VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INTELIGENTNÍCH SYSTÉMŮ
DEPARTMENT OF INTELLIGENT SYSTEMS

DOPORUČOVÁNÍ FILMŮ NA ZÁKLADĚ UŽIVATELSKÝCH PROFILŮ ČSFD

FILM SUGGESTIONS BASED ON CSFD USER PROFILES

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **PAVEL JANKO**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. et Ing. VÁCLAV UHLÍŘ**
SUPERVISOR

BRNO 2018

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav inteligentních systémů (UITS) Akademický rok 2018/2019

Zadání bakalářské práce

Student: **Janko Pavel**
Program: Informační technologie
Název: **Doporučování filmů na základě uživatelských profilů ČSFD**
Film Suggestions Based on CSFD User Profiles
Kategorie: Umělá inteligence

Zadání:

1. Prostudujte problematiku doporučování produktů využitím umělých neuronových sítí, zaměřte se na doporučení dle historie preferencí uživatele.
2. Shromážděte a zpracujte data z Česko-Slovenské filmové databáze (ČSFD) za účelem využití pro demonstraci řešení problému.
3. Navrhněte systém vytvářející doporučení filmů na základě profilů z ČSFD.
4. Implementujte navrženou aplikaci na vytvořené datové sadě.
5. Zhodnoťte dosažené výsledky na základě odezvy uživatelů a porovnejte míru relevance doporučení s již existujícími řešeními.

Literatura:

- F. Ricci, L. Rokach, B. Shapira: Recommender systems handbook, 2015 - Springer
- S. Huang: Introduction to Recommender System, Online: https://hackernoon.com/introduction-to-recommender-system-part-1-collaborative-filtering-singular-value-decomposition-44c9659c5e75
- P. Kordík: Machine Learning for Recommender systems, Online: https://medium.com/recombee-blog/machine-learning-for-recommender-systems-part-1-algorithms-evaluation-and-cold-start-6f696683d0ed

Pro udělení zápočtu za první semestr je požadováno:

- První dva body zadání.

Podrobné závazné pokyny pro vypracování práce viz http://www.fit.vutbr.cz/info/szz/

Vedoucí práce: **Uhlíř Václav, Ing. et Ing.**
Vedoucí ústavu: Hanáček Petr, doc. Dr. Ing.
Datum zadání: 1. listopadu 2018
Datum odevzdání: 15. května 2019
Datum schválení: 1. listopadu 2018

Abstrakt

Tato práce se zabývá problematikou využití neuronových sítí pro doporučování filmů. Je zde obecně popsán princip využití neuronových sítí u strojového učení a rovněž jsou zde shrnuty základní i pokročilé techniky pro tvorbu doporučovacích systémů. Jádrem práce je návrh, implementace a zhodnocení systému, jehož cílem je doporučování filmů na základě dat vydolovaných z uživatelských profilů ČSFD (Česko-Slovenské filmové databáze). Pro splnění tohoto účelu systém využívá explicitní faktorizační model založený na kolaborativním filtrování mezi položkami k co nejpřesnějšímu odhadu hodnocení, které by uživatel filmu po jeho shlédnutí udělil. Práce dále řeší souvislost obsáhlosti datové sady a přesnosti doporučení a demonstruje tuto přesnost analýzou zpětné vazby uživatelů.

Abstract

This thesis covers the topic of utilizing neural nets for recommending movies. The principle of using neural nets with machine learning and both the general and the advanced techniques of creating a recommender system are also covered in the thesis. The core of the thesis is the design, implementation and finally the evaluation of a system for movie recommendations based upon the data mined from the user profiles from the ČSFD (Czech-Slovak film database). In order to accomplish this goal the system utilizies an explicit factorization model based on collaborative filtering between items to predict an accurate rating that the user would presumably give to a movie after watching it. This thesis also describes the relation between dataset size and prediction accuracy and demonstrates this accuracy by analyzing user feedback.

Klíčová slova

doporučovací systémy, neuronové sítě, latentní faktorové modely, filmy, kolaborativní filtrování, doporučování filmů, faktorizace matic, spotlight, dolování dat, ČSFD

Keywords

recommender systems, neural networks, latent factor models, movies, collaborative filtering, movie recommendations, matrix factorization, spotlight, data mining, CSFD

Citace

JANKO, Pavel. *Doporučování filmů na základě uživatelských profilů ČSFD*. Brno, 2018. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. et Ing. Václav Uhlíř

Doporučování filmů na základě uživatelských profilů ČSFD

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. et Ing. Václava Uhlíře. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Pavel Janko
15. května 2019

Poděkování

Mnohokrát děkuji panu Ing. et Ing. Václavu Uhlířovi za ochotu a cenné rady, které mi poskytl při zpracovávání této bakalářské práce.

Obsah

Kapitola 1

Úvod

Lidé často spoléhají na doporučení ostatních při každodenním rozhodování o rutinních záležitostech, jako například kterou knihu si přečíst nebo na který film se podívat. Například zaměstnavatelé zase zohledňují doporučení z předchozích firem při hledání nových zaměstnanců. V životě člověk zkrátka čelí nespočet rozhodnutím a proto vznikla potřeba technologie, která dokáže toto rozhodování uživateli ulehčit a vybrat z nepřeberného množství informací, produktů a služeb pouze ty, které jsou pro uživatele relevantní.

Došlo tedy ke zrodu doporučovacích systémů. Tyto systémy omezují sadu možností, které jsou uživateli nabízeny, pouze na ty, které se vztahují k jeho preferencím nebo předchozímu chování. Cílem této práce je tvorba právě takového systému, který na základě uživatelova profilu doporučí vyhovující film. Tento profil sestává z informací, které uživatel poskytne při spuštění aplikace, kdy je vyzván k výběru několika filmů, které již zhlédnul a hodnotil.

Kapitola 2 se zabývá obecným popisem doporučovacích systémů a jejich významem. Také se zde lze dočíst o principu fungování neuronových sítí a některých jejich konkrétních variant, jako například konvolučních neuronových sítí. Dále kapitola nastiňuje způsob jejich užití u doporučovacích systémů. Rovněž jsou rozebrány jednotlivé techniky doporučování spolu s jejich klady i zápory, které do velké míry ovlivňují oblast jejich využití.

V rámci 3. kapitoly se potýkáme s problémem získání vhodné datové sady, která je pro správnou funkcionalitu systému naprosto klíčová. Je zde také popsán proces objevování znalostí, jehož hlavní součástí je dolování dat.

V kapitole 4 se lze dočíst o několika modelech využívaných při doporučování, spolu s informacemi o faktorizačním modelu založeném na mělké neuronové síti, který je využíván v rámci aplikace. Dále se v ní vyskytuje srovnání vybraných open-source knihoven určených pro konstrukci doporučovacích modelů a zdůvodnění výběru konkrétní knihovny pro aplikaci samotnou.

Implementací aplikace pro doporučování filmů se zabývá kapitola 5. V této části textu se vysvětluje proces trénování explicitního faktorizačního modelu a také jednotlivé parametry, které lze při vývoji ladit za účelem zvýšení přesnosti tohoto modelu.

V kapitole 6 je zhodnocena efektivita doporučovacího systému jak z objektivní stránky, kde je analyzována přesnost doporučování z matematického hlediska, tak z té subjektivní, kdy byli někteří uživatelé vybraní pro testování dotázání, zdali jim aplikace opravdu doporučuje relevantní filmy.

V závěru se nachází shrnutí výsledků a z nich vyplývajícího přínosu práce, nových poznatků a výčet případných vylepšení, které by mohly aplikaci dále obohatit.

Kapitola 2

Doporučovací systémy

Doporučovací systémy jsou softwarové nástroje a techniky, které poskytují návrhy položek, které by mohly s největší pravděpodobností uživatele zajímat. Tyto návrhy souvisí s činnostmi, které vyžadují, aby se uživatel určitým způsobem rozhodl. Například jakou hudbu si poslechnout, jaký článek si přečíst, který dárek zakoupit a podobně. Typicky se doporučovací systém zaměřuje pouze na jednu z těchto kategorií a odpovídajícím způsobem se tomu přizpůsobí návrh, uživatelské rozhraní i technika doporučování, kterou systém implementuje. [36]

Funkce doporučovacích systémů

- Prodej většího počtu položek, než by bylo možné bez využití doporučovacího systému, kterého je možné dosáhnout, protože doporučené položky zpravidla reflektují uživatelovy potřeby.
- Prodej méně známých položek, na které by uživatel bez doporučení nemusel nikdy narazit. Například streamovací platforma *Netflix*[1] má zájem na představení celého svého katalogu filmů a seriálů, nejen těch populárních. [18]
- Zlepšení zážitku uživatele tím, že mu systém poskytne relevantní a zajímavá doporučení, které často souvisí s přívětivějším uživatelským rozhraním.
- Udržování uživatelů, jelikož uživatelé mají tendence být loajální vůči aplikaci, která si jej pamatuje a přizpůsobuje se mu.

Zjednodušeně lze na personalizovaná doporučení nahlížet jako na seznamy hodnocení položek. Aby mohl systém zjistit toto hodnocení, musí od uživatelů získávat informace o jejich preferencích. Tento proces je buď explicitní, kdy je uživatel přímo dotázán na jeho názor na určitou položku, nebo implicitní, kdy se tyto informace odvozují od uživatelovy činnosti. Podle [36] se v dnešních pokročilých systémech můžeme také setkat s kombinací těchto přístupů.

Položky představují objekty, které doporučovací systém doporučuje. Tyto položky lze charakterizovat například jejich složitostí, hodnotou, nebo užitečností. Hodnota položky může být pozitivní, pokud je pro uživatele relevantní, naopak může být negativní, když se projeví jako nevhodná pro uživatele. Příkladem položek s nízkou úrovní složitosti mohou být například novinky, webové stránky, knihy, či filmy. Mezi složitější položky pak lze zařadit

[1] https://www.netflix.com/

fotoaparáty, mobilní telefony, počítače, dále extrémně složité položky jako pojištění, finanční investice, či pracovní nabídky. [32]

O uživatelích doporučovacího systému se shromažďuje pestré spektrum charakteristik. Tyto charakteristiky lze strukturovat různými způsoby a to, které informace modelovat se odvíjí od volby doporučovací techniky. Například u kolaborativního filtrování jsou uživatele modelováni jako jednoduchý seznam obsahující hodnocení položek tímto uživatelem. Uživatele lze rovněž popsat pomocí vzorů v jejich chování (behavior patterns) a data mohou zahrnovat vztahy mezi jednotlivými uživateli spolu s mírou důvěryhodnosti těchto vztahů [36].

Položky a uživatele spojují interakce, které v sobě obsahují důležité informace o vztahu mezi uživatelem a položkou. Asi nejpopulárnější způsob shromažďování dat o interakcích je formou hodnocení [13]. Hodnocení může mít několik podob, jako například celočíselné rozmezí mezi 0 až 5 hvězdičkami, ordinální hodnocení (silně souhlasí, neutrální, souhlasí) a nebo binární hodnocení, kdy je uživatel jednoduše dotázán, zdali se mu daná položka líbila nebo nikoliv.

2.1 Neuronové sítě

Umělé neuronové sítě lze nejadekvátněji charakterizovat jako výpočetní modely s obrovskou paralelní distribuovanou strukturou, které jsou schopny se učit a tudíž zobecňovat. Zobecněním se myslí poskytnutí smysluplných výstupů pro vstupy, které nebyly zpracovávány v průběhu učení. Tato schopnost zprostředkovává skrze neuronové sítě schopnost řešit složité a rozsáhlé úlohy, které by bez nich někdy nebylo možno vyřešit [28]. V praxi však není možné, aby neuronová síť docílila řešení samostatně. Namísto toho je zapotřebí síť integrovat do systému, kde bude vykonávat činnost vedoucí k řešení určitého podproblému, na kterou je daná síť stavěna.

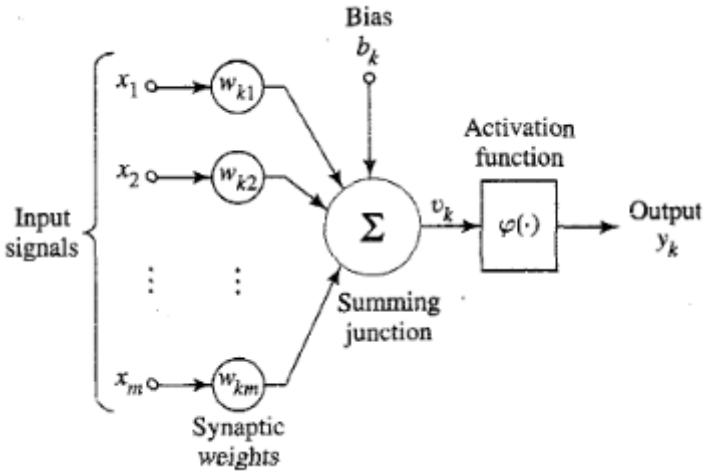
Sítě sestávají z elementárních částí (často označovaných jako buňky nebo neurony), které se skládají ze tří základních komponent. První komponentou je množina spojů, přičemž každé z těchto propojení má vlastní váhu a sílu, kde signál x_j na vstupu spoje j připojený k neuronu k je vynásoben váhovým ohodnocením w_{kj}. Druhou část představuje sčítač, který provádí součty vstupních signálu váhově ohodnocených jednotlivými spoji neuronu. Poslední základní komponenta je aktivační funkce, která omezuje amplitudu výstupu neuronu na specifikovanou konečnou hodnotu. [22]

Model neuronu znázorněný na obrázku 2.1 rovněž zahrnuje externě aplikovaný bias b_k, jehož účelem je zvýšení nebo snížení celkového vstupu aktivační funkce. Uvažujme neuron zapsaný takto:

$$Y = bias + \sum input * weight, \tag{2.1}$$

Hodnota Y může nabývat hodnot od $-\infty$ do $+\infty$. Neuron nezná meze této hodnoty, a proto neví, kdy se má aktivovat a kdy ne. Z tohoto důvodu jsou součástí neuronu aktivační funkce, které analyzují hodnotu Y a na základě toho rozhodnou, jestli by vnější spoje měly považovat neuron za aktivovaný, nebo nikoliv.

Aktivačních funkcí existuje několik druhů, v době psaní této práce mezi nejčastěji využívané patří *sigmoid* a *ReLU*. Oproti historicky užívaným aktivačním funkcím poskytují více rovnoměrně rozprostřený a kontinuální výstup. [42]



Obrázek 2.1: Nelineární model neuronu. Převzato z [22].

Sigmoid

Funkce sigmoid přetransformuje jakékoliv reálné číslo z daného rozsahu a vrátí výstup v rozsahu od 0 do 1, případně -1 do 1. Mimo oblast neuronových sítí se využívá pro binární klasifikaci u regresivních modelů a ve statistice se používá jako funkce kumulativní distribuce [34]. Příkladem funkce sigmoid je logistická funkce, zapsaná jako:

$$\varphi(v) = \frac{1}{1 + exp(-av)}, \tag{2.2}$$

kde a je parametr určující sklon funkce. Při využití této aktivační funkce nebo například hyperbolické tangenty, může při velkém počtu vrstev v neuronové síti docházet ke ztrátám gradientu, které zpomaluje a v některých případech úplně znemožňuje jakékoliv další učení sítě, protože nedochází k dostatečným úpravám váhových ohodnocení spojů [34].

ReLU

Tento problém eliminuje funkce ReLU (rectified linear unit), která umožňuje modelům založených hlubokých nebo konvolučních neuronových sítích rychlé a efektivní trénování. Její matematický zápis je velice jednoduchý:

$$f(x) = \begin{cases} x & x > 0 \\ 0 & jinak \end{cases} \tag{2.3}$$

Funkce je tedy lineární pro hodnoty větší než 0, ale na druhou stranu je nelineární, protože záporné hodnoty vždy vrací 0 [5]. Výhodou této funkce je také fakt, že je naprosto banální na implementaci. Například v knihovně Python ji lze naimplementovat pouze pomocí návratové hodnoty funkce `max(0.0, x)`.

Hluboké neuronové sítě

Jedná se o druh umělé neuronové sítě, která má více než jednu skrytou vrstvu mezi svými vstupy a výstupy. Každá skrytá vrstva využívá aktivační funkci pro mapování svého vstupu z podřazené vrstvy na skalární stav, který je poté předáván vrstvě nadřazené. [23]

Přidáním jedné nebo více skrytých vrstev do architektury sítě je možné z dat odvodit i podrobné informace a statistiky. Zdrojové uzly ve vstupní vrstvě poskytují odpovídající vstupní signály, které se předávají neuronům v druhé vrstvě, výstupy z neuronů druhé vrstvy jsou zase předávány jako vstupy třetí vrstvě a tak dále. [23]. Výstup poslední vrstvy se poté skládá z celkové odezvy sítě na vstup poskytnut první vrstvou.

Rekurentní neuronové sítě

Rekurentní neuronové sítě se od feedforward sítí liší zejména v tom, že obsahují aspoň jednu smyčku zpětné vazby (feedback loop) [22]. Taková síť se tedy může skládat například pouze z jedné vrstvy neuronů, kde každý neuron předává svůj výstup na vstup všech ostatních neuronů.

Konvoluční neuronové sítě

Konvoluční neuronové sítě jsou speciálním druhem neuronových sítí, které se používají pro zpracování dat se známou mřížkovitou topologií. Tradiční neuronové sítě využívají násobení matic mezi maticí parametrů a odlišným parametrem popisujícím interakci mezi každou vstupní a výstupní jednotkou (každá výstupní jednotka se tedy dostane do kontaktu se vstupní). [19]

Konvoluční sítě naopak zpravidla mají menší počet takových interakcí. Například při zpracování obrazu může na vstup přijít milión pixelů, detekovat však často chceme pouze malé, podstatné vlastnosti jako například okraje, které zabírají pouze desítky nebo stovky pixelů. Díky tomu lze ukládat menší počet parametrů, což snižuje jak paměťové požadavky modelu, tak jeho efektivitu. Rovněž to znamená, že pro výpočet vstupu je potřeba menšího počtu operací.

2.1.1 Způsoby učení

Primárním důvodem využívání neuronových sítí je schopnost se učit ze svého prostředí a vylepšování svého výkonu pomocí neustálých úprav. Proces učení neuronové sítě začíná stimulací svým okolím, na základě které pak podstupuje změny a upravuje své proměnlivé parametry [23]. Důsledkem těchto změn pak neuronová síť na své okolí (vstupy) reaguje novým způsobem. Pro zahrnutí schopnosti se učit do neuronových sítí existuje mnoho algoritmů učení.

Učení s učitelem

Mezi další paradigmata učení lze zařadit učení s učitelem. Učitel v tomto případě představuje sadu trénovacích příkladů, pro něž zná vstupy a odpovídající výstupy. Při předání stejných vstupů neuronové síti můžeme prostřednictvím učitele určit rozdíl mezi skutečným a sítí předpovězeným výstupem. Za účelem snížení rozdílu mezi těmito výstupy dochází iterativně k úpravám parametrů neuronové sítě a v optimálním případě se snažíme docílit, aby neuronová síť emulovala chování učitele, tedy poskytovala stejné výstupy. Nikoliv však

zapamatováním si správných výstupů v paměti, nýbrž naučení se principu, na jehož základě mapování vstupu na výstup probíhá. [22]

Učení bez učitele

V tomto případě neexistuje žádný učitel, který by dohlížel na proces učení neuronové sítě, nicméně existuje nezávislá metrika kvality reprezentace, kterou se síť musí naučit, a parametry sítě jsou tudíž optimalizovány s ohledem na tuto metriku.

2.1.2 Využití u doporučovacích systémů

Neuronové sítě se u doporučovacích systému využívají pro tvorbu vložek (embeddings), které vycházejí z informací o uživatelích a položkách. Tyto vložky jsou zpravidla vysoce dimenzionální vektory, a jejich skalární součin by měl odpovídat míře zaujatosti uživatele vůči konkrétní položce. V praxi to funguje tak, že jsou tyto vložky tvořeny z dat získaných z datové sady, což znamená, že každý uživatel a položka je reprezentována svým vlastním vektorem, který je při trénování upravován na základě uživatelových interakcí s položkami [32]. Téměř nikdy však v datové sadě nemáme informace o všech takových interakcích (například neexistuje člověk, který by hodnotil všechny filmy). Jelikož ale máme vektorovou reprezentaci uživatele a vektorovou reprezentaci položky, se kterou nepřišel do kontaktu, lze provést skalární součin těchto dvou vektorů a z výsledku lze odvodit, zdali je položka pro uživatele relevantní, či nikoliv.

2.2 Kolaborativní filtrování

Jedná se o populární doporučovací algoritmus, který zakládá své předpovědi a doporučení na hodnocení nebo chování ostatních uživatelů systému. Klíčový předpoklad pro implementaci tohoto algoritmu je možnost vybírat a shromažďovat názory ostatních uživatelů způsobem, který by umožnil smysluplnou předpověď preferencí aktivního uživatele. [37]

Hodnocení se skládá z asociace dvou prvků, a to uživatele a položky. Jeden způsob zobrazení těchto hodnocení je matice (viz tabulka 2.1), kde sloupce reprezentují filmy a řádky uživatele.

Tabulka 2.1: Matice hodnocení. Prázdné buňky znamenají, že daný uživatel film nehodnotil.

	Pán prstenů	Spiderman	Avatar	Smrtonostná past
Uživatel 1	5	2	3	
Uživatel 2		4	4	3
Uživatel 3	3	1	2	5

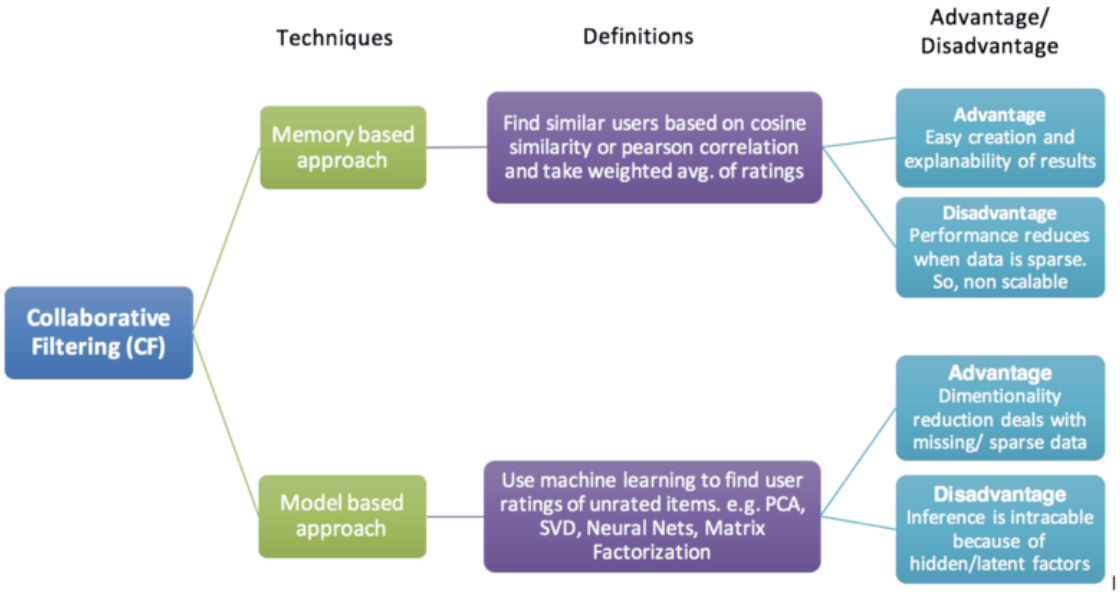
Výhodou tohoto způsobu doporučování oproti doporučováním založeném na obsahu je to, že jsou typicky do doporučení zahrnuty i subjektivní informace jako styl a kvalita položek, což v mnoha případech zvyšuje kvalitu doporučení. Rovněž doporučovací systémy založené na kolaborativním filtrováním fungují pro položky téměř jakéhokoliv typu, včetně těch, pro které je složitá automatická extrakce sémantických atributů (videa a zvukové stopy).

Asi největší nedostatek tohoto způsobu doporučování se projeví v případě, kdy se mezi obsah zařadí nová položka. Tato položka nemůže být uživateli nikdy doporučena, protože

ostatní uživatelé ještě neměli možnost položku ohodnotit. Komplikace může také nastat v případě, kdy se uživatelův vkus výrazně liší od ostatních [20]. V této situaci systém nemůže provést žádná dobrá doporučení, protože neexistují uživatelé s podobným profilem.

Přestože je kolaborativní filtrování mezi uživateli velice efektivní, tak se při narůstajícímu počtu uživatelů špatně škáluje. Za účelem rozšíření kolaborativního filtrování i na aplikace s robustnější uživatelskou základnou bylo zapotřebí vyvinout nový algoritmus. [13]

Při implementaci kolaborativního filtrování (collaborative filtering) lze vybírat z několika možných přístupů. Mezi nejpopulárnější se řadí využití nízko dimenzionálních faktorizačních modelů. Obecně se techniky kolaborativního filtrování dělí na dva proudy, vyznačené na obrázku 2.2.



Obrázek 2.2: Typy přístupů ke kolaborativnímu filtrování [20].

2.2.1 Přístup založený na paměti

Tento přístup lze rozdělit na dva hlavní druhy, a to filtrování mezi uživateli, které funguje na základě vyhledání uživatele s podobnými hodnoceními a doporučí položky, které si takový uživatel oblíbil. Druhý způsob funguje na základě doporučování položek, které se podobají těm, které uživatel v minulosti ohodnotil kladně.

Kosinová podobnost a Pearsonova korelace

Nejpoužívanějšími metodami výpočtu u tohoto druhu kolaborativního filtrování jsou kosinová podobnost a Pearsonova korelace, která je oproti kosinové podobnosti invariantní vůči přidání konstanty ke všem prvkům. Tyto metody nabývají hodnot v rozsahu od 1.0 pro uživatele, kteří se perfektně shodují, do -1.0 pro uživatele s naprosto odlišnými preferencemi [24]. Negativní hodnoty zpravidla nemají efekt na zvýšení přesnosti předpovědi při trénování (v případě využití faktorizačních modelů) a proto se často zanedbávají.

Nechť $u_{i,k}$ označuje podobnost mezi uživatelem i a uživatelem k. Dále $v_{i,j}$ označuje hodnocení, které uživatel i udělil položce j, přičemž $v_{i,j} = ?$ v případě, že uživatel položku nehodnotil. Poté může být Pearsonova korelace vyjádřena takto (převzato z [24]):

$$u_{i,k} = \frac{\sum_j (v_{i,j} - v_i)(v_{k,j} - v_k)}{\sqrt{\sum_j (v_{i,j} - v_i)^2 \sum_j (v_{k,j} - v_k)^2}} \tag{2.4}$$

A kosinová podobnost pak takto:

$$\cos(u_i, u_j) = \frac{\sum_{k=1}^{m} v_{i,k} v_{j,k}}{\sqrt{\sum_{k=1}^{m} v_{i,k}^2 \sum_{k=1}^{m} v_{j,k}^2}} \tag{2.5}$$

Nyní lze provést předpověď uživatelova názoru na neohodnocené položky pomocí následující rovnice:

$$v_{i,j}^* = K \sum_{v_{k,j} \neq ?} u_{j,k} v_{j,k} \tag{2.6}$$

K-nejbližších sousedů

Shlukování K-means se využívá pro iterativní rozdělení nepopsaných dat do jedné ze skupin reprezentovaných proměnnou K podle podobnosti v zadaných rysech. Výstupem toho algoritmu jsou také centroidy shluků, které lze využít pro popis nových dat. [39]

Algoritmus začíná s počátečními odhady pro centroidy, které lze buď náhodně vygenerovat nebo vybrat z datové sady. Následně iteruje mezi dvěma kroky. V prvním kroku dochází k přiřazení datových bodů k nejbližšímu centroidu na základě kvadratické Euklidovské vzdálenosti.

Nechť je c_i kolekcí centroidů v množině C a funkce $dist()$ vrací Euklidovskou vzdálenost mezi dvěma body. Poté je každý datový bod x přiřazen ke shluku podle funkce:

$$\underset{c_i \in C}{\operatorname{argmin}}\, dist(c_i, x)^2 \tag{2.7}$$

V dalším kroku se centroidy přepočítají podle průměru všech datových bodů přidělených k určitému shluku, formálně takto (převzato z [39]):

$$c_i = \frac{1}{|\ S_i\ |} \sum_{x_i \in S_i} x_i \tag{2.8}$$

kde S_i je soubor přiřazení datových bodů ke každému i-tému centroidu shluku.

Algoritmus mezi těmito kroky iteruje do doby, než jsou splněny podmínky pro ukončení, například lze stanovat maximální počet iterací. Je zaručeno, že tento algoritmus bude konvergovat k výsledku, který však může být lokální optimum, což nemusí nutně znamenat, že se jedná o nejlepší možný výsledek. [39]

Slope One

Slope One je další z doporučovacích algoritmů. Využívá jednoduchou formuli, která pouze odečítá průměrné hodnocení dvou položek a tento výsledný rozdíl si zapamatuje [31]. Když

pak uživatel hodnotí nějaké položky, lze tento rozdíl využít k předpovědi hodnocení dalších položek.

Existují dvě verze tohoto algoritmu. První verze zvaná Weighted Slope One při výpočtu zohledňuje váhu získanou skrze pomocnou funkci. Bi-polar Slope One pak využívá číselnou hranici odvozenou od průměrného hodnocení uživatele, která určuje, kdy má být hodnocení zadáno uživatelem interpretováno kladně a kdy už záporně.

Nechť je $dev_{j,k}$ rozdíl v průměrném hodnocení mezi položkami j a k, $P(u)_j$ je neznámé hodnocení uživatele u pro položku j a R_j je soubor položek ohodnocených daným uživatelem [31]. Poté lze zapsat:

$$dev_{j,k} = \sum_{u_i \in I_{jk}} \frac{r_{ij} - r_{ik}}{card(I_{jk})} \tag{2.9}$$

kde funkce $card()$ vrací váhu definovanou počtem uživatelů kteří ohodnotili obě položky. Předpověď hodnocení pak vychází ze vztahu (převzato z [31]):

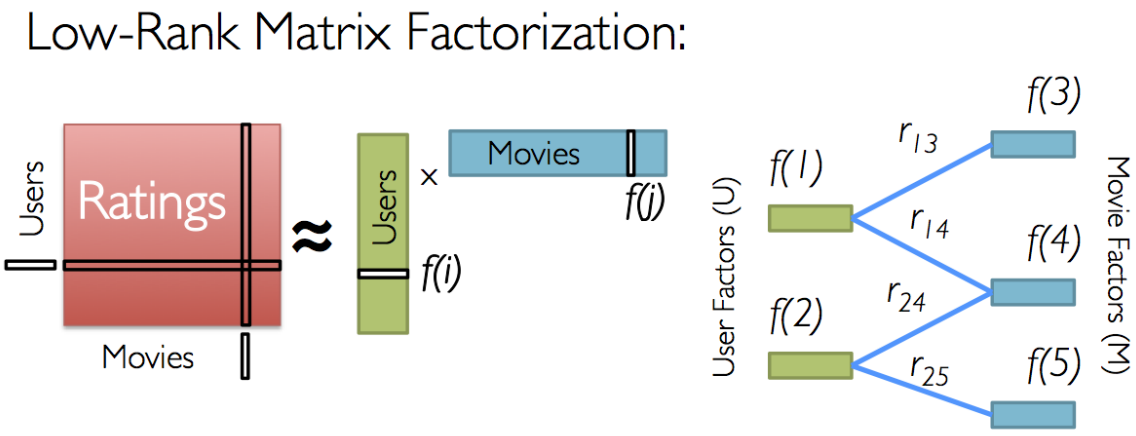
$$P^W(u)_j = \frac{\sum_{k \in R_j} (dev_{j,k} + u_k) c_{jk}}{\sum_{k \in R_j} c_{jk}} \tag{2.10}$$

2.2.2 Přístup založený na modelech

Při tomto přístupu jsou modely pro kolaborativní filtrování vyvinuty pomocí algoritmů pro strojové učení, které předvídají uživatelovo hodnocení pro jednotlivé položky. Tyto algoritmy lze rozdělit do dvou skupin, a to algoritmy založené na faktorizaci matic a pak algoritmy založené na hlubokém učení (deep learning).

Latentní faktorové modely

Některé z nejúspěšnějších implementací latentních faktorových modelů jsou založeny právě na faktorizaci matic [27]. Ve své podstatě faktorizace matic charakterizuje jak položky, tak uživatele pomocí vektorů, které se skládají z faktorů odvozených od interakcí mezi položkami a uživateli, například hodnocení.



Obrázek 2.3: Ilustrace faktorizace matic [30].

Faktorizační modely pak mapují uživatele a položky na společný latentní faktorový prostor f tak, že interakce mezi nimi jsou modelovány jako vnitřní součiny v tomto prostoru. Každá položka i je spojena s vektorem $q_i \in \mathbb{R}^f$ a každý uživatel u je spojen s vektorem $p_u \in \mathbb{R}^f$ (viz obrázek 2.3. Výsledný skalární součin těchto vektorů označován $q_i^T p_u$ zachycuje interakci mezi uživatelem u a položkou i, která vyjadřuje uživatelův zájem o položku. Rovněž lze provést aproximaci hodnocení uživatele u udělené položce i, zapsáno (převzato z [27]):

$$\hat{r}_{ui} = q_i^T p_u p_u \tag{2.11}$$

Velkou součástí tohoto výpočtu je již dříve zmíněné mapování položek na uživatele. Tuto činnost provádí doporučovací systém a výsledkem tohoto mapování je model, který lze využít pro předpověď hodnocení (mapování se tedy neprovádí pokaždé, když chceme předpovědět hodnocení položky uživatelem). Pro konstrukci takového modelu lze využít techniku dekompozice známou jako SVD.

SVD

Jedná se o techniku faktorizace matice, pomocí které se matice redukuje pouze na její podstatné složky. Často se využívá pro zjednodušení určitých maticových operací jako je inverze. U kolaborativního filtrování se využívá jako metoda zjednodušování dat při strojovém učení. SVD lze také využít u lineární regrese metodou nejmenších čtverců, u komprese obrazu a pro odstranění šumu datové sady.

Nechť A je matice m x n, kterou chceme rozložit, U je matice m x m, Σ je diagonální matice m x n a V^T je matice transponovaná k matici n x n. Pro matice s hodnotami v oboru reálných čísel pak platí:

$$A \ = U * \Sigma * V^T \tag{2.12}$$

Sloupce matice U obsahují levé singulární vektory původní matice A vyjadřující vztah mezi uživateli a latentními faktory. Sloupce matice V zase obsahují pravé singulární vektory, které reprezentují podobnost mezi položkami a latentními faktory. Diagonální hodnoty v matici Σ označujeme jako singulární hodnoty původní matice A a tyto vyjadřují sílu každého z latentních faktorů. [4]

Pro využití této techniky však nesmí v matici hodnocení chybět žádné hodnoty, což však není prakticky nikdy možné, jelikož se téměř vždy najde uživatel, který nehodnotil film hodnocený jiným uživatelem. Dřívější systémy byly závislé na imputaci pro vyplnění chybějících hodnocení [20]. Tato metoda se však prokázala jako nepříliš efektivní, jelikož často dochází ke zkreslení dat důsledkem nepřesnosti. Nynější metody se zaměřují na modelování pouze na základě vypozorovaných hodnocení, přičemž se snaží zamezit přetrénování zahrnutím regularizačního parametru [25]. Za účelem získání vektorů faktorů systém minimalizuje kvadratickou odchylku na sadě známých hodnocení:

$$\min_{q*,p*} \sum_{(u,i)\in K} (r_{ui} - q_i^T p_u)^2 + \lambda(||p_u||^2 + ||q_i||^2), \tag{2.13}$$

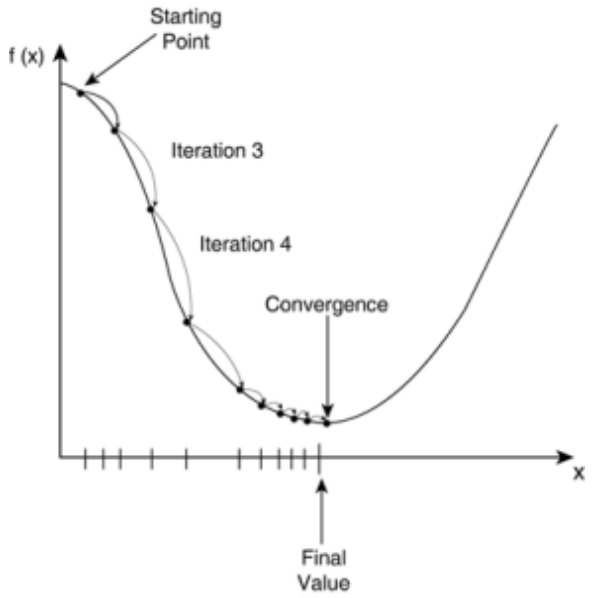
kde K je trénovací sada párů (u, i), pro které je známo hodnocení r_{ui}. Systém sestavuje model pomocí zobecňování těchto předešlých hodnocení takovým způsobem, který by umožnil jejich využití při předpovídání budoucích hodnocení. Proto je důležité, aby nedošlo

k přetrénování, k čemuž slouží právě konstanta λ. Tato konstanta má kontrolu nad mírou regularizace a její hodnota se zpravidla odvíjí od křížové validace.

Najít přesné řešení, díky kterému by hodnota sumy 2.13 byla minimální, je velmi složité. Existují však techniky, díky kterým lze nalézt přibližné řešení. Patří mezi ně například stochastický gradient descent nebo střídavá metoda nejmenších čtverců.

Stochastický gradient descent

Gradient descent je optimalizační algoritmus, který je často využíván v oboru strojového učení. Jedná se o jednoduchý iterativní proces, při kterém je nutné znát funkci ceny (cost function) a počátečních hodnot pro optimalizační proměnné. Při každé iteraci se přepočte gradient funkce ceny s ohledem na optimalizační proměnné a tyto proměnné v rámci každého kroku upraví tak, aby se blížil konvergenci funkce ceny (viz obrázek 2.4). Je však důležité zohlednit fakt, že gradient descent zaručuje konvergenci k lokálnímu minimu, nikoliv globálnímu (i když může konvergovat i tomuto minimu). [35]



Obrázek 2.4: Ukázka algoritmu gradient descent vedoucí ke konvergenci k lokálnímu minimu. Převzato z [35].

Bylo prokázáno, že gradient descent funguje dobře při optimalizaci faktorizačních modelů [16]. Avšak v případě, že je dimenzionalita původní matice hodnocení vysoká, tak počet parametrů, které je zapotřebí optimalizovat, vysoce narůstá.

Stochastický gradient descent (dále jen SGD), je jedna z nepřeberného množství variant této techniky. Zatímco u klasického gradient descentu je při přepočtu gradientu funkce ceny nutné sečíst ceny všech vzorků v sadě, u stochastické varianty je při každé iteraci využívaná cena pouze jednoho náhodného vzorku v sadě. Aby tato metoda fungovala, je však nutné zaručit, aby byl náhodný vzorek zvolen opravdu náhodně, tedy pořadí tohoto vzorku v rámci sady určit nejlépe pomocí generátoru náhodných čísel s velkou periodou.

Střídavá metoda nejmenších čtverců

Jedná se o dvou krokový iterativní optimalizační algoritmus. Na rozdíl od SGD nemůže při výpočetním kroku dojít ke vzdálení se od minima funkce ceny, v nejhorším případě zůstane hodnota stejná. Vraťme se k rovnici 2.13, kterou chceme minimalizovat. V případě, že jednou každou z matic $q*$ a $p*$ ve vzorci upevníme (učiníme neměnnými), získáme dvě kvadratické úlohy, které lze řešit přímo, přičemž výsledky těchto úloh vedou ke konvergenci k lokálnímu minimu funkce ceny. Navíc tím, že jsou na sobě úlohy nezávislé, lze celý proces paralelizovat a dosáhnout tak vysoké efektivity [16].

Překážky související s datovou sadou

Data získaná kolaborativním filtrováním jsou ovlivněna například tím, že někteří uživatele udělují obecně vyšší hodnocení než ostatní, nebo že jsou některé položky hodnoceny kladněji než ostatní, a to nemusí být zrovna z důvodu kvality. Z toho vyplývá, že by nebylo moudré využívat hodnocení z interakcí $q_i^T p_u$ bez modifikací. Namísto toho se dobře navržený doporučovací systém pokouší o identifikaci části těchto hodnot, vůči kterým by mohli uživatelé být zaujatí. Odhad míry této zaujatosti (dále bias) předpovídaného hodnocení r_{ui} lze vyjádřit takto:

$$b_{ui} = \mu + b_i + b_u, \tag{2.14}$$

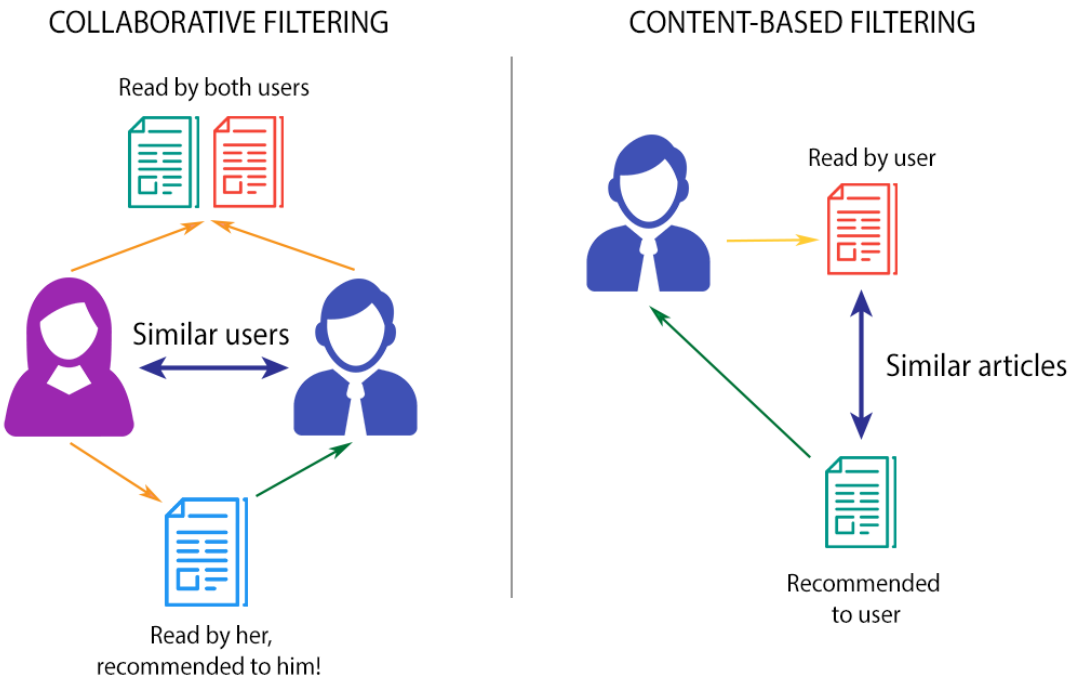
kde bias týkající se hodnocení r_{ui} je označován b_{ui} a zahrnuje v sobě jak vliv uživatelů tak položek. Odchylky od průměrného hodnocení uživatele a průměrného hodnocení položky jsou značeny b_u a b_i. Celkové průměrné hodnocení je pak značeno μ. Pro zahrnutí biasu do rovnice pro předpověď 2.11 může systém efektivněji vykonávat učení na základě minimalizace funkce kvadratické odchylky vycházející z 2.13 (převzato z [3]):

$$\min_{q*,p*,b*} \sum_{(u,i)\in K} (r_{ui} - \mu - b_u - b_i - q_i^T p_u)^2 + \lambda(||p_u||^2 + ||q_i||^2 + b_u^2 + b_i^2), \tag{2.15}$$

Dále je složité zjistit, jakým způsobem jsou jednotlivé vlastnosti hodnocených položek reprezentovány v jednoduché číselné formě, kterou uživatelé poskytují (hodnocení na stupnici od 1 do 5). Kvůli tomu nejsme většinou schopni odvodit na základě čeho je uživateli daná položka doporučena, zdali například podle žánru nebo podle popularity, ale pouze podle výstupu určité výpočetní operace. Do jisté míry lze tento problém eliminovat analýzou případných textových hodnocení přiložených k filmům, toto však značně komplikuje strukturu doporučovacího systému, a často je pro tuto činnost zapotřebí implementovat doporučovacích systému několik.

2.3 Doporučování založené na obsahu

Systémy implementující doporučování založené na obsahu analyzují popisy položek, které v minulosti uživatel hodnotil, a sestaví uživatelský profil na základě vlastností těchto položek. Tento profil je strukturovanou reprezentací uživatelových zájmů a systém jej využívá pro doporučení nových a zajímavých položek. Například pokud uživatel kladně hodnotí film zařazen mezi komedie, tak mu systém může doporučovat filmy právě z tohoto žánru.



Obrázek 2.5: Znázornění rozdílů přístupů pro doporučování [7].

Obecný rozdíl mezi tímto druhem filtrování a kolaborativním filtrování lze vidět na obrázku 2.5.

První krok doporučovacího procesu vykonává analyzátor obsahu, který extrahuje prvky z nestrukturovaných dat, převede je do strukturované podoby jako položky a následně jej uloží do repozitáře položek. V další fázi se na soubor položek označených určitým hodnocením aplikuje algoritmus učení s učitelem za účelem vygenerování prediktivního modelu. Tento model reprezentuje uživatelský profil a je uložen do profilového repozitáře. Poté filtrovací komponent pro položku, kterou aktivní uživatel ještě neohodnotil, posoudí, do jaké míry by uživatele mohla nová položka zajímat. Profil, který věrně odráží uživatelovy preference, je obrovskou výhodou pro efektivitu tohoto procesu. [32]

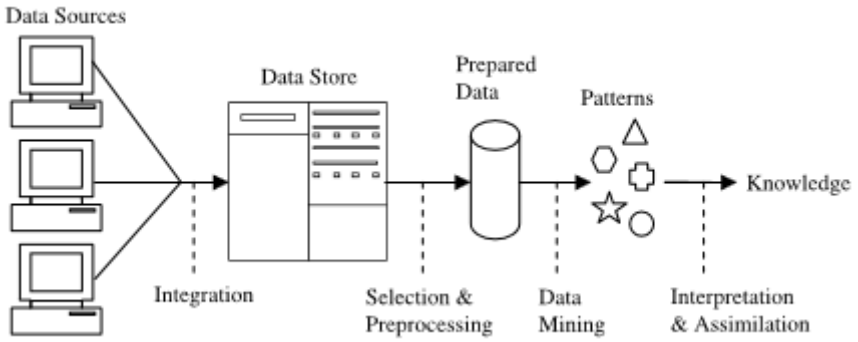
Jedná se o první velký krok v oblasti škálovatelných doporučovacích systémů [32]. Namísto podobnosti mezi uživateli využívá podobnosti mezi vzorky hodnocení položek. Tedy za předpokladu, že dvě položky kladně i záporně hodnotí stejní uživatelé, tak jsou považovány za podobné. Uživateli se poté doporučují položky, které jsou podobné těm, jež kladně hodnotil. Ve své obecné struktuře je tato metoda podobná filtrování založeném na obsahu, na rozdíl od ní však podobnost nezakládá na datech získaných z položek, nýbrž na uživatelových preferencích.

Sám o sobě však tento druh doporučování nic neřeší, opět se potýkáme s problémem časové náročnosti výpočtů podobnosti mezi položkami. Zlepšení spočívá v tom, že předzpracovaný doporučovací model je daleko efektivnější než u předzpracovaného modelu u kolaborativního filtrování mezi uživateli, jelikož uživatelské preference se mohou s časem měnit, zatímco dvě položky zůstávají podobné i se změnami v hodnocení, a to zejména pokud jim byl těchto hodnocení udělen velký počet.

Kapitola 3

Dolování dat a tvorba datové sady

Moderní počítačové systémy shromažďují data v téměř nepředstavitelné rychlosti z velmi širokého rozsahu zdrojů. Tohle obrovské množství dat je díky pokrokům v moderních technologiích daleko snadnější ukládat, a to jak z hlediska efektivity, tak ceny. Většina těchto dat však kvůli svému objemu není zpravidla nikdy analyzována způsobem, který by mohl poskytnout nové znalosti či klíčové informace. Tento nedostatek vedl ke vzniku dolování dat. Proces objevování dat, jehož je dolování dat pouze součástí, se podle [21] skládá z následujících kroků (ilustrace viz 3.1):



Obrázek 3.1: Proces objevování znalostí. Převzato z [3].

1. **Očištění dat** – V tomto kroku je z dat vyfiltrován šum a nekonzistentní data.
2. **Integrace dat** – V této fázi mohou být kombinovány různé zdroje dat. Spolu s předchozím krokem jsou využívány pro předzpracování, přičemž výsledná data jsou poté přenesena do datového uložiště.
3. **Výběr dat** – Zde se z databáze vyberou relevantní data pro analýzu, případně jinou činnost, pro kterou jsou potřeba.
4. **Transformace dat** – Při tomto kroku jsou data přeměněna tak, aby byla dobře analyzovatelná, pomocí agregačních nebo sumarizačních operací.
5. **Dolování dat** – Klíčový proces, při kterém jsou na transformovaná data aplikovány různé inteligentní metody, jejichž účelem je extrahovat vzory (patterns) z těchto dat.

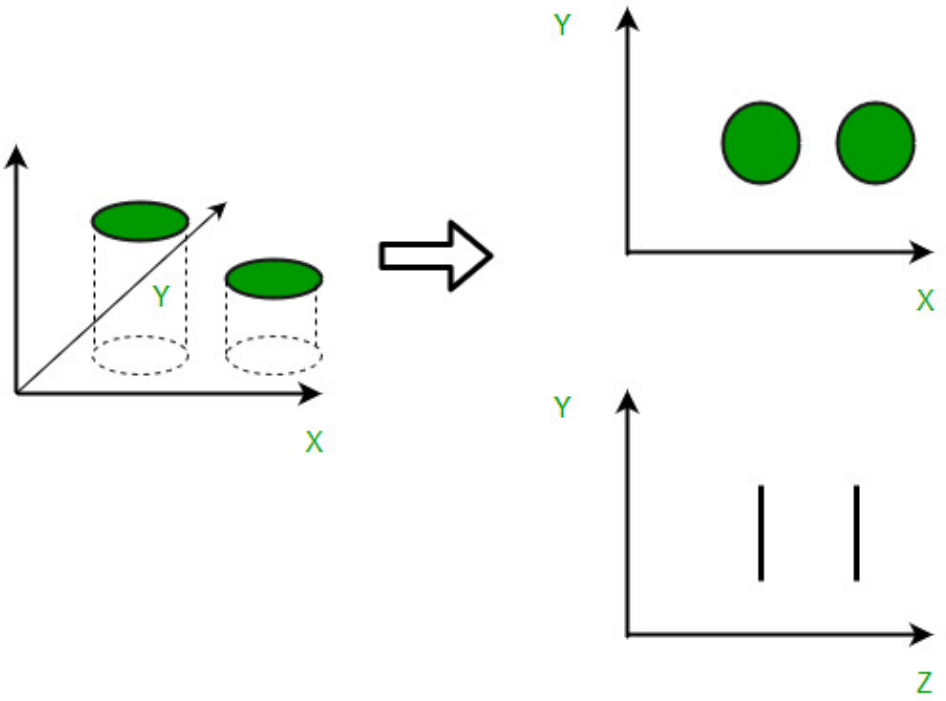
6. **Vyhodnocení vzorů** – Při této fázi jsou z extrahovaných vzorů získaných v předchozím kroku vybrány ty, které jsou relevantní pro další analýzu.

7. **Reprezentace znalostí** – Na závěr jsou analýzou získané znalosti vizualizovány a presentovány uživateli.

3.1 Předzpracování dat

Databáze jsou vysoce náchylné k šumu, ztrátám a porušení konzistence dat. V rámci procesu předzpracování dat se provádí čištění dat, které některé z těchto jevů odstraní například doplněním chybějících, nebo odstraněním nesouvisejících hodnot. Tato činnost ulehčuje analýzu získaných dat a zároveň zvyšuje důvěryhodnost znalostí objevených touto analýzou. [21]

Dále je zapotřebí odstranit nadbytečné atributy, které by mohly zkreslovat přesnost modelu. Přestože je nyní možné ukládat záznamy s velkým množstvím atributů, ne vždy je to žádoucí. S narůstajícím počtem atributů totiž úměrně narůstá riziko, že výsledky získané trénováním z těchto dat nebudou spolehlivé. K eliminaci tohoto problému slouží redukce dat. Mezi nejčastěji využívané strategie pro redukci dat se podle [3] řadí redukce dimenzionality (viz obrázek 3.2).



Obrázek 3.2: Znázornění procesu redukce dimenzionality [40].

Předpokládejme že máme k dispozici datovou sadu reprezentovanou maticí X o velikosti $n \times D$ obsahující n datových vektorů x_i $(i \in 1, 2, ..., n)$ s dimenzionalitou D. Dále předpokládejme, že tato datová sada má vnitřní dimenzionalitu d (kde $d < D$). Techniky redukce dimenzionality transformují tuto datovou sadu X s dimenzionalitou D do nové datové sady Y s dimenzionalitou d. Některé aspekty datové sady jako například vnitřní dimenzionalita

d zpravidla nejsou předem známy a proto jsme nuceni si jejich hodnoty odvodit sami na základě určitých předpokladů, které se liší podle užité techniky redukce. [41]

3.2 Dolování dat

Dolování dat představuje analýzu velkého množství dat za účelem extrakce neznámých a potencionálně zajímavých vzorů. Tento proces zpravidla zahrnuje využití různých databázových technik jako prostorové indexy (spatial indices) [8]. Takto získané vzory mohou být interpretovány jako shrnutí vstupních dat a lze je dále využít například při strojovém učení, nebo prediktivní analýze.

Klasifikace

Klasifikace je jeden z nejčastějších způsobů využití dolování dat [15]. Představuje hledání modelu (nebo funkce), který popisuje a rozděluje datové třídy či koncepty. Tento model je založen na analýze sady trénovacích dat a je použit pro předpověď zařazení objektů v datové sadě, které ještě nejsou zařazeny do žádné skupiny. Odpovídá činnostem, které se běžně často v běžném životě. Například v nemocnici je žádoucí, aby pacienti byli rozděleni do skupin podlé toho, jak velké riziko onemocnění jim hrozí. Dále ku příkladu můžeme chtít klasifikovat předměty studenta podle toho, s jakou známkou je absolvoval. Pro provedení těchto činností lze využít vícero možností, mezi které se řadí například:

- **Spojování nejbližších sousedů** – Tato metoda závisí na propojení trénovacích příkladů a jejich označení s nejbližším neklasifikovaným příznakem (například převahuje-li v okolí příznaku z označení A a B právě označení A, tak bude klasifikován do skupiny A).
- **Klasifikační pravidla** – Zde hledáme pravidla, která můžeme využít pro zařazení neklasifikovaného příznaku.
- **Klasifikační strom** – Jeden způsob generování klasifikačních pravidel je skrze stromovou strukturu nazývanou jako klasifikační či rozhodovací strom.

Regresivní analýza

Regresivní analýza se u dolování dat využívá jako spolehlivá metoda sloužící pro identifikaci proměnných, které mají vliv na zkoumané vzory. Pomocí regrese lze s poměrně vysokou přesností určit, které faktory jsou při analýze nejdůležitější, které lze naopak ignorovat a jak spolu tyto faktory souvisí. [16]

Aby bylo možné regresivní analýzu provést, je zapotřebí definovat závislou proměnnou, o které lze předpokládat, že je ovlivňována jednou nebo více nezávislými proměnnými. V rámci této práce se využívá lineární forma regrese jako ztrátová funkce při trénování modelu, o kterém se lze dočíst více v sekci 5.1.

3.3 Implementace skriptu pro shromáždění dat

Pro shromáždění dat byl využit proces extrakce a vytváření strukturované reprezentace dat z webové stránky známý jako web scraping. Využívá se především tehdy, pokud majitel těchto dat neposkytuje otevřené aplikační rozhraní (API) k přístupu k těmto datům. Před

přistoupením k této variantě bylo učiněno několik pokusů o kontaktování zodpovědných osob prostřednictvím formuláře na webové stránce ČSFD, kde byl vysvětlen účel využití dat a jaké případné užitky by mohl doporučovací systém trénovaný na těchto datech portálu přinést, bohužel však na tyto pokusy nikdy nepřišla odpověď.

Pro implementaci skriptu, který doloval data z ČSFD, byl zvolen programovací jazyk PHP, jelikož pro něj existuje široké množství open-source knihoven, které jsou tvořeny právě pro tento účel. Pro funkcionalitu dolování byla do skriptu zahrnuta knihovna *Goutte*[1], která je postavena na PHP HTTP klientovi *Guzzle*[2]. Skript dostal jako parametr, zdali má dolovat filmy, nebo uživatele spolu s jejich hodnoceními a zároveň počet položek, po jejichž zpracování se má ukončit. Dolovaná data byla ukládána do MySQL databáze, která se poté exportovala do HDF5 formátu binárních dat pro zpracování při trénování modelu.

Při následné implementaci doporučovacího systému byla pomocí tohoto skriptu získávána další data a experimentoval tak s přesností doporučování vlivem velikosti datové sady (viz kapitola 5.3).

3.4 Výsledná trénovací sada

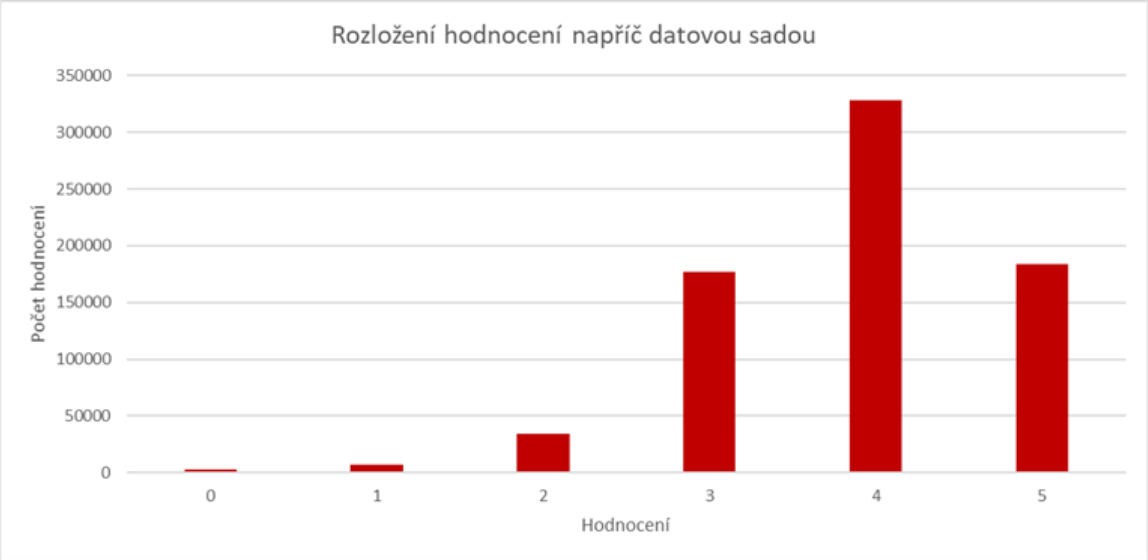
Aby byla datová sada dobře vybalancovaná, shromažďovala se data pouze o takových filmech, které měly více než 1000 hodnocení, a zároveň byly zveřejněny později než v roce 1920. Při doporučování je vždy z pohledu doporučovacího systému jistější, když doporučí film, který hodnotilo kladně větší spektrum lidí, než film, který je sice hodnocen kladně, ale malým počtem uživatelů.

Zároveň se při kolekci dat o uživatelích ignorovali uživatelé s počtem hodnocení větší než 5000 a menší než 50, a to z toho důvodu, že je pro algoritmus učení daleko lepší vycházet z dat od 50 uživatelů, kde každý provedl 100 hodnocení, než jednoho uživatele, který jich provedl 5000. V opačném případě, kdy by datová sada obsahovala velký počet uživatelů s nižším počtem hodnocení, byla by matice hodnocení velmi řídká a vyskytovalo by se v ní mnoho nevyplněných hodnot. S tímto faktem by se model složitě vypořádával, protože to zhoršuje jeho schopnost vyhledávat uživatele s podobným profilem.

O filmech byla shromažďována následující data: identifikátor ČSFD, název, režisér, rok natočení, průměrné hodnocení, odkaz na plakát a žánry. O uživatelích nebyla shromažďována téměř žádná data (aby zůstali anonymní) mimo jejich identifikátoru na ČSFD a seznamu hodnocení, které filmům udělili. Graf rozložení hodnocení napříč datovou sadou lze vidět na obrázku 3.3.

Důsledkem dolování vznikla datová sada obsahující data o 4350 filmech. Tyto filmy hodnotilo 1479 uživatelů celkem 732483 hodnoceními. Z grafu 3.3 lze vyčíst, že rozložení hodnocení není úplně ideální a převažují hlavně hodnocení v rozsahu 3 – 5 hvězdiček.

[1] https://github.com/FriendsOfPHP/Goutte
[2] https://github.com/guzzle/guzzle



Obrázek 3.3: Rozložení hodnocení v hvězdičkách od 0 do 5 napříč datovou sadou.

Kapitola 4

Systém vytvářející doporučení filmů

V této kapitole se lze dočíst o návrhu aplikace pro doporučování filmů, se kterým souvisí volba vhodného modelu pro tento konkrétní případ užití a volba vhodné knihovny, která vyhovujícím způsobem implementuje zvolený model. Nakonec zde jsou rozebrány doporučovací systémy komerčně využívány světovými technologickými firmami, které sloužily jako inspirace při návrhu a následné implementaci aplikace.

4.1 Návrh aplikace

Asi nejdůležitějším krokem při návrhu systému bylo zvolit odpovídající model, který s ohledem na vydolovanou datovou sadu a problematiku poskytování filmových doporučení bude poskytovat nejlepší výsledky. Tento výběr byl inspirován již existujícími řešeními, o kterých se lze dočíst v sekci 4.3.

Výběr modelu

V době psaní této práce jsou podle [7] téměř všechny doporučovací systémy, které doporučují filmy, z určité části založeny na kolaborativním filtrování. Důvodem je vysoká efektivita těchto systémů a poměrně vysoká přesnost doporučování. Pro kolaborativní filtrování existuje vícero přístupů (zmíněných v sekci 2.2, jelikož je však zadáním specifikováno, aby byla pro doporučování využita neuronová síť, bylo nutné využít přístup založený na modelech.

Neuronových sítí je nespočet druhů, které lze téměř donekonečna určitým způsobem upravovat, či vylepšovat. Neuronová síť, na které je model doporučovacího systému implementovaný v rámci této práce postaven, může být poměrně jednoduchá, a to proto, že data, na kterých je síť trénovaná, nejsou komplexní. Jedná se pouze o trojice složených z uživatelů, položek a hodnocení.

Bylo by možné využít i složitější modely, například sekvenční, které při trénování zohledňují časovou posloupnost interakcí. Poté by však bylo nutné, aby se v datové sadě vyskytovala informace o času vzniku interakce. Při dolování dat z ČSFD nebylo možné tato data získat.

Hluboká neuronová síť by šla pro doporučovací systém také použít, její výhodou je detailnější rozbor vztahů mezi uživateli a položkami (a tím přesnější doporučování), díky využití více skrytých vrstev. Datová sada však neobsahuje velké množství metadat a implicitní data se o uživatelích také nebudou shromažďovat. Důsledkem toho by síť neměla

dostatek informací k analýze vztahů a trénování by se výrazně prodloužilo z důvodu vyššího počtu vrstev. Proto byl tento přístup zavrhnut.

Po zvážení všech těchto skutečností byl nakonec vybrán explicitní faktorizační model využívající techniku kolaborativního filtrování.

4.2 Srovnání open-source knihoven

Poté, co byl zvolen model, na kterém bude doporučovací systém založen, bylo provedeno srovnání několika knihoven, které aplikací požadovanou funkcionalitu již částečně implementují. Některé z knihoven obsahují spíše kostru pro tvorbu modelů, zatímco ostatní už v sobě tyto modely mají naimplementovány.

Torch

Torch[1] je knihovna pro strojové učení napsána objektově v jazyce C++. Pro zjednodušení úprav již existujících algoritmů a ulehčení nových algoritmů a metod byla při návrhu zvolena modulární strategie a logika byla dle [10] rozdělena do následujících tříd:

- `DataSet` – Tato třída zpracovává jak statická a dynamická data buď uložením do paměti, případně je ukládá na disk, když je datová sada velmi rozsáhlá.
- `Machine` – Tato část knihovny představuje černou skříňku, která obdrží vstup s volitelnými parametry a na jeho základě poté poskytne výstup. Může se jednat o instanci neuronové sítě nebo například Markovovského modelu.
- `Trainer` – Tato třída se využívá pro výběr optimálních parametrů pro vstup do třídy `Machine` na základě daných kritérií a datové sady.
- `Measurer` – Objekt této třídy zapisuje do specifikovaných souborů, které nás při strojovém učení mohou zajímat, jako například hodnotu střední kvadratické odchylky, velikost klasifikační chyby apod.

Nejprve se tedy vytvoří několik trénovacích příkladů, které se předají jako vstup do stroje (machine), neboli černé skříňky. Tento stroj vypočítá výstup, který je poté využit trénovacím prvkem pro ladění parametrů, na základě nichž stroj výstup počítá. Některé stroje lze však trénovat pouze určitými typy trénovacích prvků.

V době psaní této práce je největším konkurentem této knihovny *TensorFlow*[2] od firmy Google. Na rozdíl od této knihovny nevyužívá statické grafy a tím umožňuje vývojářům a výzkumníkům změnit chování neuronové sítě za běhu. Objektivní srovnání času délky kroku jednotlivých knihoven je znázorněno v tabulce 4.1.

[1] http://torch.ch/
[2] https://www.tensorflow.org/

Tabulka 4.1: Čas délky kroku v milisekundách pro čtyři konvoluční modely pomocí rozdílných knihoven. Převzato z [17].

	AlexNet	Overfeat	OxfordNet	GoogleNet
Caffe	324	823	1068	1935
Neon	87	211	320	270
Torch	81	268	529	470
TensorFlow	81	279	540	445

PyTorch

PyTorch[3] je vědecký výpočetní balíček založený na Torch, který pro své výpočty využívá sílu grafických karet. Je to také jedna z preferovaných platforem pro hluboké učení stavěných pro podporu maximální flexibility a rychlosti [38]. Knihovna je navržena tak, aby byla intuitivní a jednoduchá pro adaptaci. Protože se jedná o nativní Python balíček, tak se velmi jednoduše zahrnuje do jiných balíčků, modulů, nebo již existujících aplikací.

LightFM

LightFM[4] vytváří reprezentace uživatelů a položek za pomocí faktorizace. Po provedení skalárního součinu těchto vektorů získáme hodnocení, které odráží kvalitu potencionálního doporučení položky danému uživateli.

Metoda lineární faktorizace, kterou LightFM využívá, je výpočetně velmi efektivní, lze při ní však narazit na problémy při zahrnutí velkého množství nesouvisejících, či v některých případech redundantních metadat. Proto v případě, kdy je zapotřebí, aby doporučovací systém při doporučování zohledňoval i detaily ve vztazích mezi jednotlivými uživateli a položkami, musíme využít nelineární reprezentační funkce, jako například hluboké neuronové sítě. [29]

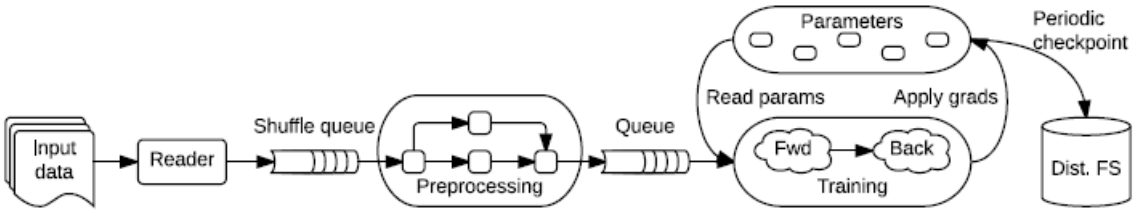
Spotlight

Spotlight[5] poskytuje širokou sadu stavebních bloků pro ztrátové funkce (využívající jak regresivní tak klasifikační přístupy), reprezentace (faktorizační reprezentace, sekvenční modely) a užitečné nástroje pro získávání i generování datových sad [30]. Cílem této knihovny je rychlé objevování a prototypování nových mělkých i hlubokých doporučovacích modelů. Využívá pro to volně dostupnou platformu pro hluboké učení PyTorch, zmíněnou v sekci 4.2.

TensorFlow

TensorFlow je systém pro strojové učení, který využívá grafy toku dat k reprezentaci výpočtu, sdíleného stavu a operací, které tento stav upravují. Mapuje uzly tohoto grafu přes mnoho zařízení ve shluku a v rámci těchto zařízení skrze jednotlivé výpočetní zařízení. Veškerá komunikace mezi výpočty je vyjádřena explicitně, což umožňuje provedení výpočtů nezávisle na sobě a zároveň paralelně (viz 4.1). [1]

[3] https://pytorch.org/
[4] https://github.com/lyst/lightfm
[5] https://github.com/maciejkula/spotlight



Obrázek 4.1: Schéma grafu toku dat využívaného systémem TensorFlow. Převzato z [1].

Všechna data v TensorFlow jsou modelována pomocí takzvaných tensorů, což jsou n-dimenzionální pole obsahující prvky konkrétního primitivního datové typu, jako například číslo s plovoucí řádovou čárkou nebo celé číslo [17]. Tensory představují vstupy a výstupy běžných matematických operací využívaných v algoritmech strojového učení.

Mnoho algoritmů učení k trénování využívají určitou formu SGD (viz kapitola 2.2.2). V případě TensorFlow provádí rozlišovací algoritmus BFS (breadth-first search) napříč stavovým prostorem za účelem identifikace všech zpětných cest ze ztrátové funkce. Poté provede součty všech částečných gradientů (vedoucí k minimu ztrátové funkce). V rámci této funkcionality mohou uživatele implementovat širokou škálu optimalizačních prvků, které vypočítají nové hodnoty parametrů v každém trénovacím kroku.

TensorRec

TensorRec[6] je doporučovací algoritmus vycházející z TensorFlow (viz podsekce 4.2) a LightFM (rozebráno v podsekci 4.2) s jednoduchým aplikačním rozhraním pro trénování a předvídání hodnocení, který se podobá jiným obecně známým nástrojům pro strojové učení v jazyce Python. Zároveň poskytuje flexibilitu pro experimentování s reprezentacemi dat a ztrátovými funkcemi, což umožňuje přizpůsobit doporučovací systém různým typům uživatelů a položek. [26]

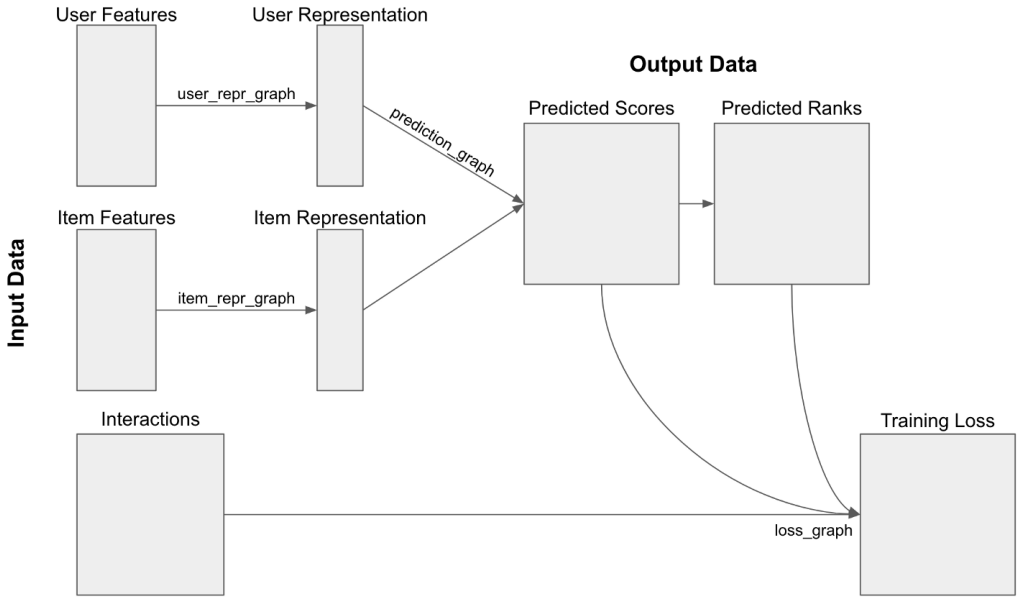
Doporučování jsou ohodnocovány zpracováním vlastností uživatelů a položek za účelem vytvoření dvou nízko dimenzionálních vektorů, neboli jejich reprezentací. Skalární součin těchto dvou vektorů je právě hodnocení vztahu mezi uživatelem a položkou, kde vyšší hodnoty znamenají lepší předpovědi (viz obrázek 4.2).

Algoritmus využit pro generování reprezentací, zvaný jako reprezentační funkce, může být libovolně upraven. Lze jej přeměnit například na lineární transformační funkci a nebo na hlubokou neuronovou síť. Volba této funkce se odvíjí zejména od využité datové sady.

Výběr knihovny

Jelikož byl jako implementační jazyk aplikace zvolen Python, vybíralo se mezi knihovnami Spotlight a TensorRec. Nakonec byla vybrána knihovna Spotlight, protože poskytuje více druhů trénovacích modelů (jak explicitní, tak implicitní, ale i sekvenční), přičemž TensorRec umožňuje využít pouze jeden předem daný typ modelu. Zároveň v sobě Spotlight obsahuje široké spektrum ztrátových funkcí, které lze při trénování modelu využít. TensorRec je v tomto ohledu značně omezen, a to právě z důvodu, že je některé ztrátové funkce velmi složité převést do TensorFlow grafů, ze kterých knihovna vychází.

[6] https://github.com/jfkirk/tensorrec



Obrázek 4.2: Princip algoritmu TensorRec. Převzato z [26].

Způsob, jakým jsou modely v rámci knihovny Spotlight implementovány, byl pečlivě analyzován, což bylo nutné minimálně pro zahrnutí funkcionality doporučování uživateli, který se ještě nevyskytuje v datové sadě. Detailní informace o tom, jakým způsobem tento explicitní faktorizační model funguje se lze dočíst v sekci 5.

4.3 Existující řešení

V této sekci se lze dočíst o srovnání s existujícími komerčně implementovanými řešeními, konkrétně firmami *YouTube*[7] a *Netflix*, které zveřejnily informace o architektuře svých doporučovacích systémů.

4.3.1 YouTube

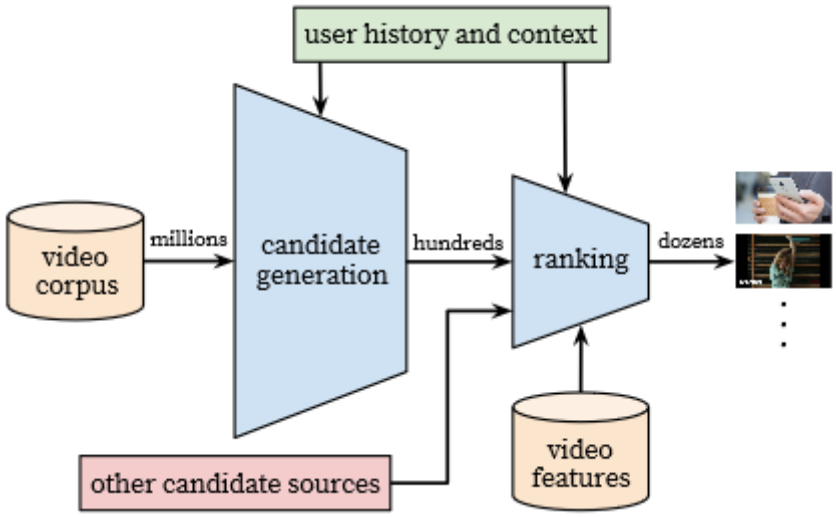
Doporučovací systém na stránce YouTube sestává ze dvou neuronových sítí, kde jedna generuje kandidáty pro doporučení a druhá tyto kandidáty ohodnocuje (viz obrázek 4.3). [12]

Generování kandidátů

Síť, která generuje kandidáty, je postavena na principu kolaborativního filtrování popsaného v kapitole 2.2. Čerpá data z uživatelovy historie aktivity na stránce a využívá je jako vstup. Výstupem jsou pak stovky videí, které by mohly uživatele zajímat, s poměrně vysokou přesností.

Asi nejdůležitějším prvkem při doporučování YouTube videí je zaměření se na nově nahraný obsah. Výzkumy opakovaně ukazují, že uživatelé upřednostňují novější obsah, avšak

[7] https://www.youtube.com/



Obrázek 4.3: Architektura doporučovacího systému využívaného na YouTube. Převzato z [11].

nikoliv za cenu relevance vůči jejich zájmům. Kromě doporučování novinek je rovněž důležité do doporučování zahrnout virální (viral) videa, která jsou v konkrétním okamžiku v uživatelově demografické skupině vysoce populární. U strojového učení se však často projevuje bias vůči minulosti, protože jsou trénovány tak, aby předpovídaly budoucí chování ze starších příkladů. Aby se tomuto dalo zabránit, předává se modelu při trénování stáří příkladu jako složka. [11]

Trénovací příklady poskytnuty modelu jsou vybírány ze zhlédnutí všech videí a nejen těch videí, které systém doporučuje. V opačném případě by se nový obsah objevoval jen s těží a doporučovací systém by byl velmi předpojatý vůči některým videím [12]. Pokud uživatelé naleznou videa i jiným způsobem než prostřednictvím doporučení, je nutno tento objev rychle propagovat ostatním uživatelům prostřednictvím kolaborativního filtrování.

Dalším důležitým postřehem, který zlepšil relevanci doporučení bylo generování statického počtu trénovacích příkladů pro každého uživatele, což zabránilo skupině vysoce aktivních uživatelů, aby byli váhově nadhodnoceni při užití ztrátové funkce [12]. Zadržování některých informací od klasifikátoru je rovněž velmi důležité, aby nedošlo k přetrénování modelu.

Hodnocení kandidátů

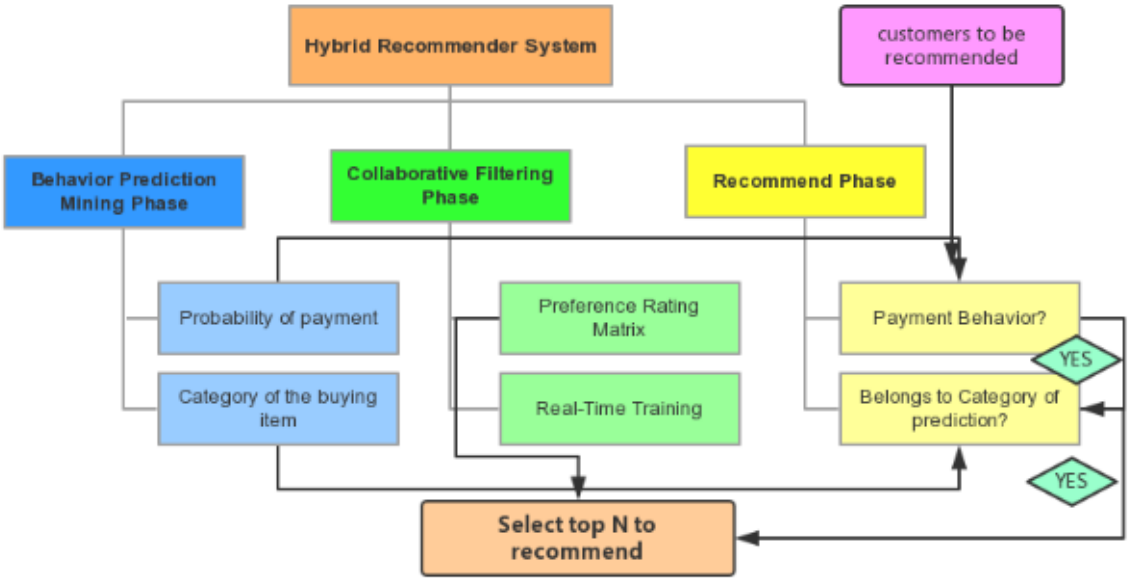
Za účelem poskytnutí co nejlepších doporučení je však zapotřebí využít druhou neuronovou síť, která výstupní videa z první neuronové sítě seřadí dle číselného hodnocení, které se odvíjí zejména od detailů o uživateli a jednotlivých videích [11]. Tento přístup zaručuje, že jsou doporučení vybírána z opravdu širokého rozsahu a zároveň jsou vybírána jen ta, která by mohla být pro uživatele zajímavá.

V modelu pro hodnocení kandidátů jsou využívány stovky vlastností, přičemž největší obtíží je určit jak posloupnost uživatelských akcí souvisí s hodnocením videa. Z pozorování vyplývá, že nejdůležitější signály jsou ty, které zahrnují uživatelovu předchozí interakci s videem samotným a videi jemu podobnými. Mezi tyto signály patří například počet videí, které uživatel zhlédnul na konkrétním kanále, kdy naposled uživatel zhlédnul video na

toto téma a podobně. Rovněž je důležité propagovat informace z generování kandidátů do tohoto modelu, patří mezi ně například zdroje, které video nominovaly a jaké hodnocení videu udělili. Systém rovněž zohledňuje, pokud uživateli bylo doporučeno video na které se nepodíval. Při znovu načtení stránky snižuje míru relevance tohoto videa a umístí jej na méně prominentní místo v seznamu doporučených položek nebo jej ze seznamu vyjme úplně. [11]

4.3.2 Netflix

Průzkum spotřebitelů ukazuje, že průměrný předplatitel Netflixu ztrácí zájem o sledování po 60 až 90 sekundách vybírání, přičemž prozkoumal mezi 10 a 20 tituly. Proto je naprosto klíčové, aby v této době uživatel našel takový titul, který jej zajímá, a pro splnění tohoto účelu využívá Netflix široký rozsah algoritmů tvořící komplexní doporučovací systém (viz 4.4). [18]



Obrázek 4.4: Architektura hybridního modelu využívaného firmou Netflix [14].

Doporučovací systém je využit převážně na hlavní stránce, která se skládá ze sekcí (řádků), jako jsou žánry, nejlepší výběry a v neposlední řadě například populární tituly. Uživatelé, kteří se rozhodnou zhlédnout určitý titul, pochází z 80% právě z této stránky a zbylých 20% pak filmy a seriály vybírá na základě vyhledávání názvů a klíčových slov [2].

Personalizovaný výběr

Algoritmus personalizovaného výběru se využívá právě v sekcích, které souvisí s konkrétními žánry. Funguje na základě seřazení celého katalogu videí podle konkrétního žánru pro každého předplatitele. Toto seřazení je převážně unikátní a zohledňuje zejména uživatelovu historii zhlédnutí, konkrétně zajímá-li se uživatel více o filmy než seriály, dále bere v úvahu oblíbené herce či režiséry a podobně. Schopnost algoritmu poskytovat osobní doporučení je však z části limitována kombinací s faktorem popularity titulů. [2]

Nejlepší videa

Sekce nejlepších videí pak na rozdíl od předešlého algoritmu nezohledňuje celý katalog, ale pouze na hlavičku katalogu (tituly s nejkladnějším ohlasem). Z tohoto výběru pak vybere ty, které nejvíce odpovídají uživatelovým preferencím.

Trendy

Na stránce rovněž existuje sekce zaměřující se na dočasné trendy v rozsahu několika minut až po dny. Tyto trendy jsou velmi efektivní pro předpověď potencionálně zajímavých titulů pro uživatele, zejména v kombinaci s malou dávkou personalizace [18]. Tento algoritmus identifikuje zejména takové trendy, které se opakují každých pár měsíců, ale mají pouze krátkodobý efekt (například zvýšený zájem o romantické filmy poblíž dne svatého Valentýna). Dále bere v úvahu jednorázové nedlouho trvající události, které jsou populární zejména v médiích, či na internetu (ku příkladu blížící se hurikán zvyšuje zájem o dokumenty pojednávající o této tématice, či jiných přírodních katastrofách).

Řešení problému studeného startu

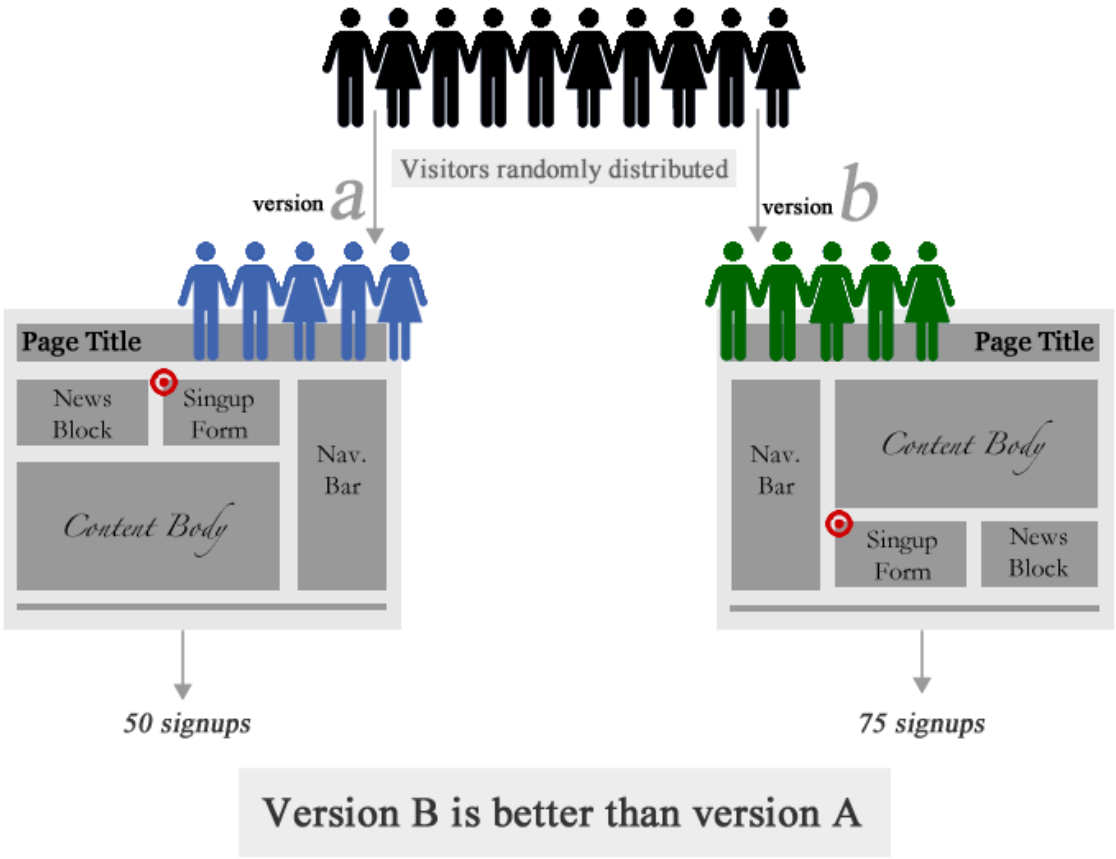
Doporučovací systém sice dosahuje uspokojivých výsledků pro uživatel s dlouhou historií využívání služby, problém však nastává v situaci, kdy přicházejí noví uživatelé, o kterých systém nemá prakticky žádné informace. Počet zrušených předplacení je ve skupině nových uživatelů největší, a to zpravidla důsledkem toho, že je první měsíc ve službě zdarma [2]. Tohle období je velmi důležité, protože když uživatel začne *Netflix* využívat i po této době, tak statisticky zůstane předplatitelem delší dobu. Proto se firma snaží neustále vylepšovat a hledat nové modely, které by zlepšily míru relevance doporučení pro nové uživatele.

V době psaní této práce je pro získání informací o uživateli při první aktivaci účtu zobrazen dotazník, který umožňuje uživateli vybrat filmy, či seriály, které se mu líbily. Stejným způsobem je řešen problém studeného startu i v aplikaci pro doporučování filmů v rámci této práce.

A/B testování

Jelikož má *Netflix* velký počet předplatitelů, je jednoduché testovat různé konfigurace modelu pomocí A/B testování. Obecně se jedná o experiment, při kterém jsou testovací subjekty rozděleny do dvou skupin a každé skupině je prezentován prakticky stejný obsah, až na jednu proměnnou. Po uběhnutí předem stanoveného časového úseku jsou výsledky obou variant srovnány a na základě této analýzy se poté určí, která varianta je vhodnější (viz obrázek 4.5).

Způsob realizace tohoto typu testování na streamovací platformě *Netflix* pak probíhá tak, že jsou předplatitelé rozděleni do několika buněk, kde doporučování uživatelům v rámci každé buňky zprostředkovává jemně upravený doporučovací model. Poté jsou sledovány metriky jako délka pobytu na stránce, délka výběru pořadu či filmu ke sledování a podobně. Na základě těchto metrik je poté vybrán takový model, který poskytoval při doporučování obsahu nejrelevantnější doporučení. Zajímavý je tedy fakt, že testování je prováděno za běhu aplikace na skutečných uživatelích, nikoliv vybraných testovacích subjektech, a vede k drastickým vylepšením přesnosti doporučení, které se odráží na neustálém růstu platformy. [18]



Obrázek 4.5: Princip A/B testování u webové stránky, kde je analyzován počet registrací u jedné a druhé varianty [9].

Kapitola 5

Implementace aplikace

Jako implementační jazyk aplikace byl zvolen Python. Tento programovací jazyk se stává jedním z nejpopulárnějších jazyků pro vědecké výpočty, zároveň však proniká i do odvětví komerčního vývoje. Vděčí za to své vysokoúrovňové interaktivitě a mimo jiné svému bohatému ekosystému vědeckých knihoven. Jedná se o dobrou volbu pro vývoj algoritmů, či například datovou analýzu. [33]

Pro tvorbu doporučovacího systému v jazyce Python existuje mnoho knihoven, které jsou prověřeny i využívány jak vědci, tak bohatou komunitou vývojařů tohoto odvětví. Proto bylo rozhodnuto využít některé z těchto knihoven při vývoji aplikace.

Jelikož Spotlight poskytuje výběr z několika možných modelů pro proces trénování, bylo zapotřebí zvolit ten nejvhodnější pro aplikaci v rámci této práce. Obecně lze využít jak faktorizační model, tak sekvenční. Zvolen byl právě faktorizační model, přičemž o zdůvodnění výběru tohoto modelu pro aplikaci se lze dočíst více v sekci 4.1.

Faktorizační modely jsou k dispozici dva, a to explicitní a implicitní. Vzhledem k tomu, že aplikace neshromažďuje informace o uživatelových interakcích za běhu aplikace, byl vybrán explicitní faktorizační model. Explicitní data pro systém uživatel zadá při prvotním spuštění aplikace, kdy je vyzván k výběru oblíbených filmů, které v historii zhlédnul, z předem definovaného počtu titulů. Na základě této volby může následně model provést doporučení a částečně se tímto eliminuje problém studeného startu.

Princip doporučování

Princip předpovědi možného hodnocení, které uživatel položce udělí, je založen na faktorizaci matic. Obecný postup je následující:

1. Začne se s trojicemi složených z uživatelů, položek a hodnocení, které předávají informaci o tom, že uživatel i hodnotil položku j hodnocením r.

2. Následně se uživatelé i položky převedou do vysoce dimenzionálních vektorů čísel. Uživatel by tedy mohl být reprezentován vektorem `[0.3, -1.2, 0.5]` a položka vektorem `[1.0, -0.3, -0.6]`, přičemž by se skalární součin těchto dvou vektorů měl rovnat původnímu hodnocení

3. Myšlenka je poté taková, že když provedeme skalární součin vektoru uživatele s vektorem položky, kterou v minulosti neohodnotil, dostaneme odhad hodnocení této položky daným uživatelem [30]

5.1 Trénování modelu

Trénování modelu založeného na tomto principu se provádí pomocí SGD (viz sekce 2.2.2). Při využití knihovny Spotlight probíhá proces následovně:

1. Začne se s reprezentací uživatelů a položek náhodně vybranými vektory. V případě, že by vektory nebyly vybrány náhodně, nebo například vynulovány, tak nebude SGD fungovat. Hodnoty uvnitř těchto vektorů se samozřejmě neustále upravují v průběhu trénování modelu.

2. Poté se procházejí trojice uživatelů, položek a hodnocení obsažené v datové sadě a pro každou takovou trojici se vypočte přibližné hodnocení skalárním součinem vektorů uživatele a položky, které se poté porovná se skutečným hodnocením z datové sady.

3. Pokud je přibližné hodnocení příliš malé, upraví se vektory uživatele a položky tak, aby se hodnocení zvýšilo. V opačném případě se upraví tak, aby se hodnocení snížilo.

4. Tento proces se opakuje do doby, než se přesnost modelu při předvídání hodnocení ustálí. [30]

Model samotný pro zpracovávání uživatelů a položek využívá bilineární neuronovou síť složenou ze čtyř vrstev. První vrstva reprezentuje uživatele a třetí vrstva bias vůči uživatelům, zatímco druhá vrstva reprezentuje položky a čtvrtá vrstva zase bias vůči položkám. Společně tyto vrstvy zprostředkovávají předpovědi hodnocení.

5.2 Metriky přesnosti modelu

Kvalitu doporučovacího systému lze odvodit srovnáním předpovědí s testovací sadou známých hodnocení. Mezi nejčastěji využívanou metriku řadíme MAE (mean absolute error), definovanou jako průměrný absolutní rozdíl mezi předpovězenými a skutečnými hodnoceními, zapsáno:

$$MAE = \frac{\sum_{u,i} |p_{u,i} - r_{u,i}|}{N}, \tag{5.1}$$

kde $p_{u,i}$ je předpověď hodnocení položky i uživatelem u, $r_{u,i}$ je skutečné hodnocení a N je celkový počet hodnocení v trénovací sadě.

Další často užívanou metrikou je RMSE (root mean square error), značící střední kvadratickou odchylku. Tato klade větší důraz na vyšší hodnoty absolutních chyb. Její zápis je následující:

$$RMSE = \sqrt{\frac{\sum_{u,i} (p_{u,i} - r_{u,i})^2}{N}}, \tag{5.2}$$

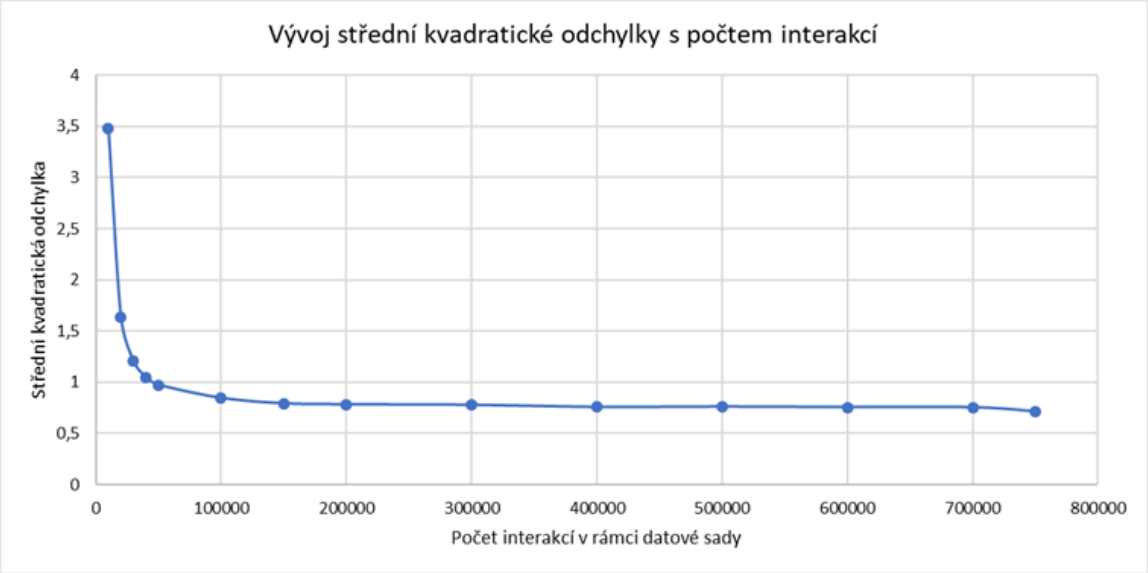
5.3 Parametry modelu

K vytrénování modelu doporučovacího systému je využita datová sada z ČSFD, o které se lze dočíst v sekci 3.4. V průběhu testování přesnosti doporučování se upravovaly parametry modelu tak, aby byla výsledná střední kvadratická odchylka co nejmenší. Výsledné hodnoty

těchto parametrů, které jsou uvedeny v následujících sekcích, jsou ideální pro případ užití a datovou sadu v rámci této práce, což nemusí znamenat, že jsou tyto hodnoty vhodné univerzálně pro různé modely.

Velikost datové sady

Faktor, který dokáže snad nejvíce ovlivnit přesnost doporučovacího modelu, je velikost datové sady. Při malých velikostech je vyvodit z interakcí mezi uživateli a filmy určitý vzor téměř nemožné, zatímco v případě velkých datových sad se můžeme potýkat s problémy jako příliš dlouhý čas trénování a nebo přetrénování.



Obrázek 5.1: Graf vývoje střední kvadratické odchylky při změně počtu interakcí.

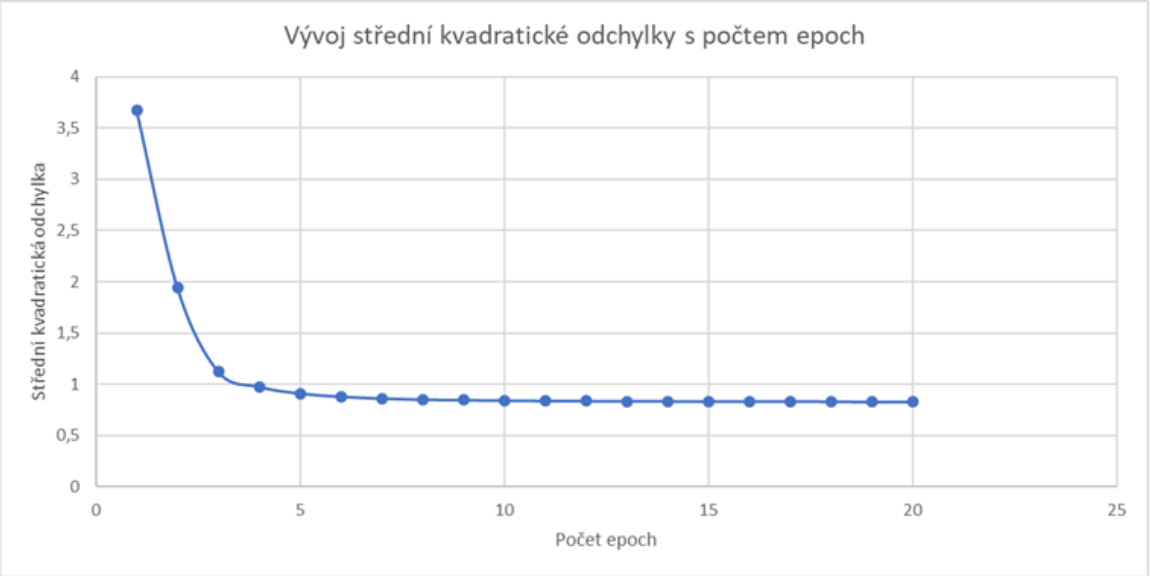
Největší vliv na přesnost doporučování modelu mělo přibližně prvních 50000 vzorků, poté se již přesnost zvětšovala pouze zanedbatelně. Při sadě o tak malém počtu prvků jako například 10000 by bylo možné přesnost vylepšit úpravou dalších zmiňovaných parametrů, zejména míry regularizace, pomocí které zabraňujeme přetrénování modelu (viz graf 5.1).

Pro výsledný model byla využita celou datová sada vydolovaná z ČSFD, tedy 732483 vzorků. Jelikož se model trénoval poměrně rychle, nebylo nutné vzorky snižovat. Zároveň mi však nepřišlo jako klíčové vzorky zvyšovat, jelikož změny v kvadratické odchylce při doporučování byly nad určitou hranici minimální.

Epocha

Epocha určuje počet průchodů celé datové sady algoritmem. Jakmile algoritmus zpracuje všechny vzorky z datové sady, tak je epocha u konce.

Při volbě malé epochy není model dostatečně přesný, v případě že zvolíme parametr epochy naopak příliš velký, tak by v případě volby nevhodného parametru regularizace mohlo dojít k přetrénování. Vhodný počet průchodů se v tomto případě pohybuje v rozsahu od 10 do 15 (viz graf 5.2).



Obrázek 5.2: Graf vývoje střední kvadratické odchylky při změně počtu epoch.

Nakonec bylo zvoleno 10 průchodů, jelikož kratší doba snižovala přesnost modelu a vyšší hodnota zvyšovala přesnost modelu pouze minimálně, přičemž výrazně prodlužovala dobu trénování.

Objem várky

Objem várky (batch size) je celkový počet trénovacích vzorů v jedné várce. Jelikož nelze za využití rozumného množství systémových prostředků vložit do modelu celou datovou sadu najednou, je zapotřebí ji rozdělit na menší části zvané várky. Várka pak reprezentuje zlomek celé datové sady, která je zpracovávaná modelem ve stejnou dobu.

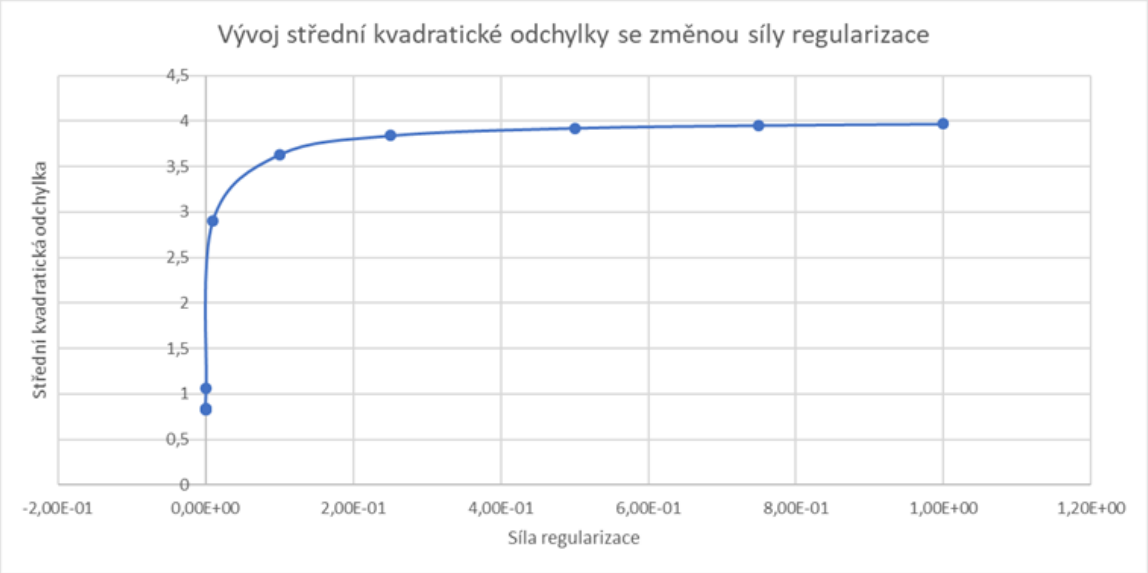
Obrázek 5.3: Graf vývoje střední kvadratické odchylky při změně objemu várky.

Z grafu na obrázku 5.3 lze vypozorovat, že při volbě velkého množství objemu várky docházelo k markantnímu zvýšení kvadratické odchylky. Důvodem je fakt, že se váhová ohodnocení upravují po zpracování každé várky, a protože při velkém objemu várky váhová ohodnocení upravila řádově až deseti-tisíckrát méně než při malém objemu, tak se přesnost modelu výrazně zhoršila.

Po tomto zhodnocení se nastavil objem várky 1024, který umožňoval optimální rychlost trénování a využití paměti, přičemž dosahoval dobré přesnosti.

Regularizace

Regularizace je technika, jejímž účelem je zabraňovat přetrénování modelu tak, že penalizuje ztrátovou funkci. Existují dva druhy regularizace. Prvním druhem je L1 regularizace, při které se většina vah rysů vstupu vynuluje. Druhá je L2 regularizace, která funguje tak, že vynutí u vah malé hodnoty a proto nejsou váhy rysů úplně zanedbány. Z tohoto důvodu je L2 regularizace efektivnější v případě, že všechny ze vstupních rysů ovlivňují výstup. [25]



Obrázek 5.4: Graf vývoje střední kvadratické odchylky při změně síly regularizace.

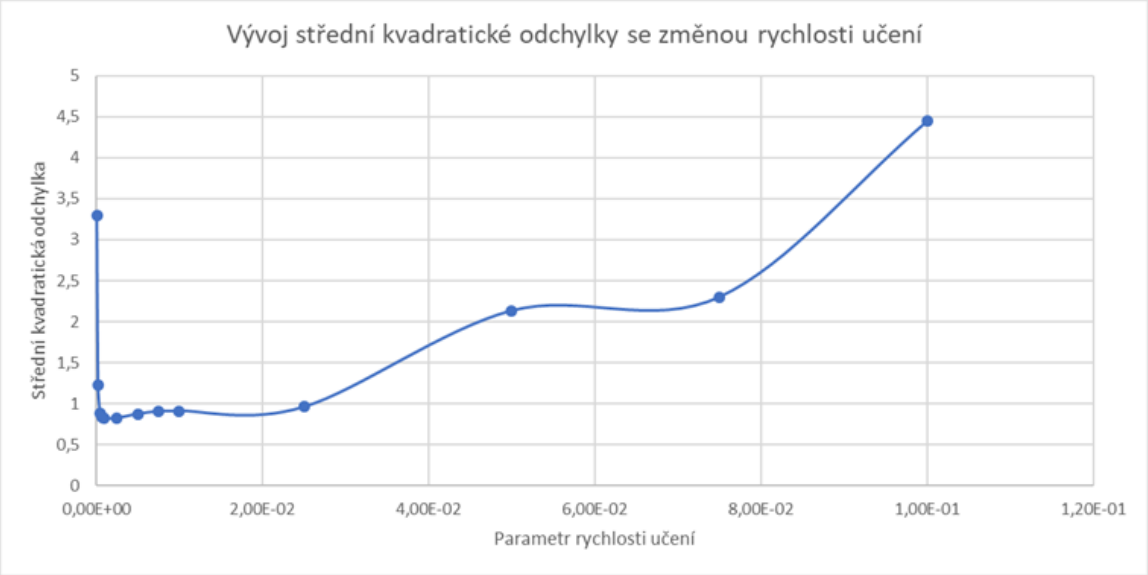
V případě, že je zvolená síla regularizace příliš nízká, tak dochází k velmi brzkému přetrénování modelu. Na druhou stranu při volbě příliš vysoké hodnoty není model vůbec přesný, jelikož je datová sada příliš zobecněna. Ideální hodnota síly regularizace se odvíjí od počtu prvku v datové sadě. V případě, že datová sada obsahuje málo dat, tak riziko přetrénování není tak vysoké a pro tento parametr lze zvolit vyšší hodnotu. V opačném případě riziko přetrénování narůstá, a proto je zapotřebí zvolit menší hodnotu.

Jelikož je datová sada zpracovávaná modelem poměrně velká a při volbě příliš vysoké síly regularizace docházelo k přetrénování (viz graf 5.4), tak byla zadaná hodnota $1 * 10^{-9}$.

Rychlost učení

Rychlost učení určuje rozsah změn, které se provedou při každém kroku vyhledávání řešení optimalizační úlohy. Jedná se o asi nejvíce podstatný parametr, který má obrovský vliv na přesnost doporučování. Během trénování je při zpětné propagaci chyby proveden odhad

toho, do jaké míry je váhové ohodnocení daného uzlu za tuto chybu zodpovědné (viz graf vývoje 5.5). [6]



Obrázek 5.5: Graf vývoje střední kvadratické odchylky při změně parametru rychlosti učení.

Obecně platí, že při výběru příliš velké hodnoty parametru učení bude model sice vytrénován rychle, ale s neoptimálním váhovým ohodnocením jednotlivých uzlů v rámci neuronové sítě. Na druhou stranu v případě volby menšího parametru učení lze dojít k optimálnějším váhovým ohodnocením, které však potrvá delší dobu získat. Konkrétní hodnota se pak odvíjí od toho, v jakém rozsahu jsou hodnoty jednotlivých interakcí. U této práce této práce se jedná o rozsah od 0 do 5 odpovídající stupnici hodnocení na ČSFD, pokud by byl rozsah jiný, musel by se tento parametr také upravit.

Serializace modelu

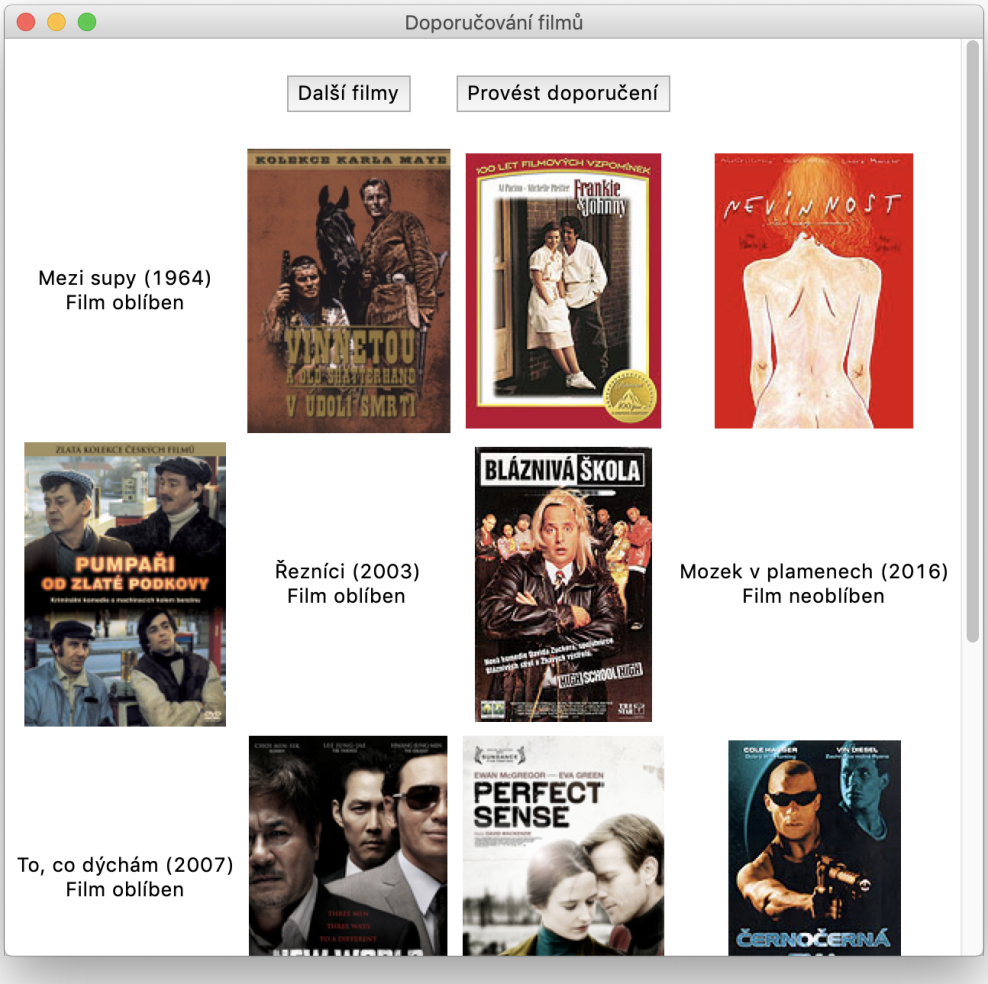
Pro provedení doporučení musí mít doporučovací systém k dispozici model, který jako vstup obdrží sadu hodnocení uživatele, pro kterého se má doporučení provést, a jako výstup poskytne uživateli filmy, které by ho mohly zajímat. Tento model však musí být již vytrénovaný, protože v případě, že by se model měl trénovat z datové sady pokaždé, když chce uživatel něco doporučit, tak by to trvalo velmi dlouhou dobu. Na druhou stranu v případě, že by model vytrénovaný nebyl, tak by neposkytoval žádná smysluplná doporučení. Proto je třeba model vytrénovat jednou a poté serializovat do formátu, kterému aplikace rozumí. Je k tomu využívána funkcionalita knihovny Spotlight.

Trénování modelu z kompletní datové sady se tedy nikdy neděje za běhu aplikace, nýbrž je k aplikaci přibalen mezi zdrojovými soubory už předem připravený model vycházející z datové sady ČSFD, který je následně aplikací pouze načten. Z tohoto důvodu nelze provádět doporučení pro filmy, které nejsou zahrnuty v této datové sadě.

K trénování modelu však dochází po poskytnutí explicitních dat uživatelem při spuštění aplikace, nikoliv však z celé datové sady, ale pouze z těch nově získaných. Děje se tak pro to, aby byl model schopen vytvořit latentní faktorovou reprezentaci uživatele pro bilineární síť, na které je založen. V opačném případě by nebylo možné zahrnout uživatelem poskytnutá data do doporučovacího procesu.

5.4 Uživatelské rozhraní

Aby bylo možné od uživatelů získat explicitní data, na základě kterých by model poté uživateli doporučil film, tak bylo zapotřebí do aplikace zahrnout grafické rozhraní. Toto rozhraní slouží uživateli k výběru několika filmů, které již viděl a hodnotil kladně nebo záporně, přičemž platí, že čím více filmů uživatel před začátkem doporučování vybere, tím relevantnější poté doporučení bude. Aby doporučovací systém nepoužíval naprosto náhodné filmy, musí uživatel vybrat alespoň dva filmy, protože například v případě, kdy by uživatel vybral při spuštění aplikace pouze jeden film, tak důsledkem toho, že model o uživateli nemá dostatek informací, bude doporučovat převážně položky, které jsou ostatními uživateli hodnoceny kladně a zároveň jsou populární.



Obrázek 5.6: Snímek grafického rozhraní aplikace pro doporučování filmů.

Pro implementaci grafického rozhraní byl využit balíček *TkInter*[1]. Rozhraní, které lze vidět na obrázku 5.6, je velmi jednoduché a skládá se pouze z hlavního okna, ve kterém jsou jako seznam mřížek s plakáty prezentovány náhodně seřazené filmy z množiny všech 4350 dostupných v rámci datové sady. Pokud chce uživatel poskytnout zpětnou vazbu i pro další filmy, je to možné prostřednictvím tlačítka, které obnoví seznam, přičemž předchozí informace o zhlédnutích zůstávají uloženy.

Doporučení filmů

Po výběru zhlédnutých filmů jsou tyto informace zpracovány předány modelu, který se poté na nově získaných datech dotrénuje. Jelikož je model již serializován a nemusí se trénovat na celé datové sadě, nýbrž pouze na uživatelově vstupu při spuštění, tak doporučení proběhne téměř okamžitě a uživateli je poskytnut seznam 16 filmů, které by jej mohly zajímat.

V případě, že uživatel některé z doporučených filmů již viděl, může opět zvolit, zdali je hodnotil kladně, či záporně, a znovu kliknout na odpovídající tlačítko pro dotrénování modelu a poskytnutých nových doporučení.

[1] https://wiki.python.org/moin/TkInter

Kapitola 6

Zhodnocení a porovnání s existujícími řešeními

Jelikož objektivní přesnost doporučovacího systému na základě hodnoty střední kvadratické odchylky (viz sekce 5.2) nemusí nutně vypovídat o relevantnosti doporučení položek uživatelům a také proto, že to vyžaduje zadání, bylo osloveno několik osob, zdali jsou s výsledkem doporučení spokojeni. O výsledcích tohoto dotazování je možno zjistit více v sekci 6.2.

6.1 Přesnost modelu

Výsledná střední kvadratická odchylka pro serializovaný model využívaný aplikací je **0,764** hvězdiček. Tato odchylka je odvozena od rozdílu mezi předpovězeným hodnocením položky daným uživatelem a skutečným hodnocením položky. Finální konfigurace modelu a jeho parametrů, která k této hodnotě RMSE vedla, je popsána v sekci 5.3.

Z této hodnoty vyplývá, že by měla aplikace teoreticky býti schopna doporučit uživateli film hodnocený na stupnici od 0 do 5 hvězdiček (stupnice hodnocení na ČSFD) s odchylkou necelé jedné hvězdičky. Přesnost doporučování se dle této hodnoty může zdát poměrně vysoká, bylo však nutné tuto přesnost prověřit na skutečných uživatelích.

6.2 Zpětná vazba uživatelů

Přestože je objektivní přesnost modelu poměrně vysoká, nemusí to nutně znamenat, že model poskytuje smysluplná doporučení. Pro zhodnocení, zdali aplikace opravdu doporučuje relevantní filmy, byla vybrána testovací sada uživatelů.

Celkem bylo osloveno 13 testovacích subjektů, kterým byla poskytnuta aplikace pro doporučování. Každý subjekt před samotným procesem doporučení při spuštění aplikace vybral průměrně 7 filmů, které již v minulosti zhlédnul a hodnotil kladně.

Na základě tohoto výběru bylo poté uživateli doporučeno 16 filmů, seřazené od nejvyššího předpokládaného hodnocení po nejnižší. Každý z testovacích subjektů pak dostal za úkol podívat se a ohodnotit libovolný z doporučených filmů v rozmezí od 0 do 5 hvězdiček. Na základě zpětné vazby poskytnuté uživatelem se poté provedl rozdíl mezi hodnocením získaným zpětnou vazbou a hodnocením, které předpověděl pro danou interakci uživatele a filmu doporučovací model. Průměr těchto rozdílů po zahrnutí hodnocení všech dotazovaných uživatelů byl roven **1,065** hvězdiček.

Rozdělení testovacích subjektů podle odchylky jejich hodnocení a hodnocení systému lze pozorovat na grafu 6.1 (odchylku 4 a 5 hvězdiček nezvolil žádný z uživatelů).

Obrázek 6.1: Graf rozdělení testovacích subjektů podle stanovené odchylky hodnocení.

Tento výsledek potvrzuje skutečnost, že přesnost modelu odvozená z metrik jako je střední kvadratická odchylka zpravidla přesně neodpovídá míře relevance doporučení, které takový model poskytuje. Daleko lepším způsobem pro testování kvality doporučení je například A/B testování, zmíněné v kapitole 4.3.2. Pro to by však byla nutná větší testovací sada uživatelů.

6.3 Srovnání s existujícími řešeními

Základem každého doporučovacího systému je datová sada, ze které jsou doporučení odvozována. Firmy, které se na trhu pohybují už několik let a shromažďují jak explicitní, tak implicitní data o uživatelích, mají k dispozici datové sady obrovských rozměrů, které jsou mnohonásobně větší a obsahují více metadat, než mnou získaná datová sada. V důsledku trénování modelů na těchto datech jsou modely užívané například firmou Netflix daleko přesnější a efektivnější, než model, který využívám v této aplikaci.

Mimo jiné YouTube ani Netflix neposkytuje nezávislý systém, který by filmy či videa doporučoval samostatně. Jedná se pouze o část (i když nedílnou) jejich online streamovací platformy. Získávání implicitních dat je pro tyto systémy jednoduché, protože uživatelé interagují s doporučovaným obsahem přímo v rámci aplikace. Je tedy možné například shromažďovat informace o tom, které filmy uživatel vyhledával, který žánr filmů má v poslední době nejraději a podobně.

Kapitola 7

Závěr

V této práci je popsáno široké spektrum postupů, pomocí nichž je možné sestrojit funkční doporučovací systém. Patří mezi ně statistické metody jako je faktorizace matic, ale i složitější přístupy, které využívají strojové učení s pomocí neuronových sítí. Rovněž zde byly analyzovány nejpoužívanější druhy filtrování využívané těmito systémy, konkrétně filtrování založené na obsahu a mezi uživateli, které odrážejí, jakým způsobem jsou vnímány vztahy mezi uživateli a položkami při trénování modelu. Také zde byly rozebrány problémy související s trénováním takového modelu, jako je získání vhodné datové sady a nebo postupy vedoucí k maximalizaci přesnosti doporučování, které odráží jemné úpravy parametrů trénovaného modelu.

Cílem práce bylo shromáždit data z Česko-Slovenské filmové databáze a na základě těchto dat navrhnout a implementovat doporučovací systém pro doporučování filmů. Výsledná datová sada v sobě obsahovala informace o 4350 filmech, které hodnotilo 1479 uživatelů napříč 732483 interakcemi. Střední kvadratická odchylka se při využití této datové sady a pečlivých změnách v parametrech modelu rovnala 0,764. To znamená, že je teoreticky doporučovací systém schopný předpovědět hodnocení, které uživatel filmu udělí, s odchylkou necelé jedné hvězdičky.

Poměrně vysoká přesnost modelu se projevila také při zpětné odezvě uživatelů na doporučení. Přestože nebyl model tak přesný, jak indikovala metrika, byl výsledný průměrný rozdíl mezi předpovězenými a skutečnými hodnoceními 1,065. Konkrétně se v 11 z 13 případů doporučený film testovacímu subjektu skutečně líbil. Přisuzuji tento úspěch také faktu, že model nezohledňoval pouze hodnocení filmu, ale zároveň jeho popularitu. Došel jsem také k závěru, že při úpravách parametrů modelu by bylo daleko efektivnější vycházet z výsledků testování na uživatelích (například A/B testování), namísto statistických metrik.

V případě budoucího vývoje aplikace by bylo zajímavé experimentovat s jinými druhy modelů pro doporučování, například sekvenčními, které při trénování zohledňují, že se vkus uživatele může s časem měnit. Možné by bylo například také rozšířit datovou sadu, domnívám se však, že zpozorovat znatelnou změnu v přesnosti doporučení by bylo pravděpodobné až při vyšších desítkách miliónů interakcí.

Literatura

[1] Abadi, M.; Barham, P.; Chen, M.; aj.: TensorFlow: A system for large-scale machine learning. In *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, 2016, s. 265–283.

[2] Amatriain, X.; Basilico, J.: Recommender systems in industry: A netflix case study. In *Recommender systems handbook*, Springer, 2015, s. 385–419.

[3] Bramer, M.: *Principles of Data Mining*, ročník 180. Springer, 2007.

[4] Brownlee, J.: *A Gentle Introduction to Singular-Value Decomposition for Machine Learning*. [Online; navštíveno 14.12.2018].
URL https://machinelearningmastery.com/singular-value-decomposition-for-machine-learning/

[5] Brownlee, J.: *A Gentle Introduction to the Rectified Linear Unit (ReLU) for Deep Learning Neural Networks*. [Online; navštíveno 11.4.2019].
URL https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/

[6] Brownlee, J.: *How to Configure the Learning Rate Hyperparameter When Training Deep Learning Neural Networks*. [Online; navštíveno 29.1.2019].
URL https://machinelearningmastery.com/learning-rate-for-deep-learning-neural-networks/

[7] Calderon, P.: *An Overview of Recommendation Systems*. [Online; navštíveno 12.3.2019].
URL http://datameetsmedia.com/an-overview-of-recommendation-systems/

[8] Chakrabarti, S.; Ester, M.; Fayyad, U.; aj.: Data Mining Curriculum: A proposal. *Intensive Working Group of ACM SIGKDD Curriculum Committee*, ročník 140, 2006.

[9] Chopra, P.: *The Ultimate Guide To A/B Testing*. [Online; navštíveno 4.5.2019].
URL https://www.smashingmagazine.com/2010/06/the-ultimate-guide-to-a-b-testing/

[10] Collobert, R.; Bengio, S.; Mariéthoz, J.: Torch: A modular machine learning software library. Technická zpráva, Idiap, 2002.

[11] Covington, P.; Adams, J.; Sargin, E.: Deep Neural Networks for YouTube Recommendations. In *Proceedings of the 10th ACM conference on recommender systems*, ACM, 2016, s. 191–198.

[12] Davidson, J.; Liebald, B.; Liu; aj.: The YouTube video recommendation system. In *Proceedings of the fourth ACM conference on Recommender systems*, ACM, 2010, s. 293–296.

[13] Ekstrand, M. D.; Riedl, J. T.; Konstan, J. A.; aj.: Collaborative filtering recommender systems. *Foundations and Trends® in Human–Computer Interaction*, ročník 4, č. 2, 2011.

[14] Fang, Z.; Zhang, L.; Chen, K.: Hybrid Recommender System Based on Personal Behavior Mining. *arXiv preprint arXiv:1607.02754*, 2016.

[15] Fayyad, U. M.; Piatetsky-Shapiro, G.; Smyth, P.; aj.: *Advances in Knowledge Discovery and Data Mining*. the MIT Press, 1996.

[16] Foley, B.: *What is Regression Analysis and Why Should I Use It?* [Online; navštíveno 16.4.2019].
URL https://www.surveygizmo.com/resources/blog/regression-analysis/

[17] Girija, S. S.: Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *Software available from tensorflow.org*, 2016.

[18] Gomez-Uribe, H. N., Carlos A: The Netflix Recommender System: Algorithms, Business Value, and Innovation. *ACM Transactions on Management Information Systems (TMIS)*, ročník 6, č. 4, 2016: str. 13.

[19] Goodfellow, I.; Bengio, Y.; Courville, A.: *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[20] Grover, P.: *Various Implementations of Collaborative Filtering*. [Online; navštíveno 18.11.2018].
URL https://towardsdatascience.com/various-implementations-of-collaborative-filtering-100385c6dfe0

[21] Han, J.; Pei, J.; Kamber, M.: *Data mining: concepts and techniques*. Elsevier, 2011.

[22] Haykin, S.: *Neural networks: a comprehensive foundation*. Prentice Hall PTR, 1994.

[23] Hinton, D. L., Geoffrey; další: Deep neural networks for acoustic modeling in speech recognition. *IEEE Signal processing magazine*, ročník 29, 2012.

[24] Huang, S.: *Introduction to Recommender System. Part 1 (Collaborative Filtering, Singular Value Decomposition)*. [Online; navštíveno 28.11.2018].
URL https://hackernoon.com/introduction-to-recommender-system-part-1-collaborative-filtering-singular-value-decomposition-44c9659c5e75

[25] Khandelwal, R.: *L1 and L2 Regularization*. [Online; navštíveno 18.1.2019].
URL
https://medium.com/datadriveninvestor/l1-l2-regularization-7f1b4fe948f2

[26] Kirk, J.: *TensorRec: A Recommendation Engine Framework in TensorFlow*. [Online; navštíveno 12.4.2019].
URL https://hackernoon.com/tensorrec-a-recommendation-engine-framework-in-tensorflow-d85e4f0874e8

[27] Koren, B. R. V., Yehuda; Chris: Matrix Factorization Techniques for Recommender Systems. *Computer*, ročník 42, č. 8, 2009: s. 30–37.

[28] Kröse, S. P., Ben: *An introduction to neural networks*. Citeseer, 1993.

[29] Kula, M.: Metadata Embeddings for User and Item Cold-start Recommendations. In *Proceedings of the 2nd Workshop on New Trends on Content-Based Recommender Systems co-located with 9th ACM Conference on Recommender Systems (RecSys 2015), Vienna, Austria, September 16-20, 2015.*, *CEUR Workshop Proceedings*, ročník 1448, editace T. Bogers; M. Koolen, CEUR-WS.org, 2015, s. 14–21.
URL http://ceur-ws.org/Vol-1448/paper4.pdf

[30] Kula, M.: Spotlight. https://github.com/maciejkula/spotlight, 2017.

[31] Li, S. W. J., Jingjiao: A Slope One collaborative filtering recommendation algorithm using uncertain neighbors optimizing. In *International Conference on Web-Age Information Management*, Springer, 2011, s. 160–166.

[32] Melville, P.; Sindhwani, V.: Recommender Systems. *Encyclopedia of Machine Learning and Data Mining*, 2017: s. 1056–1066.

[33] Pedregosa, F.; Varoquaux, G.; Gramfort, V.; aj.: Scikit-learn: Machine learning in Python. *Journal of machine learning research*, ročník 12, č. Oct, 2011: s. 2825–2830.

[34] Raschka, S.: *Logistic Regression: Why sigmoid function?* [Online; navštíveno 4.4.2019].
URL https://sebastianraschka.com/faq/docs/logistic-why-sigmoid.html

[35] Reshef, R.: *How do you build a "People who bought this also bought that"-style recommendation engine*. [Online; navštíveno 18.12.2018].
URL https://datasciencemadesimpler.wordpress.com/tag/alternating-least-squares/

[36] Ricci, F.; Rokach, L.; Shapira, B.: *Recommender systems: introduction and challenges*. Springer, 2015.

[37] Schafer, J. B.; Frankowski, D.; Herlocker, J.; aj.: *Collaborative filtering recommender systems*. Springer, 2007.

[38] Shetty, S.: *What is PyTorch and how does it work?* [Online; navštíveno 22.4.2019].
URL https://hub.packtpub.com/what-is-pytorch-and-how-does-it-work/

[39] Trevino, A.: *Introduction to K-means Clustering*. [Online; navštíveno 7.1.2019].
URL https://www.datascience.com/blog/k-means-clustering

[40] Uberoi, A.: *Introduction to Dimensionality Reduction*. [Online; navštíveno 3.3.2019].
URL https://www.geeksforgeeks.org/dimensionality-reduction/

[41] Van Der Maaten, L.; Postma, E.; Van den Herik, J.: Dimensionality reduction: a comparative. *J Mach Learn Res*, ročník 10, 2009: s. 66–71.

[42] Walia, A. S.: *Activation functions and it's types-Which is better?* [Online; navštíveno 14.3.2019].
URL https://towardsdatascience.com/activation-functions-and-its-types-which-is-better-a9a5310cc8f

Příloha A

Obsah CD

Na přiloženém CD je možné najít tyto materiály:

- **Zdrojové soubory programu**: `CSFDRecommenderSystem/src`
- **Exportovaná datová sada z MySQL**: `CSFDRecommenderSystem/dataset`
- **Zkompilovaná aplikace**: `CSFDRecommenderSystem/build`
- **Zdrojové soubory technické zprávy**: `CSFDRecommenderSystem/latex`
- **Zkompilovaná technická zpráva**: `CSFDRecommenderSystem/thesis`

Příloha B

Manuál

Zkompilovaná aplikace je obsažena ve složce `build`. Pro správnou funkci aplikace je nutné, aby se spolu ve složce s aplikací nacházel soubor `movies.tsv` obsahující informace o filmech. V opačném případě by bylo možné filmy prezentovat pouze na základě identifikátoru, což by nebylo moc užitečné. Dále v případě, že se nebude ve složce s aplikací nacházet soubor `model.dat` obsahující serializovaný model, tak proces doporučování potrvá o dost déle, jelikož bude nejprve zapotřebí model vytrénovat. Avšak aby bylo možné model takovýmto způsobem trénovat, musí se ve složce s aplikací nacházet také soubor `interactions.txt` obsahující seznam interakcí mezi uživateli a filmy v rámci datové sady.

Spuštění ze zdrojových souborů

Pokud není předem zkompilovaná aplikace spustitelná na uživatelově platformě, je zapotřebí ji spustit ze zdrojových souborů. K tomu je zapotřebí nainstalovat verzi *Pythonu* kompatibilní s verzí `3.6` a mít k dispozici následující knihovny:

- *Spotlight* - implementuje doporučovací model (viz sekce 4.2)
- *requests*[1] - slouží pro stahování plakátů filmů
- *numpy*[2] - pro matematické operace s vektory
- *sklearn*[3] - implementuje strojové učení
- *Pillow*[4] - pro upravení velikosti stažených plakátů

Všechny potřebné knihovny lze nainstalovat příkazem `pip3 install -r requirements.txt` provedeným ze složky `CSFDRecommenderSystem/src`. Je možné, že budou poté pro spuštění chybět některé systémové knihovny, které jsou však odlišné pro jednotlivé systémové instalace a nelze je zde všechny vypsat. Po stažení všech závislostí lze aplikaci spustit příkazem `python3 app.py` také ze složky `CSFDRecommenderSystem/src`.

[1] https://github.com/kennethreitz/requests
[2] https://github.com/numpy/numpy
[3] https://github.com/scikit-learn/scikit-learn
[4] https://github.com/python-pillow/Pillow

Ovládání

Aby bylo možné doporučení provést, tak uživatel musí nejprve vybrat nejméně dva filmy. Tyto filmy se vybírají ze seznamu plakátu prezentovaných ve formě mřížky. Pro výběr filmu, který se uživateli líbil, se na odpovídající plakát klikne levým tlačítkem myši. Naopak pro výběr filmu, který se uživateli nelíbil, se na odpovídající plakát klikne pravým tlačítkem myši. V případě, že uživateli prezentovaný výběr filmu nestačí, tak může načíst další filmy stiskem tlačítka `Další filmy`.

Jakmile je uživatel spokojen se svým výběrem filmů, tak pro provedení doporučení musí stisknout tlačítko `Provést doporučení`. Po stisku tohoto tlačítka budou uživateli prezentovány doporučené filmy, které lze opět označit jako již zhlédnuté, a případně požádat o nová doporučení (nová doporučení lze vyžádat i bez výběru dalších filmů opětovným stiskem tlačítka `Provést doporučení`).