COMP4449-Capstone Final Project

Evan Hollier

Situation

Overlap of audience between YouTube channels

Task

In order to build a database of YouTube channels' audiences,
used YouTube Data API to query comment threads.
Then, similarity score between channels can be measured and analyzed.

Action

First, figured out how to get usernames for a given video ID
Then, how to get a list of video IDs for a given channel name
Ran for a single channel, and realized:

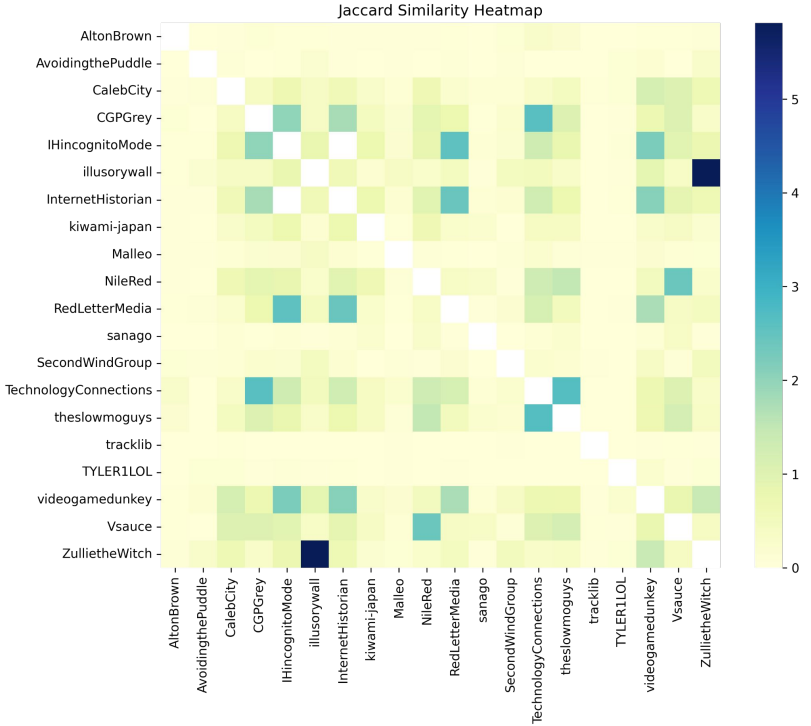
- Non UTF-8 characters could not be used
- I had to put limits in place to stay under the daily API quota

After handling those issues, I set it up to read from a .txt file of channel names and let it go through them over the course of a several days' quotas.
This left me with a folder of CSVs for 47 channels, to perform analysis on.

I started with just measuring directional % overlap but found this was misleading so I switched to using Jaccard Similarity, which is a normalized metric.
Using Jaccard Similarity, I created heatmaps for different groups of channels within my dataset

Results, pt. 1

Group 1 - random channels I subscribe to, ranging from niche interests to large audiences with mass appeal



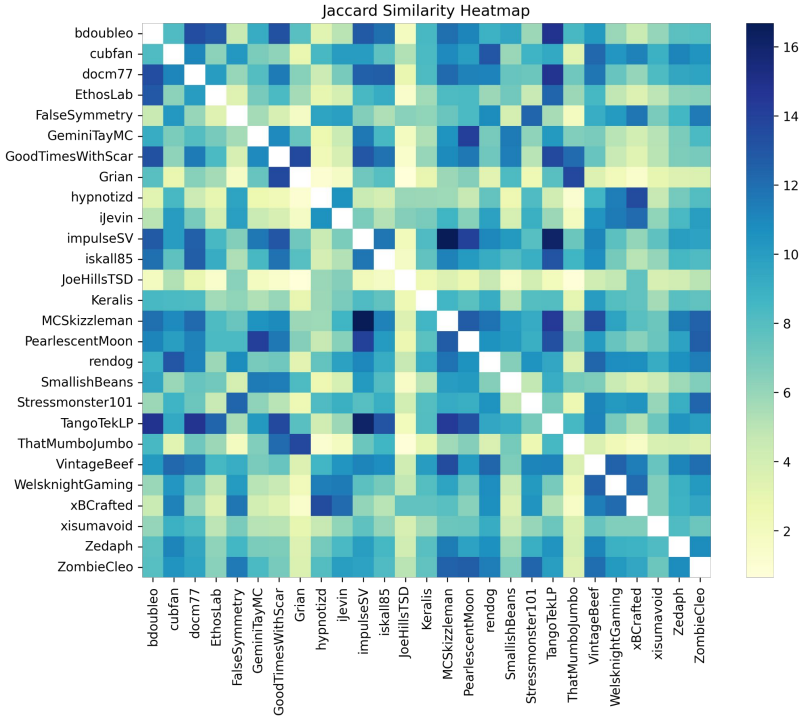
1 outlier - InternetHistorian and his 2nd channel had high overlap. Since this was to be expected, I removed this score to allow more visual variation between other channels

I would consider all of the high overlap scores as successful pairing for generating recommendations.
i.e. If you like one, you'd probably like the other.

What was more elucidating to me is the channels how varied the scores for what I consider a recommendation pair are.
More niche interests result in stronger connections.

Results, pt. 2

Group 2 - community of people that appear in each other's videos

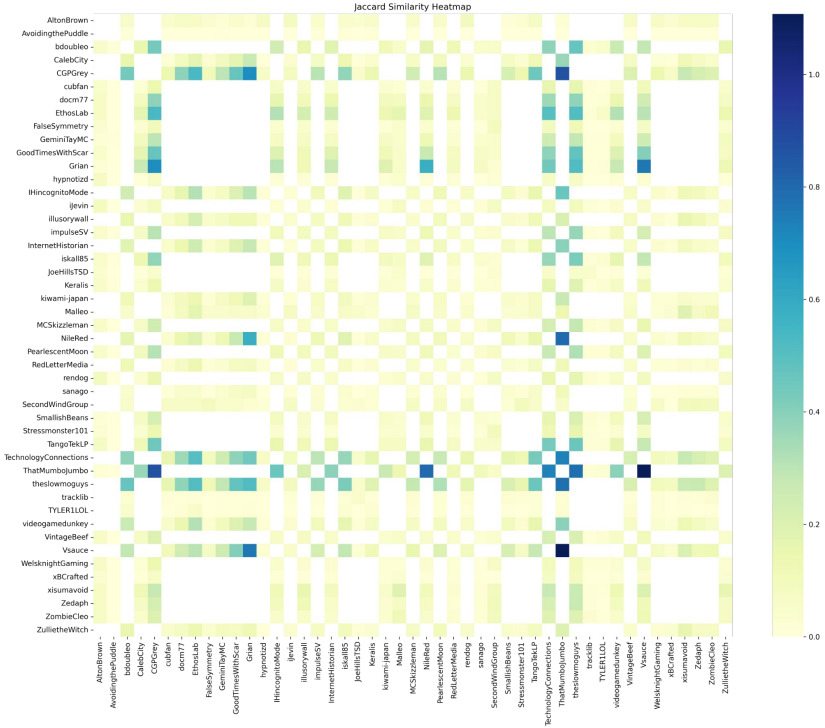


As expected, much higher range of scores compared to Group 1

I only watch a handful of these channels, so I didn't really know recommendation pairings going in.
Did whatever spot checks I could, but nothing was really that surprising.

Results, pt. 3

Combined Group 1 and Group 2, minus the scores within own group



Can see which channels "leaked" out of Group 2 into mass popularity,
as well as which popular channels are prevalent within Group 2

Data Acquisition

console.cloud.google.com
Create a project to get an API key

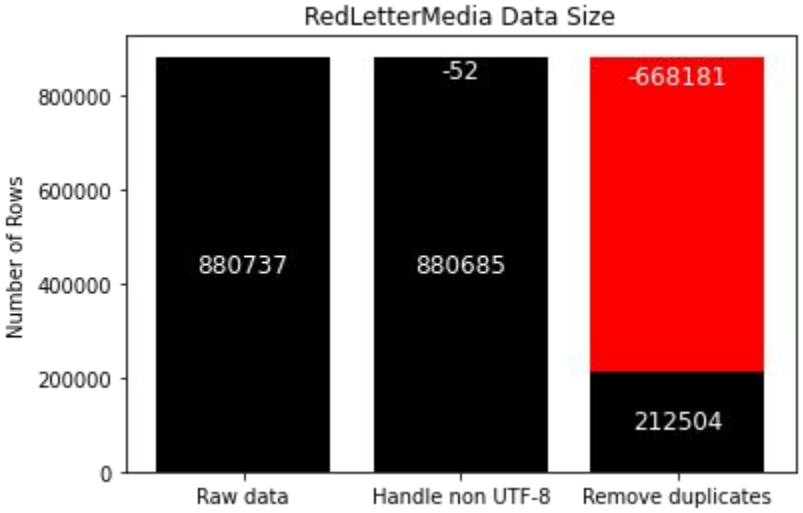
YouTube Data API v3
10,000 Queries per day = about a million usernames

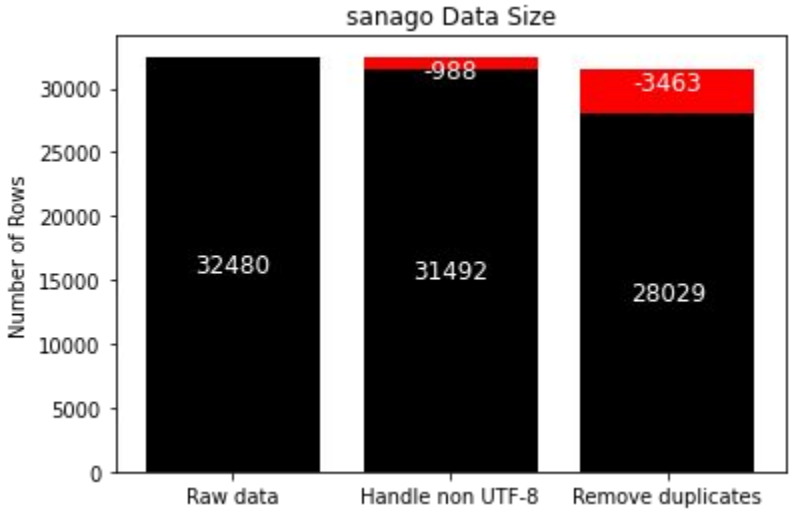
Query channel name → channel ID
Query channel ID → list of video IDs
video ID → comments
extract author name

Data Prep

Row removal for:

- Non UTF-8 characters
- Duplicates





Future Opportunities

Only 10,000 queries allowed per day - gets reached pretty quickly
Need to run over a long period of time to amass a respectable database

Bot detection: raise a flag for usernames leaving too many comments across unrelated channels
- Testing on my own dataset revealed 17 usernames that appeared in 8 of the CSVs. Within reason

Channel recommendation: input a list of channels and get back some channels that have high overlap
- Difficult to tune - even extremely related channels only had 16% Jaccard similarity
Finding channels outside of a the same community even harder - 2.7% peak

Appendix

Jaccard Similarity is Intersection over Union:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}.$$

$0 \leq J(A, B) \leq 1$
$J(A, B) = J(B, A)$

$|A \cap B|$ = overlap / $|A \cup B|$ = number of elements total, across both sets

1 CommentThread query returns a "page" of 100 top-level comments, and any replies to them.

Due to a having a limited quota for queries, I restricted:
- videos associated with a channel to recent 25
- CommentThread pages to 50 (most videos unaffected)

Store all usernames for a channel as a CSV (one column, no header).

Appendix, cont.

Important note when using an API: don't hardcode your key
Anyone with your key can use your quota

API_KEY = os.getenv('YOUTUBE_API_KEY')
.env file with "YOUTUBE_API_KEY=..."
add .env to .gitignore
Still not impossible to find, but not as trivial as looking through commit history

Also good to restrict use of that key through the portal for whoever's API you're using (Google in my case).
If I didn't this key could be used for any Google API.