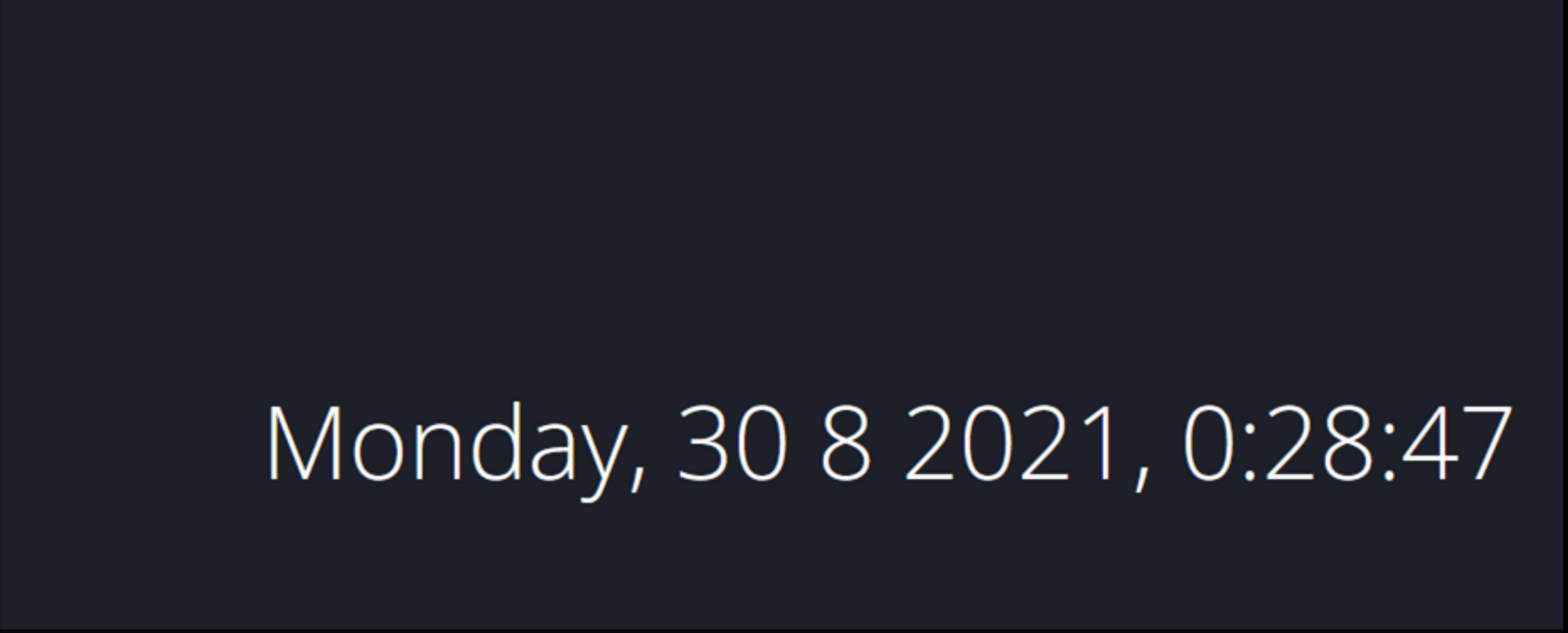


out0fCONTEXT

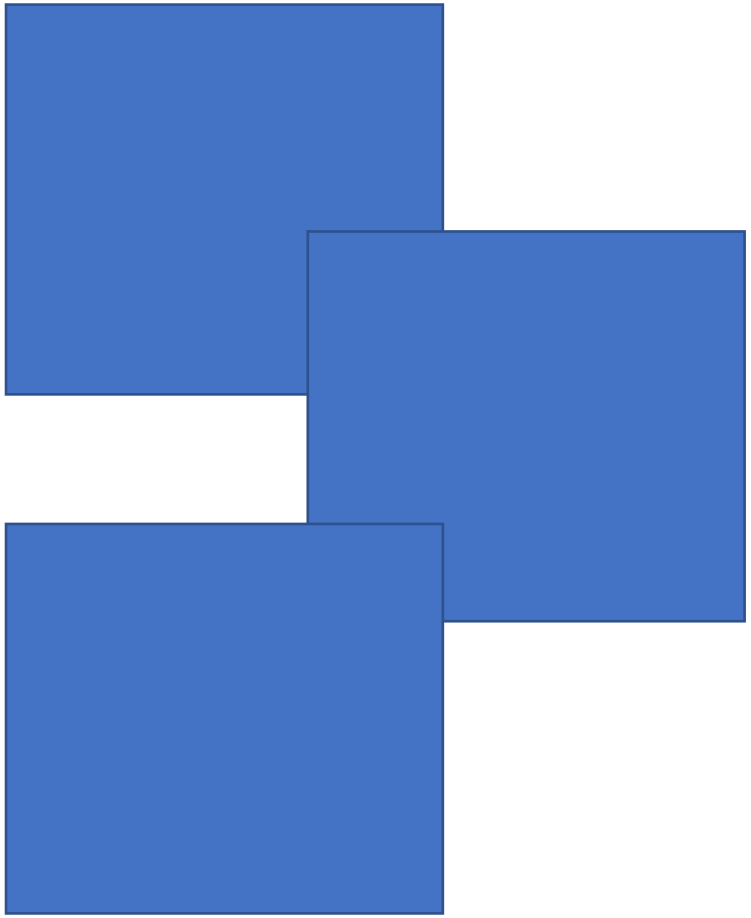
A slightly problematic
YouTube compilation

Background

The ways people consume online content varies from person to person. Some people actively engage with online content, commenting, posting, liking etc, while others just sit back and watch. One of my great fears has been that people (companies, shoulder-surfers, advertisers) would judge me based on the content I consume online.

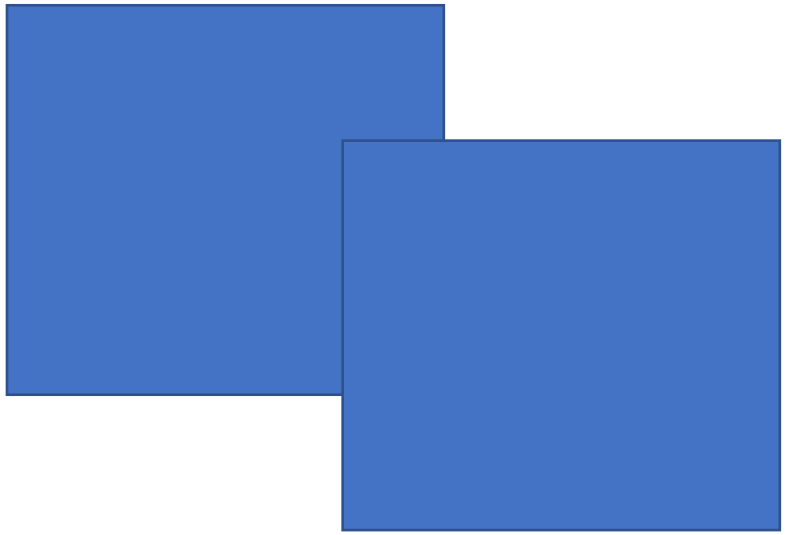
The content a user consumed is a popular tool for companies to figure out what a user is interested in, what their affiliations are etc, because this is an easily generated piece of data about a user.

The content one consumes does not necessarily match up with one's opinions and attitudes.How would anyone know, how I am interacting with the content if I don’t interact with it online by liking, commenting etc. With only the watch history, the content is put outOfCONTEXT, judging a user online (for advertisement purposes or more sinister activities), by ones consumed content will result in an incomplete and potentially problematic picture of the user.



Concept

To communicate my discomfort about information regarding decisions being made another entity, based on my consumed content, I display my consumed content, out of the context of the videos themselves, and put it in the context of how others engage with the content, which could but also could not reflect the way I engage with that content.



Summary

I requested my YouTube watch history from Google, downloaded small 10 second clips from each video, then on a separate run through scraped the comments from the YouTube-videos and ran sentiment analysis on the video. I visualized/sonified the data, playing audio samples at random intervals in chronological order, displaying the date when I watched the video. The polarity of the sentiment (positive vs negative) influenced how much I distorted the audio clips and dictated the background color.

Implementation 1: Data Preparation

Scraping the history

Google presented the YouTube history as a human readable html file.

I used python, pandas, and beautiful soup to extract the html data and create a machine-readable csv file containing the video urls, the title and the date. See getdata.py

Scraping the audio

I downloaded small clips of the video (to prevent having to download the entire video) using youtube-dl and FFMPEG. See getaudio.py

Analyzing and Scraping Comments

Scraping the comments from the YouTube videos is slightly more complicated scraping static pages, since the content is dynamically loaded. I used selenium and chromedriver to scrape the comments. [1]

For the sentiment analysis I used a plug-and-play python library called TextBlob, which is used for text and NLP-processing . I used TextBlob's built in sentiment analysis tool for polarity (whether a comment is negative or positive)

Implementation 2: Presentation

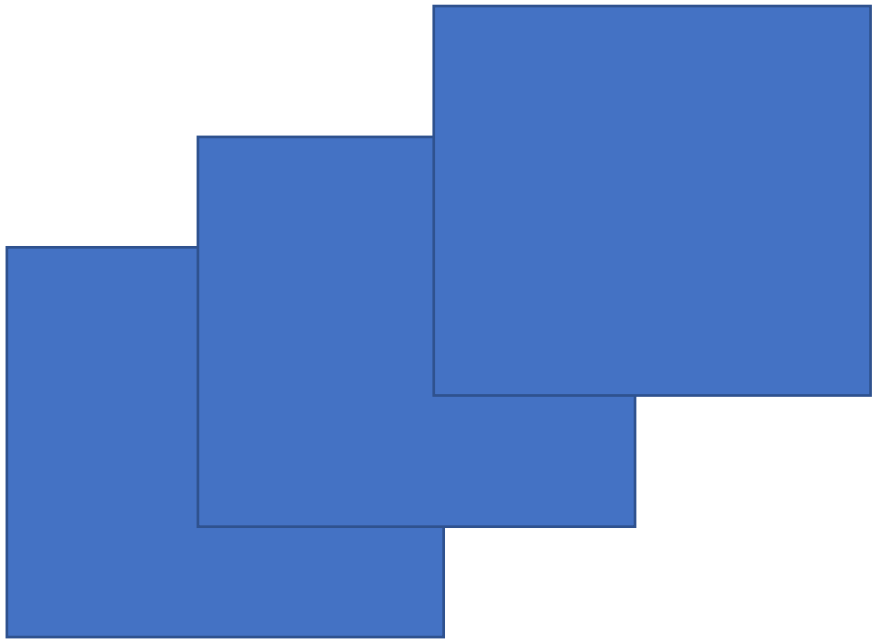
I choose to present the data using a website programmed mostly with java-script hosted on a simple http-server.

Audio

I used the Tone-js library to handle my audio playing. At randomized intervals I would trigger samples from the downloaded audio-files, with randomized fade-in and fade-out times. Depending on the sentiment of the corresponding comments, I would edit the audio. For strongly positive polarity, I would distort the audio by a varying amount depending on the polarity value. For negative and neutral polarity, I would add a Feedback/Delay effect (Echo), also varying in strength by the polarity value.

Visuals and Setup

The visuals were handled by the p5.js library, which gives me easy programmatical access to the visual website, without having to manipulate DOM elements or write CSS.



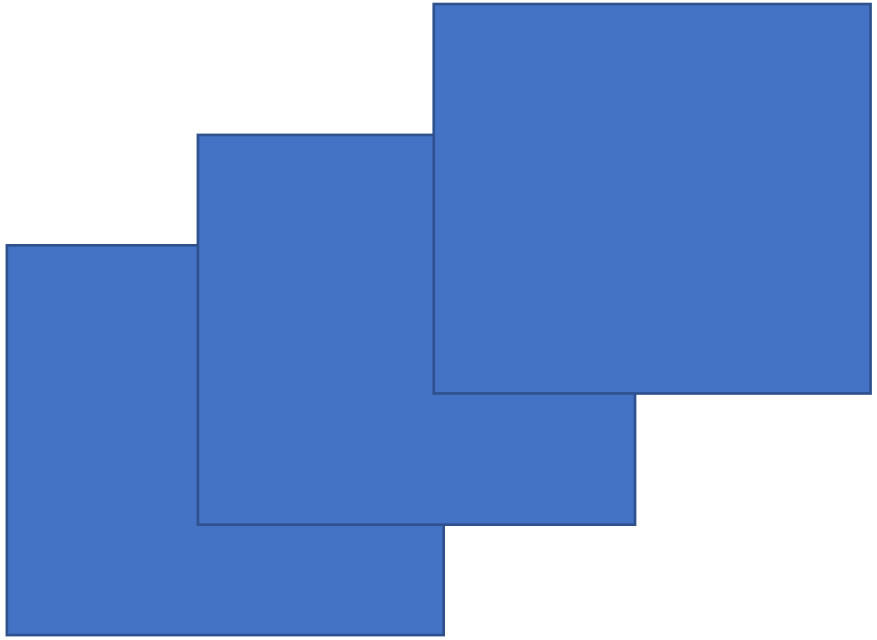
Notes:

The submission file is cleaned up. It contains a subset of my YouTube history (5 videos, that I'm okay other people knowing I watch), just to provide a working out of the box demo. You will need to download the chrome driver though and the other dependencies to run the analysis.

Specifically, you will need:

Silenium, pandas, chromedriver, chrome, FFMPEG and youtube-dl downloaded to run the analysis. The chromedriver executable is to be placed in the root folder of this repository.

For the website, you will need jQuery and tone.js (if you want to use offline sources, online sources are provided in the index.html file, p5.js (place the library in the js folder) and http-server



Implementation 3: Potential Next Steps

Streamline the data-request aspect and to allow for a more flexible experience.

Randomize from where in a video excerpts are taken

More informative and interesting visuals (maybe display parts of the comments and video descriptions)

Resources and Tutorials

https://textblob.readthedocs.io/en/dev/

https://medium.com/analytics-vidhya/how-to-scrap-youtube-comments-cf6348ef9e09

https://towardsdatascience.com/how-to-scrape-youtube-comments-with-python-61ff197115d4

https://tonejs.github.io/

https://pandas.pydata.org/

https://realpython.com/python-nltk-sentiment-analysis/

https://p5js.org/reference

https://github.com/ytdl-org/youtube-dl

https://www.ffmpeg.org/