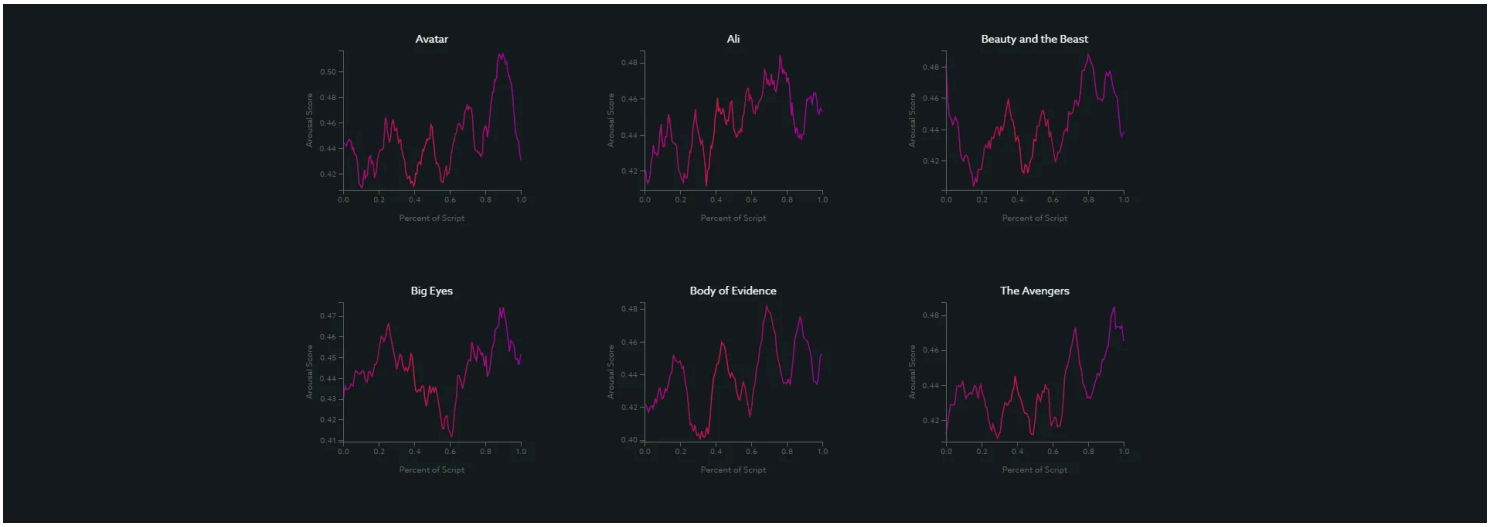



# Visualizing the Emotional Arcs of Movie Scripts Using Rule-Based Sentiment Analysis

How I used Python, D3 and Flask to create this interactive visualization

Nayomi Chibana · Follow
Published in Towards Data Science · 4 min read · Oct 17, 2019



Almost 72 years ago, acclaimed American writer Kurt Vonnegut came up with a novel method for graphing the plot lines of stories as part of his master's thesis in anthropology.

Although his work was ultimately rejected by the University of Chicago "because it was so simple and looked like too much fun," according to Vonnegut, his overlooked contribution has received some renewed attention in the last few years after a group of researchers from the University of Vermont decided to use computational methods to test his hypothesis.

What they came up with were computer-generated story arcs for nearly 2,000 books in English, categorized into one of the six main storytelling arcs outlined in Vonnegut's original thesis. These include **"Rags to Riches"** (rise), **"Riches to Rags"** (fall), **"Man in a Hole"** (fall then rise), **"Icarus"** (rise then fall), **"Cinderella"** (rise then fall then rise) and **"Oedipus"** (fall then rise then fall).

Their work deviated from Vonnegut's in that they plotted the emotional trajectory of stories, not just their plot lines. To do so, they slid 10,000-word windows through each text to score the relative happiness of hundreds of points in the story, using a lexicon of 10,000 unique words scored on a nine-point scale of happiness, resulting in the hedonometer tool for sentiment analysis.

## Using Arousal as a Proxy Measure for Action

Using a similar lexicon-based approach, I plotted the emotional arcs of more than a thousand movie scripts and used hierarchical clustering to group the most similar scripts.

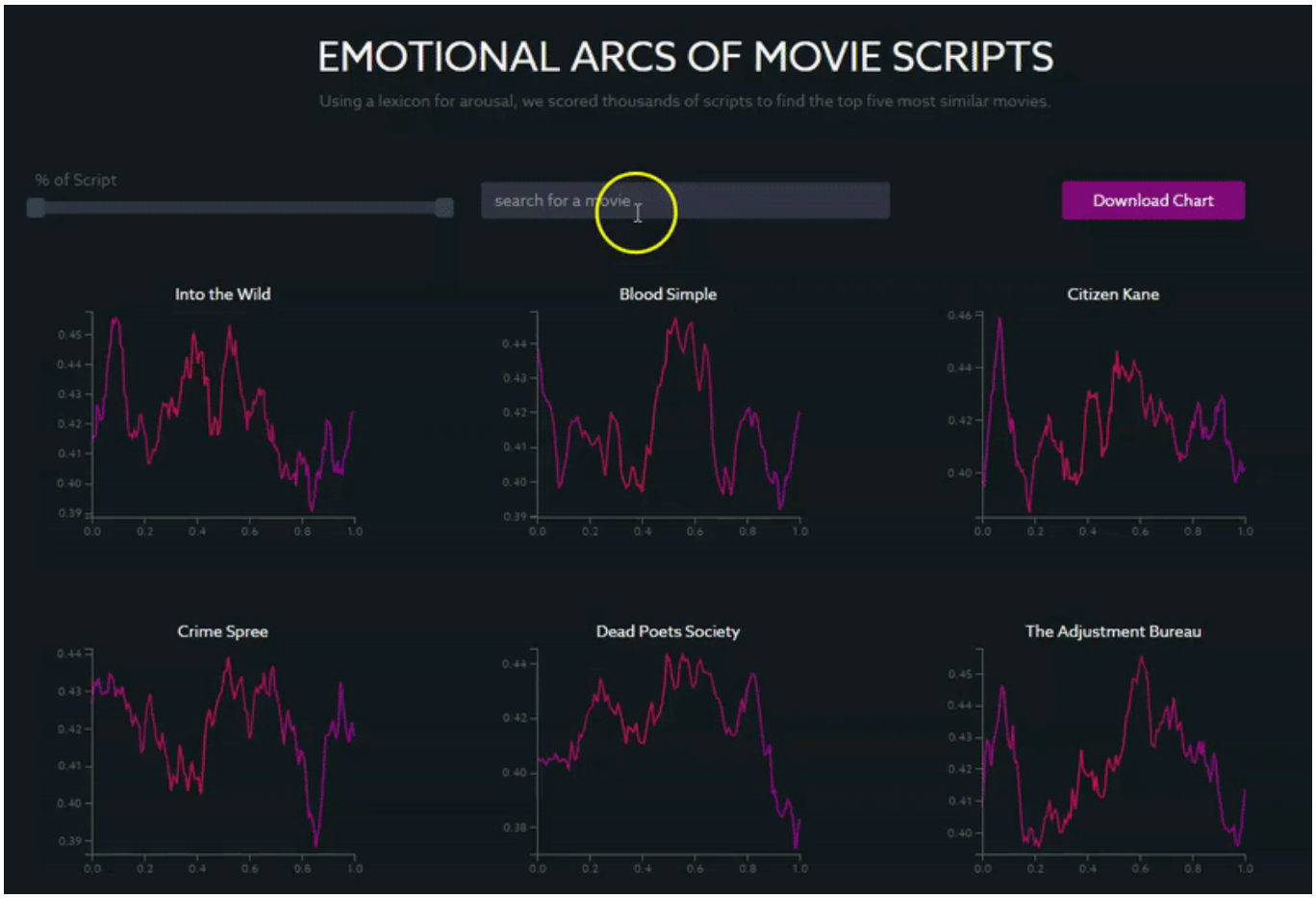
However, since previous research focuses mostly on positive and negative sentiment, rather than rising and falling action, the following method differs from UVM's approach in that it uses the NRC Valence, Arousal and Dominance Lexicon, which has scores for 20,000 English words, as a proxy measure for action or conflict in a story. Specifically, I used the arousal

dimension to score words on a spectrum from “calm” or “passive” to “excited” or “active.”

The result is an interactive visualization that can be used to search for any script already published on the Internet Movie Database to visualize its emotional story arc (for arousal) and find five of the most similar movie scripts in terms of the story’s emotional trajectory, not content.

In a search for the blockbuster “Avatar,” for example, you can see that arousal peaks towards the end of the movie, at around 90% of the length of the script, which corresponds to the point in the movie with the most tension and conflict (the final confrontation between the Na’vi and the humans to determine the fate of the planet).

Other movies with similar structures, with a clear climactic end, include “Beauty and the Beast,” “The Avengers” (2012) and “Ali.”




Here is the process I followed to arrive at these results:

## Scraping Texts and Scoring Using the VAD Lexicon

Using Beautiful Soup and a revised version of Jeremy Kunn's script, I scraped all of the movies from the Internet Movie Database, which took about 20 minutes using the following script:

```
import os
from urllib.parse import quote

from bs4 import BeautifulSoup
import requests

BASE_URL = 'http://www.imsdb.com'
SCRIPTS_DIR = 'scripts'


def clean_script(text):
    text = text.replace('Back to IMSDb', '')
    text = text.replace('''<b><!--
</b>if (window!= top)
top.location.href=location.href
<b>// -->
</b>
''', '')
    text = text.replace('''          Scanned by http://freemoviescripts.com
          Formatting by http://simplyscripts.home.att.net
''', '')
    return text.replace(r'\r', '')


def get_script(relative_link):
    tail = relative_link.split('/')[-1]
    print('fetching %s' % tail)
    script_front_url = BASE_URL + quote(relative_link)
    front_page_response = requests.get(script_front_url)
    front_soup = BeautifulSoup(front_page_response.text, "html.parser")

    try:
        script_link = front_soup.find_all('p', align="center")[0].a['href']
    except IndexError:
        print('%s has no script :(' % tail)
        return None, None

    if script_link.endswith('.html'):
        title = script_link.split('/')[-1].split(' Script')[0]
        script_url = BASE_URL + script_link
        script_soup = BeautifulSoup(requests.get(script_url).text, "html.parser")
        script_text = script_soup.find_all('td', {'class': "scrtext"})[0].get_text()
        script_text = clean_script(script_text)
        return title, script_text
    else:
```

```
        else:
            print('%s is a pdf :(' % tail)
            return None, None


if __name__ == "__main__":
    response = requests.get('http://www.imsdb.com/all%20scripts/')
    html = response.text

    soup = BeautifulSoup(html, "html.parser")
    paragraphs = soup.find_all('p')

    for p in paragraphs:
        relative_link = p.a['href']
        title, script = get_script(relative_link)
        if not script:
            continue

        with open(os.path.join(SCRIPTS_DIR, title.strip('.html') + '.txt'), 'w', encodir
            outfile.write(script)
```

download_all_scripts.py hosted with ❤️ by GitHub view raw

Then I created a new dictionary using the NRC VAD lexicon:

```
dict = pd.read_csv('./NRC-VAD-Lexicon-Aug2018Release/NRC-VAD-Lexicon-Aug2018Release/NRC-V
dict['Ranking'] = np.arange(1, len(dict)+1)
columnsTitles = ["Word","Ranking","Arousal","Valence","Dominance"]
dict = dict.reindex(columns=columnsTitles)
dict['Arousal'] = dict['Arousal'].astype(str)
newDict = dict.set_index('Word').T.to_dict('list')
```

arousal_lexicon.py hosted with ❤️ by GitHub view raw

And I adjusted the simple labMT usage script to calculate the arousal scores of a corpus, rather than happiness, by replacing the labMT dictionary and labMT vectors with my own:

```
def process():
  windowSizes = [2000]
  words = [x.lower() for x in re.findall(r"[\w\@\#\'\&\]\*\-\/\[\=\;]+",raw_text_clean,f
  lines = raw_text_clean.split("\n")
  kwords = []
  klines = []
  for i in range(len(lines)):
    if lines[i][0:3] != "<b>":
      tmpwords = [x.lower() for x in re.findall(r"[\w\@\#\'\&\]\*\-\/\[\=\;]+",lines[i],
      kwords.extend(tmpwords)
      klines.extend([i for j in range(len(tmpwords))])

  for window in windowSizes:
    breaks = [klines[window/10*i] for i in range(int(floor(float(len(klines))/window*10)
    breaks[0] = 0
    f = open("word-vectors/"+str(window)+"/"+movie+"-breaks.csv","w")
    f.write(",".join(map(str,breaks)))
    f.close()
    chopper(kwords,labMT,labMTvector,"word-vectors/"+str(window)+"/"+movie+".csv",minSiz

    f = open("word-vectors/"+str(window)+"/"+movie+".csv","r")
    fullVec = [list(map(int,line.split(","))) for line in f]
    f.close()

    # some movies are blank
    if len(list(fullVec)) > 0:
      if len(list(fullVec[0])) > 9:
        precomputeTimeseries(fullVec,labMT,labMTvector,"timeseries/"+str(window)+"/"+mov
    else:
      print("this movie is blank:")
      print(movie.title)
      movie.exclude = True
      movie.excludeReason = "movie blank"
```

process.py hosted with ❤️ by GitHub view raw

Considering that movie scripts are shorter on average than books, I set the fixed window size at 1,000 words and slid it through each script to generate *n* arousal scores, or the number of points in the resulting time series.

## Matrix Decomposition and Hierarchical Clustering

After implementing the revised version of simple labMT, which uses singular value decomposition under the hood to find a decomposition of stories onto an orthogonal basis of emotional arcs, I used linear interpolation to create word vectors of equal dimensions and used scipy's hierarchical clustering to find and group the most similar movie scripts based on the trajectory of their emotional arcs.

```
from scipy.cluster.hierarchy import fcluster

import scipy.cluster.hierarchy as hac

Z = hac.linkage(df_interpolate.iloc[:,0:266], method='ward', metric='euclidean')

# k Number of clusters I'd like to extract
results = fcluster(Z, k, criterion='maxclust')
```

cluster.py hosted with ❤️ by GitHub view raw

Using Ward's method and Euclidean distance as the distance metric, I was able to minimize the variance between clusters of movie scripts to arrive at the most accurate segmentation of observations.

## Using Flask API to Score and D3 to Visualize Results

Finally, I created a Flask API to output arousal scores for any movie found in the Internet Movie Database and return the five most similar scripts.

Using a function to call the API and filter the data, I visualized the resulting output using D3's built-in d3.json() method for loading and parsing json data objects.

```
function updateData(){
    d3.json('https://storyplotsapp.herokuapp.com/api', {
        method:"POST",
        body: JSON.stringify({
        movie_title: ItemSelect
        }),
        headers: {"Content-type": "application/json; charset=UTF-8"}
        }).then(function(data){
            console.log(data);
        // // Prepare and clean data
        filteredData = {};
        for (var movie in data) {
            if (!data.hasOwnProperty(movie)) {
                continue;
            }
            filteredData[movie] = data[movie];
            filteredData[movie].forEach(function(d){
                d["score"] = +d["score"];
                d["percent"] = +d["percent"];
                d["cluster3"] = +d["cluster3"];
            });
            console.log(filteredData);
        }
        d3.selectAll("svg").remove();

        indexSearch = d3.keys(data).indexOf(ItemSelect);
        var indexes = [0,1,2,3,4,5]
        new_array = indexes.filter(function checkIndex(index) {
            return index !== indexSearch;
        });
        console.log(new_array);

        lineChart1 = new LineChart("#chart-area1", d3.keys(data)[indexSearch]);
        lineChart2 = new LineChart("#chart-area2", d3.keys(data)[new_array[0]]);
        lineChart3 = new LineChart("#chart-area3", d3.keys(data)[new_array[1]]);
        lineChart4 = new LineChart("#chart-area4", d3.keys(data)[new_array[2]]);
        lineChart5 = new LineChart("#chart-area5", d3.keys(data)[new_array[3]]);
        lineChart6 = new LineChart("#chart-area6", d3.keys(data)[new_array[4]]);
    })
}
```

updatedata.js hosted with ❤️ by GitHub view raw

You can play around with the final interactive visualization here and view all the notebooks and code here. Let me know what you think below!

Machine Learning | Data Visualization | Data Science | Sentiment Analysis | Storytelling

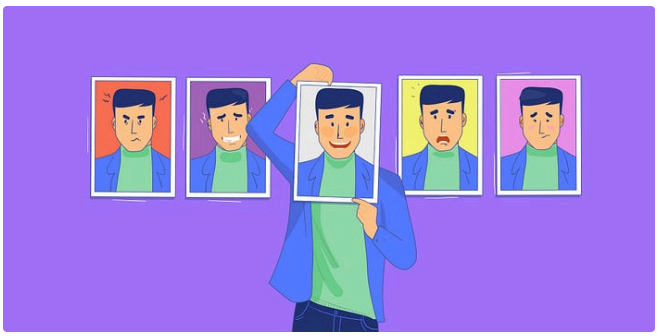
## Written by Nayomi Chibana

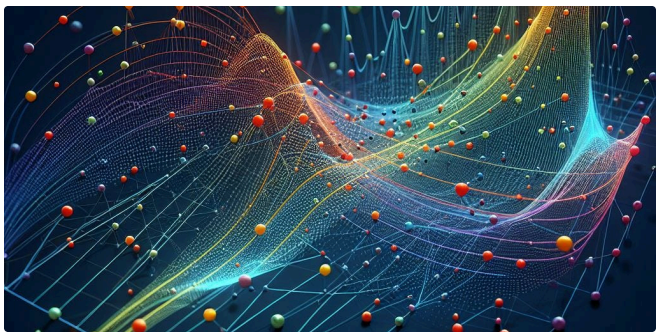
Follow

30 Followers · Writer for Towards Data Science

Data journalist interested in all things at the intersection of news, data and tech.

### More from Nayomi Chibana and Towards Data Science





Nayomi Chibana in Towards Data Science

## How Do Psychometric Test Results Vary Across Age, Race and...

Most of us have taken a personality quiz at one point or another. Whether it was for a job...

5 min read · Jun 28, 2019

154

Tim Sumner in Towards Data Science

## A New Coefficient of Correlation

What if you were told there exists a new way to measure the relationship between two...

10 min read · Mar 31, 2024

3.5K 44

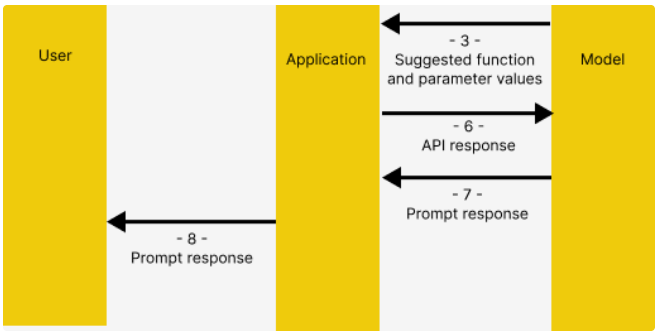
Youness Mansar in Towards Data Science

## Meet the NiceGUI: Your Soon-to-be Favorite Python UI Library

Build custom web apps easily and quickly

8 min read · Apr 16, 2024

1.2K 6




Julian Yip in Towards Data Science

## Build Autonomous AI Agents with Function Calling

Transform your chatbot into an agent that can interact with external APIs

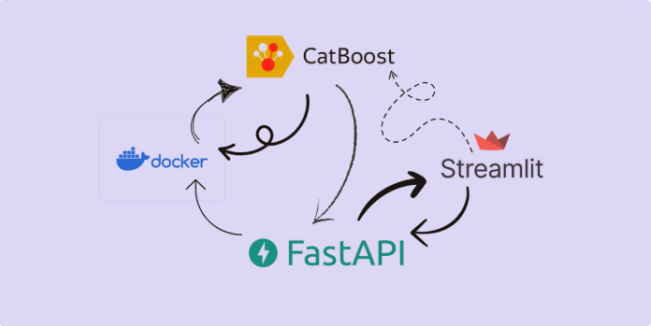
11 min read · Apr 2, 2024

1.1K 8

See all from Nayomi Chibana

See all from Towards Data Science

# Recommended from Medium



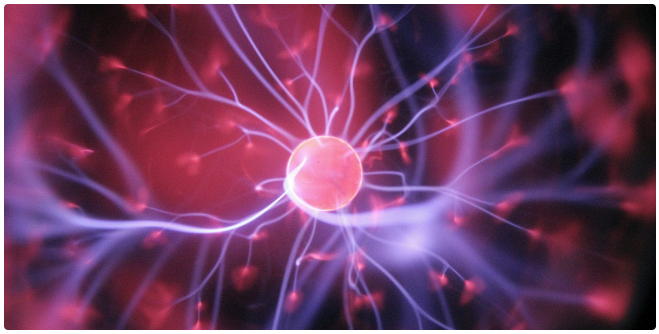

Ramazan Olmez

## End-to-End Machine Learning Project: Churn Prediction

The main objective of this article is to develop an end-to-end machine learning project. For...

18 min read · Feb 22, 2024

538 4



Rosaria Silipo in Low Code for Data Science

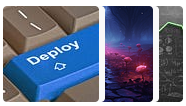
## Is Data Science dead?

In the last six months I have heard this question thousands of time: “Is data science...

6 min read · Mar 11, 2024

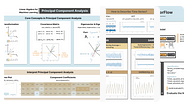
2.1K 47

## Lists




**Predictive Modeling w/ Python**

20 stories · 1161 saves



**Practical Guides to Machine Learning**

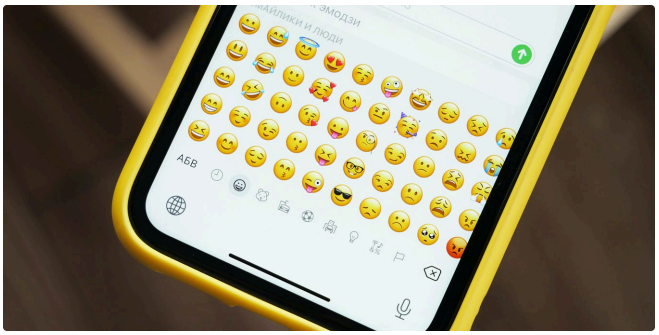
10 stories · 1402 saves


**Natural Language Processing**

1434 stories · 929 saves

**data science and AI**

40 stories · 149 saves



Bale Chen in Towards Data Science

## Emojis Aid Social Media Sentiment Analysis: Stop Cleaning Them Out!

Leverage emojis in social media sentiment analysis to improve accuracy.

14 min read · Jan 31, 2023

332 2

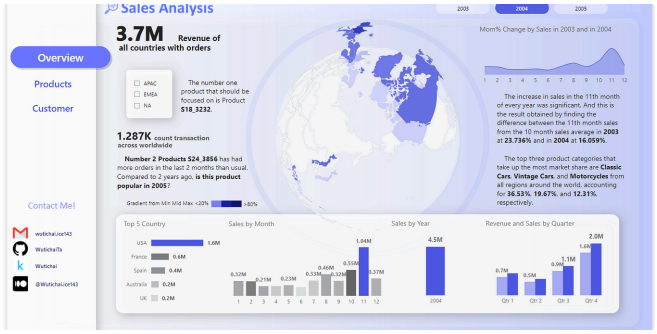

Ubaid Shah

## Twitter Sentiment Analysis — A Step by Guide

In today's digital age, social media platforms like Twitter have become a goldmine for...

6 min read · Nov 8, 2023

31



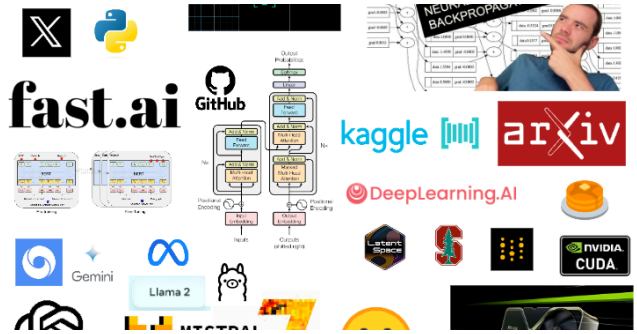

Wutichai Ta

## My First Data Analyst Project (Portfolio)—Python: Sales Analysis

This is my first data analysis project using Python. It was the beginning of my journey...

5 min read · Apr 23, 2024

275 8




Benedict Neo in bitgrit Data Science Publication

## Roadmap to Learn AI in 2024

A free curriculum for hackers and programmers to learn AI

11 min read · Mar 11, 2024

10.9K 119

See more recommendations