Automatic Sampling and Analysis of YouTube Data

The YouTube API

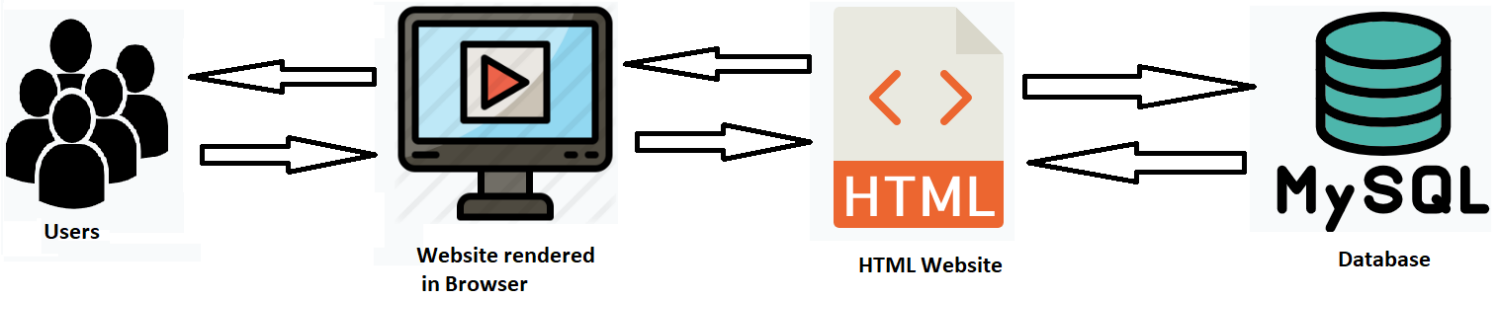
Julian Kohne
Johannes Breuer
M. Rohangis Mohseni

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 webbrowser 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**

- Webscraping

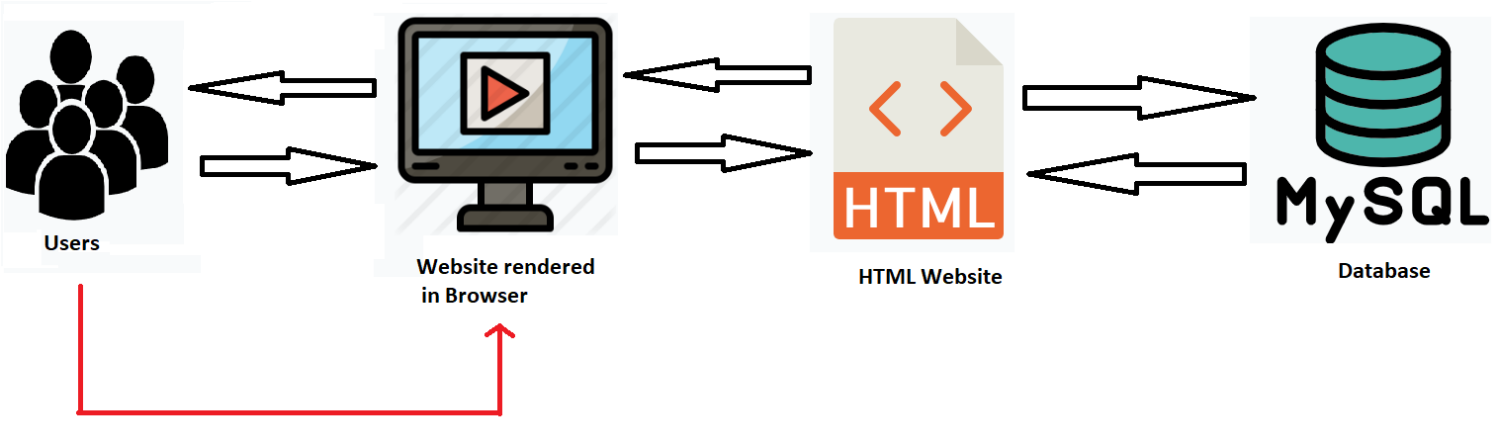
  1) **Screenscraping:** Getting the HTML-code out of your browser, parsing & formatting it, then analyzing the data

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

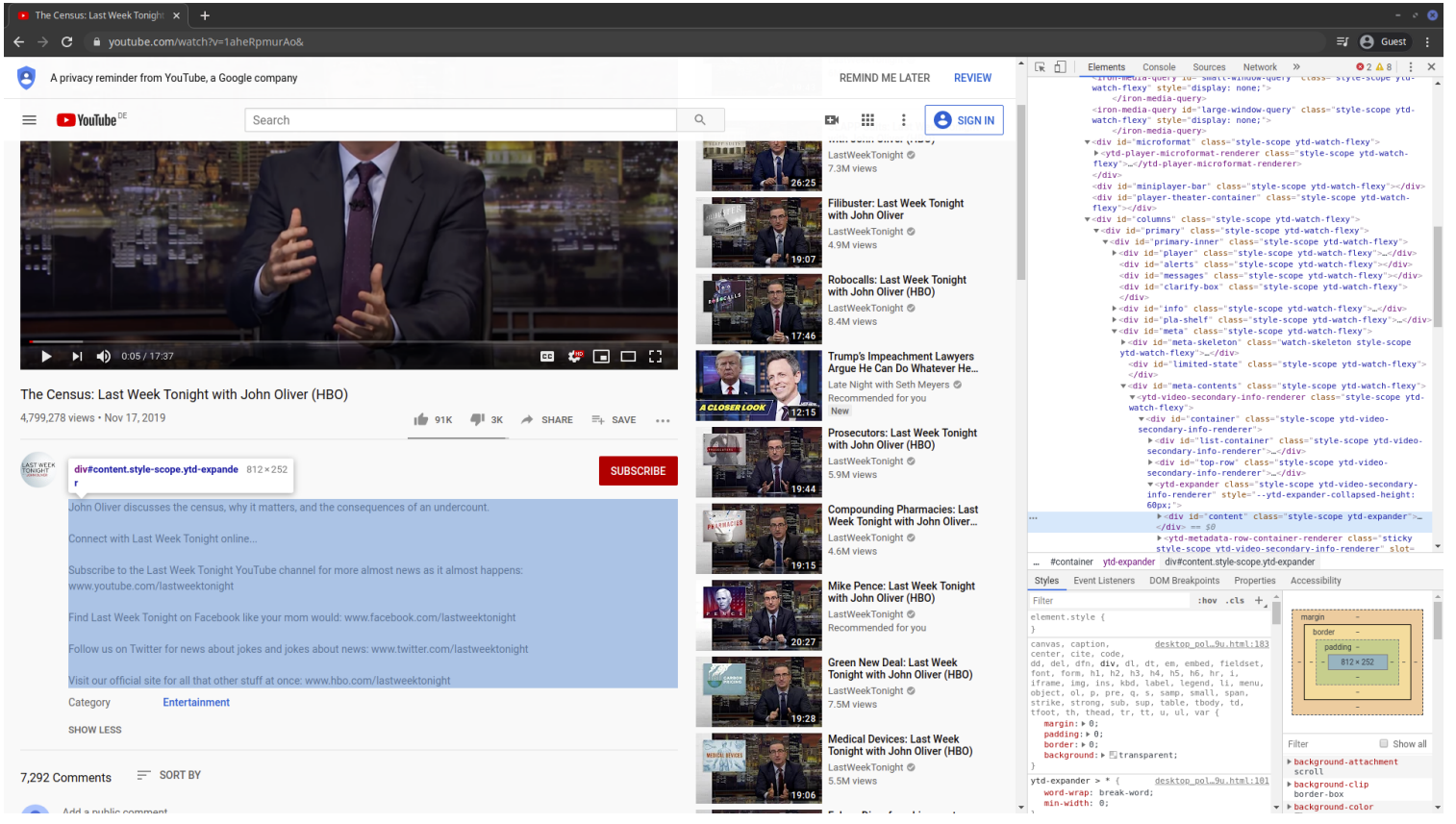
Screenscraping

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



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

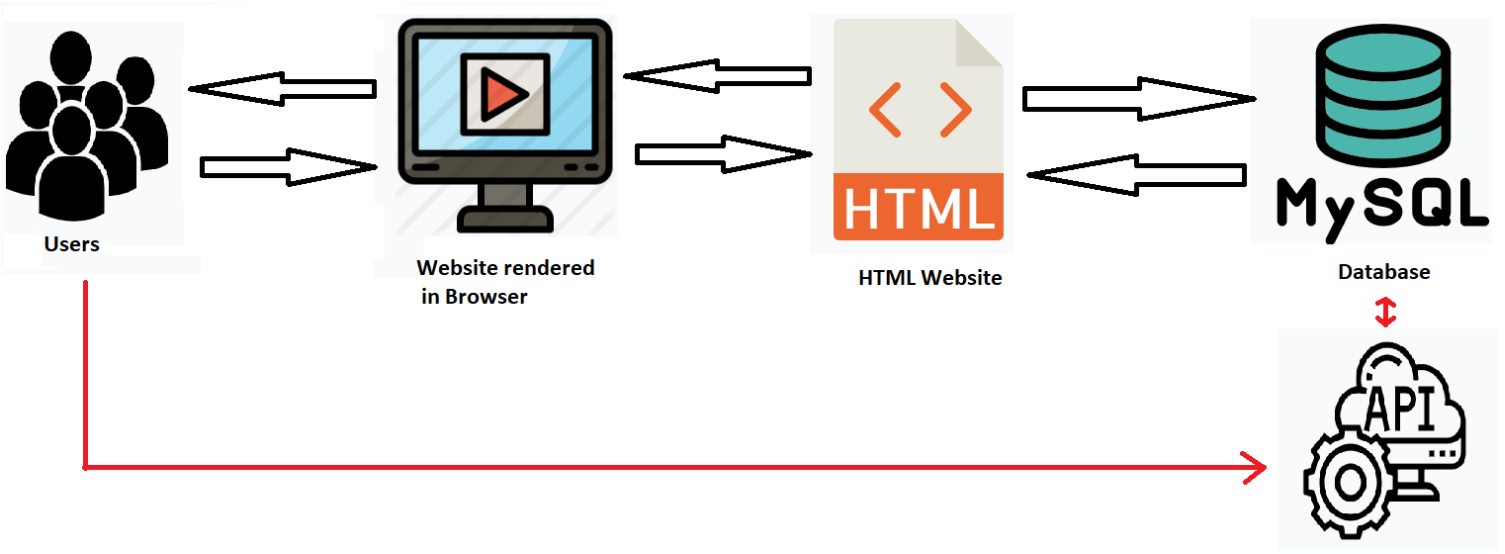
Screenscraping

- 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're getting what you want
  - If the website changes anything in their styling, your scripts won't work anymore
  - Legalality depends on country

API-Harvesting

API-Harvesting

- An Application Programming Interface) is:
  - a system build for developers
  - directly communicating with the database
  - Voluntary service of the website
  - dictating 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 do
  - 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 sure that what you do is legal and (probably) in line with Terms of Service
- 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)
    - 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 their 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 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 our created Google account to use the API
- What we get back is a JSON formatted response with the formats and information we requested in the 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) at storing large quantities of data by not declaring missing data
- Represented in R as a list of lists, 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

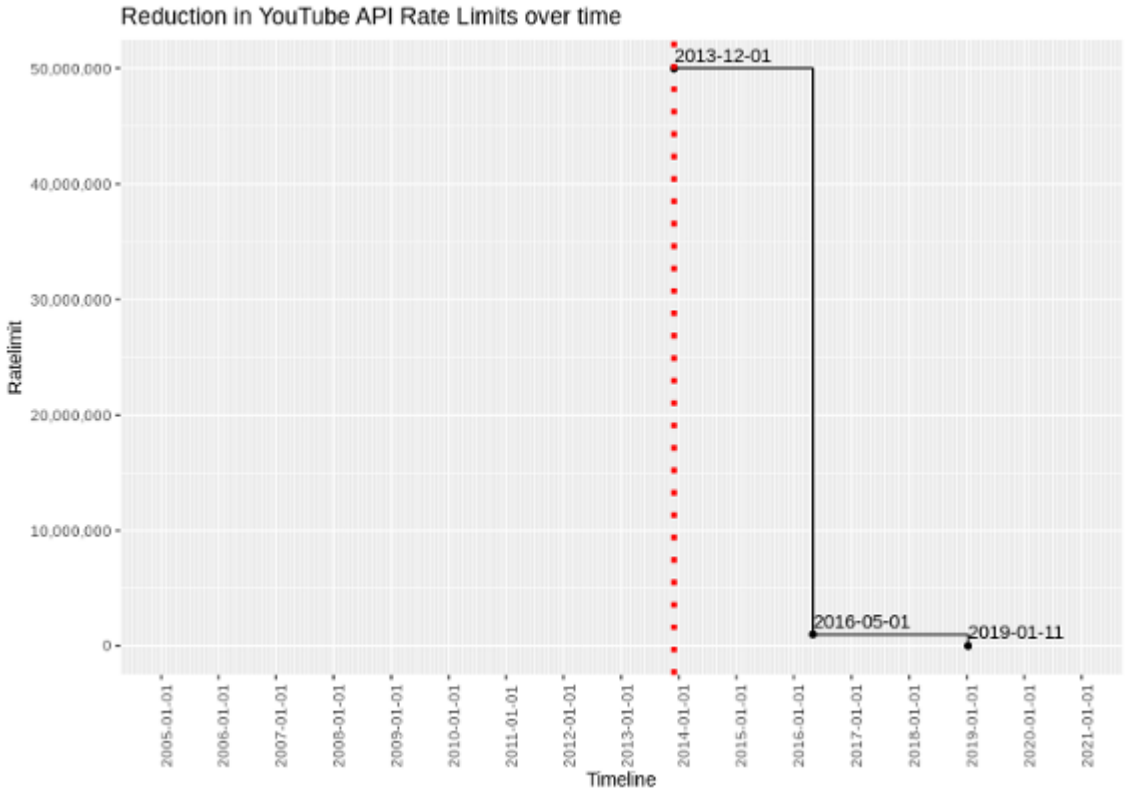
- All possible parameters are listed here
- Keep in mind that some information is only visible to owners of a channel or author of a video
- Keep in mind that not all information is necessarily available for all videos (e.g. live videos)

Using it from R

- We can simplyfiy the process of interacting with the YouTube API by using a dedicated R package
- The package handles the authentification with our credentials and translates R commands into API calls
- It also simplifies the JSON response to a standard dataframe automatically
- 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



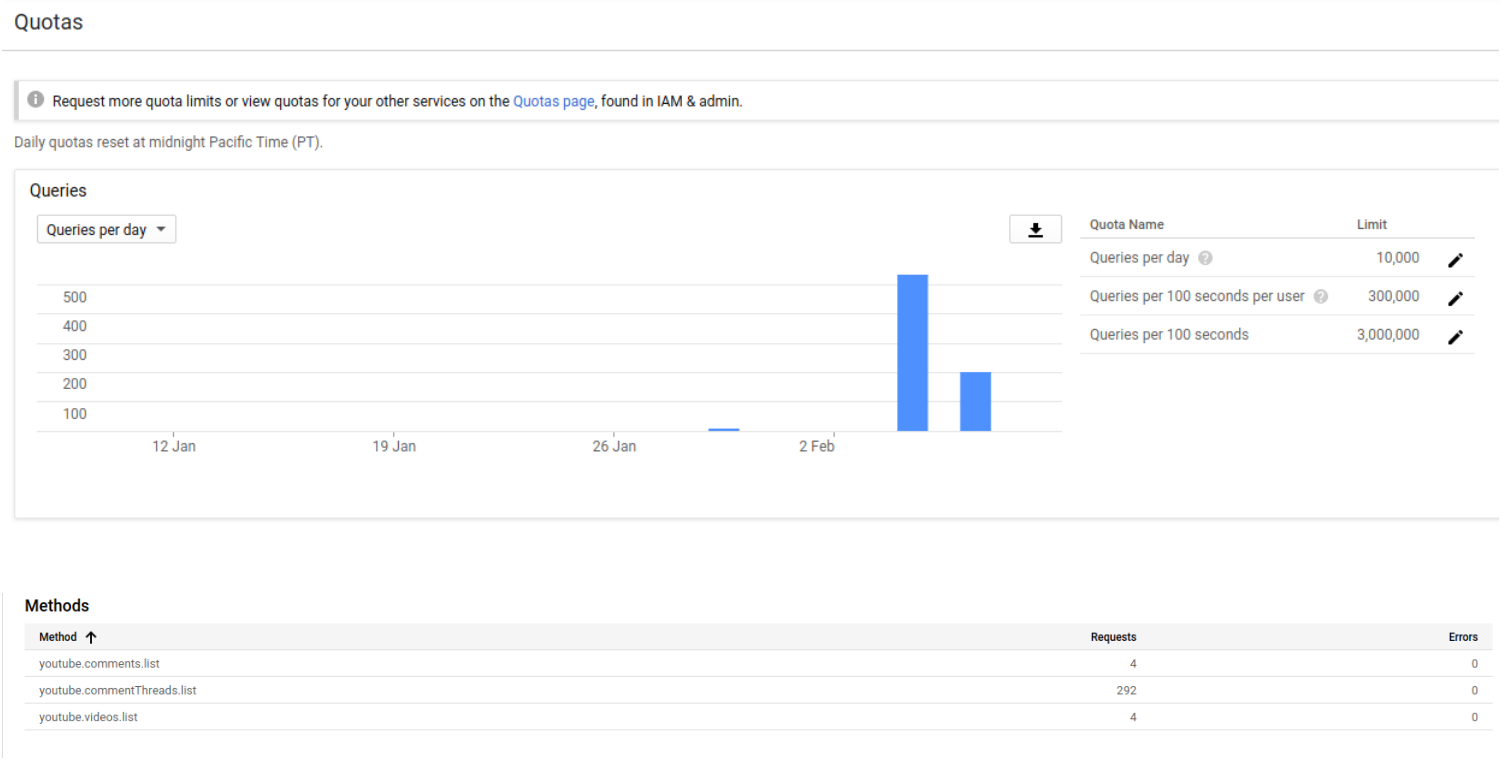
Rate Limits

- 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 (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.**

BEWARE: Sending wrong requests can fill up your daily quota

Rate Limits

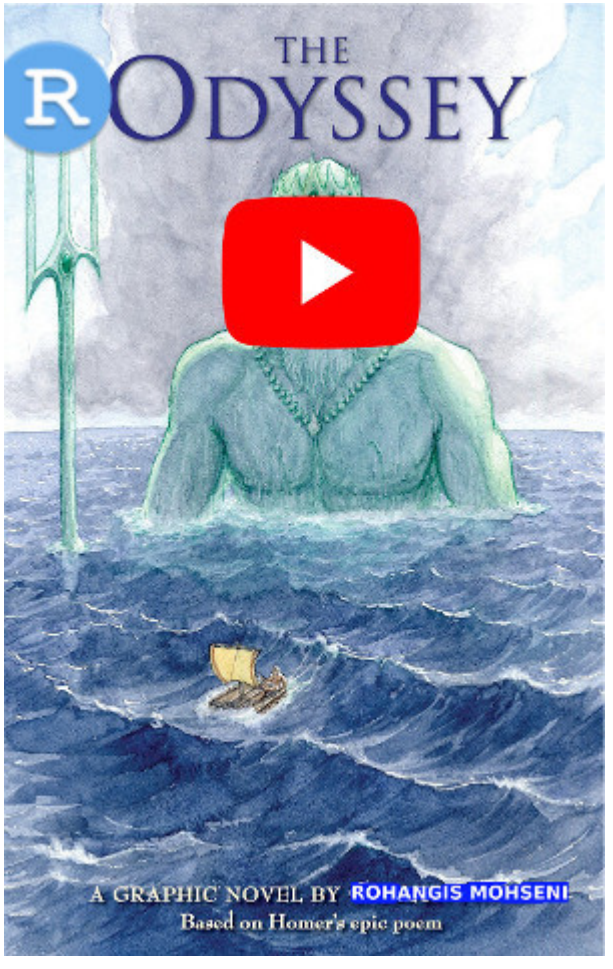
- You can check the rate limits in the *YouTube* API Documentation
- You can see how much of your quota you have already used up in the developer console



Raising the Quota Limit for YouTube 🤠

- Study planned that needs large datasets in short amounts 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

Can I Increase my Rate Limit?



Any questions?

Exercise time    

Solutions