Gymnázium, Praha 6, Nad Alejí 1952

Webová stránka Neko Movies

MATURITNÍ PRÁCE

Erica Nakada
4.A

2021/2022

Prohlašuji, že jsem tuto maturitní práci vypracovala sama a že jsou uvedeny veškeré použité zdroje.

V Praze dne 13.4.2022

Obsah

1 O aplikaci

Neko Movies je webová aplikace, která umožňuje vyhledávat informace filmů a k nim doporučené filmy.
Projekt je rozdělen do dvou částí:

- Backend – doporučovací systém a Flask, který se stará o komunikaci mezi klientem a serverem
- Frontend – samotná Neko Movies webová aplikace

2 Použité knihovny

- Python 3.10.2 - https://www.python.org/
- JavaScript - https://www.javascript.com/
- jQuery 3.6.0 - https://jquery.com/
- Bootstrap - https://getbootstrap.com/
- Flask 1.1.2 - https://flask.palletsprojects.com/en/2.1.x/
- Jinja2 2.11.2 - https://jinja.palletsprojects.com/en/3.1.x/
- Werkzeug 1.0.1 - https://werkzeug.palletsprojects.com/en/2.1.x/
- MarkupSafe 1.1.1 - https://markupsafe.palletsprojects.com/en/2.1.x/
- Pandas 1.1.3 - https://pandas.pydata.org/
- NumPy 1.19.2 - https://numpy.org/
- Scikit-learn 0.23.2 - https://scikit-learn.org/stable/

Projekt je tvořen pomocí těchto nejpodstatnějších knihoven:

Flask
Tento webový framework je klíčovou součástí aplikace. Zajišťuje přenos informací mezi danými komponenty aplikace a v důsledku generuje webovou HTML stránku. V základu je vyvinut z knihovny nástrojů Werkzeug a šablonovacího systému Jinja2.

Pandas
Tento balíček umožňuje snadnou manipulaci dat a jejich analýzu. V serverové části na doporučování filmů načítá jejich data z lokálních csv souborů do datové struktury Pandas DataFrame pro jejich další zpracování.

Scikit-learn
Tato knihovna poskytuje řadu algoritmů a užitečných nástrojů strojového učení. Doporučovací system aplikace pomocí scikit-learn provádí vektorizaci textového popisu filmů, výpočet podobností jednotlivých vektorů a následné získávání nejbližších sousedů. Podrobný popis a vysvětlení použitých funkcí lze najít v tomto dokumentu v sekci 5 Vývoj doporučovacího systému.

3 Struktura aplikace

```
movie-web-app/
   static/
      scripts.js
      style.css
   templates/
      index.html
      movie.html
   main.py
   recommendation.py
   data/
      movie_data.csv
      my_links_small.csv
      my_ratings_small.csv
```

3.1 Frontend

O samotný vzhled webové aplikace a její fungování se starají složky templates s HTML soubory a static s JS a CSS soubory.

index.html
Templates neboli šablony jsou soubory obsahující statická data a proměnné v {{ }} pro data dynamická. Veškeré dynamicky vytvořené komponenty stránky (např. informace o filmu získané z databáze) jsou předávány z Flasku v main.py, který generuje stránku pomocí render_template() (viz. 3.2). Soubor index.html je domovskou stránkou, který Flask generuje jako první na adrese http://127.0.0.1.:5000. V záhlaví souboru jsou online CDN odkazy na Bootstrap CSS a script tagy dole obsahují CDN odkaz na jQuery a odkaz na vlastní script.js.

movie.html
Jinja2 navíc umožňuje dědictví šablon. Soubor movie.html dědí vzhled stránky index.html pomocí speciálního Jinja2 syntaxu {% extends "index.html" %}. Soubor index.html pak v ¡div id="movie_container» definuje blok, který může jeho dědící movie.html přepsat. Po získání všech potřebných dynamických dat pro vytvoření movie.html Flask jej generuje na adrese http://127.0.0.1.:5000/movie. Data jsou předávána ve formátu JSON.

scripts.js
Soubor scripts.js obsahuje kombinaci obyčejného Javascriptu a jQuery, jehož příkazy jsou označeny $. Provádí četná volání ajax pomocí jQuery jak do Flasku, tak do TMDB. Sbírá tím veškerou potřebnou informaci o filmech a posílá jí ve finální podobě do Flasku pro vygenerování stránky.

- csvCheck()
  - spustí se po stisknutí tláčítka vyhledávání nebo klávesy enter
  - pomocí jQuery vyprázdní pole s id #suggestions a #movie_container a načte vstup uživatele do proměnné user_input
  - pošle user_input pomocí ajax POST request do Flasku na url "/suggestions"

  - pokud je odpověď prázdná (tedy v datovém souboru nejsou názvy filmů s hledným slovem), zobrazí o tom zprávu
  - v opačném případě předá odpověď (TMDB id nalezených filmů, viz 3.2) do funkce showResults()
- showResults(movie_search_list)
  - parametr movie_search_list je pole s TMDB ID filmů
  - zavolá funkci tmdbReq s parametrem movie_search_list, která vrátí a) pole polí se 2 proměnnými: název filmu a adresa jeho plakátu(pořadí filmů odpovídá tomu v movie_search_list), b) pořadí filmu v poli movie_search_list, u kterého nezískal výše zmíněnou informaci
  - není-li b) prázdné, odstraní příslušný film z movie_search_list
  - pomocí jQuery pro každý film v movie_search_list připojí do tagu s ID #suggestions jeho kartičku (plakát s názvem filmu)
- tmdbReq(movie_search_list)
  - parametr movie_search_list je pole s TMDB ID filmů
  - pro každý film v movie_search_list provede ajax GET request do databáze TMDB (do url potřeba TMDB ID filmu a API klíč) pro informaci o filmu
  - při úspěšném volání zapíše odpověď (název filmu a adresu plakátu) do pole
  - v opačném případě do jiného pole zapíše pořadí filmu, u kterého se vyskytla chyba
  - vrátí dvourozměrné pole s názvem a plakátem každého filmu a pole s pořadím filmů s chybou
- clickCard(e)
  - spustí se, pokud uživatel klikne na kartičku
  - předá atributy id a title této kartičky do funkce getRec()
- getRec(movie_id, movie_title)
  - parametry movie_id a movie_title jsou postupně integer s TMDB ID vybraného filmu a string s názvem tohoto filmu
  - schová tag s nabízenými filmy a objeví se tag načítání s ID #loader
  - pošle movie_id pomocí ajax POST request do Flasku na url „/recommendations“
  - při úspěšném volání předá movie_id a odpověď z Flasku (TMDB ID doporučených filmů, viz 3.2) do tmdbReqMoreDetails()
  - v opačném případě zobrazí o tom zprávu
- tmdbReqMoreDetails(movie_id, movie_rec)
  - parametry movie_id a movie_rec jsou postupně integer s TMDB ID vybraného filmu a pole s TMDB ID doporučených filmů

- provede ajax GET request do databáze TMDB pro informaci o filmu
- při úspěšném volání předá předá odpověď, movie_rec a movie_id do credits()

- credits(movie_info, movie_rec, movie_id)
  - parametr movie_info má podobu JSON formátu (jsou v něm detailní informace o filmu), movie_rec je pole s TMDB ID doporučených filmů, movie_id je integer s TMDB ID vybraného filmu
  - provede ajax GET request do databáze TMDB pro informaci o všech osobách zúčastněných natočení filmu
  - při úspěšném volání vytvoří slovník s 5 nejdůležitějšími herci a režisérem a spolu s movie_info a movie_rec jej předá do toFlask()

- toFlask(movie_info, movie_rec, credits)
  - parametr movie_info má podobu JSON formátu (jsou v něm detailní informace o filmu), movie_rec je pole s TMDB ID doporučených filmů, credits je slovník s 5 herci a režisérem
  - vyvolá infoRecM(), který funguje jako tmdbReq() a opět vyřadí z movie_rec filmy, u kterých se vyskytla chyba (rec_m_info je pak vrátí dvourozměrné pole s názvem a plakátem každého doporučeného filmu)
  - upravuje pouze potřebné informace v movie_info a přidá další (movie_poster_path, director, actors, rec_movies, rec_m_info)
  - pošle movie_info pomocí ajax POST request do Flasku na url „/“
  - při úspěšném volání provede přeadresaci na http://127.0.0.1.:5000/movie

3.2 Backend

main.py
Aplikace Flask je v souboru main.py. Veškeré vysvětlení o struktuře Flask aplikace je k dispozici na oficiálních stránkách frameworku(viz 2 použité knihovny). Tento soubor je napsán v programovacím jazyku Python 3.10 a používá knihovny flask a pandas.

- index()
  - Jako první generuje index.html na adrese „/“
  - Pokud příjme POST request (od toFlask() ve scripts.js), z jeho POST dat přeformátuje do JSON pouze potřebná data (název filmu, adresa plakátu, hodnocení, rok vydání, žánr, doba trvání, popis, režisér, 5 hlavních herců, slovník movie_cards s informacemi o doporučených filmech - TMDB ID : [adresa plakátu, název filmu])
  - Vytváří objekt session s těmito daty a provádí přeadresaci na „/movie“

- movie()
  - Data z objektu session předá do generovaného šablonu movie.html

- suggestions()
  - Z POST request (od csvCheck() ve scripts.js) přijme vstup uživatele a vrátí JSON z TMDB ID filmů, v jejichž názvu se nachází hledané slovo (filmy z datového souboru movie_data.csv)
- recommendations()
  - Z POST request (od getRec() ve scripts.js) příjme TMDB ID vybraného filmu
  - Zavolá funkci get_recommendation() v recommendation.py, která vrátí TMDB ID doporučených filmů k vybranému; tato ID jsou vrácena opět v JSON formátu

recommendation.py
Hlavní backend komponent, který se stará o doporučování filmů. Detailní vysvětlení principu doporučovacích systémů lze najít v tomto dokumentu v sekci 5.2 hybridní systémy.

Recommendation.py napsaný v Python používá knihovny Pandas, NumPy a Scikit-learn (zkráceně Sklearn) pro práci s datovými soubory. Webová aplikace používá dva doporučovací systémy kvůli rozdílu v datových souborech (viz sekce 4 předzpracování dat). Za primitivnější systém odpovídá první funkce get_recommendation1(), využívající movie_data.csv. Za složitější odpovídá druhá funkce get_recommendation2(), která byla vyvinuta z první, ale navíc je doplněna o funkci k_similar() a využívá hlavně my_ratings_small.csv. Oba doporučovací systémy přijímají jako parametr TMDB ID filmu a vrací pole s TMDB ID doporučených filmů uspořádaných podle popularity. O to, který systém je vhodné použít, se stará funkce get_recommendation(), která je volána z main.py.

Poznámka: Veškerý proces od zpracování výchozích dat až po finální podobu doporučovacích systémů s doprovázejícím komentářem je k dispozici v repozitaři aplikace ve složce jupyter notebooks.

4 Předzpracování dat

Webová aplikace používá pouze seznam filmů vyskytujících se v metadatech Full MovieLens Dataset, stažených z online zdroje dat Kaggle (odkaz zde: https://www.kaggle.com/datasets/rounakbanik/the-movies-dataset). Z tohoto datasetu konkrétně byly použity následující soubory:

- movies_metadata.csv: Hlavní metadata filmů, obsahuje 24 informací 45466 záznamů filmů. Použito bylo ale pouze 6 hlavních infromací: title, id, overview, genres, vote_average a vote_count (poslední dva jsou postupně průměrné hodnocení a celkový počet hodnotících TMDB uživatelů)
- credits.csv: obsahuje informace o filmovém štábu a hercích 45476 záznamů filmů
- links_small.csv: menší verze links.csv, obsahuje TMDB a IMDB ID k 9125 filmům z Full MovieLens Dataset (movieId sloupec je ID použitý ve Full MovieLens Dataset)
- ratings_small.csv: menší verze ratings.csv, obsahuje 100004 hodnocení 9066 filmů od 671 uživatelů

Pro lepší výsledky doporučovacích systémů je nutné mít kvalitní a zpracovaná data. Veškeré úpravy byly provedeny ve webové aplikaci Jupyter Notebook (soubor jupyter notebooks / 1 data_processing) za použití knihoven Pandas, NumPy a modulu ast. Všechna data se nejdříve načítají do Pandas Dataframů pro jejich snadné zpracování.

movies_metadata.csv
Spojí se movies_metadata a credits. Jsou ponechány pouze výše zmíněných 6 informací (sloupců) a sloupce cast a crew z credits. Ve sloupcích genres, cast a crew je obsažen rozsáhlý formát JSON, a proto 3 funkce get_genres(), get_actors() a get_director() jsou aplikovány na každý řádek, aby vytáhly pouze názvy žánrů, jména 3 hlavních herců a režiséra. U jmén je potřeba se zbavit mezer pro lepší funkcionalitu TF-IDF vectorizer později. Tyto 3 sloupce genres, cast a crew jsou poté spojeny do 1 sloupce comb (též pro budoucí výpočty). Je přidán další sloupec weighted_rating, podle kterého doporučovací systémy na konci seřazují filmy.

Weighted Rating je vzorec pro výpočet váženého hodnocení, který používá IMDB pro zařazení filmů do 250 nejlepších. Weighted rating je lepší než průmrné hodnocení, protože bere v úvahu počet hodnocení (např. film s průměrem hodnocení 8 od 100 lidí bude mít větší váhu než film s týmž průměrem od 10 lidí). Vzorec je následující: Kde:

$$WR = \frac{v}{v+m} * R + \frac{m}{v+m} * C$$

Obrázek 1: výpočet weighted rating

- WR = weighted rating
- R = průměrné hodnocení filmu
- v = počet hodnocení filmu
- m = minimální počet hodnocení potřebný pro zařazení do top 250 (IMDB používá momentálně 25,000, pro movies_metadata m byl použit kvantil 90
- C = průměr hodnocení vech filmů

Finální podoba movie_metadata má pouze 5 sloupců (title, id=tmdbId, overview, comb a weighted_rating) a je exportována do souboru movie_data.csv.

links_small.csv a ratings_small.csv
U links_small se vyřadí sloupec imdbId a u ratings_small sloupec timestamp. Datasety jsou exportovány do souborů postupně my_links_small.csv a my_ratings_small.csv.

Poznámka: Jelikož ve finální podobě hlavní datový soubor movie_data.csv má 45432 filmů, ale soubor my_ratings_small.csv pouze 9025, byly vytvořeny 2 doporučovací systémy; primitivnější používá movie_data.csv a složitější ještě navíc my_ratings_small.csv. Aby tedy film mohl být doporučován druhým systémem, musí se vyskytovat v datovém souboru s hodnocením filmů (právě tuto podmínku kontroluje get_recommendation()).

5 Vývoj doporučovacího systému

Doporučovací systémy se dělí hlavně na 2 kategorie:

- content-based filtering: Doporučuje filmy, které mají podobné charakteristiky. Vychází z logiky, že pokud se uživateli líbil 1 film, tak se mu bude líbit i druhý film s podobným námětem. get_recommendation1() využívá tento postup a počítá podobnost filmů na základě charakteristik jako děj filmu, žánr, herci a režisér.
- collaborative filtering: Doporučuje flmy na základě existujících hodnocení všech uživatelů. Tento postup nepotřebuje znát charakteristiky jednotlivých filmů, ale využívá podobnost preferencí lidí (tedy matici hodnocení předmět x uživatel). Dělí se dále na user-based a item-based collaborative filtering. Funkce get_recommendation2() je hybridním systémem, který využívá jak content-based filtering, tak item-based collaborative filtering*.

*Jelikož webová aplikace nemá fungovat jako rozhraní pro vkládání vlastních hodnocení, systém doporučuje na základě podobnosti filmů, nikoliv uživatelů.

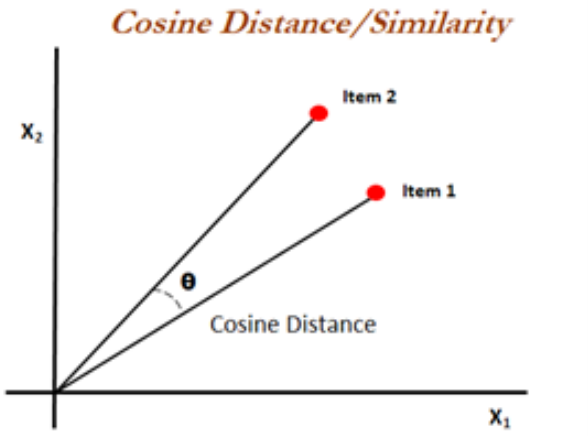
Veškeré operace byly provedeny ve webové aplikaci Jupyter Notebook za použití knihoven Pandas, NumPy, Scikit-learn a SciPy. Všechna data se nejdříve načítají do Pandas Dataframů pro jejich snadné zpracování.

5.1 Počítání podobnosti

Existuje několik metod, podle kterých se počítá podobnost předmětů, a jejich výběr závisí na mnoha faktorech, včetně podoby dat. Soubor jupyter notebooks/2 similarity metrics pojednává do podrobna výběr techniky konkrétně pro datový soubor my_ratings_small.csv. Výsledně get_recommendation1() používá cosinovou podobnost a get_recommendation2() Pearsonovu korelaci.

Cosinová podobnost
Počítá cosinus úhlu mezi 2 vektory předmětů promítnutých do multi-dimenzionálního prostoru. Čím menší úhel, tím podobné jsou dva předměty. Použitá metoda pro její výpočet: sklearn.metrics.pairwise.cosine_similarity.
Pearsonova korelace
Určuje lineární závislost mezi 2 veličinami a nabývá hodnot -1 až 1 (-1 značí nepřímou závislost, 0 žádnou a 1 přímou). V doporučovacím systému využívána k měření vzdálenosti mezi hodnoceními předmětů (odpovídá na otázku, zda 2 předměty mají tendenci být hodnoceny stejně). Použitá metoda pro její výpočet: sklearn.neighbors.NearestNeighbors s parametry metric="correlation", algorithm="brute" (NearestNeighbors realizuje strojové učení k-nejbližších sousedů bez učitele, tzn. nachází k nejpodobnějších předmětů podle příslušné metody podobnosti).

Poznámka: Před výpočtem cosinové podobnsoti textovou podobu charakteristik filmů (např. popis děje) je třeba převést na číselný vektor pomocí metod sklearn.feature_extraction.text. TfidfVectorizer a CountVectorizer (viz 2 similarity metrics).



Obrázek 2: výpočet cosinové podobnosti

5.2 Hybridní systémy

Jednotlivé doporučovací systémy (content based get_recommendation1() a collaborative filtering k_similar()) je možné kombinovat za vzniku hybridního systému. V get_recommendation2() jsou tímto způsobem odděleně vypočítány cosinové podobnosti podle popisu filmu a kombinovaných charakteristik (žánr, herce a režisér) a Pearsonova korelace podle hodnocení. Jednotlivé výsledky s jinou váhou jsou sečteny a na konec jsou doporučeny filmy s největší hodnotou podobnosti.

Soubor jupyter notebooks/3 rec_sys pojednává do podrobna sestavení jednotlivých doporučovacích systémů.

6 Zdroje

6.1 Použité knihovny

- Python 3.10.2 - https://www.python.org/
- JavaScript - https://www.javascript.com/
- jQuery 3.6.0 - https://jquery.com/
- Bootstrap - https://getbootstrap.com/
- Flask 1.1.2 - https://flask.palletsprojects.com/en/2.1.x/
- Jinja2 2.11.2 - https://jinja.palletsprojects.com/en/3.1.x/
- Werkzeug 1.0.1 - https://werkzeug.palletsprojects.com/en/2.1.x/
- MarkupSafe 1.1.1 - https://markupsafe.palletsprojects.com/en/2.1.x/
- Pandas 1.1.3 - https://pandas.pydata.org/
- NumPy 1.19.2 - https://numpy.org/

- Scikit-learn 0.23.2 - https://scikit-learn.org/stable/

6.2 Jiné

- StackOverflow - https://stackoverflow.com/
- W3Schools - https://www.w3schools.com/
- Tech With Tim - https://www.youtube.com/watch?v=mqhxxeeTbu0&list=PLzMcBGfZo4-n4vJJybUVV3Un_NFS5EOgX
- Analytics Vidhya - https://www.analyticsvidhya.com/
- Research paper by Michael Leben - http://www.lebensland.de/content/ibcf/ibcf.pdf
- TMDB - https://www.themoviedb.org/
- Kaggle - https://www.kaggle.com/datasets/rounakbanik/the-movies-dataset?select=movies_metadata.csv
- IMDB1 - https://help.imdb.com/article/imdb/track-movies-tv/ratings-faq/G67Y87TFYYP6TWAV#
- https://otik.zcu.cz/bitstream/11025/17826/1/Lochman_BP.pdf
- https://www.analyticsvidhya.com/blog/2020/11/create-your-own-movie-movie-recommendation-system/