appendix

August 13, 2025

0.1 Supplemental Links

The main code used and the notebooks can be found here:
https://github.com/deltel/anime-recommendation-documentation

The code for the web app as well as the recommendation service can be found here:
https://gitlab.com/anime-recommendation-capstone

The deployment can be found here:
https://anime-recommendation.apps.deltel.online/login
You will need to allow insecure content in order to allow the response from the server.
Otherwise a mixed content response blocks the response.

0.2 Supplementary code snippets

[ ]:
```python
#anime-narrowing/anime-reduced.py

import pandas as pd
import requests
import time
from datetime import datetime

from exceptions.no_data_exception import NoDataException

animeDf = pd.read_csv("E:\\applied data science␣
 ↪capstone\\data\\combined\\anime_list_11_Jun.csv")

keys = ['mal_id', 'url', 'title', 'type', 'source', 'episodes', 'status',␣
 ↪'premiered', 'duration', 'rating', 'synopsis', 'broadcast', 'producers',␣
 ↪'licensors', 'studios', 'genres', 'themes', 'demographics']

def get_anime(id):
    url = "https://api.jikan.moe/v4/anime/{}".format(id)
    response = requests.get(url)

    if not response.ok:
        raise NoDataException(id, response.status_code)

    responseJson = response.json()
```

```
    return responseJson

def get_names(lst):
    return ', '.join([item["name"] for item in lst])

def extract_data(animeData, index):
    try:
        data = animeData["data"]
    except KeyError:
        id = animeData["error"].split('/')[-2]
        raise NoDataException(id, animeData["status"])

    malId = data["mal_id"]
    url = data["url"]
    title = data["title"]
    work_type = data["type"]
    source = data["source"]
    episodes = data["episodes"]
    status = data["status"]
    premiered = data["aired"]["from"]
    duration = data["duration"]
    rating = data["rating"]
    synopsis = data["synopsis"]
    broadcast = data["broadcast"]["string"]
    producers = get_names(data["producers"])
    licensors = get_names(data["licensors"])
    studios = get_names(data["studios"])
    genres = get_names(data["genres"])
    themes = get_names(data["themes"])
    demographics = get_names(data["demographics"])

    keys = ['mal_id', 'url', 'title', 'type', 'source', 'episodes', 'status',␣
 ↪'premiered', 'duration', 'rating', 'synopsis', 'broadcast', 'producers',␣
 ↪'licensors', 'studios', 'genres', 'themes', 'demographics']
    values = [malId, url, title, work_type, source, episodes, status,␣
 ↪premiered, duration, rating, synopsis, broadcast, producers, licensors,␣
 ↪studios, genres, themes, demographics]
    tmpDict = {key: value for key, value in zip(keys, values)}

    return pd.DataFrame(data=tmpDict, index=[index])


failedIds = []
index = 1
newAnimeDf = pd.read_csv("E:\\applied data science␣
 ↪capstone\\data\\combined\\anime_list_12_Jun.csv")
starttime = datetime.now()
```

```
for id in animeDf.loc[:, "anime_id"]:
    if index % 3 == 0:
        print("sleeping for 0.5 seconds")
        time.sleep(0.5)

    if index % 60 == 0:
        endtime = datetime.now()
        timeelapsed = endtime - starttime
        if timeelapsed.microseconds / 1000000 > 59:
            print("sleeping for 2 seconds")
            time.sleep(2)
        starttime = datetime.now()

    print(f"processing - {id}")

    try:
        animeData = get_anime(id)
        df = extract_data(animeData, index)
        newAnimeDf = pd.concat([newAnimeDf, df])
    except NoDataException as e:
        print(f"failed for id: {id}")
        failedIds += [(e.id, e.status_code)]

    if index % 50 == 0:
        newAnimeDf.to_csv("E:\\applied data science␣
 ↪capstone\\data\\combined\\anime_list_12_Jun.csv", index=False)
        print("saving to file")
    index += 1

failedDf = pd.DataFrame(failedIds, columns=["anime_id", "status_code"])
failedDf.to_csv("E:\\applied data science␣
 ↪capstone\\data\\combined\\missing_anime_12_Jun.csv", index=False)
newAnimeDf.to_csv("E:\\applied data science␣
 ↪capstone\\data\\combined\\anime_list_12_Jun.csv", index=False)
print("finished")
```

[ ]:
```
#anime-narrowing/reviews.py

import pandas as pd
import requests
import time
from datetime import datetime

from exceptions.no_data_exception import NoDataException

def get_anime(id, page):
```

```
    url = f"https://api.jikan.moe/v4/anime/{id}/reviews?
 ↪page={page}&preliminary=true"
    response = requests.get(url)

    if not response.ok:
        raise NoDataException(id, response.status_code)

    responseJson = response.json()
    return responseJson

def extract_review(data, id):
    malId = data["mal_id"]
    work_type = data["type"]
    date = data["date"]
    review = data["review"]

    keys = ('anime_id', 'mal_id', 'type', 'date', 'review')
    values = (id, malId, work_type, date, review)
    tmpDict = {key: value for key, value in zip(keys, values)}
    return pd.DataFrame(data=tmpDict, index=[index])

def extract_data(animeData, id):
    try:
        data = animeData["data"]
    except KeyError:
        raise NoDataException(id, animeData["status"])

    df = pd.DataFrame(columns=['anime_id', 'mal_id', 'type', 'date', 'review'])
    for review in data:
        dfReview = extract_review(review, id)
        df = pd.concat([df, dfReview])

    return df

def has_more_pages(animeData, id):
    return animeData["pagination"]["has_next_page"]

def sleep_if_necessary(index, starttime):
    if index % 3 == 0:
        print("sleeping for 0.5 seconds")
        time.sleep(0.5)

    if index % 60 == 0:
        endtime = datetime.now()
        timeelapsed = endtime - starttime
        if timeelapsed.microseconds / 1000000 > 59:
            print("sleeping for 2 seconds")
```

```
            time.sleep(2)
        starttime = datetime.now()

failedIds = []
index = 1

filelocation = "E:\\applied data science␣
 ↪capstone\\data\\combined\\anime_reviews_11_Jun.csv"
reviewsDf = pd.read_csv(filelocation)
starttime = datetime.now()

animeDf = pd.read_csv("E:\\applied data science␣
 ↪capstone\\data\\combined\\anime_list_11_Jun.csv")
animeIds = animeDf["anime_id"]
for id in animeIds:
    sleep_if_necessary(index, starttime)

    reuse_id = True
    j = 1
    while reuse_id:
        sleep_if_necessary(index, starttime)
        print(f"processing - {id}")

        try:
            animeData = get_anime(id, j)
            df = extract_data(animeData, id)
            reviewsDf = pd.concat([reviewsDf, df])
        except NoDataException as e:
            print(f"failed for id: {id}")
            failedIds += [(e.id, e.status_code)]

        reuse_id = has_more_pages(animeData, id)
        index += 1
        j += 1


    if index % 50 == 0:
        reviewsDf.to_csv(filelocation, index=False)
        print("saving to file")

    index += 1

failedDf = pd.DataFrame(failedIds, columns=["anime_id", "status_code"])
failedDf.to_csv("E:\\applied data science␣
 ↪capstone\\data\\combined\\missing_reviews_11_Jun.csv", index=False)
reviewsDf.to_csv(filelocation, index=False)
print("finished")
```