FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Zadání diplomové práce

Název:	Doporučovací algoritmy pro hierarchická data
Student:	Bc. Kryštof Zindulka
Vedoucí:	Ing. Tomáš Řehořek, Ph.D.
Studijní program:	Informatika
Obor / specializace:	Znalostní inženýrství
Katedra:	Katedra aplikované matematiky
Platnost zadání:	do konce letního semestru 2023/2024

Pokyny pro vypracování

Seznamte se s algorithmy kolaborativního filtrování používané v doporučovacích systémech (např. ItemKNN, UserKNN) a s metrikami vyhodnocování offline úspěšnosti doporučovacích modelů. Proveďte rešerši přístupů k doporučování hierarchicky uspořádaných objektů (např. seriál-epizoda, herec-film).

Navrhněte a implementujte úpravu standardních algoritmů tak, aby bylo možné pomocí nich doporučovat nikoli pouze jednotlivé položky, ale také segmenty, do kterých položky hierarchicky spadají. V situacích, kdy lze doporučovat smíšený obsah (filmy + seriály jakožto segmenty epizod) se zaměřte na balancování zastoupení segmentů a položek.

Dále navrhněte a naimplementujte framework na měření offline úspěšnosti těchto algoritmů a proveďte experimenty na několika dodaných průmyslových datasetech. Dosažené výsledky vhodným způsobem prezentujte a diskutujte.

Elektronicky schválil/a Ing. Magda Friedjungová, Ph.D. dne 24. února 2023 v Praze.

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Diplomová práce

Doporučovací algoritmy pro hierarchická data

Bc. Kryštof Zindulka

Katedra aplikované matematiky
Vedoucí práce: Ing. Tomáš Řehořek, Ph.D.

4. května 2023

Poděkování

Rád bych poděkoval především svému vedoucímu, Ing. Tomáši Řehořkovi, Ph.D., za čas, který mi věnoval a cenné rady, které mi poskytl. Také děkuji firmě Recombee za poskytnutá průmyslová data a poskytnutý hardware, na provádění experimentů.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů, zejména skutečnost, že České vysoké učení technické v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 citovaného zákona.

V Praze dne 4. května 2023 . . . . . .

Odkaz na tuto práci

Zindulka, Kryštof. *Doporučovací algoritmy pro hierarchická data.* Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2023.

Abstrakt

Klasické doporučovací přístupy se zabývají doporučováním položek uživatelům. Nad těmito položkami často existuje nějaká hierarchická struktura, případně více takových struktur. V této práci je představen způsob doporučování prvků hierarchické struktury, které označuje jako segmenty. Existuje několik praktických situací, kdy má smysl doporučovat prvky hierarchické struktury, například doporučování seriálů místo epizod. V práci je navržen nový způsob převodu položek na segmenty. Pro doporučování segmentů se používají upravené klasické doporučovací algoritmy. Kvalita navrženého řešení je prezentována pomocí experimentů provedených na dvou průmyslových datasetech ze streamingových platforem.

Klíčová slova doporučovací systémy, hierarchická data, doporučování segmentů, strojové učení, KNN algoritmy.

Abstract

Classic recommendation approaches consist of recommending items to users. Often there is some hierarchical structure, or multiple such structures, above these items. This paper presents a method for recommending elements of a hierarchical structure. Parts of such structure are refered to as segments. There are several practical situations where it makes sense to recommend elements of a hierarchical structure, such as recommending series instead of episodes. In this paper, a new way of converting items into segments is proposed. For recommending segments, modified classical recommendation algorithms are used. The quality of the proposed solution is presented through experiments conducted on two industrial datasets from streaming platforms.

Keywords recommender systems, hierarchical data, segment recommendation, machine learning, KNN algorithms.

Obsah

Seznam obrázků

Úvod

Doporučovací systémy jsou stále důležitějším prvkem v digitálním světě. V posledních letech se staly neodmyslitelnou součástí online nakupování, streamování hudby a videí, zpravodajství, sociálních sítí a mnoha dalších oblastí. Tyto systémy umožňují perzonalizovanou navigaci po platformě na základě chování uživatele s využitím informací o jiných uživatelích s podobnými zájmy.

Motivace

Klasické doporučovací přístupy, jako například kolaborativní filtrování, využívají často interakční data. Interakční data reprezentují interakce mezi uživateli a položkami. Na různých platformách mohou uživatelé s položkami různě interagovat. Jako interakce se označuje přidání produktu do košíku, přehrání filmu, ohodnocení příspěvku, přečtení článku a další akce, které uživatel může na dané platformě provést.

V datech existuje často určitá hierarchie. Produkty na e-shopu jsou rozděleny do kategorií, jednotlivé epizody seriálu patří do sérií a série přísluší nějakému seriálu. V praxi bývá někdy užitečné doporučit segment, který je v hierarchii výše než koncová položka. Na streamingové platformě se uživateli doporučuje seriál jako celek, nikoli jedna epizoda. Interakční data neobsahují interakce mezi uživateli a vyššími segmenty hierarchie, interakční data obsahují pouze interakce mezi uživateli a koncovými položkami.

Cíl práce

S doporučováním objektů hierarchické struktury se často nesetkáváme. Nejčastěji je tento přístup využíván na už zmíněných streamingových platformách při doporučování seriálů. Mimo seriálů existují i jiné hierarchické segmenty, které má smysl doporučovat. Doporučovací systém může v případě streamovací platformy doporučit uživateli filmový žánr, který sledují podobní uživatelé.

Podobně je možné doporučit nějakého herce a s ním filmy, ve kterých hraje. Na jiných platformách je zase možné doporučovat jiné segmenty.

Cílem této práce je upravit klasické doporučovací algoritmy tak, aby pomocí nich bylo možné doporučovat mimo jednotlivých položek i segmenty hierarchické struktury. Dále najít obecné řešení pomocí kterého bude možné doporučovat segmenty libovolné hierarchie, která nad daty existuje a zaměřit se jak na doporučování samostatných segmentů — například při doporučování herců, tak na doporučování smíšeného obsahu (položek i segmentů) — například při doporučování filmů a seriálů.

Struktura práce

Práce je rozdělena do 4 kapitol. V kapitole 1 jsou představeny teoretické základy důležité pro pochopení práce. V kapitole 1 jsou také formálně nadefinována hierarchicky uspořádaná data. V kapitole 2 je představena obecná struktura FRAMEWORKu, který implementuje algoritmy na doporučování segmentů a vyhodnocuje jejich úspěšnost. V kapitole 3 je popsána implementace navrženého FRAMEWORKu včetně použitých technologií. V kapitole 4 jsou prezentovány experimenty provedené pomocí navrženého FRAMEWORKu. Větší část kapitoly je věnována doporučování smíšeného obsahu, konkrétně filmů a seriálů, kde se diskutuje jak spolehlivost doporučení, tak poměr mezi počtem doporučených filmů a seriálů. Zbytek kapitoly je věnován doporučování samotných segmentů, konkrétně se experimentuje s doporučováním žánrů a také s doporučováním herců.

Kapitola 1

Analýza

V této kapitole jsou představeny základní pojmy a definice, které jsou v práci používány. Některé definice jsou oproti obecně používaným definicím zjednodušené, protože pro použití v této práci nemusí být tolik obecné. Ke konci kapitoly jsou formálně nadefinována hierarchicky uspořádaná data.

1.1 Kolaborativní filtrování

V této práci se využívá přístup k doporučování zvaný kolaborativní filtrování. Kolaborativní filtrování bylo poprvé představeno v článku [1]. Hlavní myšlenka kolaborativního filtrování je, že existují skupiny uživatelů, kteří se zajímají o stejné položky. Doporučovací systém se snaží uživatele zařadit do nějaké skupiny uživatelů a doporučit mu položky, se kterými interagovali uživatelé v dané skupině. Tento přístup ke kolaborativnímu filtrování je založený na uživatelské podobnosti. Dále existuje přístup založený na položkové podobnosti, kdy doporučovací systém hledá položky podobné těm, se kterými uživatel interagoval a ty mu doporučí [2].

1.2 Interakční data

Data v doporučovacích systémech se primárně skládají z uživatelů a jejich atributů, položek a jejich atributů, interakcí uživatelů a položek. Existují různé doporučovací přístupy používající různá data. Tato práce se zaměřuje na doporučování pomocí interakčních dat.

Interakční data v doporučovacích systémech jsou informace o tom, jak uživatelé interagují s obsahem na dané platformě. Tato data mohou obsahovat například informace o tom, na které položky uživatelé klikají, jak které položky uživatelé hodnotí, které položky uživatelé nakupují, na které položky se uživatelé dívají. Interakční data jsou klíčovým prvkem pro perzonalizaci doporučení v doporučovacích systémech.

Interakční data se skládají z jednotlivých interakcí mezi uživatelem a položkou. Existují různé druhy interakcí v závislosti na platformě. Na e-shopu mezi interakce patří kliknutí na položku, přidání položky do košíku, koupení položky. Na streamovací platformě mezi interakce patří shlédnutí filmu nebo nějaké jeho části, ohodnocení filmu. Interakční data se rozdělují na explicitní a implicitní. Explicitní interakce jsou interakce, které uživatel explicitně vytvoří, kupříkladu hodnocení počtem hvězdiček, nebo „like". Mezi implicitní interakce patří interakce, které platforma vyčte z uživatelského chování. V této práci se pracuje výhradně s implicitními interakcemi. Shromažďování implicitních interakcí bývá jednodušší, protože nevyžadují uživatelovu explicitní zpětnou vazbu.

1.2.1 Formální definice

V této práci se používají data ze streamovacích platforem a jako interakce se uvažuje pouze takzvaná *view portion* (zhlédnutá část) [3]. Následující značení bude zjednodušovat běžně užívané značení a bude brát v potaz pouze tento jeden druh interakcí [4].

Mějme množinu n položek $I = \{i_1, \ldots, i_n\}$ a množinu m uživatelů $U = \{u_1, \ldots, u_m\}$. Interakční data tvoří množinu jednotlivých interakcí mezi položkou a uživatelem. Mějme k interakcí $Y = \{y_1, \ldots, y_k\}$, kde $y_j = (u_j, i_j, d_j)$, jednotlivé prvky trojice mají následující význam:

- $u_j \in U$ - uživatel, který provedl interakci
- $i_j \in I$ - položka se kterou bylo interagováno
- $d_j \in (0, 1]$ - míra interakce, neboli číslo udávající jak velkou část média si uživatel přehrál

1.2.2 Interakční matice

Interakční data lze uložit do interakční matice, se kterou pak pracují doporučovací algoritmy. Bez újmy na obecnosti budeme předpokládat, že množina identifikátorů uživatelů $U = \{1, \ldots, m\}$ a množina identifikátorů položek $I = \{1, \ldots, n\}$, tedy identifikátory uživatelů i položek jsou celá čísla od 1 do počtu prvků dané množiny. Interakční matici označíme jako $\mathbf{R} \in \mathbb{R}^{|U|,|I|}$,

$$\mathbf{R}_{ui} \begin{cases} d & \text{pokud } (u, i, d) \in Y \\ 0 & \text{jinak.} \end{cases}$$

Interakční matice zaznamenává interakce, v našem případě *view portion*, do matice, kde řádky reprezentují uživatele a sloupce reprezentují položky.

1.3 Algoritmy

V této práci jsou používány dva základní algoritmy kolaborativního filtrování — **userKNN** a **itemKNN**. Jedná se o algoritmy doporučující předměty uživateli. Tyto dva algoritmy jsou v praxi velmi často používány pro jejich schopnost pracovat s velkými daty. Oba algoritmy pracují s interakční maticí. Oba algoritmy používají k výpočtu doporučení podobnost dvou vektorů. V této práci je použita cosinová podobnost, která je v doporučovacích systémech často využívána [5]. Uživatel je reprezentován jako řádek interakční matice a položka jako sloupec interakční matice.

Mějme vektory a a b délky n, kde a_i a b_i značí i-tý prvek vektoru a a b. Cosinová vzdálenost mezi a a b, se počítá následovně:

$$\mathrm{sim}(a,b) = \frac{\sum_{i=1}^{n} a_i b_i}{\sqrt{\sum_{i=1}^{n} a_i^2}\sqrt{\sum_{i=1}^{n} b_i^2}} \tag{1.1}$$

$\mathrm{sim}(a,b)$ je číslo v intervalu $[-1,1]$ a pro nezáporné vektory v intervalu $[0,1]$.

Pro nějaké dva uživatele $u, v \in U$, je cosinová podobnost mezi jejich vektory počítána jako $\mathrm{sim}(\mathbf{R}_{u:}, \mathbf{R}_{v:})$, kde $\mathbf{R}_{u:}$ značí u-tý řádek interakční matice. V následujícím značení budeme tuto podobnost zapisovat zkráceně jako $\mathrm{sim}(u,v)$. Podobně pro nějaké $i, j \in I$ budeme $\mathrm{sim}(\mathbf{R}_{:i}, \mathbf{R}_{:j})$, kde $\mathbf{R}_{:i}$ značí i-tý sloupec interakční matice, zjednodušovat jako $\mathrm{sim}(i,j)$.

Následující dva algoritmy dostanou na vstupu uživatele $u \in U$ a číslo $k \in \mathbb{N}$. Oba algoritmy pro každou položku $i \in I$ spočítají skóre. $\mathrm{score}(u,i)$ určuje relevantnost položky i pro uživatele u.

1.3.1 UserKNN

User-Based k-Nearest Neighbors, zkráceně **userKNN** je algoritmus, který využívá k výpočtu skóre k nejpodobnějších uživatelů danému uživateli.

S využitím sim funkce označíme množinu k nejpodobnějších uživatelů uživateli u, jako $\mathrm{NN}_k(u)$.

$$\begin{gathered} \mathrm{NN}_k(u) = \{v_1, \ldots, v_k\} \\ \forall s \in U \backslash \{u\} : s \notin \mathrm{NN}_k(u) \implies \forall v \in \mathrm{NN}_k(u) : \mathrm{sim}(u,s) \leq \mathrm{sim}(u,v) \end{gathered} \tag{1.2}$$

S pomocí této množiny ukážeme, jak se počítá skóre mezi uživatelem u a libovolnou položkou $i \in I$. Existuje nevážená a vážená varianta. Nevážená varianta počítá skóre následovně:

$$\mathrm{score}(u,i) = \sum_{v \in \mathrm{NN}_k(u)} \mathbf{R}_{vi} \tag{1.3}$$

Tato varianta nezohledňuje ve výpočtu podobnost k nejbližších uživatelů a uživatele u. Oproti tomu vážená varianta tuto podobnost zohledňuje a počítá skóre následovně:

$$\text{score}(u, i) = \sum_{v \in \text{NN}_k(u)} \text{sim}(u, v) \cdot \mathbf{R}_{vi} \tag{1.4}$$

1.3.2 ItemKNN

Item-Based k-Nearest Neighbors, zkráceně **itemKNN** je algoritmus, který využívá k výpočtu skóre k nejpodobnějších položek každé položce, se kterou uživatel u interagoval. Nyní si jako $\text{NN}_k(i)$ označíme množinu k nejpodobnějších položek položce i.

$$\begin{gathered} \text{NN}_k(i) = \{j_1, \ldots, j_k\} \\ \forall l \in I \backslash \{i\} : l \notin \text{NN}_k(i) \implies \forall j \in \text{NN}_k(i) : \text{sim}(i, l) \leq \text{sim}(i, j) \end{gathered} \tag{1.5}$$

Podobně jako u **userKNN** existuje nevážená a vážená varianta. Nevážená varianta počítá skóre následovně:

$$\text{score}(u, i) = \sum_{\substack{j \in I \\ i \in \text{NN}_k(j)}} \mathbf{R}_{uj} \tag{1.6}$$

Tato varianta však nezohledňuje ve výpočtu podobnosti mezi k nejbližšími položkami. Oproti tomu vážená varianta tuto podobnost zvažuje a počítá skóre následovně:

$$\text{score}(u, i) = \sum_{\substack{j \in I \\ i \in \text{NN}_k(j)}} \text{sim}(i, j) \cdot \mathbf{R}_{uj} \tag{1.7}$$

Pro **userKNN** i **itemKNN** existují i varianty, kde dochází k určitému normování skóre při výpočtu [6]. To se často využívá při práci s explicitními interakcemi, aby například velké množství špatných hodnocení neznamenalo velké skóre. My uvažujeme pouze interakce typu *view portion* a ty považujeme ve všech případech za kladné. Malá *view portion* sice nemusí znamenat kladnou interakci, ale o tom je více zmíněno v podkapitole 2.1.4.1.

1.3.3 Regularizace populárních položek

Algoritmy **userKNN** a **itemKNN** mohou mít tendenci upřednostňovat populárnější položky a doporučovat tak pouze malé množství všech možných položek. Tomu se dá zabránit tak, že se populární položky budou penalizovat pomocí parametru $\beta \in \mathbb{R}$ [4]. Pro $\beta = 0$ se žádná penalizace nekoná, dále čím vyšší beta, tím vyšší penalizace populárních položek. Záporná β je naopak penalizace nepopulárních položek. Popularitu položky určuje součet interakcí

s danou položkou v interakční matici. Při použití této regularizace bude skóre pro každou položku přepočítáno následovně:

$$\text{score}^{\beta}(u,i) = \frac{\text{score}(u,i)}{\left(\sum_{v \in U} \mathbf{R}_{vi}\right)^{\beta}} \tag{1.8}$$

1.3.4 Top-N doporučení

Představené doporučovací algoritmy spočítají pro uživatele u a každou položku i skóre. Ale nebylo zatím řečeno, které položky uživateli algoritmus doporučí. V praxi se využívá Top-N doporučení, kdy se doporučí N položek s nejvyšším skóre [7]. Jak velké N se volí pak už záleží čistě na použité platformě. E-shop může doporučovat například 5 položek při prohlížení nějaké položky. Streamovací platforma může automaticky spustit další film po dokoukání filmu, v takovém případě se N rovná 1. Pro různé praktické případy se volí různé N. Tento přístup budeme používat při vyhodnocování metrik, kterými budeme měřit vlastnosti modelu.

1.4 Metriky

Přesnost doporučení se dá vyhodnocovat offline nebo online. Online vyhodnocování se měří na modelu spuštěném přímo v produkci. Online vyhodnocování je nejlepší přístup k měření přesnosti modelu v doporučovacích systémech, ale představuje několik překážek: špatné modely poškozují uživatelskou zkušenost, je časově náročné je nasadit do produkce. Proto se modely v první řadě testují offline na historických datech.

Vyhodnocování na historických datech nemusí být vždy objektivním odhadem online výkonu. Offline vyhodnocování měří přesnost modelu na historických datech, ale nemá jak zaznamenat uživatelovu reakci na poskytnuté doporučení. Historická data mohou být velmi ovlivněna algoritmem, který v době sběru historických dat data doporučoval.

Vzhledem k tomu, že vytváříme nový model, tak budeme jeho přesnost vyhodnocovat offline na historických datech pomocí dvou metrik, které se v doporučovacích systémech využívají [8, 9]. Jedná se o metriky Recall@N a Catalog Coverage, kterou dále budeme označovat jako coverage.

Pro vyhodnocování metrik, se množina uživatelů U náhodně rozdělí na trénovací množinu U^{tr} a testovací množinu U^{te}. Pro tyto množiny musí platit

$$(U^{tr} \cup U^{te} = U) \wedge (U^{tr} \cap U^{te} = \emptyset).$$

Množinu relevantních položek pro uživatele u označíme jako

$$\text{RI}(u) = \{i \mid i \in I : \mathbf{R}_{u,i} > 0\}.$$

Jedná se o položky, se kterými uživatel u někdy interagoval.

Dále předpokládáme existenci modelu naučeném na množině uživatelů U^{tr}, který dokáže doporučit N položek nějakému uživateli na základě množiny jeho relevantních položek. Množinu těchto doporučených položek označíme jako $\mathrm{Top}(N, \mathrm{RI}(u))$.

1.4.1 Recall

Mějme množinu relevantních položek R a množinu D doporučených předmětů. Recall je poměr počtu doporučených relevantních položek a všech relevantních položek [9].

$$\mathrm{Recall} = \frac{|D \cup R|}{|R|} \tag{1.9}$$

Recall budeme měřit pro celou množinu testovacích uživatelů a využijeme u toho Top-N doporučení. Recall při použití Top-N doporučení označíme jako $\mathrm{Recall}\,@N$.

$$\mathrm{Recall}\,@N = \frac{\sum\limits_{u \in U^{te}} |\mathrm{Top}(N, \mathrm{RI}(u)) \cap \mathrm{RI}(u)|}{\sum\limits_{u \in U^{te}} |\,\mathrm{RI}(u)|} \tag{1.10}$$

Takto nastavený recall však neměří schopnost modelu doporučit uživateli položky, se kterými zatím neinteragoval. Vysokého recallu by dosáhl model, který by pouze doporučil položky, se kterými uživatel interagoval. Pro měření schopnosti modelu doporučovat uživateli nové položky se používá přístup *leave-one-out* [10].

$$\mathrm{Recall}\,@N_{LOO} = \frac{\sum\limits_{u \in U^{te}} \sum\limits_{i \in \mathrm{RI}(u)} |\mathrm{Top}(N, \mathrm{RI}(u) \backslash \{i\}) \cap \{i\}|}{\sum\limits_{u \in U^{te}} |\,\mathrm{RI}(u)|} \tag{1.11}$$

S využitím *leave-one-out* přístupu už musí umět model dobře doporučit položku bez znalosti toho, že s ní uživatel interagoval, aby dosáhl vysokého recallu.

Uživatel s velkým množstvím relevantních položek recall výrazně ovlivní oproti uživateli s malým množstvím relevantních položek. Proto ještě recall normalizujeme tak, aby každý uživatel ovlivnil recall stejným dílem.

$$\mathrm{Recall}\,@N^{n}_{LOO} = \sum_{u \in U^{te}} \frac{\sum\limits_{i \in \mathrm{RI}(u)} |\mathrm{Top}(N, \mathrm{RI}(u) \backslash \{i\}) \cap \{i\}|}{|\,\mathrm{RI}(u)|} \tag{1.12}$$

V této práci se bude recall počítat vždy posledním představeným způsobem a budeme ho označovat pouze jako *recall*.

1.4.2 Coverage

Mějme množinu všech položek I a dále množinu všech doporučených položek po jednom nebo více doporučeních $I_p \subseteq I$. Coverage je poměr počtu doporučených položek a všech položek.

$$\text{Coverage} = \frac{|I_p|}{|I|} \tag{1.13}$$

Coverage neměří spolehlivost modelu, ale poskytuje užitečnou informaci o tom, jak velkou část všech položek model doporučuje. Coverage můžeme počítat také pomocí Top-N doporučení.

$$\text{Coverage}\,@N = \frac{\left|\bigcup_{u \in U^{te}} \text{Top}(N, \text{RI}(u))\right|}{|I|} \tag{1.14}$$

U měření coverage lze také využít *leave-one-out* přístupu. Není to tolik zapotřebí jako u recallu, protože u coverage se neměří žádná přesnost. Lze toho však využít, pokud se v algoritmu počítá recall a coverage zároveň.

$$\text{Coverage}\,@N_{LOO} = \frac{\left|\bigcup_{\substack{u \in U^{te} \\ i \in \text{RI}(u)}} \text{Top}(N, \text{RI}(u) \backslash \{i\})\right|}{|I|} \tag{1.15}$$

Tímto posledním představeným způsobem se bude měřit coverage v této práci a dále tento způsob budeme označovat pouze jako *coverage*.

1.5 Hierarchicky uspořádaná data

V této podkapitole jsou nadefinována hierarchicky uspořádaná data. Hierarchicky uspořádaná data jsou hlavním tématem této práce a jedná se o málo prozkoumanou oblast doporučovacích systémů. Následující definice nejsou převzaté, jsou nově zavedeny přímo pro účely této práce.

1.5.1 Definice

Nad množinou položek může existovat nějaké hierarchické uspořádání. Dokonce může existovat několik různých hierarchických uspořádání nad stejnými položkami. Například nad položkami na streamovací platformě může existovat uspořádání, ve kterém spadají epizody do sérií a série do seriálů. Dále nad stejnými položkami může existovat uspořádání přiřazující jednotlivým položkám herce nebo žánry.

Mějme posloupnost n položek $I = \{i_1, \ldots, i_n\}$. Nad položkami může být nadefinováno h různých hierarchických uspořádání $H = \{H_1, \ldots, H_h\}$, $h \in \mathbb{N}$.

$$\begin{aligned} \forall H_j \in H : H_j &= (S, E) \\ S &= \{s_1, \ldots, s_p\},\ p \in \mathbb{N} \\ E &= \{(k, l) \mid k \in I \cup S,\ l \in S\}, \end{aligned} \tag{1.16}$$

neboli každé hierarchické uspořádání je dvojice, kde první prvek je množina segmentů a druhý je množina uspořádaných dvojic, kde první prvek je položka nebo segment a druhý je segment. Tím se definuje, pod jaký segment položka spadá. Položka může spadat pod více segmentů, nebo pod žádný segment. Položka nemůže spadat pod jinou položku.

1.5.2 Doporučování segmentů

Při doporučování segmentů v nějakém hierarchickém uspořádání $H_j \in H$ můžeme doporučovat celou množinu segmentů S, nebo nějakou její podmnožinu $S^r \subset S$. Tuto podmnožinu budeme označovat jako doporučované segmenty. Doporučovat celou množinu S dává smysl například v případě herců, kde každý segment je nějaký herec. Doporučovat podmnožinu S^r dává smysl v případě seriálů, kde chceme doporučovat filmy a seriály, ale ne série nebo epizody. Když vybereme nějakou podmnožinu S^r, kterou chceme doporučovat (například podmnožinu filmů a seriálů), tak musíme zjistit, jaké položky spadají pod jaký segment. Následující značení uvažuje již jednu zvolenou hierarchii $H_j \in H$, která se bude doporučovat.

$$\begin{aligned} \forall i \in I, s \in S^r : i \to s \iff \exists (k_1, l_1), \ldots, (k_p, l_p) \in E : \\ k_1 = i \wedge l_p = s \wedge \forall a \in 2, \ldots, p : l_{a-1} = k_a, \end{aligned} \tag{1.17}$$

kde symbol $\to$ značí, že položka i spadá pod segment s. Neboli položka spadá pod segment právě tehdy, když existuje posloupnost orientovaných hran začínající v položce, končící v segmentu a navazující na sebe.

Dále definujeme funkce $\mathrm{M} : S \to \mathcal{P}(I)$ a $\mathrm{N} : I \to \mathcal{P}(S)$, kde $\mathcal{P}$ značí potenční množinu.

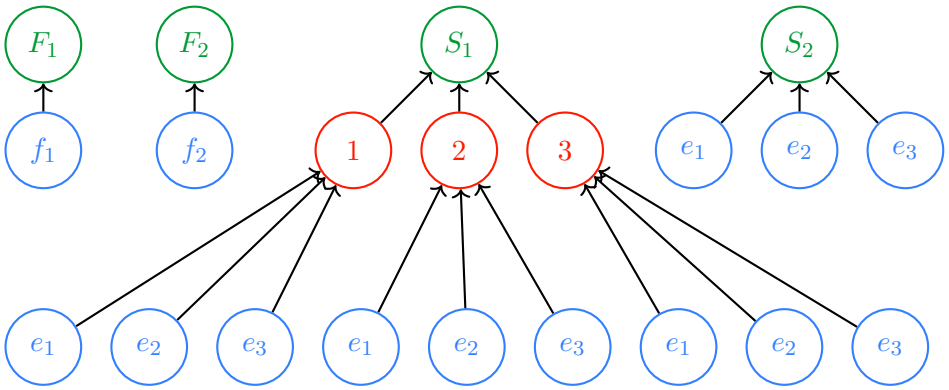
$$\begin{aligned} \forall s \in S^r : \mathrm{M}(s) &= \{i \mid i \in I,\ i \to s\}, \\ \forall i \in I : \mathrm{N}(i) &= \{s \mid s \in S^r,\ i \to s\}, \end{aligned} \tag{1.18}$$

funkce M tedy pro nějaký segment vrací všechny položky, které pod daný segment spadají a funkce N naopak vrací pro nějakou položku všechny segmenty, pod které položka spadá.

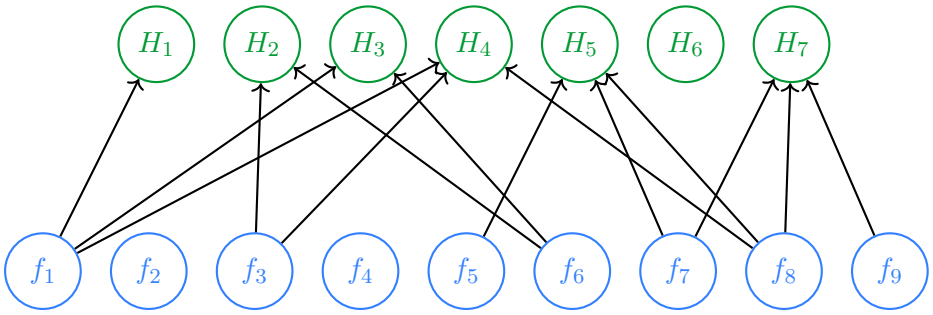
Při doporučování se doporučují pouze segmenty. Při doporučování smíšeného obsahu je pro každou položku, která se bude doporučovat, vytvořen vlastní segment, pod který spadá pouze daná položka.

Každé hierarchické uspořádání lze reprezentovat pomocí orientovaného grafu. Na obrázku 1.1 je znázorněno hierarchické uspořádání filmů, epizod, sérií a seriálů. Pro každý film byl vytvořen samostatný segment. Zeleně jsou zobrazeny doporučované segmenty — filmy a seriály. Červeně jsou zobrazeny segmenty příslušející sériím. Modře jsou zobrazeny položky — filmy a epizody.

Na obrázku 1.2 je znázorněno hierarchické uspořádání mezi filmy a herci, kteří v nich hrají. Zeleně jsou zobrazeny segmenty — herci a modře položky — filmy.



Obrázek 1.1: Znázornění hierarchické struktury definující vztahy mezi epizodami, sériemi, seriály a filmy.



Obrázek 1.2: Znázornění hierarchické struktury definující vztahy filmy a herci, kteří v nich hrají.

1.5.3 Související literatura

Téma doporučování hierarchicky uspořádaných dat není příliš prozkoumané. Samotná hierarchie v datech se někdy využívá pro zlepšení predikce.

Článek [11] se zabývá použitím hierarchie v datech pro zlepšení přesnosti doporučení. Pracuje se vzdáleností dvou položek v hierarchii a tuto vzdálenost používá k výpočtu podobnosti dvou položek zároveň s cosinovou podobností.

Nedochází zde však k žádnému doporučování segmentů, doporučují se pouze položky.

Článek [12] se podobně jako článek [11] zabývá použitím hierarchie v datech pro zlepšení přesnosti doporučení. V případě, že data nemají explicitní hierarchii, tak ukazuje způsob, jak vytvořit implicitní hierarchii. Opět v tomto článku nedochází k doporučování segmentů.

Článek [13] ukazuje přístup k doporučování seriálů uživateli na základě filmů, které viděl. Myšlenka tohoto přístupu vychází z toho, že uživatel pravděpodobně viděl více různých filmů, než kolik viděl různých seriálů, tedy jeho interakce s filmy lépe definují jeho vkus. Jedná se o atributový model, který doporučuje seriály na základě uživatelových oblíbených žánrů. Dále se k doporučení využívá hodnocení seriálů z IMDb a preferuje se doporučování seriálů s menším množstvím epizod. Jedná se o doporučování segmentů, ale tento způsob není nějak přenositelný na jiné segmenty. Je zaměřen přímo na seriály.

K doporučování seriálů existuje velké množství různých přístupů. Některé přístupy jsou atributové modely, kde se seriál nijak neliší od filmu. Některé přístupy používají naivní způsob, kde se každá interakce s nějakou epizodou seriálu započítává jako interakce se seriálem samotným. V článcích [14, 15, 16] jsou ukázány další různé přístupy.

Kapitola 2

Návrh

V této kapitole ukážeme návrh FRAMEWORKu na doporučování položek nebo segmentů uživatelům na základě jejich historických interakcí. Ukážeme celý proces FRAMEWORKu, od načtení dat až po evaluaci metrik. FRAMEWORK umí doporučovat položky, segmenty i smíšený obsah — kombinaci položek a segmentů.

2.1 Struktura

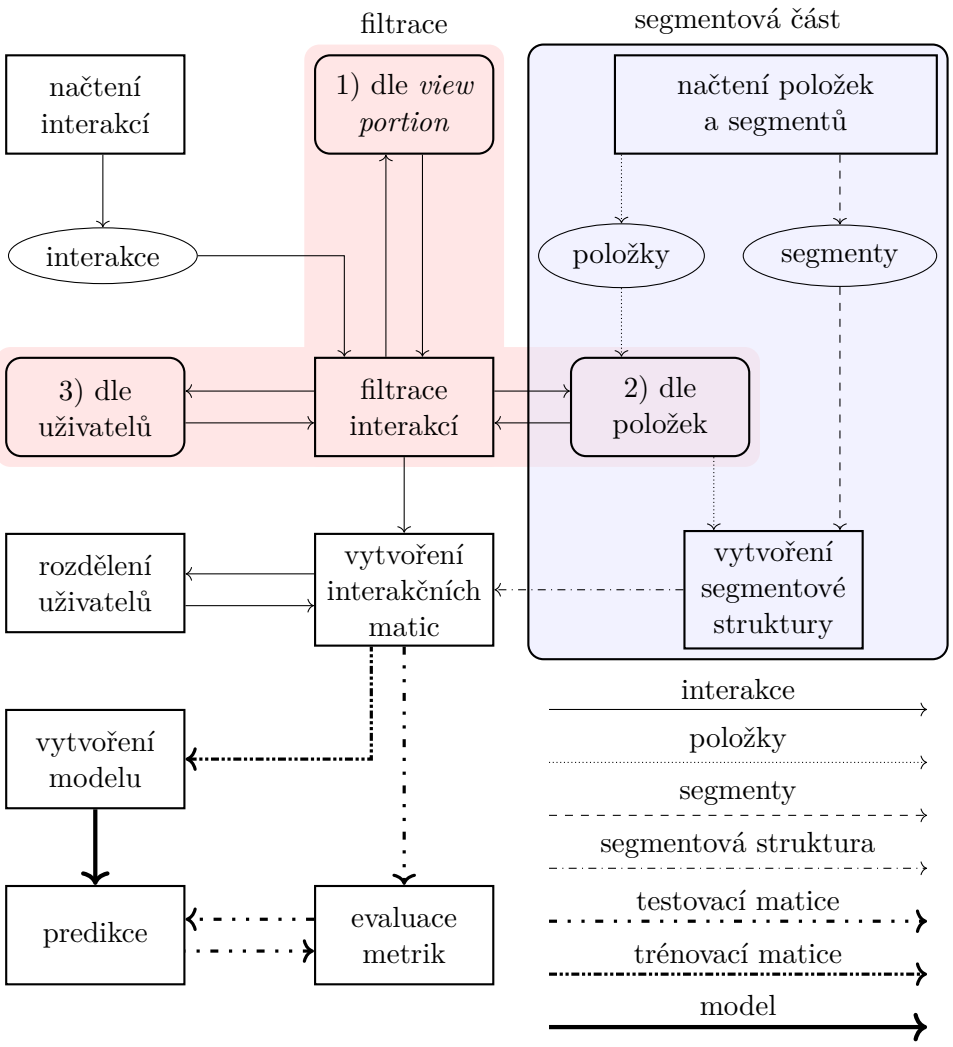
Na obrázku 2.1 je zobrazena obecná struktura FRAMEWORKu popisující každou akci, kterou FRAMEWORK vykonává. Akce v modrém boxu, pojmenovaném segmentová část, jsou vykonané při doporučování segmentů. Při doporučování pouze položek tyto akce provedeny nejsou. Pohyb různých dat mezi jednotlivými procesy je označen různými šipkami, které jsou vpravo dole vysvětleny.

2.1.1 Načtení dat

FRAMEWORK si z databáze nebo nějakého jiného úložiště načte potřebná interakční a segmentová data.

2.1.2 Načtení interakcí

FRAMEWORK načte interakce mezi uživateli a položkami. Načtena bude množina interakcí $Y' = \{y_1, \ldots, y_{k'}\}$, $k' \in \mathbb{N}$, kde $y_i = (u_j, i_j, d_j)$ pro každé $j \in \{1, \ldots, k'\}$. Jednotlivé prvky této trojice jsou popsány v kapitole 1.2.1. Množinu všech uživatelů, kteří interagovali s nějakou položkou, označíme jako $U' = \{u \mid (u, i, d) \in Y'\}$. Podobně množinu všech položek, se kterými interagoval nějaký uživatel, označíme $I' = \{i \mid (u, i, d) \in Y'\}$.



Obrázek 2.1: Struktura FRAMEWORKu.

2.1.3 Načtení položek a segmentů

FRAMEWORK načte definici segmentové struktury. Načtena bude hierarchie v podobě množiny h dvojic $H = \{H_1, \ldots, H_h\}$ popsaná v kapitole 1.5.1. V segmentových datech musí být rozlišeno co je položka a co je segment. Pro množinu segmentů musí být rozlišeno jaké segmenty se budou doporučovat. Množina položek bude označena jako I_s a množina doporučovacích segmentů bude označena jako $S^{r\prime}$. Některé položky mohou být zároveň doporučovací segmenty, například filmy v případě segmentace epizod do seriálů. V takovém případě bude pro každý film vytvořen samostatný segment, pod který bude spadat pouze daný film.

2.1.4 Filtrace interakcí

FRAMEWORK načetl velké množství interakcí Y', které profiltruje. Filtrování provede ve třech krocích, aby se zbavil všech interakcí, které nebude potřebovat. Díky tomu tyto nepotřebné informace nebudou zpomalovat běh programu. Výsledkem bude množina interakcí Y, $Y \subseteq Y'$.

2.1.4.1 1) Dle view portion

FRAMEWORK odfiltruje interakce s malou *view portion*. Smyslem této filtrace je zbavit se interakcí s malou informační hodnotou. Pokud uživatel viděl pouze velmi malou část filmu nebo seriálu, tak si ho pravděpodobně pouze prohlížel, nebo se podíval na začátek a poté ho vypnul. Hranice *view portion* k odfiltrování je definována vstupním parametrem `min_vp`. Bude vytvořena nová množina interakcí $Y'' = \{y \mid y = (u, i, d) \in Y',\ d \geq \texttt{min\_vp}\}$, nová množina uživatelů $U'' = \{u \mid (u, i, d) \in Y''\}$ a nová množina položek $I'' = \{i \mid (u, i, d) \in Y''\}$.

2.1.4.2 2) Dle položek

V tomto kroku FRAMEWORK vytvoří množinu položek, ke kterým má jak interakční, tak segmentová data $I = I'' \cap I_s$. Dále budou odfiltrovány interakce s položkami, které nejsou v této množině. Při doporučování segmentů nemá smysl ponechávat interakce s položkami, které nespadají pod žádný segment. Například při doporučování žánrů není důvod ponechávat položky, u kterých chybí definice žánru. V ideálním případě by nechyběla žádná data, ale při práci s průmyslovými datasety často nějaká data chybí. Bude vytvořena nová množina interakcí $Y''' = \{y \mid y = (u, i, d) \in Y'',\ i \in I\}$ a nová množina uživatelů $U = \{u \mid (u, i, d) \in Y'''\}$.

2.1.4.3 3) Dle uživatelů

Mezi jedním uživatelem a jednou položkou může existovat více interakcí, pokud uživatel interagoval s položkou vícekrát. V tomto kroku FRAMEWORK vyfiltruje tyto opakované interakce a ponechá pouze tu nejvýznamnější, tedy tu s největší *view portion*. Po tomto kroku bude každá interakce jednoznačně definována uživatelem a položkou. Bude vytvořena finální množina interakcí

$$Y = \{y \mid y = (u, i, d) \in Y''', \\ d = max\{d_m \mid (u_m, i_m, d_m) \in Y''' : u_m = u, i_m = i\}\}.$$

Množina uživatelů ani položek se nemění.

2.1.5 Vytvoření segmentové struktury

V tomto kroku provede FRAMEWORK filtraci segmentů k doporučování. Budou odfiltrovány segmenty, pod které nespadá žádná položka. Bude tedy vytvořena nová množina segmentů $S^r = \{s \mid s \in S^{r\prime}, \exists i \in I : i \to s\}$.

Dále FRAMEWORK vytvoří vhodné datové struktury pro to, aby bylo možné získat všechny položky spadající pod nějaký segment, tedy pro každý segment $s \in S^r$ vytvoří množinu $\mathrm{M}(s)$. Také pro každou položku musí být možné získat všechny segmenty, pod které položka spadá, tedy pro každou položku $i \in I$ vytvoří množinu $\mathrm{N}(s)$. Zmíněné dvě množiny jsou nadefinovány v podkapitole 1.5.1.

2.1.5.1 Rozdělení uživatelů

Množina uživatelů U bude náhodně rozdělena na množinu trénovacích uživatelů U^{tr} a množinu testovacích uživatelů U^{te} v poměru nadefinovaným parametrem `test_size`. Po rozdělení bude platit: $(U^{tr} \cap U^{te} = \emptyset) \wedge (U^{tr} \cup U^{te} = U)$.

2.1.6 Vytvoření interakčních matic

V tomto kroku vytvoří FRAMEWORK dvě interakční matice. Jednu trénovací, která se později použije k trénování modelu a jednu testovací, pomocí které se budou provádět experimenty. Množina interakcí Y bude rozdělena na dvě množiny — trénovací $Y^{tr} = \{y \mid y = (u, i, d) \in Y, u \in U^{tr}\}$ a testovací $Y^{te} = \{y \mid y = (u, i, d) \in Y, u \in U^{te}\}$.

Dále dojde k přeindexování uživatelů, položek a segmentů. Uživatelům v trénovací množině budou přiřazena čísla $\{1, \ldots, |U^{tr}|\}$, uživatelům v testovací množině budou přiřazena čísla $\{1, \ldots, |U^{te}|\}$, položkám budou přiřazena čísla $\{1, \ldots, |I|\}$ a segmentům budou přiřazena čísla $\{1, \ldots, |S^r|\}$. S tímto přeindexováním se bude nadále pracovat, aby bylo možné interakce jednoduše zaznamenat do interakčních matic. FRAMEWORK si pamatuje přeindexovávací tabulky, aby bylo možné případně uživatele, položky nebo segmenty přeindexovat zpět na původní indetifikátory.

FRAMEWORK umí doporučovat položky, jako klasické doporučovací algoritmy, segmenty, nebo smíšený obsah. Při doporučování segmentů se interakční matice vytvářejí jiným způsobem, než při doporučování položek. Při doporučování smíšeného obsahu jsou interakční matice vytvořeny stejně jako při doporučování segmentů, protože položky k doporučování jsou převedeny na segmenty.

2.1.6.1 Interakční matice pro položky

V případě doporučování položek budou vytvořeny dvě interakční matice — trénovací $\mathbf{R}^{tr} \in \mathbb{R}^{|U^{tr}|,|I|}$ a testovací $\mathbf{R}^{te} \in \mathbb{R}^{|U^{te}|,|I|}$. Matice mají stejný

počet sloupců jako je počet položek. V případě doporučování položek FRAMEWORK nabízí dvě možnosti na vytvoření interakčních matic. První možnost zaznamenává do matic *view portion*:

$$\begin{aligned}\mathbf{R}^{tr}_{ui} &= \begin{cases} d & \text{pokud } (u,i,d) \in Y^{tr} \\ 0 & \text{jinak} \end{cases} \\ \mathbf{R}^{te}_{ui} &= \begin{cases} d & \text{pokud } (u,i,d) \in Y^{te} \\ 0 & \text{jinak} \end{cases}\end{aligned} \tag{2.1}$$

Druhá možnost zaznamenává pouhou informaci o tom, že byla provedena interakce:

$$\begin{aligned}\mathbf{R}^{tr}_{ui} &= \begin{cases} 1 & \text{pokud } (u,i,d) \in Y^{tr} \\ 0 & \text{jinak} \end{cases} \\ \mathbf{R}^{te}_{ui} &= \begin{cases} 1 & \text{pokud } (u,i,d) \in Y^{te} \\ 0 & \text{jinak} \end{cases}\end{aligned} \tag{2.2}$$

2.1.6.2 Interakční matice pro segmenty

V případě doporučování segmentů budou vytvořeny matice, kde sloupce budou reprezentovat segmenty místo položek. Je několik různých způsobů, jak jednotlivé položky agregovat do segmentů. Budou vytvořeny dvě matice — trénovací $\mathbf{R}^{tr} \in \mathbb{R}^{|U^{tr}|,|S^r|}$ a testovací $\mathbf{R}^{te} \in \mathbb{R}^{|U^{te}|,|S^r|}$.

Nejjednodušší způsob agregace položek do segmentů je zaznamenávat pouhou informaci o tom, zda uživatel s daným segmentem interagoval. V takovém případě budou interakční matice vytvořeny následujícím způsobem:

$$\begin{aligned}\mathbf{R}^{tr}_{us} &= \begin{cases} 1 & \text{pokud } \exists i \in \mathrm{M}(s) : (u,i,d) \in Y^{tr} \\ 0 & \text{jinak} \end{cases} \\ \mathbf{R}^{te}_{us} &= \begin{cases} 1 & \text{pokud } \exists i \in \mathrm{M}(s) : (u,i,d) \in Y^{te} \\ 0 & \text{jinak} \end{cases}\end{aligned} \tag{2.3}$$

Ukázaný způsob zaznamenává informaci o tom, jestli uživatel někdy interagoval s daným segmentem, ale už nezaznamenává žádnou informaci o tom, s kolika položkami spadajícími pod daný segment uživatel interagoval.

Další možností je do matice sečíst interakce mezi uživatelem a všemi položkami spadají pod daný segment. V takovém případě už prvky interakčních matic nebudou pouze z intervalu $[0, 1]$ jako doposud, ale mohou dosahovat i vyšších hodnot. Matice budou vytvořeny následovně:

$$\begin{aligned}\mathbf{R}^{tr}_{us} &= \sum_{\substack{(u,i,d) \in Y^{tr} \\ i \in \mathrm{M}(s)}} d \\ \mathbf{R}^{te}_{us} &= \sum_{\substack{(u,i,d) \in Y^{te} \\ i \in \mathrm{M}(s)}} d\end{aligned} \tag{2.4}$$

Interakční matice vytvořené tímto způsobem lépe zaznamenají míru interakce mezi uživatelem a segmentem, protože s čím více položkami spadajícími pod daný segment uživatel interagoval, tím vyšší bude hodnota v interakční matici.

Pokud jde však o filmovou — seriálovou hierarchii, tak tento způsob agregace bude velmi upřednostňovat seriály, protože interakce uživatele s filmem může mít maximálně hodnotu 1 a interakce uživatele se seriálem může dosáhnout až takového čísla, kolik má seriál epizod. Proto následující způsob agregace interakcí do interakčních matic tento součet normuje na 1 podělením sumy počtem položek spadajícím pod daný segment:

$$\begin{aligned} \mathbf{R}^{tr}_{us} &= \sum_{\substack{(u,i,d) \in Y^{tr} \\ i \in \mathrm{M}(s)}} \frac{d}{|\,\mathrm{M}(s)|} \\ \mathbf{R}^{te}_{us} &= \sum_{\substack{(u,i,d) \in Y^{te} \\ i \in \mathrm{M}(s)}} \frac{d}{|\,\mathrm{M}(s)|} \end{aligned} \tag{2.5}$$

Tento způsob vrací prvky interakčních matic do intervalu $[0, 1]$. Agregace položek tímto způsobem se dá interpretovat jako *view portion* nad celým segmentem. Problém s tímto způsobem agregace je, že segmenty s velkým množstvím položek budou mít ve většině případů malou hodnotu v interakční matici, protože hodnota bude ve výpočtu vydělena velkým číslem.

V případě doporučování smíšeného obsahu — segmentů a položek, se tato práce zaměřuje i na balancování zastoupení položek a segmentů. FRAMEWORK převedl položky na segmenty, aby se dalo pracovat s doporučováním smíšeného obsahu stejným způsobem jako s doporučováním samotných segmentů. Ale při balancování položek a segmentů se budeme zaměřovat na to, o jaký objekt se jedná na vstupu a na výstupu. Tímto se budeme zabývat zejména v případě doporučování filmů a seriálů. Proto pro tento případ zavádíme ještě další dva způsoby, jak agregovat data do interakčních matic. Tyto způsoby jsou založené na předchozích dvou způsobech, ale jsou více parametrizovatelné a rozlišují mezi agregováním čistého segmentu a segmentu, který byl původně pouze položka. Zavedeme množinu segmentů, které byly uměle vytvořeny pro položky k doporučování. Jedná se o průnik segmentů a položek, protože pro každou takovou položku byl uměle vytvořen vlastní segment, proto se tyto položky nacházejí v obou množinách. Nejčastějším příkladem této množiny v této práci jsou filmy, proto je označena jako $F = S^r \cap I$.

Následující způsob agregace rozlišuje mezi agregací čistých segmentů a segmentů, které jsou zároveň položky. U segmentů, které jsou zároveň položky, do interakční matice jednoduše zaznamená hodnotu *view portion* interakce uživatele s jedinou položkou, která pod segment spadá, pokud taková interakce existuje. Pokud se jedná o čistý segment, tak bude provedena agregace podle

vzorce 2.4 s přidáním dvou parametrů. Vzorec této agregace vypadá takto:

$$\mathbf{R}^{tr}_{us} = \begin{cases} \min\left(\sum\limits_{\substack{(u,i,d)\in Y^{tr}\\ i\in \mathrm{M}(s)}} \mathtt{c}\cdot d,\ \mathtt{max\_val}\right) & \text{pokud } s \notin F \\ d & \text{pokud } s \in F \wedge \mathrm{M}(s) = \{i\} \\ & \qquad \wedge (u,i,d) \in Y^{tr} \\ 0 & \text{jinak} \end{cases}$$
$$\mathbf{R}^{te}_{us} = \begin{cases} \min\left(\sum\limits_{\substack{(u,i,d)\in Y^{tr}\\ i\in \mathrm{M}(s)}} \mathtt{c}\cdot d,\ \mathtt{max\_val}\right) & \text{pokud } s \notin F \\ d & \text{pokud } s \in F \wedge \mathrm{M}(s) = \{i\} \\ & \qquad \wedge (u,i,d) \in Y^{te} \\ 0 & \text{jinak} \end{cases} \tag{2.6}$$

Zavedli jsme dvě konstanty, pomocí kterých je možné ovlivňovat váhy segmentů v interakčních maticích. Konstanta `c` je nejčastěji v intervalu $[0, 1]$, pro různé experimenty však lze volit i mimo tento interval. Tato konstanta reguluje váhu, kterou interakce s jednou položkou spadající pod nějaký segment přispěje do interakční matice. Touto konstantou se dá nastavit poměr významnosti interakce s filmem a významnosti interakce s epizodou seriálu. Konstanta `max_val` zastřešuje maximální hodnotu, která se může v interakční matici vyskytnout. Hodnoty interakčních matic budou díky této konstantě v intervalu $[0, \mathtt{max\_val}]$.

Poslední představený způsob agregace položek do interakční matice funguje podobně jako předchozí způsob 2.6, ale v případě agregace interakcí do segmentů využívá vzorec 2.5 s přidáním jednoho parametru:

$$\mathbf{R}^{tr}_{us} = \begin{cases} \sum\limits_{\substack{(u,i,d)\in Y^{tr}\\ i\in \mathrm{M}(s)}} \frac{\mathtt{c}\cdot d}{|\,\mathrm{M}(s)|} & \text{pokud } s \notin F \\ d & \text{pokud } s \in F \wedge \mathrm{M}(s) = \{i\} \\ & \qquad \wedge (u,i,d) \in Y^{tr} \\ 0 & \text{jinak} \end{cases}$$
$$\mathbf{R}^{te}_{us} = \begin{cases} \sum\limits_{\substack{(u,i,d)\in Y^{tr}\\ i\in \mathrm{M}(s)}} \frac{\mathtt{c}\cdot d}{|\,\mathrm{M}(s)|} & \text{pokud } s \notin F \\ d & \text{pokud } s \in F \wedge \mathrm{M}(s) = \{i\} \\ & \qquad \wedge (u,i,d) \in Y^{te} \\ 0 & \text{jinak} \end{cases} \tag{2.7}$$

Znovu je použita konstanta `c`. V tomto případě je však tato konstanta nejčastěji větší než 1. Tato konstanta balancuje snížení váhy segmentů, které

přináší jejich normalizace. Konstanta `max_val` zde není potřeba, protože maximální hodnota v interakčních maticích je určena konstantou `c`. Všechny prvky interakčních matic budou v množině $[0, \mathtt{c}]$.

Ukázali jsme několik různých způsobů agregace položek do segmentů, které FRAMEWORK používá pro různá segmentová data.

2.1.7 Vytvoření modelu

Vytvoření modelu při použití **KNN** algoritmů je jednoduchá operace. Pro model **userKNN** je trénovací interakční matice modelem sama o sobě. Pro model **itemKNN** stačí trénovací interakční matici transponovat. FRAMEWORK si v tomto kroku vybere model a předá mu matici $\mathbf{R}^{tr}$, případně matici $(\mathbf{R}^{tr})^T$.

2.1.8 Predikce

Tento podmodul FRAMEWORKu dostane na vstupu nějakého uživatele definovaného řádkem testovací interakční matice $\mathbf{R}^{te}$. Nemusí to však být přímo řádek testovací matice, může se jednat o upravený řádek testovací matice. Upravený řádek interakční matice znamená zejména řádek s vynulovaným jedním prvkem, protože se používá *leave-one-out* přístup při měření metrik. Na základě řádku na vstupu spočítá FRAMEWORK pomocí modelu natrénovaného na trénovacích datech pro každou položku skóre.

2.1.9 Evaluace metrik

FRAMEWORK vyhodnocuje zároveň metriky *recall* a *coverage*. Pro každého uživatele u a každou jemu relevantní položku $i \in \mathrm{RI}(u)$ vytvoří dočasného uživatele bez dané položky a nechá si pro něj spočítat skóre pro všechny položky $i \in I$. Položky následně seřadí podle skóre a na N položkách s nejvyšším skóre postupně vypočítává *recall* a *coverage*.

FRAMEWORK také nabízí možnost vyhodnotit metriky pouze pro podmnožinu segmentů. Toho se v této práci využívá zejména pro vyhodnocení metrik zvlášť pro filmy a seriály při doporučování médií. Označme podmnožinu segmentů $B \subseteq S^r$, může se jednat například o podmnožinu filmů nebo seriálů. Dále označme množinu všech testovacích uživatelů, kteří interagovali s alespoň jedním segmentem z množin B jako

$$U_B^{te} = \{u \mid u \in U^{te} \wedge \exists s \in B : \mathbf{R}_{us}^{te} \geq 0\}.$$

Výpočet segmentového *recallu* vychází z vzorečku 1.12, ale zvažuje pouze segmenty v množině B. Zde už se nepracuje s položkami, ale se segmenty.

$$\mathrm{Recall}\,@N_{LOO}^{n}(B) = \sum_{u \in U_B^{te}} \frac{\sum\limits_{s \in \mathrm{RI}(u) \cap B} |\mathrm{Top}(N, \mathrm{RI}(u) \backslash \{s\}) \cap \{s\}|}{|\,\mathrm{RI}(u) \cap B|} \tag{2.8}$$

Výpočet segmentové *coverage* vychází z vzorečku 1.15, ale také zvažuje pouze segmenty v množině B.

$$\text{Coverage}\,@N_{LOO}(B) = \frac{\left|\bigcup_{\substack{u \in U_B^{te} \\ s \in \text{RI}(u)}} \text{Top}(N, \text{RI}(u) \backslash \{s\}) \cap B\right|}{|B|} \tag{2.9}$$

Jak bylo zmíněno, tak tato segmentová evaluace metrik se využívá při počítání metrik jednotlivě pro filmy a seriály. *Recall* spočítaný pouze na množině filmů budeme označovat jako *filmový recall*, podobně *coverage* spočítanou na množině filmů budeme označovat jako *filmovou coverage*. Podobně pro seriály budeme tyto metriky označovat jako *seriálový recall* a *seriálová coverage*.

Mimo *recallu* a *coverage* FRAMEWORK zaznamenává při vyhodnocování metrik další užitečné informace. Například počty doporučených segmentů spadajících do nějakých skupin. Což se znovu dá využít při doporučování filmů a seriálů k zjištění toho, jaký je poměr mezi počtem doporučených filmů a seriálů.

Kapitola 3

Implementace

V této kapitole ukážeme implementační detaily FRAMEWORKu představeného v předchozí kapitole a zmíníme důležité parametry implementovaných algoritmů.

3.1 Použité technologie a knihovny

Celý FRAMEWORK je implementován v programovacím jazyce Python. Jazyk Python byl zvolen především z důvodu existence velkého množství užitečných knihoven pro strojové učení, které nabízí. Mezi nejdůležitější použité knihovny patří:

Scipy: [17, 18] Knihovna Scipy je ve FRAMEWORKu využita k uložení matic. Knihovna nabízí několik možností, jak efektivně uložit řídké matice a také implementuje algoritmy, které nad těmito maticemi provádějí maticové operace. Efektivní uložení řídkých matic je důležité, protože interakční matice jsou velké a velmi řídké matice, které by se v husté reprezentaci nevešly do operační paměti.

Numpy: [19] Knihovna Numpy je jedna z nejpoužívanějších Python knihoven. Tato knihovna se používá k manipulaci s vektory. Velká část Numpy knihovny je implementována v programovacím jazyce C a díky tomu je Numpy velice efektivní. Operace nad vektory jsou ve FRAMEWORKu prováděny pomocí knihovny Numpy.

Sklearn: [20] Knihovna Sklearn implementuje mnoho algoritmů pro strojové učení. Ve FRAMEWORKu je využita k rozdělení dat na trénovací a testovací a také k hledání nejbližších sousedů v algoritmech **userKNN** a **itemKNN**.

Pandas: [21] Knihovna Pandas se používá pro manipulaci a analýzu dat. Poskytuje užitečné datové struktury na pracování s datasety a k tomu je také ve FRAMEWORKu využita především při předzpracování dat.

Matplotlib: [21] Knihovna Matplotlib je vhodná pro vizualizaci dat. Grafy v této práci jsou vytvořeny pomocí knihovny Matplotlib.

3.2 Formát vstupních data

V této práci pracujeme se dvěma průmyslovými datasety ze streamingových platforem, které označíme jako dataset *D1* a dataset *D2*. Stuktura obou datasetů je velice podobná.

První tabulka, kterou FRAMEWORK používá, obsahuje interakční data. Z této tabulky jsou využity tři sloupce. Indentifikátor uživatele — userid, indentifikátor položky — itemid a *view portion* — portion. Tabulka určuje, jaký uživatel interagoval s jakou položkou, neboli přesněji, jak velkou část média si uživatel přehrál. Deset řádků této tabulky je ukázáno na obrázku 3.1.

	userid	itemid	portion
0	NIDINFDIFD	63105	1.00
1	MDFINFDINI	74705	0.01
2	DNIDNFWOJD	137928	0.50
3	FDKDNFINFE	27440	1.00
4	FDNFIWNIWI	127592	0.90
5	VCDNIDOWOJ	135473	1.00
6	DNIWNOENON	126638	1.00
7	FNDINDWOND	140270	0.01
8	MONFINWONC	56110	1.00
9	CDIONDIWNW	99205	0.01

Obrázek 3.1: Ukázka interakčních dat.

Dále FRAMEWORK pracuje se segmentovými daty. Segmentová data nejsou v datasetech zpracována do jednotné formy. K získání segmentových dat se využívají primárně dva způsoby. Prvním způsobem je data extrahovat z atributové položkové tabulky. Ze zmíněné tabulky stačí použít sloupce s požadovaným atributem. Ukázka deseti sloupců atributové tabulky pro atribut žánr je zobrazena na obrázku 3.2. Můžeme vidět, že nějaká atributová data mohou chybět.

Druhý způsob se používá pro práci se seriály. Seriálová hierarchie je uložena ve speciální tabulce. Tabulka určuje vztah přímo mezi epizodou a seriálem a je ukázána na obrázku 3.3. Tato tabulka má jiné vlastnosti než atributová tabulka. Seriálová tabulka nemá chybějící hodnoty, ale není nadefinována pro všechny položky (pro filmy není nadefinována). Také v seriálové tabulce je pro každou položku právě jeden segment, v atributové tabulce to může být množina.

	itemid	genres
0	131324	['animation', 'kids', 'family']
1	135982	['family', 'thriller', 'animation']
2	131332	['animation', 'kids', 'family']
3	169496	NaN
4	76241	['action', 'drama']
5	40913	['documentary']
6	169497	NaN
7	131350	['animation', 'kids', 'family']
8	136008	['family', 'thriller', 'animation']
9	136010	['family', 'thriller', 'animation']

Obrázek 3.2: Ukázka segmentových dat ve formě atributů.

	itemid	seriesid
0	5500	5503
1	2572	3814
2	32028	31912
3	19877	17203
4	5665	5510
5	2568	3814
6	19582	11110
7	24205	24293
8	30532	15889
9	33040	17203

Obrázek 3.3: Ukázka seriálových segmentových dat.

3.3 Předzpracování dat

FRAMEWORK provede filtrace interakcí zmíněné v kapitole 2.1.4. Parametr `min_vp` určí hranici při filtrování interakcí podle *view portion*. Dále dojde k rozdělení uživatelů na trénovací a testovací. Množina uživatelů bude extrahována z množiny vyfiltrovaných interakcí. Parametr `test_size` určí velikost testovací množiny. `test_size` může být reálné číslo v intervalu (0, 1), v takovém případě určí poměr testovacích uživatelů vůči všem uživatelům — $\frac{|U^{te}|}{|U|} =$ `test_size`. Pamatetr `test_size` může být i celé číslo rovno nebo vyšší 1, v takovém případě určuje počet testovacích uživatelů — $|U^{te}| =$ `test_size`.

3.4 Interakční matice

FRAMEWORK vytvoří dvě interakční matice. Jednu trénovací a jednu testovací. Počet řádků trénovací interakční matice je roven počtu trénovacích uživatelů — $|U^{tr}|$, počet řádků testovací interakční matice je roven počtu

testovacích uživatelů — $|U^{te}|$. Počet sloupců bude pro obě matice stejný a bude záležet na tom, jestli se budou doporučovat položky nebo segmenty. Při doporučování položek bude počet sloupců roven počtu položek — $|I|$ a při doporučování segmentů bude roven počtu doporučovaných segmentů — $|S^r|$. Počet sloupců interakčních matic označíme obecně c, kde $c = |I|$ nebo $c = |S^r|$ podle toho, s čím se právě pracuje. Trénovací matice tedy bude mít tvar $\mathbf{R}^{tr} \in \mathbb{R}^{|U^{tr}|,c}$ a testovací $\mathbf{R}^{te} \in \mathbb{R}^{|U^{te}|,c}$. Množinu segmentů nebo položek budeme pro jednoduchost dále označovat $I = \{i_1, \ldots, i_c\}$ a budeme o nich mluvit jako o položkách, i když se může jednat o segmenty, protože po vytvoření interakčních matic už nezáleží na tom, jestli se pracuje s položkami nebo se segmenty.

3.4.1 Formát

Interakční matice jsou velké a řídké, proto jsou uloženy pomocí řídké reprezentace. Jako formát uložení interakčních matic je zvolen CSR (compressed sparse row) formát, protože tento formát umožňuje efektivní iteraci přes řádky (uživatele) [22].

3.4.2 Vytvoření interakčních matic

FRAMEWORK implementuje všechny způsoby vytvoření interakčních matic ukázané v kapitole 2.1.6. Pokud se budou doporučovat pouze položky, tak se vytvoří interakční matice do kterých budou zaznamenány *view portion* podle vzorečku 2.1, nebo matice, kde budou pouze nuly a jedničky podle vzorečku 2.2. Tento výběr určuje booleovský parametr `constant_matrix`. Pokud je nastaven na True, tak je zvolen způsob 2.2, pokud na False, tak je zvolen způsob 2.1.

Při doporučování segmentů je na výběr ze tří možností agregace. Výběr určuje parametr `aggregation`. Pokud `aggregation` = constant, tak bude použit způsob 2.3. Tento způsob agregace budeme označovat jako konstantní agregace. Pokud `aggregation` = sum, tak je využit způsob 2.6. V takovém případě je nutné ještě nastavit parametry `c` a `max_val`. Tento způsob agregace budeme označovat jako sčítací agregace. Pokud `aggregation` = mean, tak je využit způsob 2.7 a je nutné nastavit ještě parametr `c`. Tento způsob agregace budeme označovat jako průměrovací agregace.

3.5 Modely

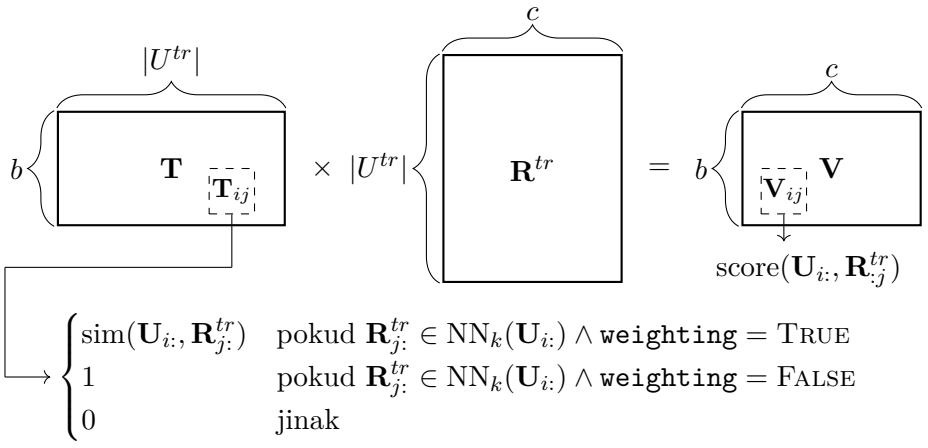
KNN algortimy jsou implementovány s využitím modelu `NearestNeighbors` z knihovny `sklearn.neighbors`. Model `NearestNeighbors` se využívá k hledání nejbližších sousedů v obou algoritmech. Tento model pracuje ve dvou fázích — uložení trénovacích dat a hledání sousedů. Trénovací data jsou matice nějakého rozměru, označíme ji jako $\mathbf{R} \in \mathbb{R}^{m,n}, m, n \in \mathbb{N}$. Řádky matice

R jsou vektory délky n, mezi kterými následně model umí hledat nejbližší sousedy. Pro vektor u délky n model najde k nejbližších sousedů mezi řádky matice **R**. Je možné nadefinovat, podle jaké metriky se bude počítat podobnost mezi vektory, my využíváme cosinovou podobnost. Model vrátí nejbližší sousedy formou dvou polí. Jedno pole je pole indexů sousedů v matici **R**, druhé pole má stejnou délku a jsou v něm uloženy hodnoty cosinových podobností pro každého z nejbližších sousedů. Model dokáže hledat nejbližší sousedy pro více vektorů najednou. Z důvodu efektivity jsou modely implementovány tak, aby dokázaly doporučovat položky více uživatelům zároveň. Počet uživatelů, kterým se v jednom okamžiku doporučují položky, označíme jako $b \in \mathbb{N}$.

3.5.1 UserKNN

Model **userKNN** využívá k doporučování položek podobnosti uživatelů, proto je nutné umět hledat podobné uživatele. K tomu se využije zmíněný model `NearestNeighbors`. Do modelu bude uložena trénovací interakční matice $\mathbf{R}^{tr} \in \mathbb{R}^{|U^{tr}|,c}$, jejíž řádky (uživatele) označíme jako $\{\mathbf{R}^{tr}_{1:}, \ldots, \mathbf{R}^{tr}_{|U^{tr}|:}\}$.

Při doporučování dostane model **userKNN** na vstupu matici $\mathbf{U} \in \mathbb{R}^{b,c}$. Tato matice reprezentuje interakce b různých uživatelů, pro které se počítá doporučení. Tyto uživatele, respektive řádky matice **U**, označíme $\{\mathbf{U}_{1:}, \ldots, \mathbf{U}_{b:}\}$.



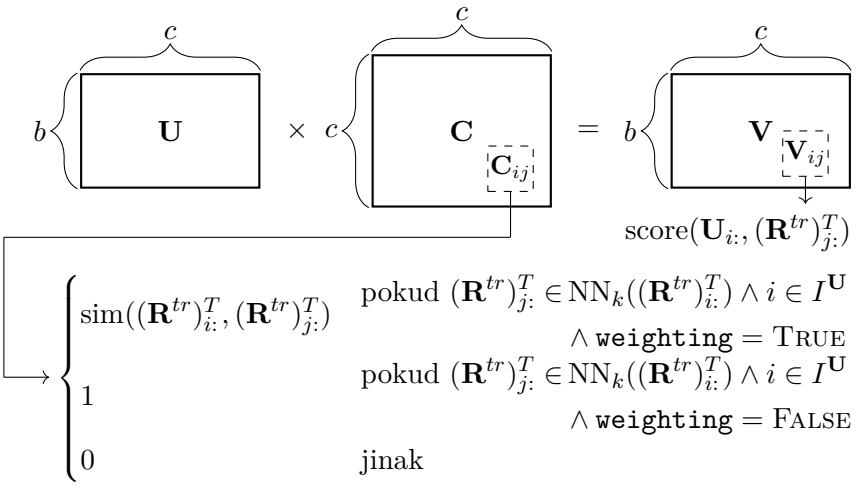
Obrázek 3.4: Práce s maticemi v **userKNN**.

Matice **U** je vstupem do modelu `NearestNeighbors`. Model `NearestNeighbors` najde pro každého z b uživatelů k nejpodobnějších trénovacích uživatelů z $|U^{tr}|$. Počet sousedů definuje parametr `neighbors_num` $= k$. Dále bude vytvořena matice $\mathbf{T} \in \mathbb{R}^{b,|U^{tr}|}$. Do matice **T** je zaznamenán výstup z modelu `NearestNeighbors` tak, že každý řádek je vektorem pro jednoho z b vstupních uživatelů. Tento vektor má až k nenulových hodnot, které zaznamenávají cosinovou podobnost k nejbližším sousedům z matice $\mathbf{R}^{tr}$, pokud parametr `weighting` = True. Pokud parametr `weighting` = False, tak

bude matice $\mathbf{T}$ místo cosinových podobností zaznamenávat pouze hodnotu 1 u k nejbližších sousedů. Násobením matic $\mathbf{T}$ a $\mathbf{R}^{tr}$ podle obrázku 3.4 vznikne matice $\mathbf{V} \in \mathbb{R}^{b,c}$, která už obsahuje pro každého vstupního uživatele a každou položku spočítané skóre.

3.5.2 ItemKNN

Model **itemKNN** využívá k doporučování položek podobnosti položek, z toho důvodu je nutné umět hledat podobné položky. K tomu se také využije model `NearestNeighbors`. Vzhledem k tomu, že se budou hledat nejpodobnější položky místo uživatelů a hledání probíhá v řádcích, tak se musí trénovací interakční matice transponovat. Do modelu se uloží transponovaná trénovací interakční matice $(\mathbf{R}^{tr})^T \in \mathbb{R}^{c,|U^{tr}|}$, jejíž řádky (položky) označíme jako $\{(\mathbf{R}^{tr})^T_{1:}, \ldots, (\mathbf{R}^{tr})^T_{c:}\}$.



Obrázek 3.5: Práce s maticemi v **itemKNN**.

Při doporučování dostane model **itemKNN**, stejně jako **userKNN**, na vstupu matici $\mathbf{U} \in \mathbb{R}^{b,c}$. Označme $I^{\mathbf{U}} \subseteq I$ množinu položek, se kterými interagoval minimálně jeden uživatel ze vstupních uživatelů:

$$I^{\mathbf{U}} = \{i \mid i \in I \wedge \exists u \in \{1, \ldots, b\}, j \in \{1, \ldots, c\} : \mathbf{U}_{uj} \geq 0\}.$$

Množina řádků transponované trénovací interakční matice $\{(\mathbf{R}^{tr})^T_{i:} \mid i \in I^{\mathbf{U}}\}$ bude vstupem do modelu `NearestNeighbors`. Jedná se o řádky reprezentující položky z množiny $I^{\mathbf{U}}$. Model `NearestNeighbors` najde ke každé položce z $I^{\mathbf{U}}$ k nejbližších položek a také jejich podobnosti. Počet sousedů znovu definuje parametr `neighbors_num` $= k$. Dále bude vytvořena matice $\mathbf{C} \in \mathbb{R}^{c,c}$ zaznamenávající zmíněné podobnosti pokud `weighting` = True, nebo pouze hodnotu 1 u k nejbližších sousedů, pokud `weighting` = False. Řádky matice

C reprezentující položky, které nejsou v $I^{\mathbf{U}}$, budou celé nulové. Vynásobením matic **U** a **C** podle obrázku 3.5 vznikne hledaná matice $\mathbf{V} \in \mathbb{R}^{b,c}$, která obsahuje pro každého vstupního uživatele a každou položku spočítané skóre.

3.5.3 Modifikace výstupu

Oba algoritmy dále nabízí dva parametry na další modifikaci výstupu. Číselný parametr `beta` určuje míru penalizace populárních položek. Fungování tohoto parametru je vysvětleno v kapitole 1.3.3. Při použití parametru `beta` bude každý prvek výstupní matice **V** vydělen součtem prvků sloupce se stejným indexem v interakční trénovací matici umocněným na hodnotu `beta`.

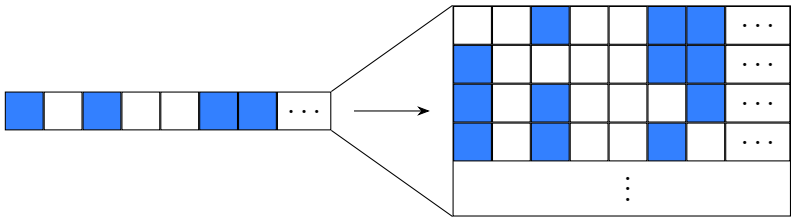
$$\texttt{beta} : \mathbf{V}_{ij} = \frac{\mathbf{V}_{ij}}{(\text{sum}(\mathbf{R}^{tr}_{:j}))^{\texttt{beta}}}$$

Druhým parametrem je booleovský parametr `recommend_seen`, který když je nastaven na False, tak se nebudou doporučovat uživateli položky, se kterými někdy už interagoval.

$$\texttt{recommend\_seen} = \text{False} : \mathbf{V}_{ij} = \begin{cases} \mathbf{V}_{ij} & \text{pokud } \mathbf{U}_{ij} = 0 \\ 0 & \text{jinak} \end{cases}$$

3.6 Vyhodnocení metrik

Metriky se vyhodnocují na množině testovacích uživatelů $|U^{te}|$. Z paměťových důvodů probíhá vyhodnocení po dávkách. Velikost dávky, neboli počet testovacích uživatelů, se kterými se bude v jednom okamžiku pracovat, určuje parametr `user_batch`. U měření metrik se využívá *leave-one-out* přístup. Využití tohoto přístupu bude dosaženo tak, že z každého řádku testovací interakční matice bude vytvořeno tolik řádků, kolik je v původním řádku nenulových hodnot, kde v každém řádku bude jedna hodnota vynulována. Na obrázku 3.6 je toto rozdělení graficky znázorněno. Obrázek zobrazuje expanzi jednoho řádku reprezentujícího jednoho uživatele. Modrá políčka znázorňují nenulové hodnoty. Takto vznikne pro každého uživatele několik virtuálních uživatelů, na kterých se budou vyhodnocovat metriky. Počet virtuálních uživatelů pro každého uživatele bude stejný jako počet jeho interakcí s položkami. Vstupem do tohoto kroku je matice rozměru $\mathbb{R}^{\texttt{user\_batch},c}$ reprezentující podmnožinu řádků testovací interakční matice. Označme počet všech nenulových hodnot v této matici jako b. Výstupem tohoto kroku bude matice rozměru $\mathbf{U} \in \mathbb{R}^{b,c}$ a ta bude také vstupem do doporučovacího modelu.



Obrázek 3.6: Expanze řádku reprezentujícího interakce uživatele do více řádků pro využití *leave-one-out* přístupu.

Po tom, co doporučovací model spočítá skóre pro každého virtuálního uživatele a každou položku, tak pro každého virtuálního uživatele seřadí pole položek podle skóre. Parametr `topN` určí, kolik položek s nejvyšším skóre se bude uvažovat v Top-N doporučení. Tento parametr musí být celé číslo nebo pole celých čísel. V případě pole čísel FRAMEWORK vyhodnotí metriky pro každý prvek pole jednotlivě. Počítání *recallu* a *coverage* je vysvětleno v kapitolách 1.4.1 a 1.4.2. U *recallu* se pro každého virtuálního uživatele kontroluje, zda zakrytá položka patří mezi N doporučených. U *coverage* se porovnává velikost množiny doporučených položek a všech položek.

Kapitola 4

Experimenty

Tato kapitola představuje provedené experimenty. Vyhodnocení experimentů probíhá převážně pomocí metrik *recall* a *coverage* a vizualizace těchto metrik je provedena pomocí *recall-coverage* roviny.

První je otestována schopnost algoritmů doporučovat samotné položky. Experimenty na samotných položkách jsou provedeny, aby bylo možné porovnat průběh algoritmů na položkách a na segmentech.

Dále se experimentuje s doporučováním smíšeného obsahu, přesněji filmů a seriálů. Jsou otestovány různé způsoby agregace položek do segmentů. Testuje se vliv parametrů na měřené metriky a také na poměr mezi počtem doporučených filmů a seriálů. Také se testuje schopnost algoritmů doporučovat více seriálů uživatelům, kteří více interagovali se seriály a filmů uživatelům, kteří více interagovali s filmy.

Poslední experimenty jsou provedeny na samotných segmentech. Přesněji se experimentuje s doporučováním žánrů a herců. U žánrů je i vizualizována podobnost mezi jednotlivými žánry spočítaná pomocí interakční matice.

4.1 Datové sady

Experimenty probíhají na dvou datových sadách. Formát vstupních dat je popsán v kapitole 3.2. V obou datových sadách jsou interakce 100000 uživatelů za rok. Tabulka níže ukazuje počet uživatelů, položek a interakcí po filtraci interakcí podle *view portion*, kde jsme nastavili `min_vp` = 0.5. Hustota značí poměr počtu interakcí a součinu počtu uživatelů a položek. Tato filtrace podle *view portion* je provedena ve všech experimentech. Pokud není řečeno jinak, tak je počet testovacích uživatelů v experimentech nastaven na 5000.

datová sada	počet uživatelů	počet položek	počet interakcí	hustota
D1	94474	64427	10675264	0.17%
D2	97793	33381	3616047	0.11%

4.2 Doporučování položek

V prvních experimentech ukážeme fungování obou algoritmů na doporučování položek, zatím bez využití segmentů. Ukážeme, jak se algoritmy **userKNN** a **itemKNN** chovají na obou datasetech a jaký vliv mají jednotlivé parametry na kvalitu doporučení. Naměřené hodnoty metrik *recall* a *coverage* pro různé parametry algoritmů jsou vizualizovány v *recall-coverage* rovině [4]. V této rovině je přehledně vidět, jakého *recallu* a jaké *coverage* dosáhl algoritmus pomocí jakých parametrů. Na každém grafu je více různých křivek. Každá křivka reprezentuje vývoj metrik v závislosti na počtu sousedů pro jednu hodnotu parametru `beta`. Čísla u křivek značí počet sousedů použitý při doporučení, neboli `neighbors_num`. Metriky jsou měřené pomocí Top-10 doporučení. Ve všech experimentech je nastaven parametr `recommend_seen` na False, aby algoritmy doporučovaly pouze nové položky.

U **userKNN**, obrázek 4.1, je při nastavení parametru `beta` = 0.25 nebo `beta` = 0.5 dosaženo dobrého kompromisu mezi hodnotami *recall* a *coverage*, zejména pro hodnoty parametru `neighbors_num` okolo 50 až 100. Výsledky jsou podobné na obou datasetech. U datasetu *D2* bylo dosahováno vyšší *coverage*, tento dataset má však méně položek.

U **itemKNN**, obrázek 4.2, bylo dosaženo pro oba datasety podobných hodnot metrik jako u **userKNN**. Lepších výsledků bylo v tomto případě dosaženo pomocí nižších hodnot parametrů. Při nastavení hodnoty parametru `beta` = 0 bylo dosaženo nejvyššího *recallu* a rozumné *coverage*. Při nastavení `beta` = 0.25 bylo dosaženo mírně nižšího *recallu* a mírně vyšší *coverage*. Jako optimální hodnoty parametru `neighbors_num` se ukázaly hodnoty 5 až 10.

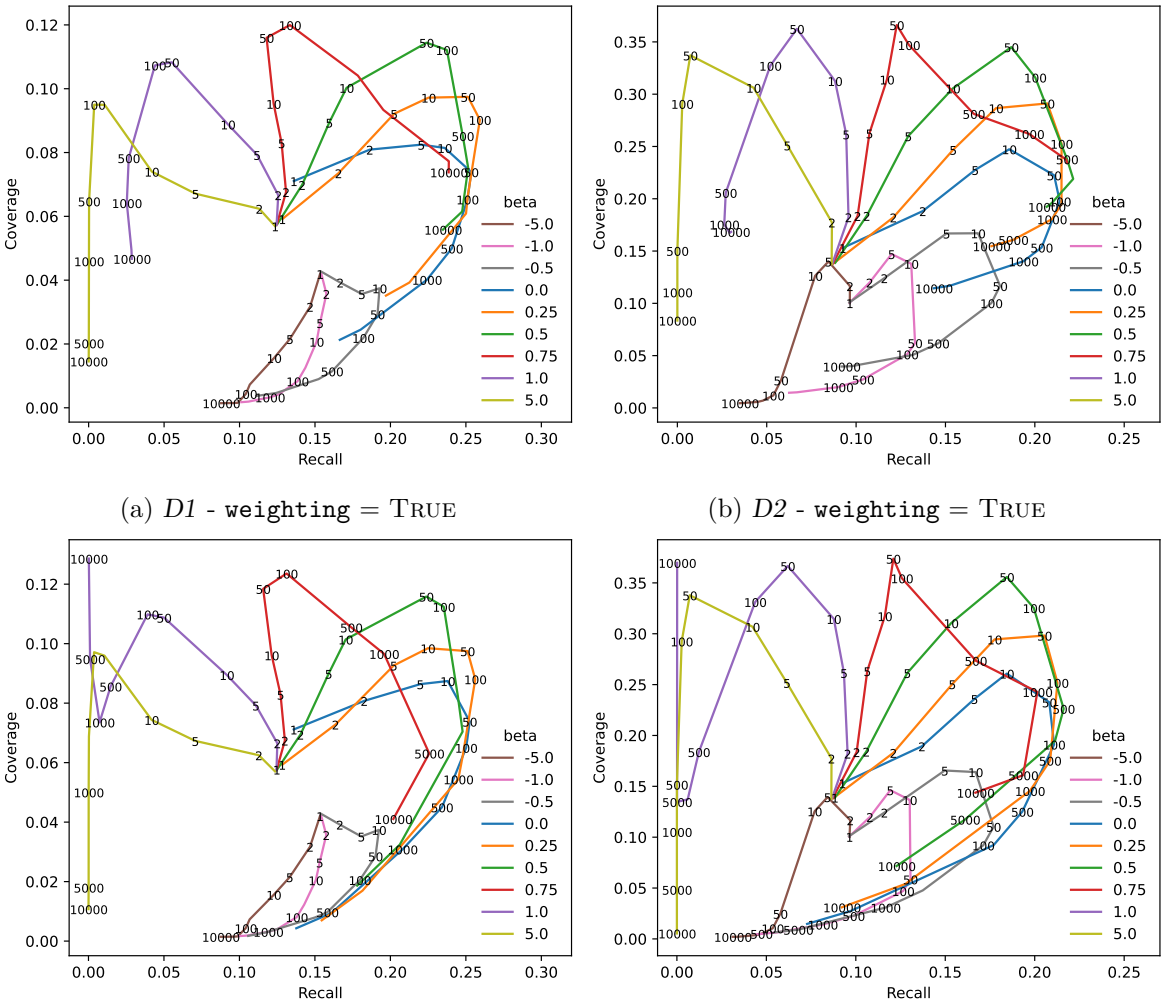
Hodnoty dosaženého *recallu* mohou být zavádějící, protože pro dosažení vysokého *recallu* stačí uživateli doporučovat epizody seriálu na základě jeho interakcí s jinými epizodami stejného seriálu. Pro přesnější odhad *recallu* by bylo dobré místo využití *leave-one-out* přístupu zakrývat všechny epizody jednoho seriálu při vyhodnocování metrik. K tomu při segmentaci seriálů v následující podkapitole dochází automaticky s pouhým využitím *leave-one-out* přístupu.

Průběh algoritmů byl velice podobný pro váženou i neváženou variantu obou algoritmů. Jediný výraznější rozdíl při použití nevážené varianty byl ten, že u algoritmu **itemKNN** se při nastavení `beta` = 0 pro velké množství sousedů podařilo dosáhnout vysoké hodnoty *coverage*, ale za cenu téměř nulového *recallu*. Podobný, ale mírnější, výkyv je možné pozorovat u nevážené varianty algoritmu **userKNN** při nastavení `beta` = 1. Při zamyšlení se nad fungováním algoritmů **userKNN** a **itemKNN** je zřejmé, že při ukázaném nastavení parametrů se v obou případech s rostoucím množstvím sousedů doporučení přibližuje doporučení náhodných položek, proto je dosaženo vyšší hodnoty *coverage*, ale téměř nulové hodnoty *recallu*.

Vzhledem k tomu, že průběh obou algoritmů na obou datasetech je pro váženou i neváženou variantu podobný, tak budeme dále používat pouze váženou variantu obou algoritmů. Vážené variantě algoritmu **itemKNN** se

podařilo dosáhnout mírně vyššího *recallu*, než neváženého variantě. V dalších experimentech bude tedy nastaven parametr `weighting` = TRUE.

Dále se ukázalo, že pro hodnoty parametru `beta` mimo interval $[0, 1]$ není dosaženo vysokých hodnot ani jedné z metrik. Při nastavení `beta` > 1 dochází k moc velké penalizaci populárních položek a při nastavení `beta` < 0 dochází naopak ke zvýhodnění populárních položek, které už jsou tak zvýhodněny svoji popularitou. Pro větší přehlednost vizualizace *recall-coverage* roviny budou v dalších experimentech vizualizovány hodnoty parametru `beta` pouze v intervalu $[0, 1]$.



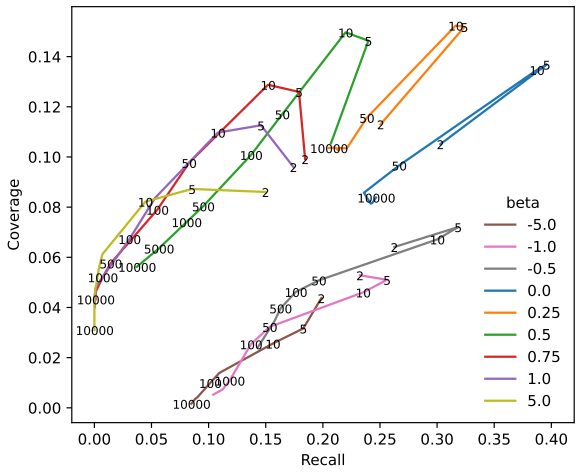
(a) *D1* - `weighting` = TRUE

(b) *D2* - `weighting` = TRUE

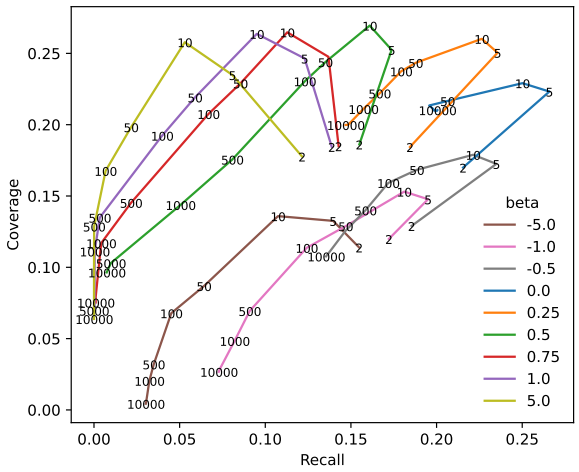
(c) *D1* - `weighting` = FALSE

(d) *D2* - `weighting` = FALSE

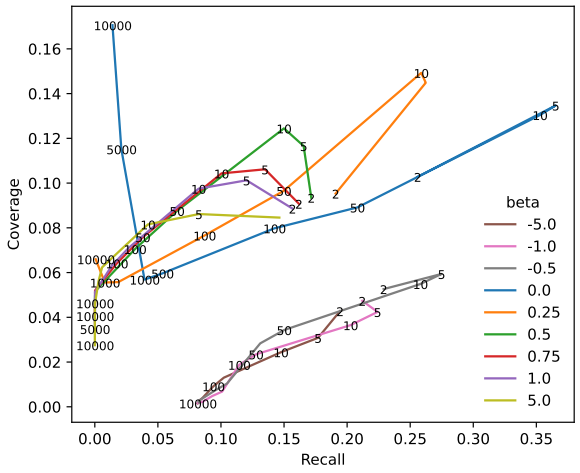
Obrázek 4.1: Vizualizace běhu algoritmu **userKNN** při doporučování položek v *recall-coverage* rovině pro různé hodnoty parametrů `beta` a `neighbors_num` (hodnoty přímo v grafu). `topN` = 10.



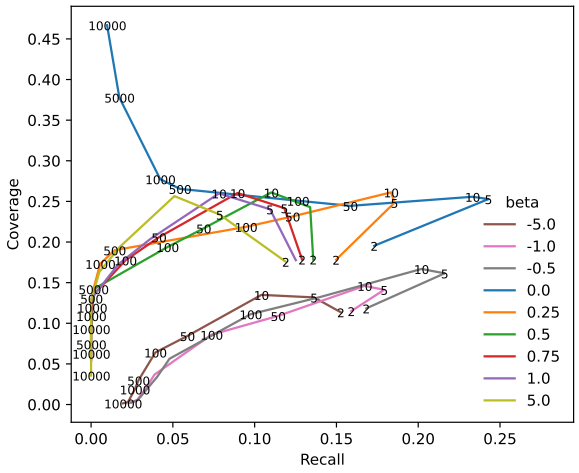
(a) *D1* - `weighting` = True



(b) *D2* - `weighting` = True



(c) *D1* - `weighting` = False



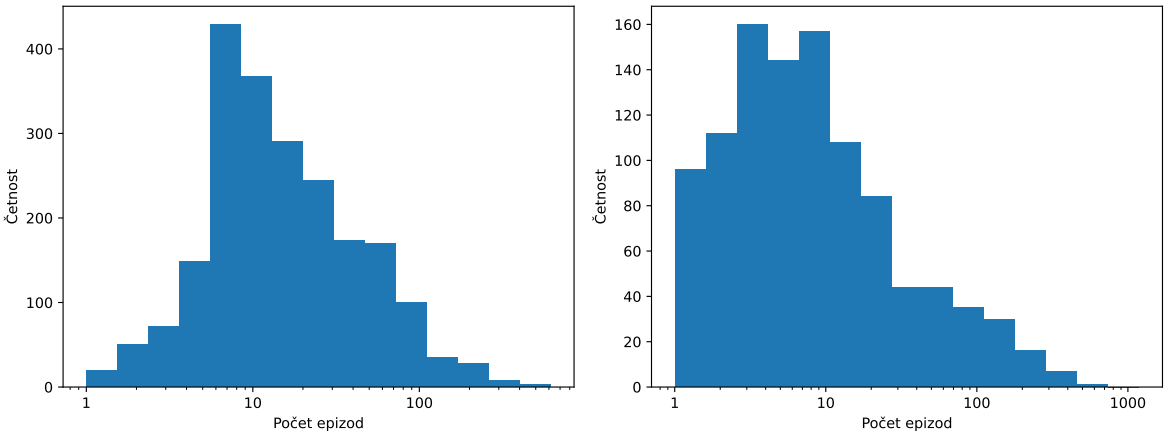
(d) *D2* - `weighting` = False

Obrázek 4.2: Vizualizace běhu algoritmu **itemKNN** při doporučování položek v *recall-coverage* rovině pro různé hodnoty parametrů `beta` a `neighbors_num` (hodnoty přímo v grafu). `topN` = 10.

4.3 Doporučování filmů a seriálů

Nyní budeme experimentovat s doporučováním filmů a seriálů. Budeme testovat různé druhy způsobu agregace hodnot do interakčních matic a jejich vliv na kvalitu doporučení. Tabulka níže ukazuje rozdíly mezi datasety *D1* a *D2* v počtech filmů, seriálů a interakcí.

datová sada	*D1*	*D2*
počet uživatelů	94474	97793
počet položek	64427	33381
počet filmů	5816	7940
počet seriálů	2139	1039
počet epizod	58611	25441
počet seriálů a filmů (položek)	7955	8979
počet interakcí s filmy	762808	756444
počet interakcí se seriály	725279	772981
počet interakcí s epizodami	9912456	2859603
průměrný počet epizod na seriál	27	24
průměrný počet interakcí s filmem	131	95
průměrný počet interakcí se seriálem	339	743
průměrný počet interakcí s epizodou	169	112
průměrný počet zhlédnutých epizod	13.6	3.6

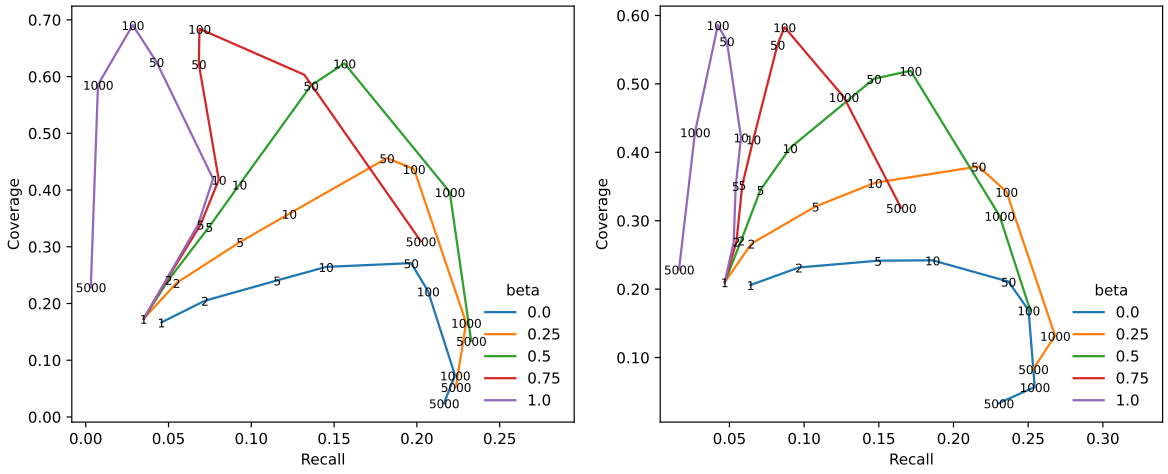


(a) *D1* (b) *D2*

Obrázek 4.3: Rozložení počtu epizod jednotlivých seriálů.

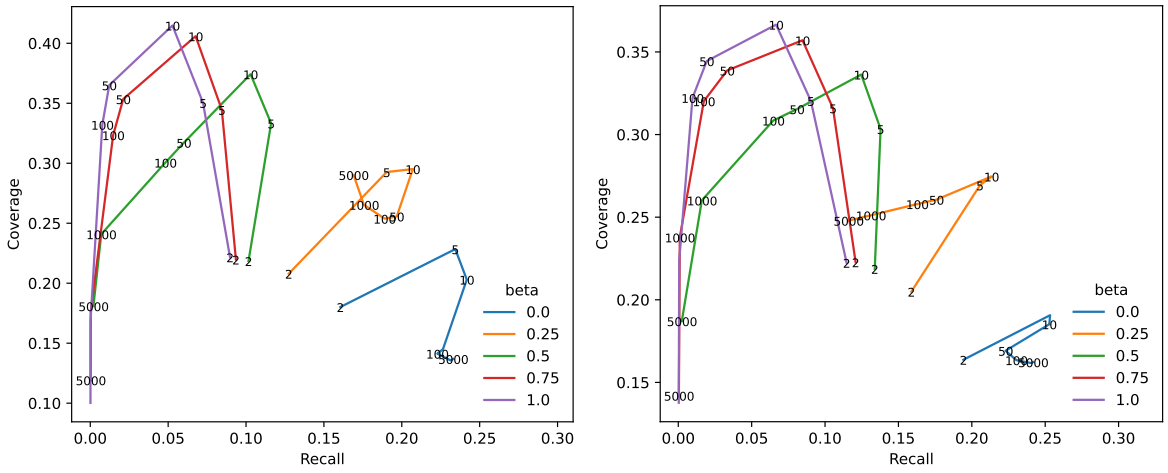
Dataset *D2* obsahuje více filmů a méně seriálů než dataset *D1*. Počet interakcí s filmy i se seriály je podobný pro oba datasety. Počet interakcí se seriály znamená počet unikátních interakcí mezi uživatelem a seriálem, kde

interakce jednoho uživatele s více epizodami jsou už agregovány do jedné interakce. Dataset *D1* má významně vyšší počet interakcí s epizodami, z čehož vyplývá největší rozdíl mezi těmito datasety, což je průměrný počet interakcí jednoho uživatele s různými epizodami jednoho seriálu, označený v tabulce jako průměrný počet zhlédnutých epizod. Při agregování interakcí do interakčních matic bude tedy v průměru u datasetu *D1* agregováno více hodnot do jedné hodnoty, než u datasetu *D2*. Přesněji řečeno, při použití sčítací nebo průměrovací agregace se bude provádět výpočet jedné hodnoty v interakční matici v případě seriálu z výrazně více hodnot u datasetu *D1*, než u datasetu *D2*. Průměrný počet epizod seriálu je však pro oba datasety podobný.



(a) *D1* - **userKNN**

(b) *D2* - **userKNN**



(c) *D1* - **itemKNN**

(d) *D2* - **itemKNN**

Obrázek 4.4: Vizualizace běhu algoritmů při segmentaci podle seriálů v *recall-coverage* rovině pro různé hodnoty parametrů `beta` a `neighbors_num` (hodnoty přímo v grafu). `aggregation` = CONSTANT, `topN` = 10.

Obrázek 4.3 ukazuje histogram rozložení počtu epizod pro oba datasety s logaritmickou osou x. Velké množství seriálů má do 100 epizod, ale jsou i seriály, které mají více epizod. Dataset *D2* na rozdíl od datasetu *D1* obsahuje i takzvané „nekonečné seriály“ nebo „soap opery“, neboli seriály, které mají značné množství epizod, někdy i přes 1000.

Pro první expetimenty provedeme agregaci do matice způsobem 2.2, neboli `aggregation` = CONSTANT. Průběh experimentů je zobrazen na obrázku 4.4. Experimenty znovu probíhají na Top-10 doporučení. Znovu bylo dosaženo poměrně vysokého *recallu*. Nyní už to není ovlivněno tím, že je triviální doporučovat epizody na základě interakcí s ostatními epizodami stejného seriálu. Nyní už se doporučují pouze celé seriály. Na druhou stranu se výrazně snížil počet položek, díky čemuž se také dařilo dosáhnout vyšší *coverage*. Nejvyšších hodnot *recall* a *coverage* se podařilo dosáhnout s podobným nastavením parametrů jako při doporučování položek.

4.3.1 Sčítací agregace

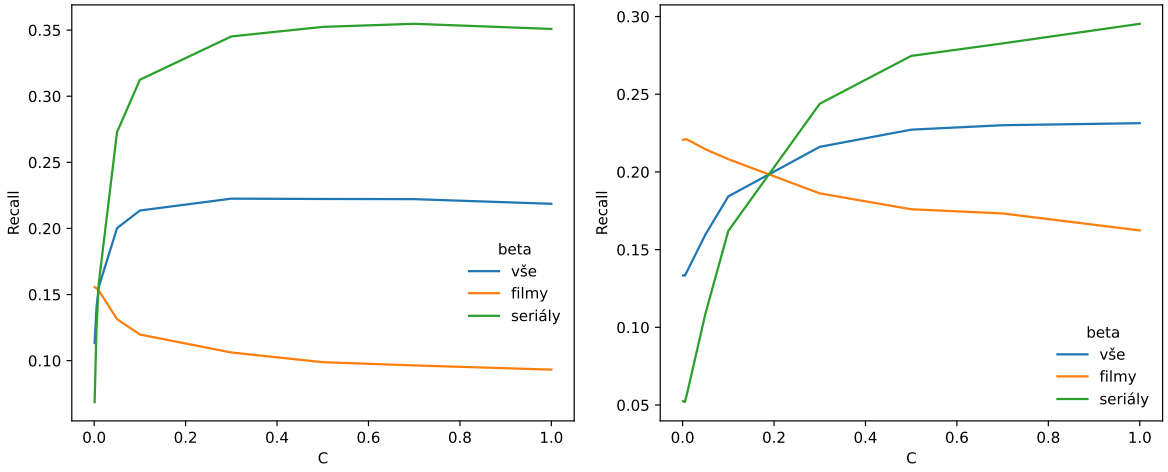
V této podkapitole testujeme agregaci položek do interakční matice podle vzorce 2.6, neboli `aggregation` = SUM. Testujeme vliv parametru `c` na *recall* a *coverage*. Parametr `c` testujeme na množině hodnot $(0, 1]$. Čím vyšší hodnota parametru `c`, tím více přispívá jedna epizoda seriálu do interakční matice. Pro vyšší hodnoty parametru `c` budou mít vyšší hodnoty v interakční matici seriály oproti filmům a naopak. Pro tento experiment jsme nastavili parametr `max_val` = 1, aby se nestávalo, že seriály budou mít v interakčních maticích vyšší maximální hodnoty než filmy.

Experimenty probíhají na algoritmech **userKNN** a **itemKNN**. Aby se v těchto experimentech mohl ladit parametr `c`, tak ostatní parametry budou zafixovány na hodnoty, pomocí kterých bylo dosaženo vysokého *recallu* a *coverage* v předchozím experimentu. Pro **userKNN** jsou parametry nastaveny jako `beta` = 0.5, `neighbors_num` = 1000. Pro **itemKNN** jsou parametry nastaveny jako `beta` = 0.25 a `neighbors_num` = 10.

Závislost *recallu* a *coverage* na parametru `c` pro algoritmus **userKNN** je zobrazena na obrázku 4.5. Je zobrazen celkový *recall*, *filmový recall* i *seriálový recall*, stejně tak pro *coverage*. *Filmové* a *seriálové* metriky jsou vysvětleny v kapitole 2.1.9. S rostoucím parametrem `c` roste *seriálový recall* a klesá *filmový recall*. Celkový *recall* zprvu roste, ale od určité hodnoty je už skoro konstantní. U datasetu *D1* je výrazně vyšší *seriálový recall* než *filmový recall*, u datasetu *D2* se *seriálový* a *filmový recall* více řídí parametrem `c`. U datasetu *D2* se nabízí jako optimální volba `c` = 0.2, protože v tomto bodě se rovná *seriálový* a *filmový recall* a celkový. U datasetu *D2* není volba `c` tak jednoznačná, protože *seriálový* a *filmový recall* se rovnají přibližně okolo hodnoty `c` = 0.005, ale celkový *recall* je výrazně vyšší pro vyšší hodnoty `c`.

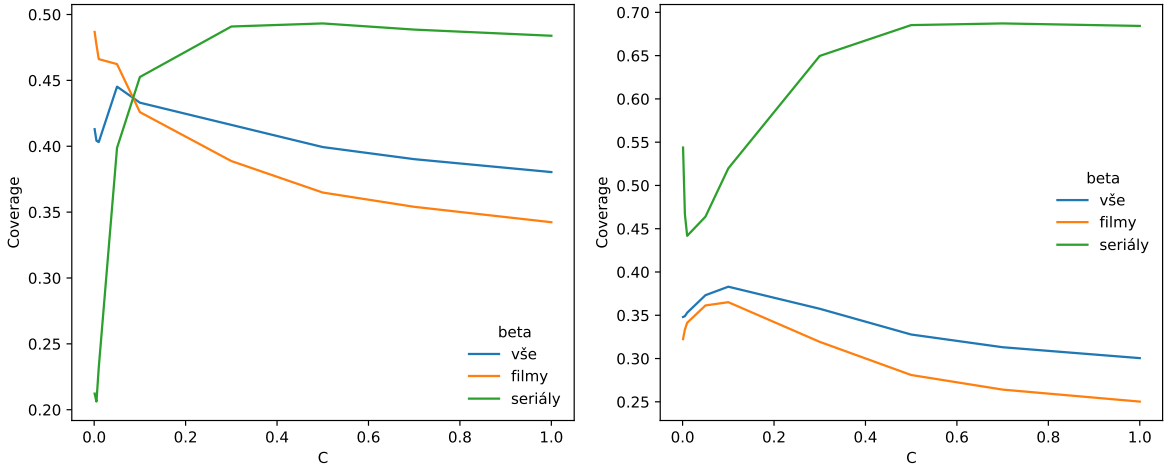
Pro oba datasety platí, že *seriálová coverage* roste s rostoucím parametrem `c` a *filmová* klesá. U datasetu *D2* to však platí až od určité hodnoty `c`.

Nejvyšší *coverage* a nejlepšího kompromisu mezi *filmovou* a *seriálovou coverage* je dosaženo kolem hodnoty `c` = 0.05 pro oba datasety.



(a) *D1 - recall*

(b) *D2 - recall*



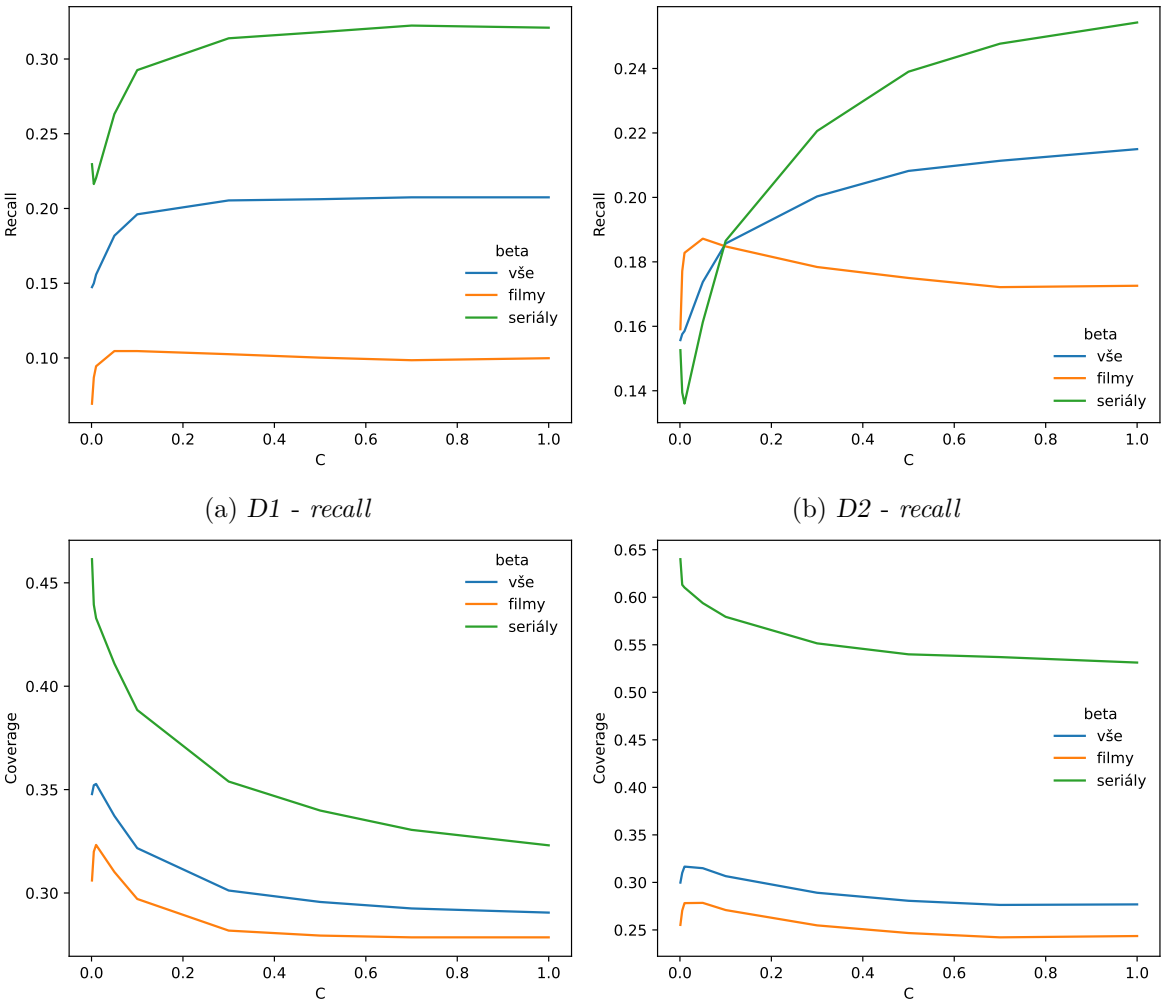
(c) *D1 - coverage*

(d) *D2 - coverage*

Obrázek 4.5: Závislost *recallu* a *coverage* na parametru `c` při segmentaci podle seriálů a použití algoritmu **userKNN**. `aggregation` = SUM, `beta` = 0.5, `neighbors_num` = 1000, `topN` = 10.

U algoritmu **itemKNN** je průběh *recallu* v závislosti na parametru `c` podobný jako u algoritmu **userKNN** pro oba datasety, obrázek 4.6. Rozdíl je však v *coverage*. Všechny *coverage* s rostoucím parametrem `c` pro oba datasety mírně klesají od určité hodnoty. To je způsobeno častějším doporučováním populárních seriálů.

Ve většině případů seriály dosahují vyššího *recallu* i *coverage*. U datasetu *D1* je u *recallu* tento rozdíl výraznější, než u datasetu *D1*. Je to způsobeno



(a) *D1 - recall*

(b) *D2 - recall*

(c) *D1 - coverage*

(d) *D2 - coverage*

Obrázek 4.6: Závislost *recallu* a *coverage* na parametru `c` při segmentaci podle seriálů a použití algoritmu **itemKNN**. `aggregation` = SUM, `beta` = 0.25, `neighbors_num` = 10, `topN` = 10.

tím, že dataset *D1* obsahuje vyšší množství interakcí s epizodami. Agregace epizod do segmentů je pak kvalitnější, protože obsahuje více informací. Celkově *seriálový recall* u datasetu *D1* dosahuje vyšších hodnot než u datasetu *D2*.

Naopak u *coverage* je větší rozdíl mezi *seriálovou* a *filmovou coverage* u datasetu *D2*. To způsobuje větší nepoměr mezi počtem filmů a seriálů v datasetu *D2*. Dataset *D2* obsahuje více filmů a méně seriálů než dataset *D1*.

U datasetu *D2* se nabízí jako jedna z možných voleb `c` = 0.1 z důvodu rovnosti *recallů*. U datasetu *D2* je volba znovu méně jednoznačná.

4.3.2 Průměrovací agregace

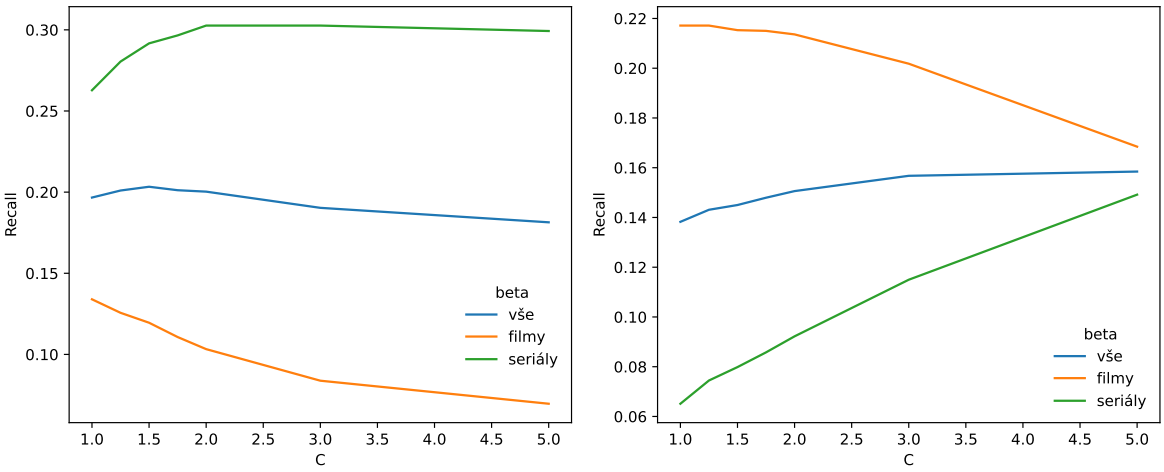
Další způsob agregace položek do interakční matice, který FRAMEWORK nabízí, je agregace podle vzorce 2.7, neboli `aggregation` = MEAN. Tento způsob agregace bere v potaz i počet epizod jednotlivých seriálů. Znovu otestujeme vliv parametru `c` na metriky *recall* a *coverage*. Nyní budeme testovat `c` pro hodnoty 1 a vyšší. Pro hodnotu `c` = 1 bude do matice agregována průměrná hodnota *view portion* mezi uživatelem a epizodou daného seriálu. Pro vyšší hodnoty `c` se tato agregovaná hodnota pro seriály zvyšuje. V tomto způsobu agregace může dosáhnout maximální hodnota v interakční matici až hodnoty `c` pro seriály. Hodnoty parametrů pro doporučovací algoritmy jsou nastaveny stejně jako v předchozím experimentu. Tedy pro **userKNN** jsou parametry nastaveny jako `beta` = 0.5, `neighbors_num` = 1000 a pro **itemKNN** jsou parametry nastaveny jako `beta` = 0.25, `neighbors_num` = 10.

Obrázek 4.7 zobrazuje závislost *recallu* a *coverage* na parametru `c` pro algoritmus **userKNN**. Pro dataset *D1* dosahuje *recall* velice podobných výsledků jako v předchozím způsobu agregace, hodnoty jsou však mírně nižší. Pro dataset *D2* významně klesnul *seriálový recall*. *Coverage* dosahuje podobných hodnot jako v předchozím experimentu. S rostoucím parametrem `c` roste *seriálová coverage* a klesá *filmová coverage*.

Pro algoritmus **itemKNN** vývoj *recallu* a *coverage* zobrazuje obrázek 4.8. *Recall* má znovu velice podobný vývoj jako u předchozí sčítací agregace. Pro dataset *D1* dosahuje podobných, ale mírně nižších hodnot a pro dataset *D2* znovu *seriálový recall* klesnul. Podobně jako u **userKNN** s rostoucím parametrem `c` roste *seriálová coverage* a klesá *filmová coverage*, což v minulém experimentu neplatilo, tam všechna *coverage* pouze klesala pro vyšší `c`.

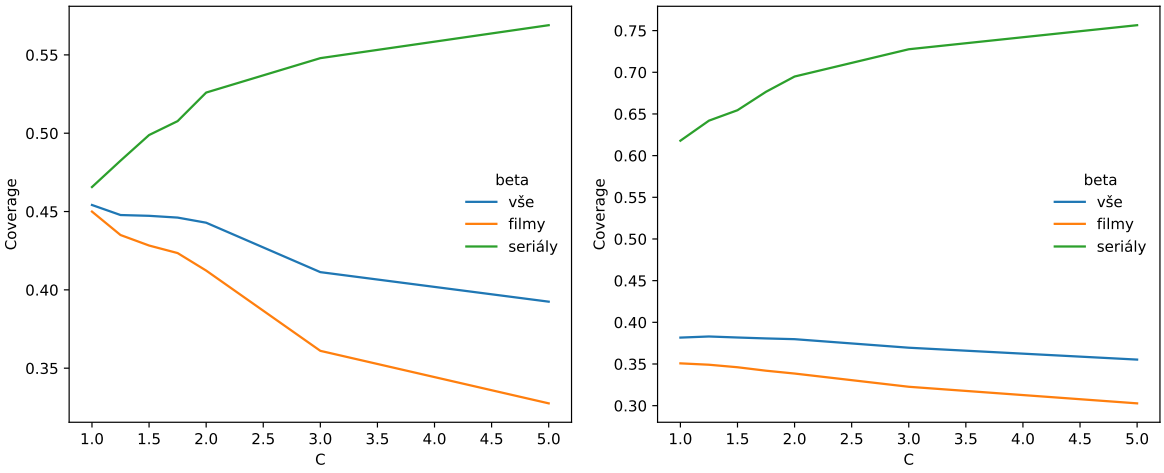
Důvod, proč u datasetu *D2* tolik klesl *seriálový recall* pro oba algoritmy je ten, že tento průměrovací způsob agregace penalizuje seriály s větším počtem epizod. Jak je ukázáno na obrázku 4.3, tak dataset *D2* obsahuje i seriály s opravdu velkým množstvím epizod. Hodnota, dosazena do interakční matice, je vydělena počtem epizod, proto seriály s tak velkým množstvím epizod budou mít malou hodnotu v interakční matici a budou málo doporučovány.

Po experimentálním porovnání sčítací agregace podle vzorečku 2.6 a průměrovací agregace podle vzorečku 2.7, se sčítací agregace jeví jako lepší způsob agregace. Pro oba datasety dosahuje tento způsob vyšších hodnot metrik *recallu* a *coverage*. Nejedná se však o velké rozdíly. Sčítací agregace nebere v potaz počty epizod jednotlivých seriálů. Tím upřednostňuje seriály s větším množstvím epizod, protože čím více epizod, tím více hodnot se při agregaci sčítá. Naopak průměrovací agregace bere v potaz počet epizod a tímto počtem vydělí agregovanou hodnotu v interakční matici. Tento způsob penalizuje seriály s velkým množstvím epizod, protože jejich hodnotu v interakční matici dělí větším číslem. Tato penalizace se ukázala jako horší přístup u datasetu *D2*, vzhledem k tomu, že tento dataset obsahuje i seriály, které mají přes 1000 epizod. Pro offline evaluaci se ukazuje sčítací způsob agregace jako



(a) *D1 - recall*

(b) *D2 - recall*



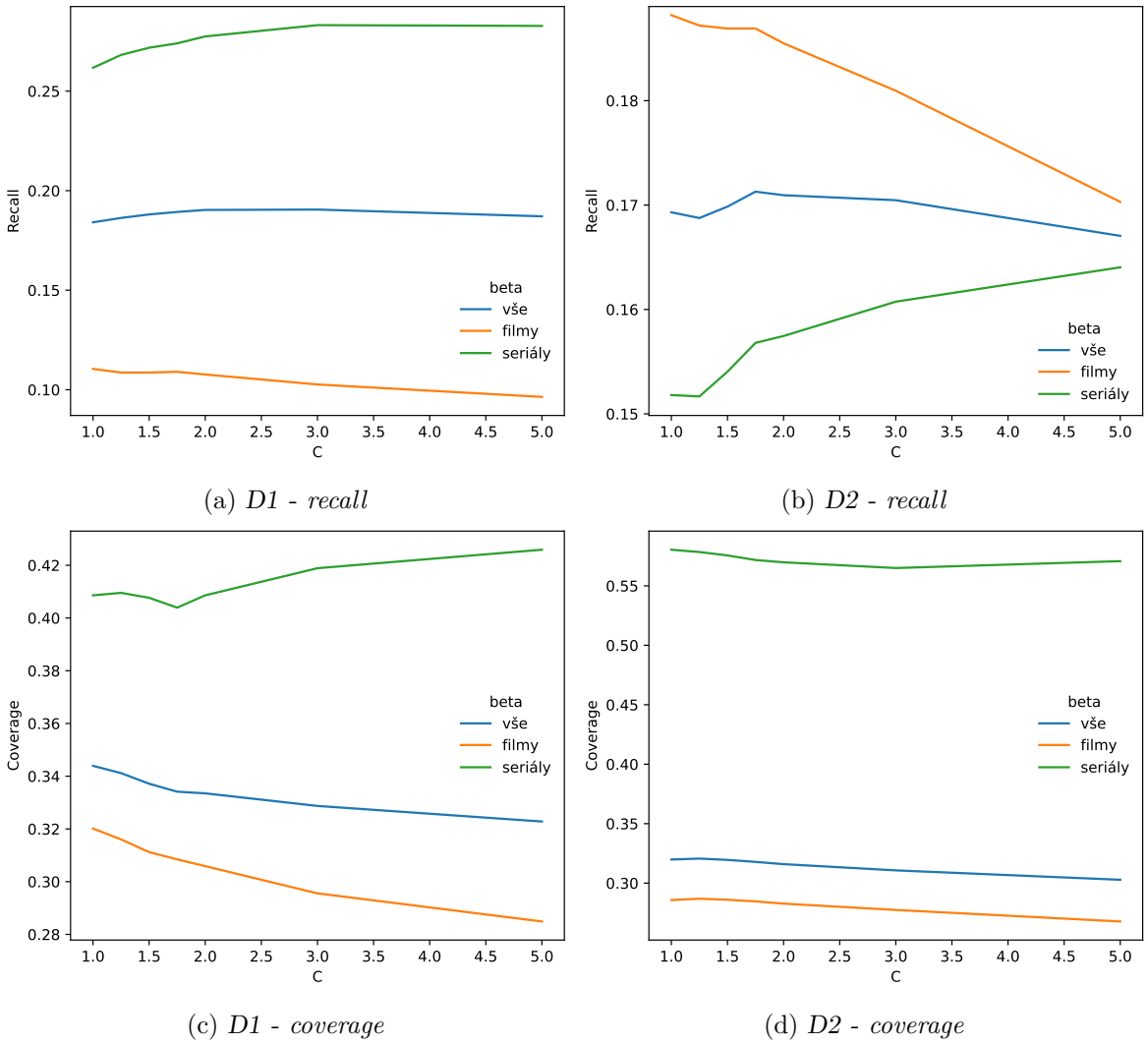
(c) *D1 - coverage*

(d) *D2 - coverage*

Obrázek 4.7: Závislost *recallu* a *coverage* na parametru `c` při segmentaci podle seriálů a použití algoritmu **userKNN**. `aggregation` = MEAN, `beta` = 0.5, `neighbors_num` = 1000, `topN` = 10.

lepší způsob. Průměrovací způsob by mohl být pro lepší výsledky upraven dalšími parametry určujícími například maximální hodnotu, kterou se bude agregovaná hodnota dělit. Pro online evaluaci by však tento způsob mohl fungovat dobře. Některé doporučovací přístupy penalizují seriály s velkým počtem epizod, protože větší počet epizod znamená pro uživatele větší „závazek“, což může nového uživatele odradit [13]. Průměrovací agregace tuto penalizaci dělá automaticky, tento způsob by tedy mohl dobře fungovat v online evaluaci, tento aspekt se však špatně měří offline.

Pro oba testované způsoby agregace se ukázalo, že pomocí parametru `c` je



(a) *D1 - recall*

(b) *D2 - recall*

(c) *D1 - coverage*

(d) *D2 - coverage*

Obrázek 4.8: Závislost *recallu* a *coverage* na parametru `c` při segmentaci podle seriálů a použití algoritmu **itemKNN**. `aggregation` = MEAN, `beta` = 0.25, `neighbors_num` = 10, `topN` = 10.

možné hledat kompromis mezi *filmovým* a *seriálovým recallem*, nebo *coverage*. Při použití algoritmů v praxi by bylo dobré odhadnout optimální hodnotu parametru `c` pomocí A/B testování na základě požadavků na kvalitu doporučení filmů nebo seriálů.

V průběhu vývoje FRAMEWORKu bylo testováno více různých možností agregace epizod do seriálu. Jsou prezentovány pouze ty, které dosahovaly nejvyšších hodnot *recallu* a *coverage* a pomocí kterých bylo možné úspěšněji parametricky ovlivňovat *filmový* a *seriálový recall* a *coverage*.

4.3.3 Balancování filmů a seriálů

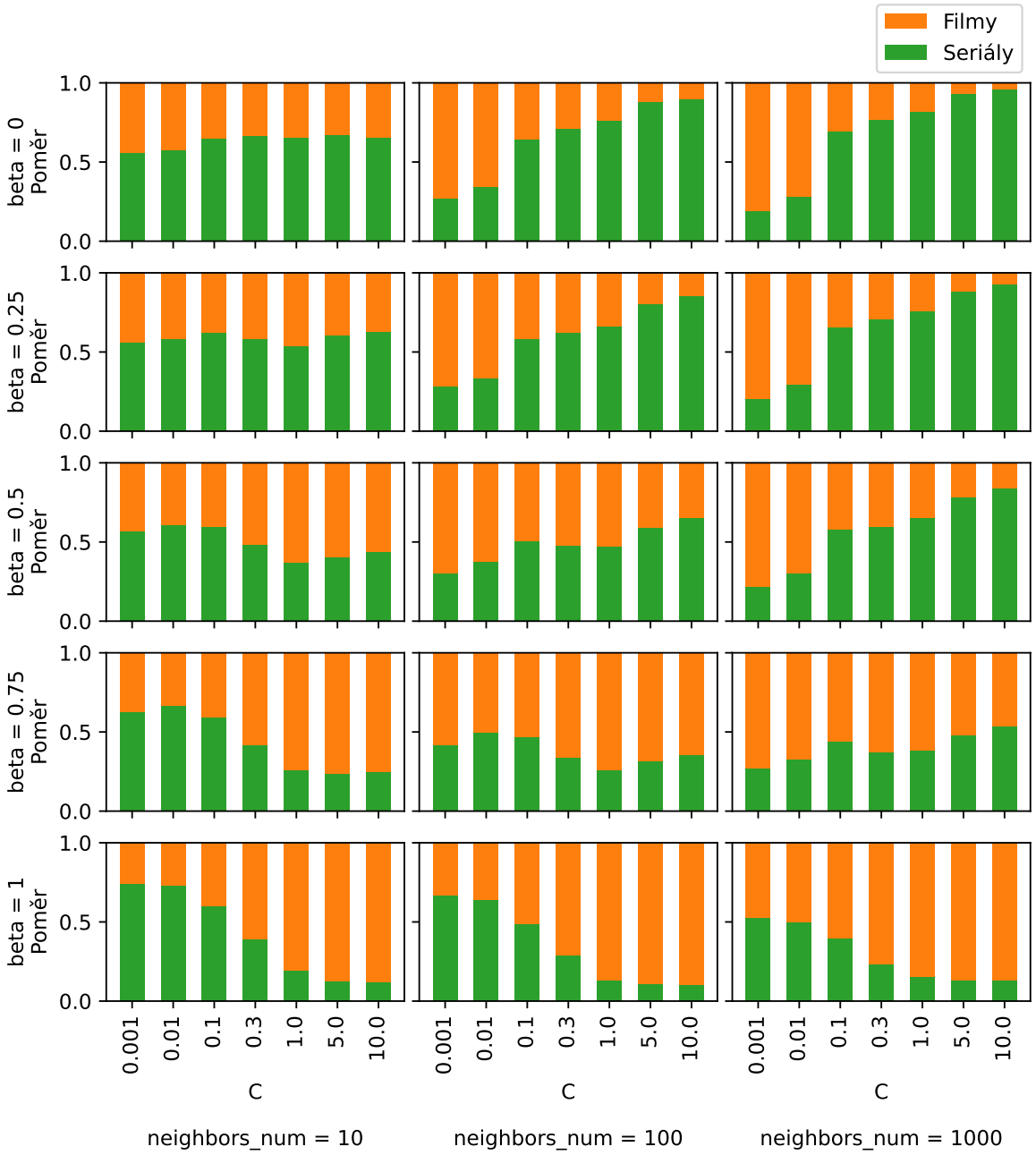
V této podkapitole se zaměříme na poměr mezi počtem doporučených filmů a a počtem doporučených seriálů. V první části podkapitoly budeme zkoumat tento poměr pro obecně všechny uživatele. V druhé části podkapitoly budeme zkoumat závislost tohoto poměru a poměru mezi zhlédnutými filmy a seriály v minulosti pro jednotlivé uživatele.

4.3.3.1 Vliv parametrů na poměr mezi počtem doporučených filmů a seriálů

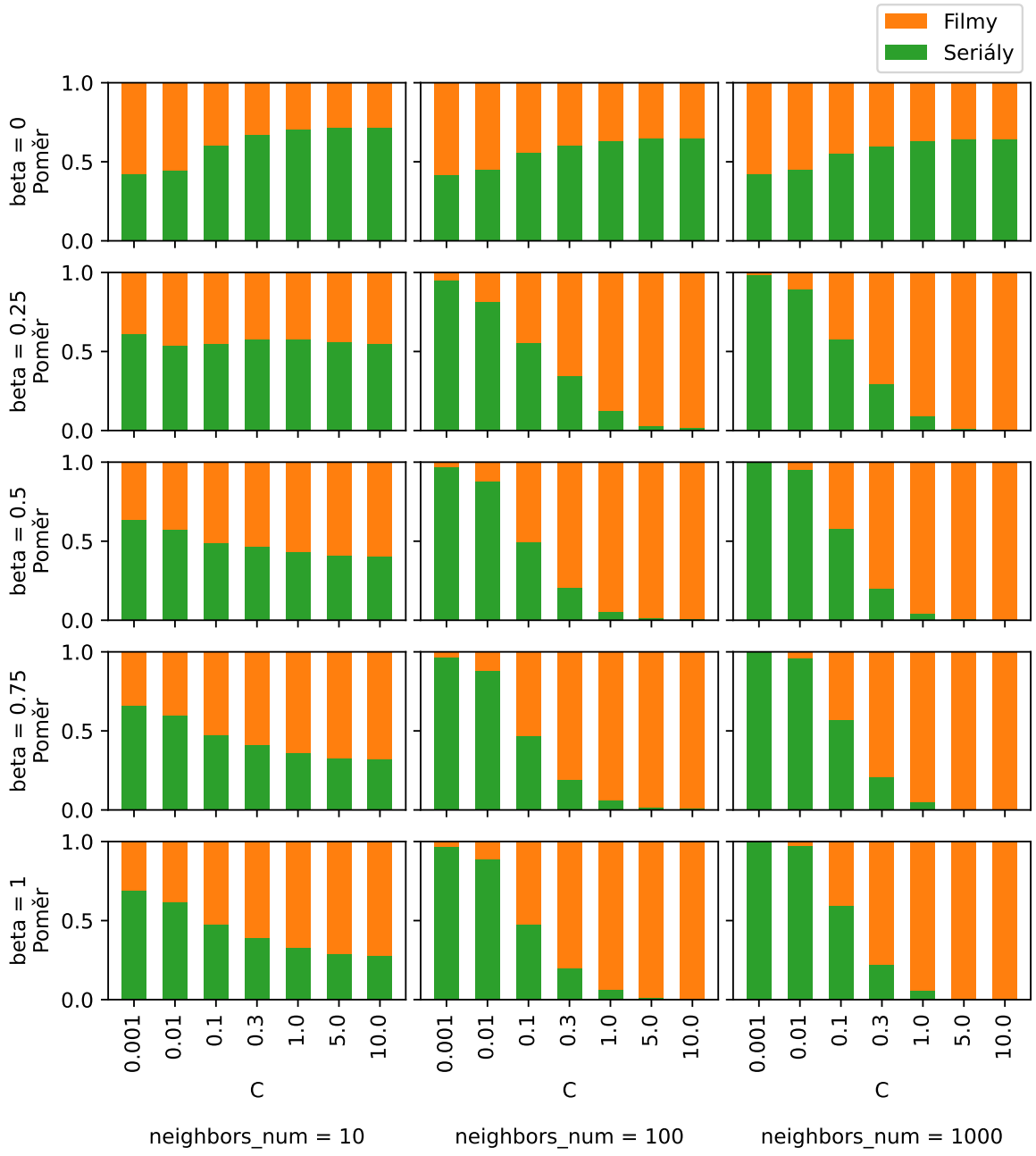
Nyní se zaměříme na poměr mezi počtem doporučených filmů a seriálů. Budeme využívat sčítací metodu agregace, která se v předchozích experimentech více osvědčila. Následují experiment se zaměřuje na poměr počtu doporučených filmů a seriálů v závislosti na agregačním parametru `c`. Parametr `max_val` v tomto experimentu není nijak omezený. Experiment je proveden pro více různých hodnot parametrů `beta` a `neighbors_num`. Experimenty se vyhodnocují na Top-10 doporučení. Vizualizace experimentů je tentokrát předvedena pouze pro dataset *D1*, protože pro oba datasety bylo dosaženo velice podobných výsledků.

Průběh experimentu s algoritmem **userKNN** je zobrazen na obrázku 4.9. Rostoucí parametr `c` zvyšuje hodnotu, kterou mají seriály v interakční matici. Intuitivně by tedy měl počet doporučených seriálů růst s rostoucím parametrem `c`. Očekávané chování můžeme pozorovat pro nižší hodnoty parametru `beta`. Při vyšších hodnotách parametru `beta` jsou položky s vyšší hodnotou v interakční matici více penalizovány, což způsobuje, že poměr doporučených seriálů už neroste s parametrem `c`. Pro `beta` = 1 dokonce poměr doporučených seriálů klesá s parametrem `c`.

Závislost poměru doporučených seriálů na parametru `c` ať už přímá, nebo pro větší hodnoty parametru `beta` nepřímá, je významnější s větším počtem sousedů. Čím vyšší parametr `neighbors_num`, tím je tato závislost významnější. Vážený algoritmus **userKNN** při počítání skóre pro nějakou položku sčítá hodnoty interakce všech nejbližších sousedů s touto položkou v trénovací interakční matici, vzoreček 1.4. Agregované hodnoty v interakční matici se přímo promítnou do výpočtu skóre. Když budou seriály mít v interakční matici velké nebo naopak malé hodnoty, tak jim bude spočítáno velké nebo naopak malé skóre. Při malém množství sousedů se však stává, že velikost množiny všech položek, se kterými uživatelé interagovali, je menší než parametr `topN` a jsou pak doporučeny i položky, kterým bylo spočítáno libovolně malé nenulové skóre. K tomu při hodnotě parametru `neighbors_num` = 10 dochází často, proto je závislost poměru doporučených seriálů na parametru `c` v tomto případě minimální. Pro hodnoty parametru `neighbors_num`, které jsou výrazně vyšší než hodnota parametru `topN`, se už stává minimálně, že by byla doporučena položka s malým skóre, proto pro hodnoty parametru



Obrázek 4.9: Závislost poměru počtu doporučených filmů a seriálů na parametru `c` pro různé parametry `beta` a `neighbors_num` při použití algoritmu **userKNN**. `aggregation` = SUM, `topN` = 10. Dataset = *D1*.



Obrázek 4.10: Závislost poměru počtu doporučených filmů a seriálů na parametru `c` pro různé parametry `beta` a `neighbors_num` při použití algoritmu **itemKNN**. `aggregation` = SUM, `topN` = 10. Dataset = *D1*

`neighbors_num` = 100 a `neighbors_num` = 1000 už je závislost poměru doporučených seriálů na parametru `c` významná.

Na obrázku 4.10 je zobrazen průběh experimentu s algoritmem **itemKNN**. Při použití váženého algoritmu **itemKNN** se skóre počítá podle vzorečku 1.7. Na rozdíl od **userKNN** se v **itemKNN** ve výpočtu skóre pro nějakou položku nikdy nezapočítává hodnota v interakční matici pro danou položku. Hodnota interakční matice se ve vzorečku započítává pouze za každou položku, ke které se hledají nejbližší sousedi. Mezi nejbližší sousedy nějakého seriálu může patřit i film a stejně tak naopak. Z toho důvodu je závislost poměru doporučených seriálů na parametru `c` méně významná než u **itemKNN**. Tato závislost začíná být významná pro vyšší hodnoty parametru `beta`, ale v tomto případě se jedná o nepřímou úměru mezi poměrem doporučených seriálů a parametrem `c` způsobenou penalizací položek s vysokými hodnotami v interakční matici. Znovu je míra zmíněné závislosti závislá na parametru `neighbors_num`, avšak méně než u algoritmu **userKNN**.

Parametr `c` se ukázal jako velice efektivní nástroj pro řízení poměru mezi počtem doporučených filmů a počtem doporučených seriálů. Při použití algoritmu **userKNN** roste poměr doporučených seriálů s rostoucí hodnotou parametru `c`. Tento efekt je významnější při použití vyšších hodnot parametru `neighbors_num` a nižších hodnot parametru `beta`. Při použití algoritmu **itemKNN** je pro efektivní manipulaci s poměrem doporučených filmů a seriálů vhodné volit vyšší hodnoty parametru `beta`. Na rozdíl od **userKNN** však s rostoucím parametrem `c` roste poměr doporučených filmů.

4.3.3.2 Závislost poměru doporučených seriálů na poměru zhlédnutých seriálů

Praktickým požadavkem při doporučování filmů a seriálů může být důraz na doporučování více seriálů uživatelům, kteří interagovali více se seriály a doporučování více filmů uživatelům, kteří interagovali více s filmy. Jinými slovy, snaha doporučit uživateli filmy a seriály v podobném poměru v jakém uživatel filmy a seriály v minulosti sledoval.

Provedli jsme experiment s cílem najít konfiguraci algoritmů **userKNN** a **itemKNN**, která doporučuje filmy a seriály uživatelům v co nejpodobnějším poměru, v jakém uživatelé s filmy a seriály v minulosti interagovali. Experiment probíhal na 1000 testovacích uživatelích. Pro každého testovacího uživatele byl spočítán poměr počtu seriálů, se kterými interagoval, vůči počtu všech interakcí daného uživatele s filmy i seriály. Zmíněný poměr označujeme jako poměr zhlédnutých seriálů. Dále byl pro různé modely pro každého uživatele počítán poměr doporučených seriálů. Tento poměr je poměr počtu doporučených seriálů vůči všem doporučeným položkám danému uživateli. Při měření je znovu využit *leave-one-out* přístup.

Oba zmíněné poměry nadefinujeme s dříve zavedeným značením. Nově zavedeme množinu Ser jako množinu seriálů $Ser = S^r \backslash F$. Mějme nějaký

model, který doporučuje N položek na základě uživatelových interakcí. Toto doporučení je reprezentováno funkcí Top. Pro každého testovacího uživatele $u \in U^{te}$ definujeme dva poměry:

$$\text{poměr zhlédnutých seriálů} = \frac{|\,\mathrm{RI}(u) \cap Ser|}{|\,\mathrm{RI}(u)|}$$

$$\text{poměr doporučených seriálů} = \frac{\sum\limits_{i \in \mathrm{RI}(u)} |\mathrm{Top}(N, \mathrm{RI}(u) \backslash \{i\}) \cap Ser|}{|\,\mathrm{RI}(u)| \times N}$$

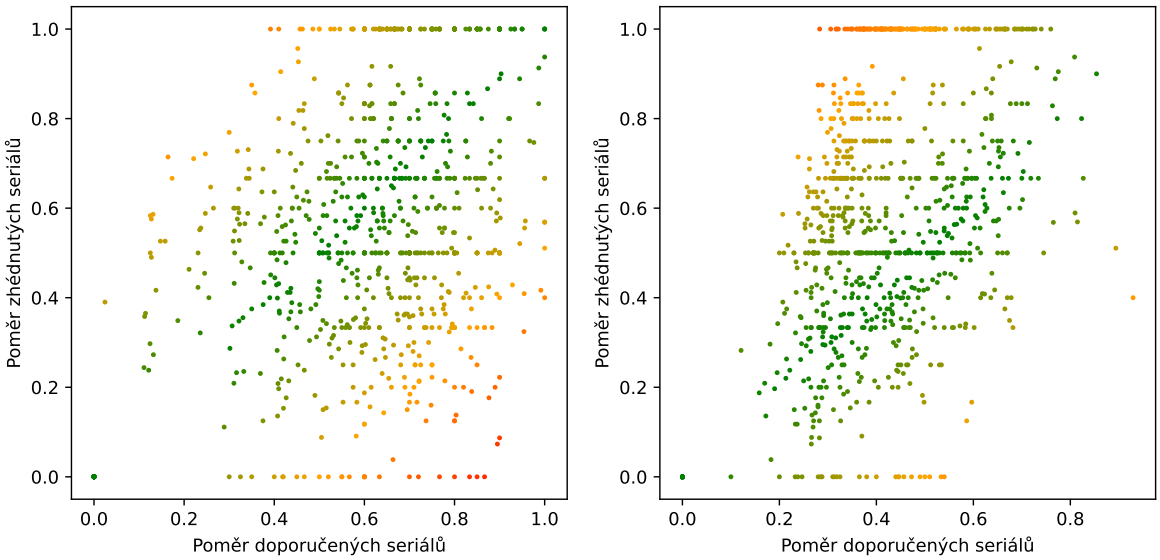
První poměr je definován pouze uživatelem, druhý poměr závisí i na použitém modelu a hodnotě N.

Závislost poměru doporučených seriálů na poměru zhlédnutých seriálů je vizualizována pomocí bodového grafu, ve kterém každý bod reprezentuje jednoho z 1000 testovacích uživatelů v rovině definované dvěma představenými poměry. Při stejném poměru zhlédnutých seriálů a doporučených seriálů je bod reprezentující uživatele umístěn na diagonále. Čím více se tyto dva poměry liší, tím dále je bod od diagonály. Body blíže diagonály mají zelenější barvu a body dále mají červenější barvu. Pro každé prezentované nastavení algoritmů jsou zobrazeny dva grafy. Jeden graf zobrazující závislost poměru doporučených seriálů na poměru zhlédnutých seriálů při Top-10 doporučení, druhý při Top-100 doporučení se stejným nastavením parametrů algoritmu.

Na obrázku 4.11 jsou zobrazeny zmíněné grafy pro dataset *D1* a algoritmus **userKNN** s nastavením parametrů, které zatím není nijak laděno pro dodržení co nejvyšší podobnosti poměrů. Parametry byly nastaveny na hodnoty, které dosáhly v experimentu zobrazeném na obrázku 4.4a vysokého *recallu*, tedy `aggregation` = CONSTANT, `beta` = 0.25, `neighbors_num` = 1000. Z vizualizace je možné pozorovat, jak moc je podobnost dvou poměrů pro různé uživatele dodržena bez optimalizace jednotlivých parametrů pro co nejvyšší podobnost poměrů. Při Top-10 doporučení se doporučovali více filmy než seriály a při Top-100 doporučení bylo doporučováno podobné množství seriálů i filmů. Není však možné pozorovat nějakou výraznou závislost poměru doporučených seriálů na poměru zhlédnutých seriálů pro většinu uživatelů.

Během experimentu bylo testováno, při jakém nastavení parametrů algoritmy **userKNN** a **itemKNN** nejlépe udržují rovnost mezi dvěma představenými poměry. Přesněji, při jakém nastavení parametrů je průměrný kvadratický rozdíl mezi poměrem zhlédnutých seriálů a poměrem doporučených seriálů pro každého testovacího uživatele nejmenší. Laděné parametry jsou `beta`, `neighors_num` a protože je využita sčítací agregace tak i parametr `c`. Parametr `max_val` = 1. Ladění parametrů probíhá na Top-10 doporučení.

Vizualizace závislosti mezi dvěma poměry pro algoritmus **userKNN** je zobrazena pro dataset *D1* na obrázku 4.12 a pro dataset *D2* na obrázku 4.13 a pro algoritmus **itemKNN** pro dataset *D1* na obrázku 4.14 a pro dataset *D2* na obrázku 4.15. Přesné hodnoty parametrů pro oba algoritmy a pro oba datasety jsou uvedeny u příslušných obrázků. Pro algoritmus **userKNN** bylo

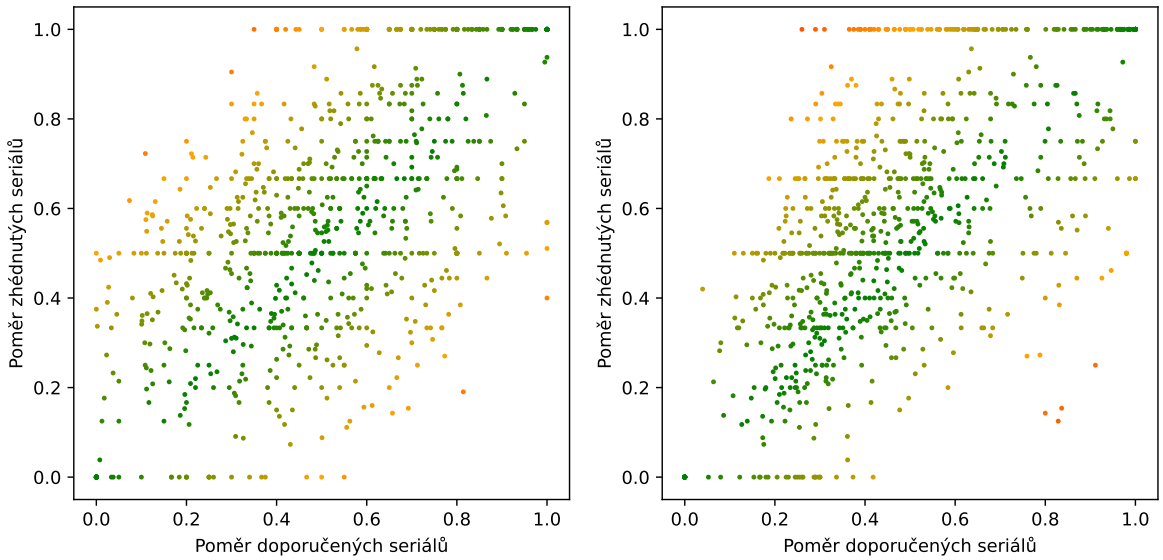


(a) `topN` = 10 (b) `topN` = 100

Obrázek 4.11: Závislost poměru doporučených seriálů na poměru zhlédnutých seriálů pro 1000 testovacích uživatelů. Algoritmus: **userKNN**, `beta` = 0.25, `neighbors_num` = 1000, `aggregation` = CONSTANT. Dataset = *D1*.

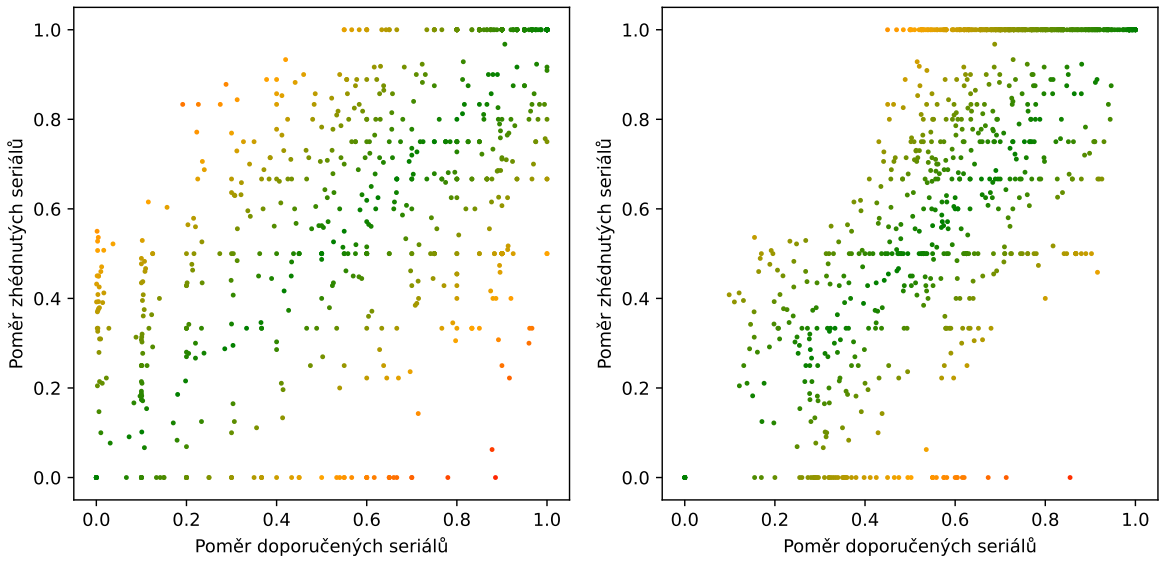
lepších výsledků dosaženo pomocí nižších hodnot parametru `beta`, vyšších hodnot parametru `neighbors_num` a nižších hodnot parametru `c`. Pro algoritmus **itemKNN** bylo lepších výsledků dosaženo pomocí nižších hodnot parametru `beta`, nižších hodnot parametru `neighbors_num` a vyšších hodnot parametru `c`. Pro oba algoritmy je rozdíl mezi nastavením experimentálně optimálních parametrů pro oba datasety minimální.

Na těchto grafech už můžeme pozorovat vyšší závislost mezi poměry než na grafu 4.11. Pro všechny uživatele sice není dosaženo dokonale stejného poměru doporučených seriálů, jako je poměr zhlédnutých seriálů, ale to ani není přesně cílem. Není na škodu doporučit uživateli, co sleduje pouze filmy, nějaký seriál, nebo naopak. Z grafů je však zřejmé, že při správném nastavení parametrů je dosaženo určité závislosti mezi těmito dvěma poměry. Velké množství uživatelů je umístěno blízko diagonály. V rozích grafu reprezentujících největší rozdíl mezi poměry je minimum uživatelů. Algoritmy při doporučování nijak nerozlišují mezi filmem a seriálem, závislost poměru doporučených seriálů na poměru zhlédnutých seriálů vzniká z interakčních dat. Značí to, že pouze na základě interakcí jsou obecně filmy více podobné filmům a seriály seriálům. Vyšší podobnosti mezi dvěma poměry dosahuje algoritmus **itemKNN**, jak je i z grafů viditelné, který dokáže více využít podobnosti mezi filmy a seriály, protože využívá při doporučování podobnosti položek.



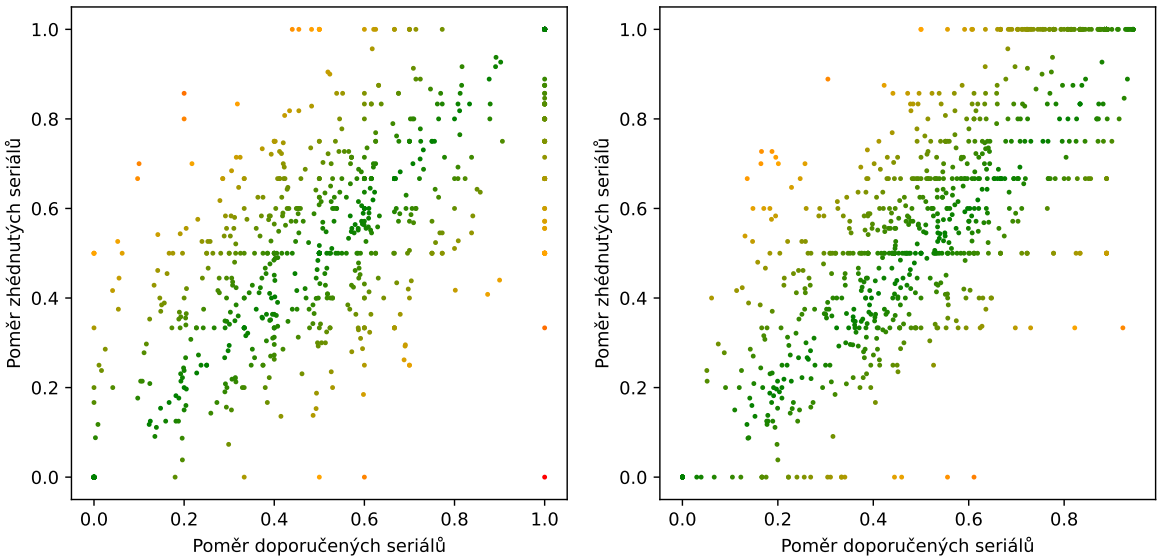
(a) `topN` = 10 (b) `topN` = 100

Obrázek 4.12: Závislost poměru doporučených seriálů na poměru zhlédnutých seriálů pro 1000 testovacích uživatelů. Algoritmus: **userKNN**, `beta` = 0.25, `neighbors_num` = 1000, `aggregation` = SUM, `c` = 0.1. Dataset = *D1*.



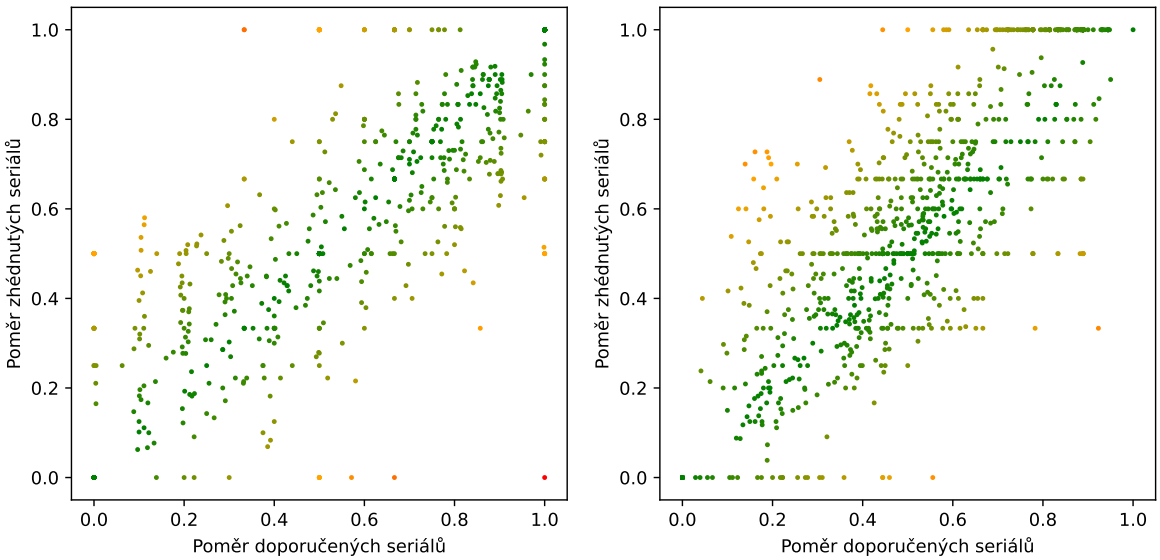
(a) `topN` = 10 (b) `topN` = 100

Obrázek 4.13: Závislost poměru doporučených seriálů na poměru zhlédnutých seriálů pro 1000 testovacích uživatelů. Algoritmus: **userKNN**, `beta` = 0, `neighbors_num` = 1000, `aggregation` = SUM, `c` = 0.1. Dataset = *D2*.



(a) `topN` = 10 (b) `topN` = 100

Obrázek 4.14: Závislost poměru doporučených seriálů na poměru zhlédnutých seriálů pro 1000 testovacích uživatelů. Algoritmus: **itemKNN**, `beta` = 0, `neighbors_num` = 10, `aggregation` = SUM, `c` = 0.7. Dataset = *D1*.



(a) `topN` = 10 (b) `topN` = 100

Obrázek 4.15: Závislost poměru doporučených seriálů na poměru zhlédnutých seriálů pro 1000 testovacích uživatelů. Algoritmus: **itemKNN**, `beta` = 0, `neighbors_num` = 10, `aggregation` = SUM, `c` = 0.5. Dataset = *D2*.

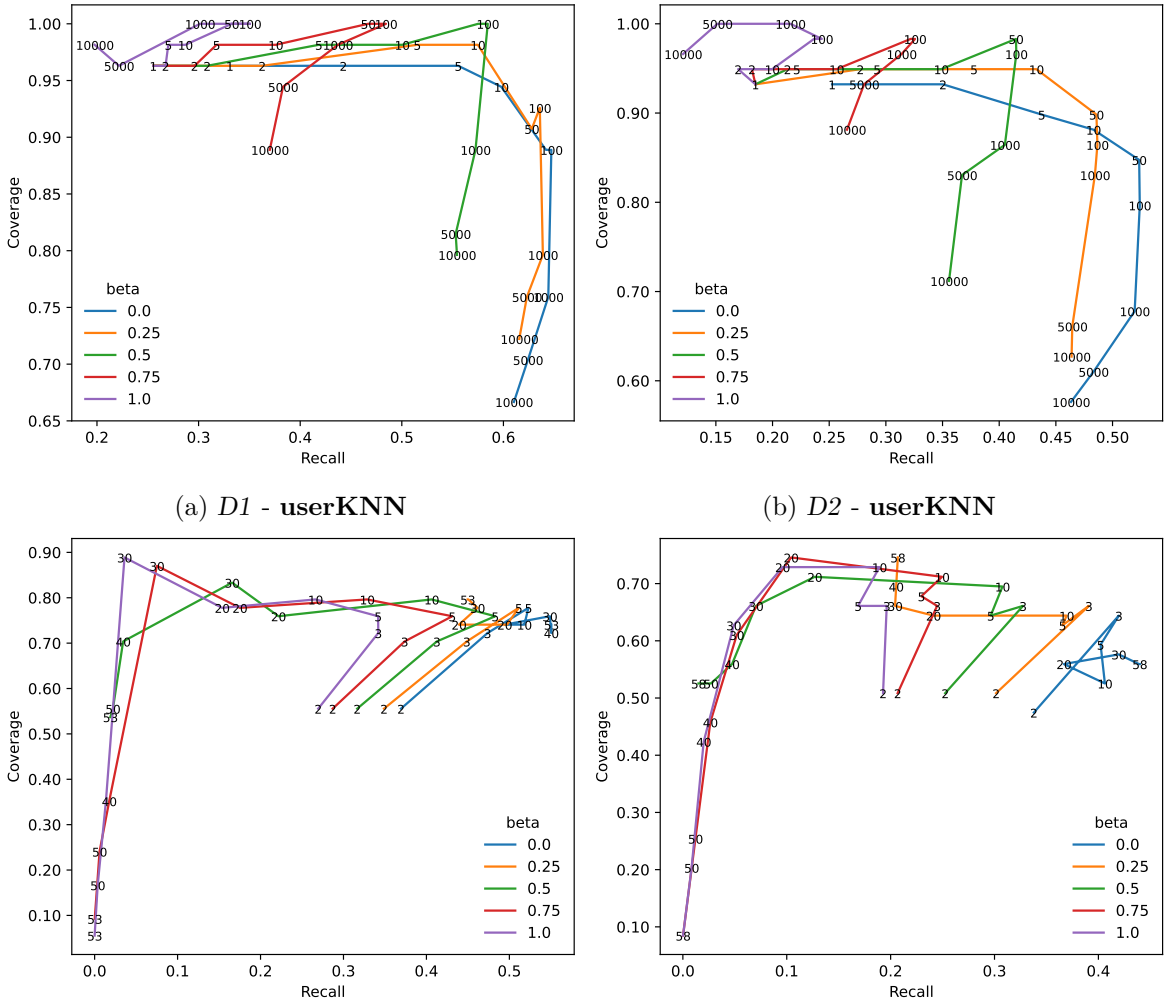
4.4 Doporučování žánrů

Následující experimenty se zaměřují na doporučování žánrů. Jedná se o další praktické využití segmentů. Jedna položka může mít jeden nebo více žánrů. U některých položek informace o žánrech chybí. Při práci s průmyslovými daty se často stává, že nějaká data chybí. Takové položky budou vyfiltrovány, protože neposkytují žádnou informaci o žánrech. Uživatelé, kteří neinteragovali s žádnou položkou s nadefinovaným žánrem, budou také vyfiltrováni. Naštěstí však velké množství položek má žánry nadefinované a není třeba vyfiltrovávat velké množství dat. Tabulka níže ukazuje pro oba datasety počty žánrů, položek s žánrem a uživatelů, kteří interagovali s alespoň jednou položkou s nadefinovaným žánrem.

datová sada	*D1*	*D2*
počet uživatelů	94474	97793
počet položek	64427	33381
počet různých žánrů	54	59
počet položek s nadefinovanými žánry	55585	31470
počet uživatelů, kteří interagovali nějakým žánrem	91258	97566

Běh algoritmu budeme znovu demonstrovat pomocí *recall-coverage* roviny. Pro následující experimenty byla provedena konstantní agregace, neboli `aggregation` = CONSTANT. Vzhledem k malému počtu segmentů využíváme při měření *recall* a *coverage* Top-1 doporučení. Pro **itemKNN** jsou nejsou experimenty provedeny pro vyšší hodnoty parametru `neighbors_num` než 50, protože počet sousedů nesmí překročit počet položek.

Průběh experimentů je zobrazen na obrázku 4.16. **UserKNN** dosahuje mírně lepších výsledků, dokonce je dosažena pro nějaké kombinace parametrů maximální *coverage*. *Recall* dosahuje i hodnot vyšších než 0.5, což není špatné při doporučování pouze jedné položky.



(a) *D1* - **userKNN**

(b) *D2* - **userKNN**

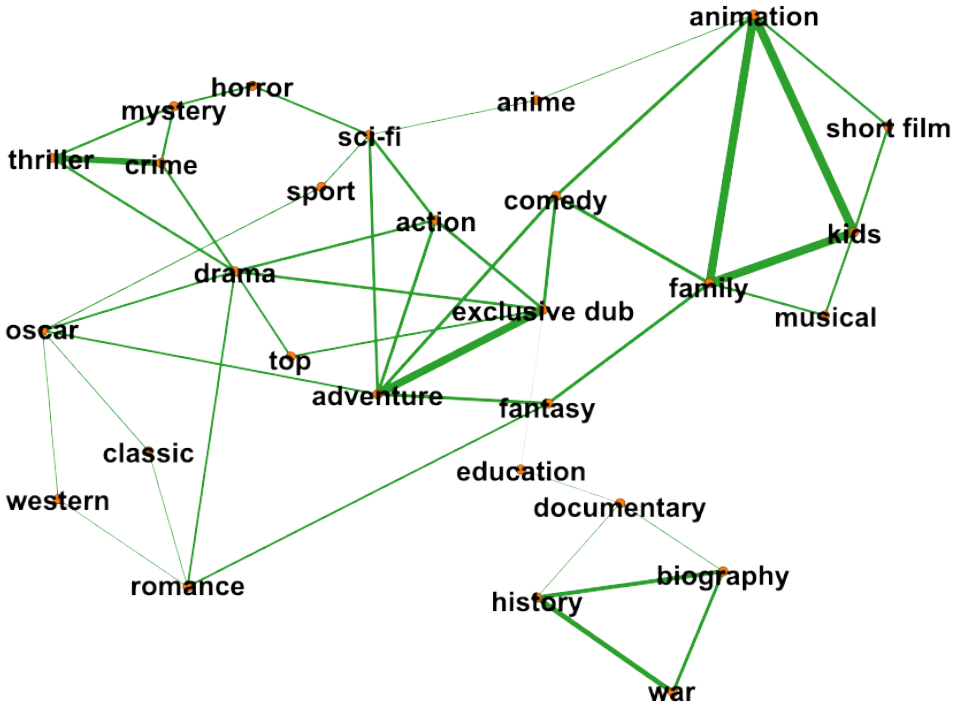
(c) *D1* - **itemKNN**

(d) *D2* - **itemKNN**

Obrázek 4.16: Vizualizace běhu algoritmů při segmentaci podle žánrů v *recall-coverage* rovině pro různé hodnoty parametrů `beta` a `neighbors_num` (hodnoty přímo v grafu). `aggregation` = CONSTANT, `topN` = 1.

4.4.1 Podobnost segmentů na základě interakčních dat

Nyní vizualizujeme podobnost mezi jednotlivými žánry pouze na základě interakčních dat. K vizualizaci je využito softwaru gephi [23]. Software gephi umí vizualizovat grafy. Jednotlivé žánry jsou vrcholy vizualizovaného grafu. Z každého vrcholu reprezentujícího nějaký žánr vede hrana ke dvěma nejpodobnějším žánrům, kde podobnost je počítána jako cosinová podobnost v trénovací interakční matici vytvořené v předchozím experimentu. Tloušťka hrany reprezentuje hodnotu cosinové podobnosti, případně součet podobností, patří-li oba žánry, které hrana spojuje, mezi dva nejbližší žánry toho druhého žánru. Pro přehlednost nejsou vizualizovány všechny žánry, ale pouze takové žánry, pod které spadá minimálně 500 položek.

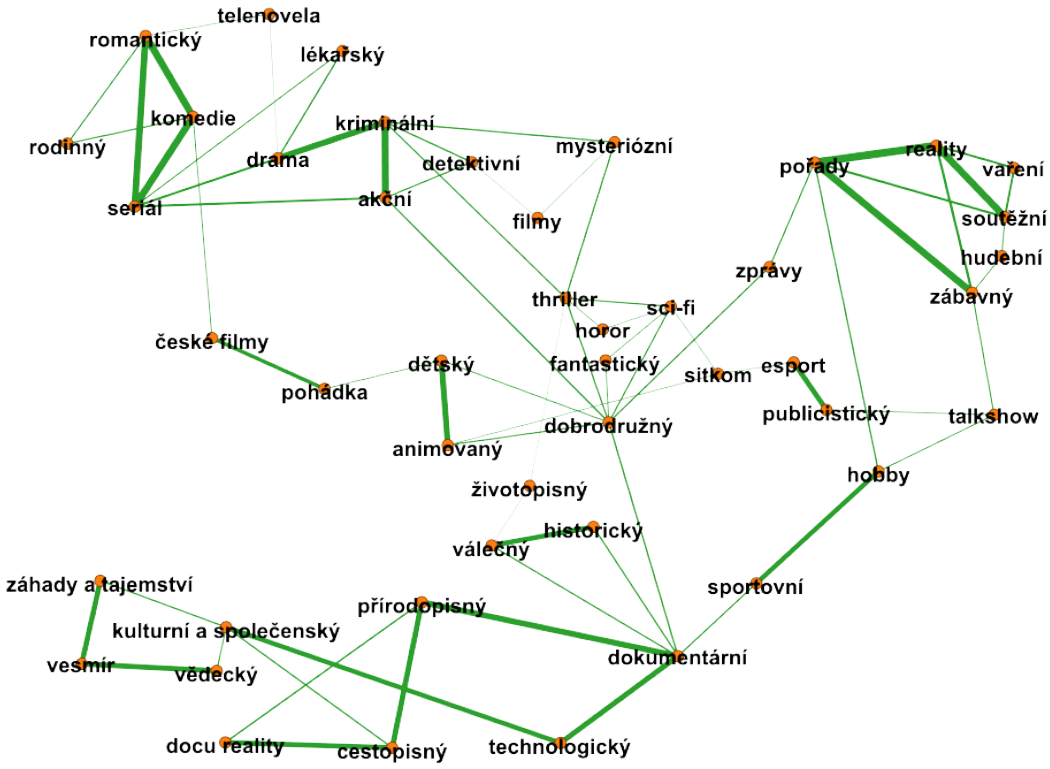


Obrázek 4.17: Vizualizace podobností mezi žánry na základě interakčních dat, Dataset = *D1*.

Podobnost mezi žánry je pro dataset *D1* vizualizována na obrázku 4.17 a pro dataset *D2* na obrázku 4.18. Z vizualizací je zřejmé, že z interakčních dat je možné dobře odvodit logickou podobnost mezi žánry. Podobné žánry

jsou spojeny hranou a podobné skupiny žánrů jsou blízko u sebe a vytvářejí logické shluky. Žánry ke kterým vedou pouze dvě hrany, se více odlišují od ostatních žánrů. Mezi takové žánry patří v obou datasetech například horor. Žánry ke kterým vede více hran, mají hodně podobných žánrů, například žánr dobrodružný v obou datasetech.

Z tohoto experimentu vyplývá, že pouze na základě interakčních dat lze hledat podobnosti mezi segmenty. Vizualizace ukazují další náhled na funkčnost segmentů mimo pouhé vyhodnocování metrik *recall* a *coverage*.



Obrázek 4.18: Vizualizace podobností mezi žánry na základě interakčních dat, Dataset = *D2*.

4.5 Doporučování herců

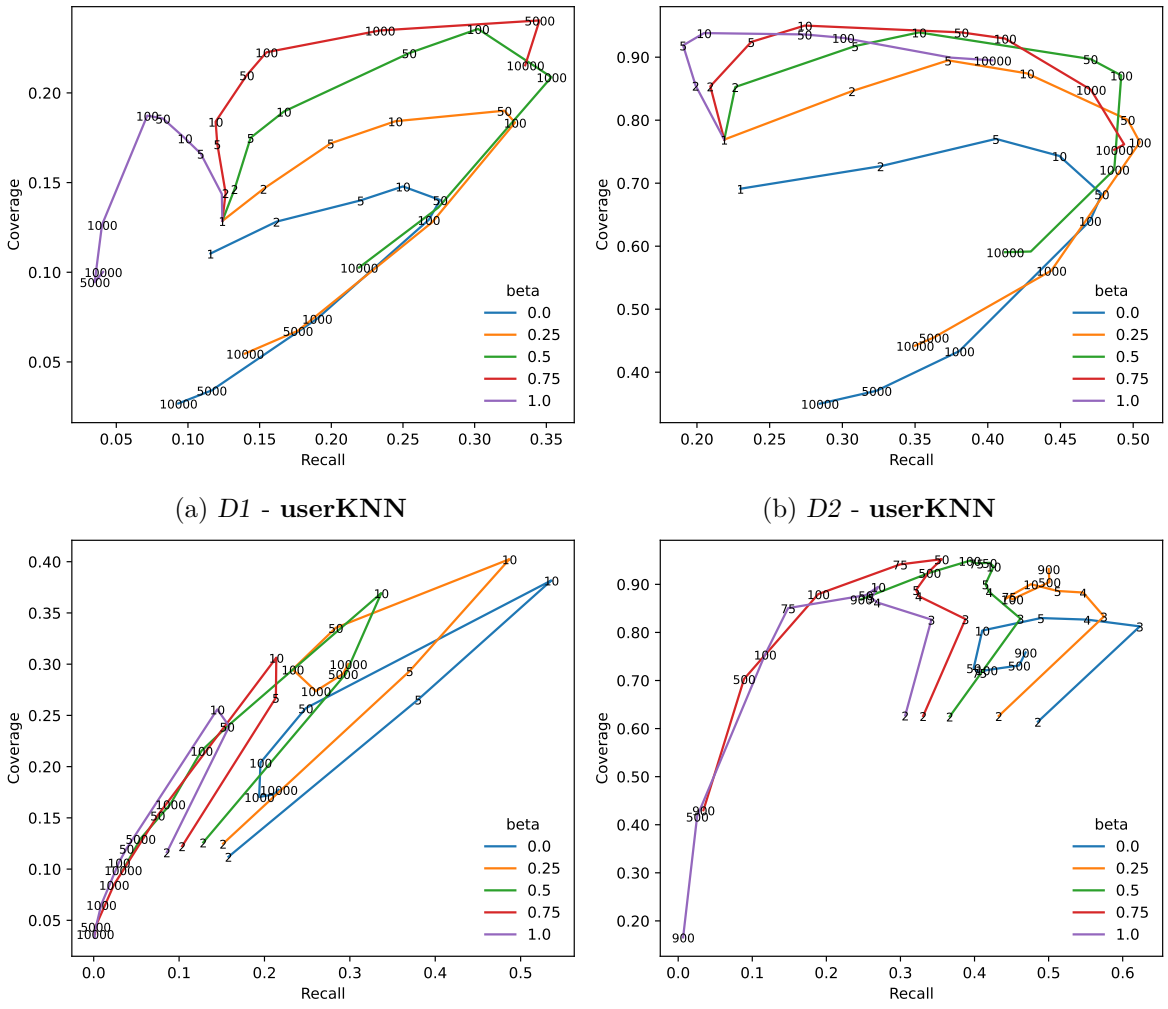
Posledním hierarchickým uspořádáním, se kterým budeme experimentovat, jsou herci. Velké množství položek bohužel nemá herce nadefinováno, experimenty tedy jsou prováděny na menším množství dat. Na rozdíl od předchozích hierarchických uspořádání, se kterými bylo experimentováno, je u herců větší rozdíl mezi oběma datasety. Dataset *D1* má nadefinovány herce pro větší množství položek než dataset *D2*. Dataset *D1* obsahuje mezinárodní obsah a dataset *D2* obsahuje primárně český obsah, z toho důvodu je u datasetu *D1* řádově více různých herců.

datová sada	*D1*	*D2*
počet uživatelů	94474	97793
počet položek	64427	33381
počet různých herců	38206	923
počet položek s nadefinovanými herci	11627	1925
počet uživatelů, kteří interagovali nějakým hercem	71682	16874

Průběh experimentů je znovu demonstrován pomocí *recall-coverage* roviny. Parametry algoritmů jsou nastaveny na stejné hodnoty jako u experimentů s doporučováním žánrů, aby bylo možné doporučování žánrů a doporučování herců porovnávat. Oproti předchozím experimentům byl počet testovacích uživatelů zmenšen na 1000 z důvodu malého počtu uživatelů interagujících s nějakým hercem v datasetu *D2*.

Experimenty probíhají pomocí Top-1 doporučení, aby bylo možné průběh experimentů porovnat s experimenty provedenými na doporučování žánrů. Průběh experimentů pro oba datasety a pro algoritmy **userKNN** a **itemKNN** je zobrazen na obrázku 4.19. Na datasetu *D2* bylo dosaženo vyššího *recallu* a výrazně vyšší *coverage*. To je způsobeno menším množstvím segmentů. Při porovnání průběhu algoritmů při doporučování žánrů a při doporučování herců je možné pozorovat, že pro jiné segmenty se mohou lišit hodnoty parametrů, pomocí kterých je dosaženo nejvyšších hodnot metrik *recall* a *coverage*. Ale při doporučování stejného segmentu pro různé datasety je dosaženo nejvyšších hodnot metrik pomocí velice podobného nastavení parametrů.

Při doporučování žánru dosahoval vyššího *recallu* algoritmus **userKNN** a při doporučování herců dosahoval vyššího *recallu* algoritmus **itemKNN**, to platí stejně pro oba datasety. Při doporučování žánrů pomocí algoritmu **userKNN** bylo nejvyššího *recallu* dosaženo při nastavení parametrů `beta` = 0 nebo `beta` = 0.25 a `neighbors_num` kolem hodnot 50 až 1000. Při doporučování herců algoritmem **userKNN** bylo dosaženo nejvyšších hodnot *recallu* pomocí nastavení algoritmů `beta` = 0.25, `beta` = 0.5 a `beta` = 0.75 s různým nastavením parametru `neighbors_num` pro různé hodnoty parametru `beta` a nejlepší nastavení parametrů je velice podobné pro oba datasety. Při doporučování žánrů algoritmem **itemKNN** bylo nejvyšších hodnot *recallu* dosa-



(a) *D1* - **userKNN**

(b) *D2* - **userKNN**

(c) *D1* - **itemKNN**

(d) *D2* - **itemKNN**

Obrázek 4.19: Vizualizace běhu algoritmů při segmentaci podle herců v *recall-coverage* rovině pro různé hodnoty parametrů `beta` a `neighbors_num` (hodnoty přímo v grafu). `aggregation` = CONSTANT, `topN` = 1.

ženo pomocí nastavení parametrů `beta` = 0 a vyšších hodnot parametru `neighbors_num` dosahujících až maximální možné hodnoty, což je počet segmentů bez jedné. Při doporučování herců algoritmem **itemKNN** bylo nejvyšších hodnot *recallu* dosaženo pomocí nastavení parametrů také `beta` = 0, ale nižších hodnot parametru `neighbors_num`, konkrétně `neighbors_num` = 10 a `neighbors_num` = 3. Při porovnání doporučování žánrů a doporučování herců s doporučováním seriálů na obrázku 4.4, je možné pozorovat, že při doporučování seriálů je optimální nastavení parametrů znovu rozdílné, ale znovu podobné pro oba datasety.

Ukázalo se, že pro stejné segmenty je dosaženo nejvyšší kvality doporučení na různých datasetech pomocí podobného nastavení parametrů doporučovacích algoritmů. Pro jiné segmenty může být optimální nastavení doporučovacích algoritmů rozdílné. Při použití segmentačních algoritmů v praxi by bylo dobré nastavit parametry algoritmů doporučujících nějaký segment například pomocí A/B testování. Takto nastavený algoritmus by měl fungovat dobře i na jiných platformách při doporučování stejných logických segmentů.

Závěr

V klasických doporučovacích přístupech se doporučují uživatelům položky. Tato práce se zaměřuje na málo prozkoumané téma v doporučování, kterým je doporučování prvků hierarchické struktury, do které položky spadají. Tyto prvky hierarchické struktury se označují jako segmenty.

V práci je zavedena formální definice hierarchické struktury. Dále je navržen FRAMEWORK, který upravuje klasické doporučovací algoritmy tak, aby pomocí nich bylo možné doporučovat i segmenty. Pomocí upravených algoritmů je možné doporučovat položky, segmenty, nebo smíšený obsah — položky a segmenty najednou. Algoritmy jsou otestovány na dvou průmyslových datasetech streamingových platforem. Testování algoritmů probíhá offline. Úspěšnost se měří pomocí evaluačních metrik.

Experimentuje se s více různými představenými způsoby převedení položek na segmenty. Testuje se vliv různých parametrů doporučovacích algoritmů na kvalitu doporučení.

Velká část experimentální kapitoly je věnována experimentům na doporučování smíšeného obsahu, přesněji filmů a seriálů, protože se jedná o velice praktické využití doporučování segmentů. Jsou provedeny experimenty testující vliv parametrů algoritmů na zastoupení položek a segmentů v doporučení. Ukazuje se, že pomocí způsobu agregace položek do segmentů a pomocí parametrů doporučovacích algoritmů se dá velice dobře ovlivňovat poměr mezi počtem doporučených filmů a seriálů.

Při doporučování segmentů se experimentuje s doporučováním žánrů a herců. Doporučovací algoritmy v kombinaci se způsoby převodu položek na segmenty fungují uspokojivě. Pro různé segmenty bylo dosaženo nejlepších výsledků pomocí různých nastavení parametrů, ale v rámci jednoho segmentu bylo vždy dosaženo nejlepších výsledku pro oba datasety s velice podobným nastavením parametrů. To značí dobrou přenositelnost algoritmů i na jiné datasety. Dále byla vizualizována podobnost žánru na základě interakčních dat a ukázalo se, že interakční data definují v segmentech logické shluky při správně provedeném převodu položek na segmenty.

Mezi nejdůležitější možné rozšíření do budoucna patří přidání podpory více algoritmů pro doporučování segmentů. FRAMEWORK je navržený tak, aby bylo možné pro doporučování jednoduše použít jakýkoliv algoritmus pracující s interakční maticí. Toto rozšíření by nemělo být tedy náročné na implementaci, ale bylo by nutné otestovat, jestli doporučování segmentů pomocí jiných algoritmů funguje stejně dobře. Dalším důležitým rozšířením je otestovat navržený FRAMEWORK v praxi na online datech.

Literatura

[1] Goldberg, D.; Nichols, D.; Oki, B. M.; aj.: Using Collaborative Filtering to Weave an Information Tapestry. *Commun. ACM*, ročník 35, č. 12, dec 1992: str. 61–70, ISSN 0001-0782, doi:10.1145/138859.138867. Dostupné z: https://doi.org/10.1145/138859.138867

[2] Ning, X.; Desrosiers, C.; Karypis, G.: *A Comprehensive Survey of Neighborhood-Based Recommendation Methods*. Boston, MA: Springer US, 2015, ISBN 978-1-4899-7637-6, s. 37–76, doi:10.1007/978-1-4899-7637-6_2. Dostupné z: https://doi.org/10.1007/978-1-4899-7637-6_2

[3] Nakhli, R. E.; Moradi, H.; Sadeghi, M. A.: Movie Recommender System Based on Percentage of View. In *2019 5th Conference on Knowledge Based Engineering and Innovation (KBEI)*, 2019, s. 656–660, doi:10.1109/KBEI.2019.8734976.

[4] Řehořek, T.: *Manipulating the Capacity of Recommendation Models in Recall-Coverage Optimization.* Dizertační práce, Ceské vysoké učení technické v Praze, 2019. Dostupné z: https://dspace.cvut.cz/handle/10467/81823

[5] Jannach, D.; Zanker, M.; Felfernig, A.; aj.: *Recommender Systems: An Introduction*. Cambridge University Press, 2010, ISBN 9781139492591. Dostupné z: https://books.google.cz/books?id=eygTJBd_U2cC

[6] Cañamares, R.; Castells, P.: A Probabilistic Reformulation of Memory-Based Collaborative Filtering: Implications on Popularity Biases. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '17, New York, NY, USA: Association for Computing Machinery, 2017, ISBN 9781450350228, str. 215–224, doi:10.1145/3077136.3080836. Dostupné z: https://doi.org/10.1145/3077136.3080836

[7] Cremonesi, P.; Koren, Y.; Turrin, R.: Performance of Recommender Algorithms on Top-n Recommendation Tasks. In *Proceedings of the Fourth ACM Conference on Recommender Systems*, RecSys '10, New York, NY, USA: Association for Computing Machinery, 2010, ISBN 9781605589060, str. 39–46, doi:10.1145/1864708.1864721. Dostupné z: https://doi.org/10.1145/1864708.1864721

[8] Ge, M.; Delgado-Battenfeld, C.; Jannach, D.: Beyond Accuracy: Evaluating Recommender Systems by Coverage and Serendipity. In *Proceedings of the Fourth ACM Conference on Recommender Systems*, RecSys '10, New York, NY, USA: Association for Computing Machinery, 2010, ISBN 9781605589060, str. 257–260, doi:10.1145/1864708.1864761. Dostupné z: https://doi.org/10.1145/1864708.1864761

[9] Herlocker, J. L.; Konstan, J. A.; Terveen, L. G.; aj.: Evaluating Collaborative Filtering Recommender Systems. *ACM Trans. Inf. Syst.*, ročník 22, č. 1, jan 2004: str. 5–53, ISSN 1046-8188, doi:10.1145/963770.963772. Dostupné z: https://doi.org/10.1145/963770.963772

[10] Campochiaro, E.; Casatta, R.; Cremonesi, P.; aj.: Do Metrics Make Recommender Algorithms? In *2009 International Conference on Advanced Information Networking and Applications Workshops*, 2009, s. 648–653, doi:10.1109/WAINA.2009.127.

[11] Wei, S.; Ye, N.; Zhang, S.; aj.: Item-Based Collaborative Filtering Recommendation Algorithm Combining Item Category with Interestingness Measure. In *2012 International Conference on Computer Science and Service System*, 2012, s. 2038–2041, doi:10.1109/CSSS.2012.507.

[12] Wang, S.; Tang, J.; Wang, Y.; aj.: Exploring Hierarchical Structures for Recommender Systems. *IEEE Transactions on Knowledge and Data Engineering*, ročník 30, č. 6, 2018: s. 1022–1035, doi:10.1109/TKDE.2018.2789443.

[13] Ahmed, M.; Paul, A.; Imtiaz, M. T.; aj.: TV series recommendation using fuzzy inference system, K-Means clustering and adaptive neuro fuzzy inference system. In *2017 13th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD)*, 2017, s. 1540–1547, doi:10.1109/FSKD.2017.8392994.

[14] Oh, J.; Kim, S.; Kim, J.; aj.: When to recommend: A new issue on TV show recommendation. *Information Sciences*, ročník 280, 2014: s. 261–274, ISSN 0020-0255, doi:https://doi.org/10.1016/j.ins.2014.05.003. Dostupné z: https://www.sciencedirect.com/science/article/pii/S0020025514005210

[15] Martinez, A. B. B.; Arias, J. J. P.; Vilas, A. F.; aj.: What's on TV tonight? An efficient and effective personalized recommender system of TV programs. *IEEE Transactions on Consumer Electronics*, ročník 55, č. 1, 2009: s. 286–294, doi:10.1109/TCE.2009.4814447.

[16] Aharon, M.; Hillel, E.; Kagian, A.; aj.: Watch-It-Next: A Contextual TV Recommendation System. In *Machine Learning and Knowledge Discovery in Databases*, editace A. Bifet; M. May; B. Zadrozny; R. Gavalda; D. Pedreschi; F. Bonchi; J. Cardoso; M. Spiliopoulou, Cham: Springer International Publishing, 2015, ISBN 978-3-319-23461-8, s. 180–195.

[17] Scipy. 2023. Dostupné z: `https://github.com/scipy/scipy`

[18] Virtanen, P.; Gommers, R.; Oliphant, T. E.; aj.: SciPy 1.0: fundamental algorithms for scientific computing in Python. *Nature Methods*, ročník 17, Únor 2020: s. 261–272, doi:10.1038/s41592-019-0686-2, 1907.10121.

[19] Numpy. 2023. Dostupné z: `https://github.com/numpy/numpy`

[20] Sklearn. 2023. Dostupné z: `https://github.com/scikit-learn/scikit-learn`

[21] Pandas. 2023. Dostupné z: `https://github.com/pandas-dev/pandas`

[22] Elafrou, A.; Goumas, G.; Koziris, N.: A lightweight optimization selection method for Sparse Matrix-Vector Multiplication. 11 2015.

[23] Gephi. 2023. Dostupné z: `https://gephi.org`

Příloha **A**

Obsah elektronické přílohy

Mezi přiloženými soubory je soubor, ve kterém jsou všechny implementované funkce a modely. Dále notebook ukazující použití modelů při různých experimentech na datasetu *D1*. Poslední notebook obsahuje vizualizace provedených experimentů. Soubory se vstupními daty nejsou přiložené z důvodu toho, že se jedná o soukromá data, které Recombee nemůže zveřejnit.

`functions.py` ..... Implementované funkce
`experiments_data.ipynb` ..... Notebook s experimentama
`experiments_plots.ipynb` ..... Notebook s grafama