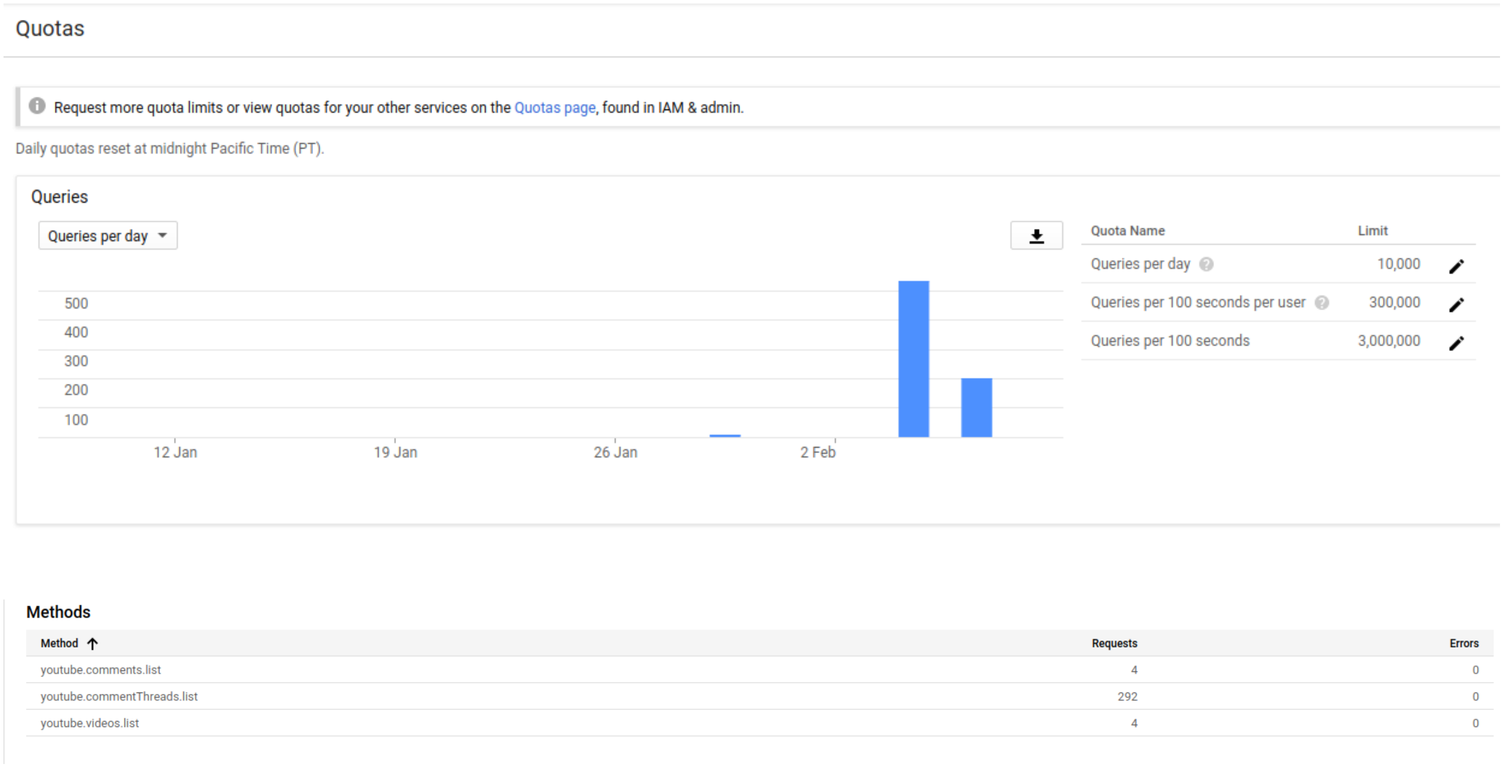
Rate Limits

- Currently (02.2021), you have a quota of **10.000** units per day
- Each request (even invalid ones) costs a certain amount of units
- There are two factors influencing the quota cost of each request:
  - different types of requests (e.g., write operation: 50 units; video upload: 1600 units)
  - how many parts the requested resource has (playlist:2 ; channel:6 ; video:10)
- **You should only request parts that you absolutely need to make the most of your units. More on that in the data collection session.**

NB: Also sending incorrect requests can fill up your daily quota

Rate Limits

- You can check the rate limits in the *YouTube* API Documentation
- You can see how much of your quota you have already used up in the *Google* Developer Console



Quotas

Request more quota limits or view quotas for your other services on the Quotas page, found in IAM & admin.

Daily quotas reset at midnight Pacific Time (PT).

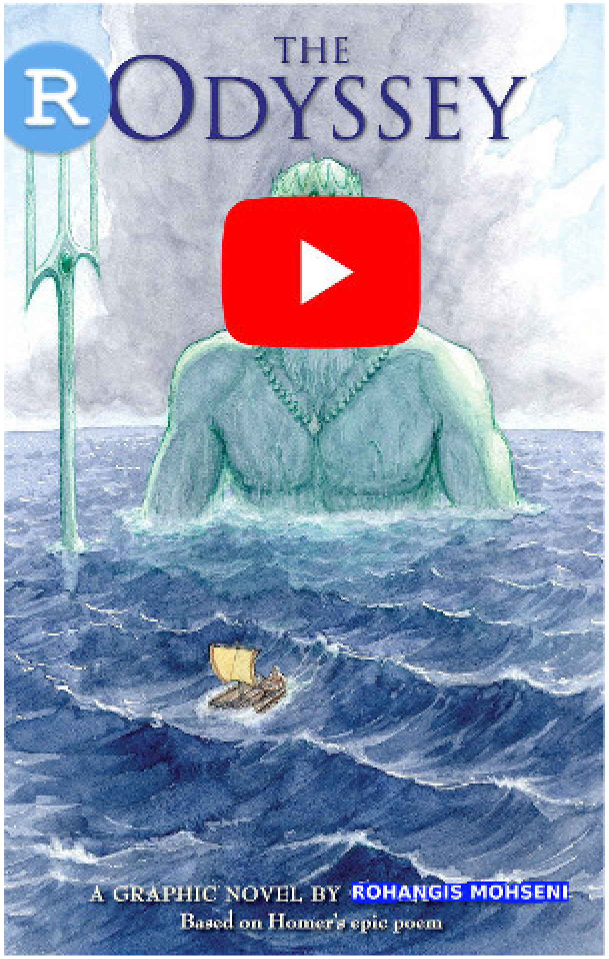
Queries

Quota Name	Limit
Queries per day	10,000
Queries per 100 seconds per user	300,000
Queries per 100 seconds	3,000,000

Methods

Method	Requests	Errors
youtube.comments.list	4	0
youtube.commentThreads.list	292	0
youtube.videos.list	4	0

Can I Increase my Rate Limit?


THE
R ODYSSEY
A GRAPHIC NOVEL BY ROHANGIS MOHSENI
Based on Homer's epic poem

Trying to Raise the *YouTube* API Quota

- Study that needs large datasets in a short amount of time
- RQ: Is there a u-shaped relationship between success and number of uploads?
- Sample: 600 popular channels (identified via SocialBlade)
- Request for higher quota (October 11, 2019)
- Problem: Same application form for (web) apps and research
- Hard to figure what applies to research and what to write into the form
- Experience: Stuck in an infinite loop with e-mails from *Google* support on this issue

Any questions?

Exercise time 🏋 💪 🏃 🚴

Solutions