Jihočeská univerzita

Přírodovědecká fakulta

Ústav aplikované informatiky

Analýza a návrh modulu doporučovacího systému

Diplomová práce

Bc. Lukáš Kortus

Vedoucí práce: doc. Ing. Ladislav Beránek, CSc., MBA

České Budějovice 2016

NÁZEV:

Kortus L., 2016 : Analýza a návrh modulu doporučovacího systému [Recommendation system module analysis and design. Mgr. Thesis, in Czech.] - 138.p, Faculty of Science, University of South Bohemia, České Budějovice, Czech Republic

ABSTRAKT:

Doporučovací systémy slouží uživatelům E-commerce aplikací k individuálnímu doporučení určitých produktů nebo služeb na základě jejich preferencí. Cílem této práce je vytvoření modulu doporučovacího systému. Součástí práce je analýza doporučovacích systémů a metod využívaných v těchto systémech včetně popisu výpočtu. Práce dále také řeší problém studeného startu, což je problém, kdy je potřeba dát určité kvalitní doporučení pro nového uživatele, o kterém ale doporučovací systém nemá žádné nebo málo informací. Na základě analýzy je v práci navržen modul doporučovacího systému použitelného např. pro Internetový e-shop nebo jinou aplikaci založenou na Internetu. Součástí tohoto modulu je realizování platformy Apache Mahout, jejíž některé části jsou postaveny na distribuovaném výpočetním projektu Apache Hadoop. Dále jsou v této práci otestovány, na již zmíněné platformě Mahout, vybrané metody výpočtu podobností pomocí zvolených kritérií (např. průměrný čas na jedno doporučení a počet uživatelů, kterým nebylo možné vygenerovat doporučení).

KLÍČOVÁ SLOVA:

uživatelská preference, doporučovací systémy, Mahout

TITLE:

Kortus L., 2016 : Analýza a návrh modulu doporučovacího systému [Recommendation system module analysis and design. Mgr. Thesis, in Czech.] - 138.p, Faculty of Science, University of South Bohemia, České Budějovice, Czech Republic

SUMMARY:

Recommendation systems serve to users of e-commerce applications for individual recommendations to certain products or services based on their preferences. The aim of this thesis is to create a module of recommender system. The work includes analysis of recommendation systems and the methods used in these systems, including a description of the calculations. This work also solves the cold start problem, which is a problem when generation of some good recommendations for the new user is needed, but the recommendation system has no or little information about this user. Based on analysis is in this thesis designed module for recommender system, which is applicable e.g. internet for e-commerce or other internet-based application. Part of this module is the realization of a platform Apache Mahout, which some parts are built on a distributed computing platform Apache Hadoop project. Furthermore, in this work, on the aforementioned platform Mahout, selected methods of calculating the similarity using selected criteria (e.g. the average time for a recommendation, and the number of users for who have not been able to generate recommendations) are tested.

KEYWORDS:

user preferences, recommendation system, Mahout

Prohlašuji, že svoji diplomovou práci jsem vypracoval samostatně pouze s použitím pramenů a literatury uvedených v seznamu citované literatury. Prohlašuji, že v souladu s § 47b zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své diplomové práce, a to v nezkrácené podobě elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách, a to se zachováním mého autorského práva k odevzdanému textu této kvalifikační práce. Souhlasím dále s tím, aby toutéž elektronickou cestou byly v souladu s uvedeným ustanovením zákona č. 111/1998 Sb. zveřejněny posudky školitele a oponentů práce i záznam o průběhu a výsledku obhajoby kvalifikační práce. Rovněž souhlasím s porovnáním textu mé kvalifikační práce s databází kvalifikačních prací Theses.cz provozovanou Národním registrem vysokoškolských kvalifikačních prací a systémem na odhalování plagiátů.

České Budějovice 10. 12. 2015 …..…….

Obsah

1 Úvod

Dnes již žijeme v době, kdy osobní počítač a internetové připojení jsou již dostupné pro každého bez ohledu na věk, postavení ve společnosti či hodnotě majetku jako to bylo v dřívějších dobách. Z pohledu obchodníků se zde otevřela nová příležitost v podnikání a nabízení zboží. V případě, že si chce zákazník koupit zboží, už není pro něj nutné navštívit kamennou prodejnu kde si zboží objednat, ale může využít internetových služeb, které daný obchod poskytuje a objednat si zboží v tzv. e-commerce. Patří sem ale nejen elektronický prodej zboží ale i nabízení služeb, elektronická výměna nejrůznějších dat, vedení bankovních účtů, obchodní aukce a další.

Tyto systémy přináší pro zákazníka mnoho výhod, který je dnešní hektické době využívá hlavně kvůli jejich menší časové náročnosti v porovnání s kamennými obchody. Dalšími výhodami pro uživatele těchto systémů jsou dostupnost kdykoliv a odkudkoliv, kde je stabilní internetové připojení a odpovídající technika, dále rychlost vyhledání požadované položky a někdy dokonce mu tyto systémy pomáhají při rozhodování pomocí vestavěných nástrojů či hodnocení jiných uživatelů.

Hlavním zájmem provozovatele je rychlé a efektivní zobrazení těch položek, které by si zákazník potenciálně chtěl koupit. Tomu pomáhají moduly, které využívají různých technik k predikci uživatelových preferencí a vybrání takové množiny položek, u kterých je předpokládán zájem uživatele. Tyto moduly se nazývají doporučovací systémy a je možné je i kombinovat. Doporučovací systémy využívají uživatelovi zpětné vazby k doporučení položek. Nejen že pomáhají uživateli při výběru, ale navíc umožnují individuálně cílit zboží na jednotlivé zákazníky a tím zvyšovat počet prodaných položek.

Tato práce se zabývá problematikou doporučovacích systémů. Činí tak po stránce teoretické i praktické. Zabývá se analýzou doporučovacích systémů, metodami výpočtu, dále pak návrhem a naprogramováním modulu. Vzhledem k tomu, že doporučovací systémy jsou pro web efektivní jen při nízké chybovosti, je součástí

práce také řešení problému studeného startu a nalezení nejoptimálnější kombinace metod podobností k maximální přesnosti.

2 Cíle práce

Cíle práce, které byly stanoveny, jsou následující:

- Analýza doporučovacích systémů
    1. Popsat cíle uživatele a jeho preference
    2. Vybrat zastoupení doporučovacích systémů a provést jejich popis
    3. Popsat druhy a principy doporučovacích systémů
    4. Popsání Apache Mahout

- Návrh doporučovacího modulu
    1. Návrh a naprogramování doporučovacího systému
    2. Řešení problému studeného startu
    3. Testování a nalezení optimální kombinace metod výpočtu kolaborativního doporučení pro 100 000 hodnocení.

3 Chování uživatele

3.1 Cíle uživatele

Cíl uživatele je soubor očekávaných interakcí s webovou stránkou, případně s konkrétním webovým serverem. Od uživatele se očekává formulace svého cíle nejsnadnějším a nepřirozenějším způsobem tedy v přirozeném jazyce. Například „Chci auto značky Audi a chci o něm veškeré informace“. Uživatelské cíle při dotazování na webu můžeme přibližně rozdělit do 3 kategorií: [1]

- **Navigační cíle**

  Uživatel zná, nebo předpokládá existenci webu nebo objektu. Cílem je tento objekt pomocí interakce nalézt. („AAA auto“, „Lumia“, „Asus“, atd.)

- **Informační cíle**

  Uživatel předpokládá existenci informace na webu. Jeho cílem je tuto informaci nalézt („otevírací doba Národního muzea“, „datum vydaní Windows 10“, „odjezdy autobusu do Prahy“, atd.).

- **Transakční cíle**

  Uživatel chce nalézt web, či objekt za účelem provádění dalších operací (shlédnout film, poslechnout si hudbu, objednat službu, koupit produkt aj.).

Při podrobnějším pohledu na tyto kategorie můžeme předpokládat, že navigační cíle obvykle slouží k dosažení webu jako takového (případně některé jeho součásti). Po dosažení následuje úprava uživatelova cíle buď na informační, nebo transakční

a dále se orientuje pomocí navigačních prvků webu, nebo web opouští, pokud neodpovídá jeho požadavkům.[1]

Z pohledu provozovatele prodejního webu jsou pro okamžitý zisk zajímavější uživatelé s transakčním cílem, přesněji s cílem nákup/objednávka objektu, který prodejní web nabízí. Uživatelé s informačním cílem okamžitý zisk obvykle nepřináší, je však možné, že uživatel časem svůj cíl přehodnotí.[1]

3.2 Preference uživatele

Aby systém mohl doporučit uživateli novou potencionálně zajímavou položku, je třeba vědět, jak se uživatel choval v minulosti, zda a jaké interakce provedl v minulosti. Jinými slovy je třeba znát jaké má uživatel preference.

Uživatel je osoba navštěvující webové stránky, které obsahují doporučovací systém, aby provedla interakci se systémem k nákupu produktů, nebo služeb, které stránky nabízí.

- Uživatel vyjadřuje, co se mu líbí
- Každý chce něco jiného

Objekt je produkt nebo služba, která je nabízena webovými stránkami. Například můžeme mluvit o konkrétní značce mobilního telefonu v e-shopu, čili se jedná o jeho zboží.

- O čem chci rozhodnout, zda se mi to líbí nebo ne

Uživatelská preference je obvykle definovaná jako $PU(o): OxU \rightarrow R$, která pro konkrétního uživatele U a objekt o z množiny objektů O vrací míru „oblíbenosti“

objektu u uživatele. R je rating, který uživatel přiřadil položce. Značení ratingu závisí na zvoleném druhu ohodnocení. Při zvoleném hodnocení líbí se/nelíbí se bude rating nabývat hodnoty {0,1}. Při pěti-hvězdičkovém hodnocení bude rating nabývat hodnoty {1,2,3,4,5} apod. „Oblíbenost" objektu je doménově závislá vlastnost (oblíbenost hudební skladby vs. oblíbenost notebooku). V případě prodejních webů budeme dále předpokládat, že uživatel má transakční cíl „nákup/objednávka objektu s určitými atributy" (ostatní cíle zanedbáváme, neboť nejsou příliš důležité pro provozovatele prodejního webu). Oblíbenost objektu pak můžeme definovat jako míru ochoty uživatele objekt koupit.[1]

Uživatelská preference je pro každého jedinečná, z toho důvodu, že každému se líbí něco jiného, a je proměnlivá. Uživatel se postupem času vyvíjí. Je ovlivnitelný vnějším světem (knihy, reklama, názory blízkých lidí), nebo svou nejistotou, důsledkem toho mění své nároky a požadavky na objekt, který hledá. Nejdříve chce co nejvyšší rozlišení, pak spíše menší hmotnost a nakonec největší rozsah zoomu. Záleží i na aktuálním rozpoložení uživatele. Při stresu nebo při časové tísni se člověk chová jinak.[2]

3.2.1 Předmět preference

Předmět preference vyjadřuje všechny atributy toho, o čem se uživatel rozhoduje. Každý je jiný, a tak každý má jinou prioritu atributů. Atributy položky se většinou nemění. Když bude uživatel preferovat určitý model notebooku, tak jeho vlastnosti jako výkon, velikost pamětí atd. jsou neměnitelné, tedy krom ceny. Jaký atribut je důležitý a jaký atribut naopak můžeme zanedbat, záleží na situaci. Když si uživatel chce koupit výkonný notebook a na barvě mu nezáleží, v tomto případě je pro systém důležitější parametr výkonu, než parametr barvy. Atributy můžeme rozdělit na následující skupiny[2]:

1. Nominální – Barva, výrobce, typ obrazovky, …

2. Numerické – Hmotnost, rozlišení, váha, výdrž baterie, rozměry, …

3. Speciální – Množina hodnot některého z atomických typů (herci ve filmu)

   – Těžko zachytitelné atributy (tvar, oblost, zvuk, …)

Před tím, než bude spuštěn doporučovací proces, je vyžadováno tyto atributy předzpracovat využitím například diskretizace, normalizace, atd., aby více odpovídali uživateli.

3.2.2 Krátkodobá preference

Krátkodobá preference uživatele představuje preferenci k objektům na základě aktuálního cíle a na základě určitého aktuálního ovlivnění. Pokud aktuální uživatel, který za normálních okolností preferuje výkon a rychlost počítače, má v úmyslu koupit levný, nevýkonný počítač pro manželku, tak preference výkonného PC se bude blížit 0, i přes to, že v jiné situaci by měla hodnotu 1.[1] Tato preference se nedá předpovědět.

3.2.3 Dlouhodobá preference

Dlouhodobá preference vyjadřuje obecná pravidla, kterými se uživatel většinou při nákupu řídí. Například preference vyššího výkonu, preference notebooků před stolními počítači, preference chytrých telefonů s OS Android před OS Windows Mobile, atp.[1] Podle této preference se dá odhadnout, jaké bude chování uživatele v budoucnu.

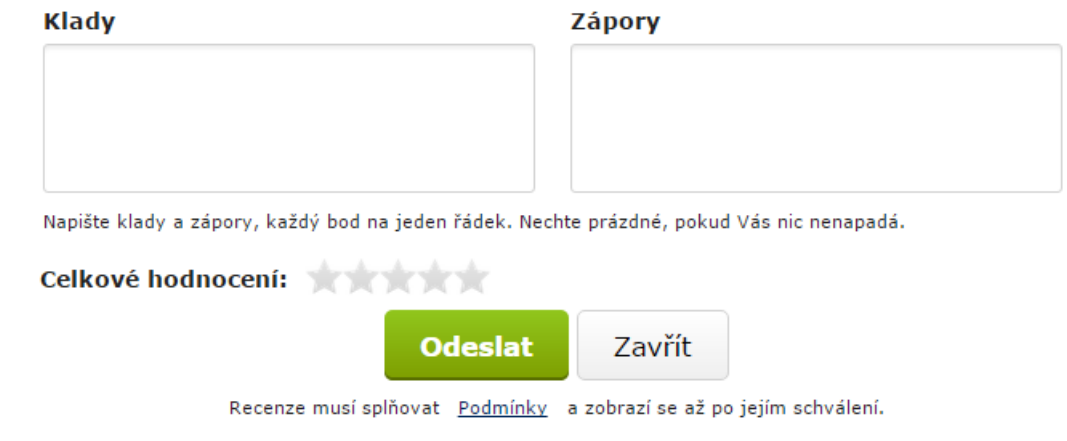
3.2.4 Způsoby získání uživatelských preferencí

Získávání uživatelských preferencí lze uskutečnit pomocí informace poskytnuté při interakci s webovým serverem uživatelem ať už vědomě či nevědomě. Tato informace se nazývá zpětná vazba. Na základě zpětné vazby je možno činit závěry a předpoklady o uživatelově preferenci vůči některému objektu [1,3]. Zpětné vazby lze rozdělit na explicitní zpětnou vazbu a implicitní zpětnou vazbu.

3.2.4.1 Explicitní přístup

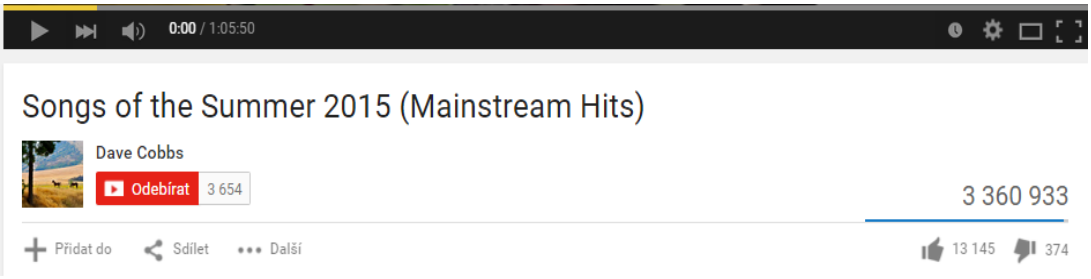
Explicitní vyžaduje interakci s určitým nástrojem webové stránky. Uživatel musí vědomě vyvinout určité úsilí, například použít nástroj pro hodnocení položky, který uživatele vyzve k přidělení hodnocení diskrétní škály k objektu. Každý systém má zavedenou svou stupnicí. Při hodnocení objektu je z pohledu správce webu zásadní zvolit dostatečně jednoduchou stupnici tak, aby nedocházelo k přílišným nekonzistencím v hodnocení a zároveň uživateli srozumitelně stupnici zobrazit. [1,3]

Například internetový obchod ALZA.cz vyžaduje hodnocení položky v rozsahu od 1 do 5 ve formě hvězd.



Obrázek 1: Hodnocení položky na webu Alza.cz

YouTube má hodnoceni pod videem jen binárního druhu „To se mi líbí“, „To se mi nelíbí“ v podobě palců (thumbs).



Obrázek 2: Hodnocení videa na portálu youtube.com

Pokud doporučovací systém využívá pro doporučení pouze explicitní hodnocení, uživatel potřebuje explicitně ohodnotit co nejvíc položek, aby získal relevantní doporučení[3].

Zásadním problémem explicitní zpětné vazby je neochota některých uživatelů ji poskytovat. Tento problém zapříčiňuje zkreslování výsledků kolaborativních algoritmů z důsledku nedostatku hodnocení a častějšího projevu šumu. Tato problematika nejvíce postihuje prodejní weby s velkou nabídkou a s častou obměnou nabízených objektů, které pak mohou zůstat neohodnoceny. Proto musí být kladen důraz na motivaci uživatele k poskytnutí zpětné vazby. Ať už ve formě bonusů/slev, či jiných výhod, nebo pomoci ostatním uživatelům (obvykle také s možností napsání komentáře k hodnocení v prostém rodném jazyce). [1]

3.2.4.2 Implicitní přístup

Implicitní přístup je založen na aktivitě a chování uživatelů na webu bez jejich aktivní účasti. Uživatel nemusí vyplňovat žádné recenze, ani hodnotit položky v obchodě. Systém sám sbírá data o chování uživatele na stránce reprezentující položku.

Mezi hlavní skenované chování patří „Jakou stránku otevřel“, „Jak dlouho se uživatel na stránce zdržel.“, „Na jaké odkazy klikl“, „Jestli aktuální položka byla koupena, či ne.“, atp. Systém pak na základě tohoto chovaní vyhodnotí jak velký zájem má uživatel o tuhle položku a přiřazuje ohodnocení pro dvojici uživatel-položka. K dosáhnutí vyšší přesnosti v přiřazování ohodnocení je tento implicitní přístup kombinován s explicitním.[3]

Co je třeba si uvědomit je závislost implicitní zpětné vazby na doméně – pro různé domény se uvažuje různé implicitní uživatelské chování. Nemá smysl zpracovávat takové uživatelské chování, které nesouvisí s typem webové stránky. Například systém nebude skenovat, jak dlouho uživatel strávil v „nastavení“ nebo editací jeho účtu. Tyto informace jsou v internetovém obchodě zbytečné na rozdíl od informace, jaké zboží uživatel navštívil.[1] Podle Keely a Teevan [4] lze rozdělit pozorovatelné implicitní úkony podle rozsahu na úkony spojené se segmentem objektu, s objektem a se třídou objektů, a jednak podle druhu činnosti na průzkum, uchovávání, reference, anotace a tvorba:

	Segment	**Objekt**	**Třída objektů**
Průzkum	Zobrazení, Poslechnutí, Vyhledávání, „Scrollování“, Dotazování	Vybrání	Prohlížení
Uchovávání	Tisk	Přidání do záložek, Uložení, Smazaní, Koupě, Poslání e-mailem	Odebírání novinek
Reference	Copy/paste, Citování	Přeposlání, Odpověď, Odkazovat na, Citování	
Anotace	Označení	Ohodnocení, Publikování	Setřídění

Tvorba	Napsání, Upravení	Být autorem	

Tabulka 1: Rozdělení implicitních úkonů

Tabulka ukazuje přehled doménově nezávislých implicitních faktorů. Neobsahuje všechny možné implicitní akce jako například přehrání videa, vložení objektu do košíku, atd. a takové akce, které v současnosti nejsou zjistitelné, nebo jen velmi obtížně.

Výhody a nevýhody implicitního přístupu

Každý přístup má své výhody a nevýhody. Výhodou implicitního přístupu, která je nejčastěji zmiňována, oproti explicitnímu je množství získaných dat[1]. Hlavním důvodem tohoto je, že uživatel nemusí dělat nic jiného než to, proč na daný web přisel. Jak již bylo zmíněno hlavním problémem u explicitního přístupu je neochota uživatele jakkoliv provádět interakci hodnotícím nástrojem za účelem ohodnocení položky. Chce si například jen koupit položku a nezdržovat se (pro něj zbytečnými) úkony. Například ve studii Jawaheer, Szomszor, Kostkova[5], ve které při zkoumání explicitního přístupu náhodně vybrali 16 394 uživatelů hudebního doporučovacího systému, museli z dalšího zkoumání vyřadit 6 382 uživatelů z důvodu, že neposkytli žádnou zpětnou vazbu.[1,5]

Dalším problémy jsou[1]:

- **Neexistence určení negativní zpětné vazby.** Vše co zákazník na webu obchodu udělá, je báno jako jeho preference. Není možné pomocí implicitního přístupu zjistit, zda je zákazník s položkou spokojen, nebo nikoliv.
- **Nejednoznačnost interpretace implicitních dat.** Na rozdíl od explicitních přístupů, kde jsou vztahy mezi daty jednoznačně a přímo

definovatelné, je v implicitních obtížná interpretace dat. Může být složité určit vztah mezi získanými implicitními daty a preferencí uživatele.

- **Nutnost použít jiné vyhodnocovací algoritmy, než u explicitních dat.**

I přes tyto nevýhody patří použití implicitního přístupu mezi nejvýznamnější způsoby získání preferencí hlavně v internetových obchodech s vysokým poměrem produktů/zákazníků, s vysokou obměnou produktů, kde jsou rychle staré nahrazovány novými, a v neposlední řadě v obchodech, které nechtějí nebo nemohou motivovat uživatele k poskytnutí zpětné vazby.[1]

Příklady možnosti interpretace implicitních dat

Hu, Koren a Volinsky[6] ve studii, která se zabývá využitím implicitních uživatelských dat ze sledování digitálních TV pro doporučování dalších zajímavých pořadů pro diváka, navrhli preferenční model uživatele k jím nenalezenému objektu s binární proměnnou preference

$$pui = \begin{cases} 1\, r_{ui} > 0 \\ 0\, r_{ui} = 0 \end{cases},$$

kde pokud divák *u* položku *i* nikdy nezobrazil ($r_{ui} = 0$) $p_{ui} = 0$, pokud zobrazil $r_{ui} > 0$ $p_{ui} = 0$ a proměnnou konfidence

$$cui = 1 + \alpha r_{ui},$$

kde c_{ui} roste, se získáváním dalších zobrazení, které indikují pozitivní preferenci diváka k položce. [1,6]

Holub a Bieliková [7] se ve své práci se zabývají návrhem systému pro doporučování relevantních odkazů na konkrétním webu. Model je zaměřený na využití akcí jako „času, který uživatel strávil na webové stránce“, „počtu scrollování“, a využití toho „kolikrát uživatel zkopíroval text do schránky (clipboardu)“. Jejich metoda

je založena na porovnání chování aktuálního uživatele se všemi ostatními uživateli. První dvě akce jsou porovnány s akcemi ostatních uživatelů, kteří navštívili stejnou webovou stránku. Když hodnota aktuálního uživatele je o více než X % vyšší než průměr, je to znak pozitivní preference uživatele na tu určitou stránku. Na druhou stranu pokud je o více než X % nižší než průměrná hodnota, je to tento případ brán jako známka negativní preference. V případě, že se hodnota pohybuje okolo průměru (± X %) je to známka neutrálního zájmu.[7]

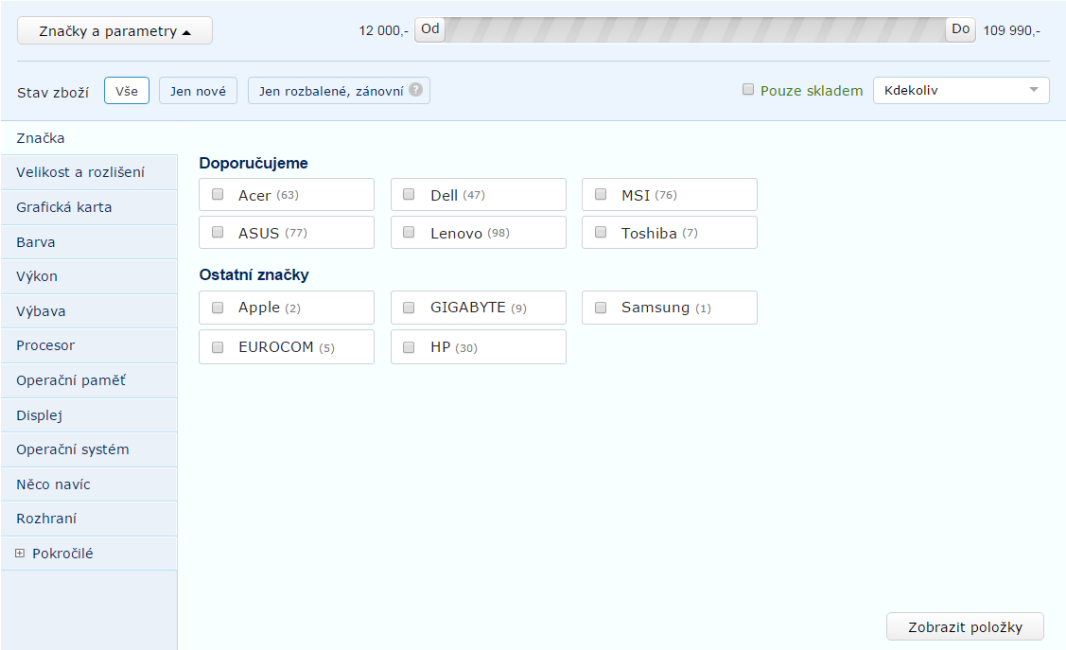
3.2.4.3 Přímá preference

Přímou preferencí se rozumí situace, kdy uživatel prodejního webu pomocí rozhraní specifikuje z množiny vlastností objektu nebo atributů, jaké vlastnosti u hledané položky vyžaduje (očekává, nevyžaduje, upřednostňuje, atd.). Tento přístup je obvykle považován za jeden z nejpřesnějších přístupů. [1]

Přímá preference je obvykle vyjádřená interakcí se stránkou. Může se jednat kupříkladu o procházení se seznamem produktů, kde zobrazení určité skupiny produktů značí zvýšenou preferenci k této kategorii.[1]

Obrázek 3: Výběr zboží na stránce Alza.cz

Dalšími příklady jsou vyhledávání podle klíčového slova, v takovém případě je vyhledávané slovo znakem preference, nebo vyhledávání položky podle atributů a vlastností, kde vyplněné hodnoty značí preferenci aktuálního uživatele.[1]



Obrázek 4: Vyhledávání položky podle atributů na stránce Alza.cz

Přímá preference funguje tak, že vstup od uživatele je přeložen jako dotaz do databáze se všemi objekty a výsledek tohoto dotazu je pak uživateli zobrazen na stránce. Do budoucna se uvažuje o využití přímých preferencí k dedukci dlouhodobých preferencí uživatele, nebo preferencí na jednotlivých atributech.[1]

I když tento přístup je uznávaný a hojně používaný, má i několik negativ. Jedno z nich je, že uživatel musí znát, co chce a tedy i objekt, za jehož koupením vstoupil na dané webové stránky, čili musí znát své preference u jednotlivých vlastností položky a musí tyto preference umět vyjádřit. Dalším negativem je, že webové rozhraní musí být navrženo tak, aby uživatel mohl jednoznačně a vhodně vyplnit své preference. Je tedy kladen důraz na to, aby rozhraní bylo jednoduché, pochopitelné a pro uživatele intuitivní, ale i natolik komplexní a rozsáhlé, aby umožnilo vyjádřit preference co

nejpřesněji. Stále může ovšem nastat situace, kdy uživatelova preference bude mít takové aspekty, které webový server nedokáže vyjádřit (důležitost atributu, jiné než rozsahové zadání, zadání vlastností, které položka nesmí obsahovat, atd.).[1]

Jak již bylo řečeno, vyhodnocení přímých preferencí je realizováno dotazem na databázi a ty položky, které splňují uživatelovi preference a jsou označené jako vyhovující, jsou zobrazeny, a naopak položky nesplňující kritéria zobrazeny nejsou. Některé vlastnosti položky ale mohou být pro uživatele hraniční, proto se doporučuje zařadit do výsledku i takové položky, které uživatelově preferenci jen těsně nevyhovují. Popřípadě může být uvažováno o zařazení do výsledku i pro takové položky, které se liší jen jedním atributem, ale takovým, který je vyhodnocen jako nejméně důležitý.[1]

4 Doporučovací systémy

Doporučovacím systémem se rozumí jakákoliv kolekce softwarových nástrojů poskytující návrhy, které budou uživateli užitečné. Tyto návrhy se mohou týkat mnoha různých položek například počítačů, knih, nábytku, ale i služeb jako hotelů, restaurací apod.[9]

Cílem doporučovacích systémů je tedy vytvářet smysluplné doporučení položek, nebo produktů pro uživatele, které by je mohli v budoucnu zajímat a tím pomoci jak webovému obchodu, tak i uživatelům [8]. Databáze obchodů obvykle obsahuje tisíce záznamů rozdělených do mnoha kategorií.

Systém doporučení poskytuje uživateli položky, se kterými by chtěl provádět interakci bez toho, aby je uživatel musel zdlouhavě a obtížně hledat mezi tisíci dalšími. Toto setří uživatelův čas a pomáhá mu strávit více času interakcí se zajímavými položkami, než aby se zdržoval bezvýsledným prohlížením nezajímavých položek. Systém také pomáhá najít nové položky pro uživatele. Vlastností uživatele je i to, že má tendenci při hledání a prohlížení položek se zaseknout v relativně malé skupině položek, a tomu systém tímto zabraňuje.[10]

Z pohledu vlastníka webového obchodu doporučovací systém pomáhá zvyšovat konkurenceschopnost i zisky. Spokojený zákazník by měla být priorita pro všechny business strategie. Jestliže zákazník rychle nalezne to, co potřebuje, nebo to, co ho zajímá, pak je vetší pravděpodobnost, že si položku koupí a dokonce má v budoucnu vetší tendenci se vrátit znovu, což zapříčiňuje zvyšování návštěvnosti obchodu. Zvyšuje se také doba strávená na stránce, kde na uživatele může být cílena přesná reklama.

Návrh těchto doporučovacích enginů závisí hlavně na oblasti a na konkrétní specifikaci dostupných dat. Například pokud zákazníci ohodnotí položku na stupnici od 1 (spokojen) do 0 (nespokojen), pak systém zaznamená jejich hodnocení a uloží jej. Takto charakterizovaný zdroj dat zaznamenává kvalitu interakce mezi zákazníkem a položkou, u kterých navíc může mít přístup k atributům jak zákazníka, tak položky, jako jsou demografie, popis zákazníka, popis produktu aj. a dále s nimi pracovat a identifikovat nejlepší dvojice zákazník-produkt. Doporučovací systémy se liší ve způsobu analýzy dat k vytvoření představy o příbuznosti mezi zákazníkem a konkrétní položkou, která může být použita k identifikaci párů. [9]

Z důvodu velké variability uživatelů a jejich preferencí jsou návrhy generované doporučovacím systémem obvykle individuální (každý uživatel dostane jiné návrhy). Jsou používané ale i neindividualizované návrhy, jejichž znakem je mnohem jednodušší vytvoření a jsou obvyklé v novinách, či časopisech. Typickým příkladem takovýchto návrhů je nejlepší desítka knih, nebo DVD, podle prodeje, nebo hodnocení zákazníků. [9]

4.1 Příklady doporučovacích systémů

V této části byly vybrány reálné příklady doporučovacích systémů. Byly vybrány takové, které splňují podmínky velké uživatelské základny a velkého objemu nabízeného zboží.

4.1.1 Amazon.com

Amazon.com je internetový obchod, který vlastní americká společnost Amazon.com, Inc. ve státě Washington. Tento internetový obchod patří mezí největší a nejstarší vůbec. Jeho začátek se datuje od roku 1994, kdy Jeff Bezos začal provozovat

knihkupectví Cada-bra.com, které bylo přejmenováno, inspirováno řekou amazonkou, na Amazon později téhož roku. Příjmy společnosti se dnes pohybují okolo 7 miliard dolarů ročně a její sílu podtrhává i to, že jako jedna z mála společností dokázala růst i v časech ekonomické krize. Amazon dnes nabízí velké množství druhů zboží počínaje online knihami, hudbou, filmy, ale nachází se zde i takové segmenty jako jsou prodeje hraček, elektronických strojů, oblečení, léků, dokonce i potraviny až po vlastní elektronických knih Amazon Kindle a tablet Kindle Fire.

Zákazníkům Amazon.com nabízí mnoho druhů doporučení. Z pohledu prodeje knih například:

Customers who Bought

Jako mnoho stranek e-commerce, tak také Amazon.com je strukturován tak, že každá kniha má svou informační stránku dávající zákazníkovi detaily o obsahu knihy a informace o koupi. Doporučení „Customers who Bought“ se nalézá v dolní části a skládá se ze dvou doporučení: První obsahuje doporučení knih, které si často zákazníci kupují společně s touto knihou. Druhé je trochu odlišné a nabízí knihy toho samého autora aktuálně prohlížené knihy, které jsou frekventovaně kupovány s touto knihou.[11]

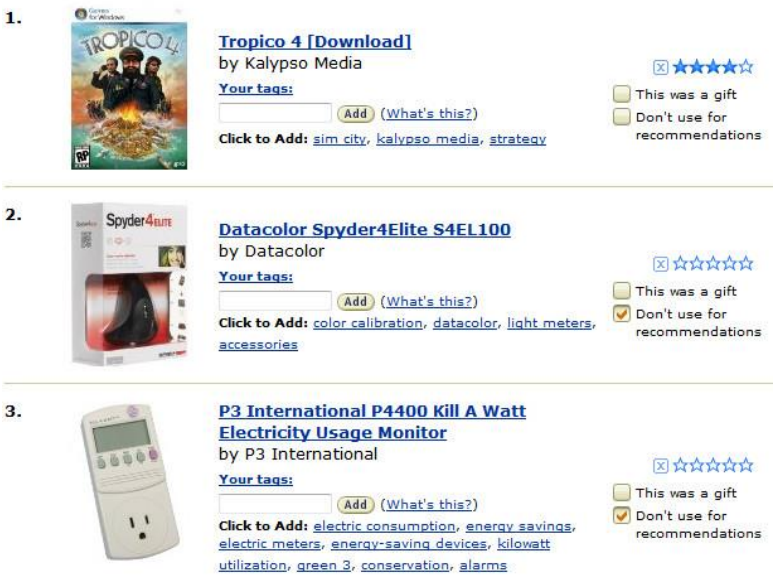
Obrázek 5: Doporučení „Customers who Bought“ na stránce Amazon.com

Eyes

Další formou doporučení je vlastnost zvaná Eyes. Eyes umožňuje zákazníka informovat přes email v momentě, kdy byla přidána nová položka do nabídky obchodu. Pro doporučení musí buď zákazník vložit požadavek, který obsahuje, co ho zajímá např.: ISBN, autor, téma, atd. nebo tento požadavek je generován systémem na základě textového vyhledávání na webu zákazníkem.[11]

Book Matcher

Book Matcher umožňuje zákazníkovi dát zpětnou vazbu knihy, kterou si koupil a přečetl. Zákazník hodnotí knihu na 5 bodové škále od „nesnáším ji“ do „miluji ji“. Po několika hodnocení zákazník dostává doporučení knih, které by ho mohli zajímat. Zpětná vazba je na stránkách obchodu realizována vlastností „ohodnoť tuto knihu“, kde zákazníci mohou vyjádřit názor na jednu nebo více knih.[11]



Obrázek 6: Hodnocení položek na stránce Amazon.com

Customer Comments

Tato vlastnost nabízí zobrazení textových doporučení založených na názoru jiných zákazníků. Pro každou knihu je na její informační stránce k dispozici hodnocení v podobě hvězd a textový komentář.[11]

Obrázek 7: Hodnocení od zákazníků na stránce Amazon.com

Amazon.com	Doporučovací interface	Dororučovací technologie
Customers who Bought	Podobná položka	Item to Item Cerrelation
Eyes	Email	Attribute Based
Book Matcher	Top N	People to People Correlation
Customer Comments	Průměrné hodnocení Textové komentáře	Aggregated Rating

Tabulka 2: Doporučení na Amazon.com

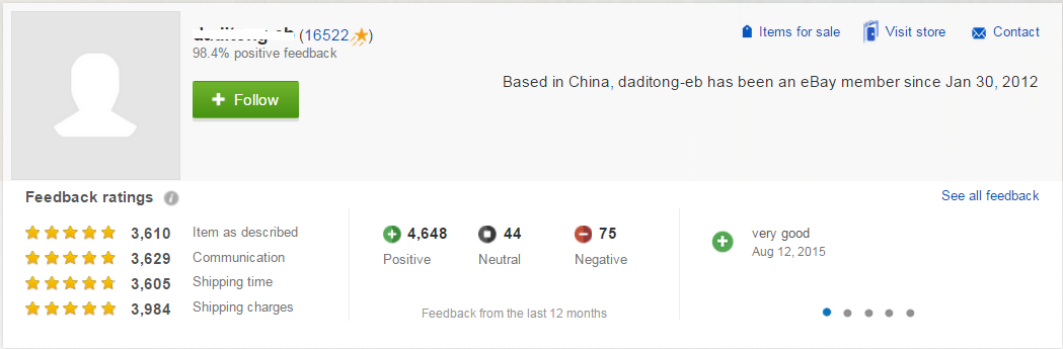
4.1.2 eBay

Tato firma patří mezi nejúspěšnější a nejznámější v kategorii virtuálních tržišť, nebo-li internetových aukčních síní. Společnost eBay byla založena v roce 1995 Američanem íránského původu. Aukční síň zahrnuje dále především jednu z nejoblíbenějších služeb bezhotovostní platby v USA PayPal nebo systémem Bill Me Later, který představuje alternativu k placení prostřednictvím PayPalu, kde hlavní rozdíl je v tom, že Bill Me Later je zaměřen na firemní klientelu a možností postupného splácení. Hlavním příjmem společnosti jsou především poplatky spojené s obchodováním a samozřejmě i reklama. Na aukcích lze pořídit nebo prodat téměř vše, od kamionu přes posezení s celebritou až např. k obyčejnému jídlu za astronomické sumy.[12]

Feedback Profile

Co se týče doporučovacích systémů, je zde provozován modul s názvem Feedback Profile, který umožňuje oběma stranám, jak zákazníkovi, tak prodejci, vyjádřit zpětnou vazbu k obchodování složenou z ohodnocení spokojenosti

(spokojen/neutrální/nespokojen) a textového komentáře popisující druhou stranu. Tato zpětná vazba je použita doporučovacím systémem u nákupů, kde je možné vidět statistiky uživatele a tabulku vyjadřující jeho ohodnocení za 7 dní, měsíc a za posledních 6 měsíců.



Obrázek 8: Statistiky uživatele na stránce Ebay.com

eBay	Doporučovací interface	Dororučovací technologie
Feedback Profile	Průměrné hodnocení Textové komentáře	Aggregated Rating

Tabulka 3: Doporučení na Ebay.com

4.1.3 Alza.cz

Alza.cz a.s. je český obchod s počítači a spotřební elektronikou. Provozuje stejnojmenný internetový obchod a síť kamenných poboček. Společnost byla založena Alešem Zavoralem 29. listopadu 1994 pod jménem Alzasoft. Společnost působí v České republice a na Slovensku s centrálou umístěnou v Praze. První webové stránky společnosti, které ještě nesloužily k nákupu, vznikly na jaře. V roce 2000 vznikl elektronický obchod propojený s webovými stránkami. V roce 2006 firma prošla změnou obchodní značky a byla přejmenována Alza.cz. Nyní působí jako ryze česká

akciová společnost. Je vlastněna skupinou investorů, kteří ji ovládají přes holdingovou společnost L.S. Investments Limited.

Doporučení k předchozím nákupům

Na své úvodní stránce Alza.cz nabízí různé kategorie zboží. Jednou z nich je kategorie, která se zakládá na doporučení na základě předchozích nákupů uživatele. Doporučení je založeno na základě podobnosti, kdy uživateli jsou doporučeny položky se stejnými vlastnostmi již koupených výrobků nebo na základě doporučení příslušenství k již koupeným položkám.

Obrázek 9: Doporučení k předchozím nákupům na stránce Alaz.cz

Hodnocení výrobků

Toto doporučení výrobku se nalézá ve spodní části stránky výrobku. Zaznamenává se zde hodnocení v podobě hvězd a k němu textové komentáře k výrobku od uživatelů, kteří si jej koupili.

Obrázek 10: Hodnocení výrobků na stránce Alza.cz

Zákazníci nejčastěji porovnávají

Při pohledu na úplný konec stránky výrobku nalezneme doporučení, podobné jako výše zmíněné Customers who Bought, s názvem Zákazníci nejčastěji porovnávají. Rozdíl je v tom, že toto doporučení nepracuje s již koupenými položkami, ale s těmi, které nejčastěji porovnávají s touto položkou.

Obrázek 11: Zákazníci nejčastěji porovnávají na stránce Alza.cz

Alza.cz	Doporučovací interface	Dororučovací technologie
Doporučení k předchozím nákupům	Doporučení k předchozím nákupům	Item to Item Cerrelation
Hodnocení výrobků	Průměrné hodnocení Hodnocení	Aggregated Rating

Zákazníci nejčastěji porovnávají	Zákazníci nejčastěji porovnávají	Item to Item Cerrelation

Tabulka 4: Doporučení na Alza.cz

4.2 Druhy doporučovacích systémů

4.2.1 Doporučení vyhledáváním

Je jeden z nejjednodušších doporučovacích systémů. Vyhledání potencionálně zajímavé položky probíhá zadáním vyhledávacího dotazu zákazníkem a posléze nalezením všech položek, odpovídajících dotazu, systémem. Například uživatel zadá dotaz o zobrazení nejprodávanější knihy měsíce. Systém doporučí některou z knih z nabídky, která splňuje zadané podmínky na základě všeobecného, neosobního hodnocení (podle prodejní pozice, popularity, atd.). Výhodou tohoto doporučení je jednoduchost na implementaci a možnost rychlého a jednoduchého nasazení na prodejní web. Naopak mezi nevýhody jednoznačně patří malá účinnost doporučení a to, že uživatel dostane jen ty položky, na které se ptal.

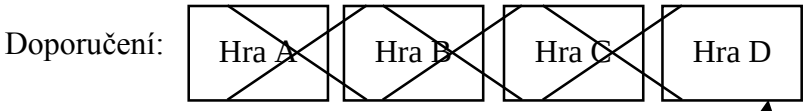
4.2.2 Doporučení kategoriemi

Další velmi jednoduché doporučení, kde položky jsou rozděleny do kategorií a každá položka musí patřit do jedné nebo více těchto kategorií. Zákazník si vybere kategorii, která ho nějakým způsobem zajímá, tím zpřesní vyhledávání. Systém vybere kategorie zájmů pro zákazníka na základě prohlížené aktuální položky, předchozích nákupů atd., a doporučí určité pro uživatele potencionálně zajímavé položky ze zadané kategorie.

4.2.3 Kolaborativní doporučení (Collaborative filtering)

Hlavní myšlenkou kolaborativního doporučení je využití informací o uživatelské komunitě webu, jako je její chování, nebo její volby v minulosti, pro předpovídání, o jaké položky by se aktuální uživatel systému mohl zajímat, nebo by se mu mohly líbit. Dnes je tento druh systému hojně využíván hlavně v průmyslovém odvětví, zejména jako nástroj v online obchodech k přizpůsobení obsahu potřebám zákazníka, čímž dochází k podpoře dalších položek obchodu a zvyšování prodejů. Tento přístup nevyžaduje k doporučení žádné znalosti systému o položkách a jejich vlastnostech. Není tedy potřeba udržovat nebo vkládat žádná data o položkách samotných, čímž také odpadá údržba databáze položek.[9,14]

Příkladem může být tato situace: Aktivní uživatel s historií nakoupených her A, B přichází na web e-shopu. Prvním krokem systému je, že vyhledá uživatele s podobnou historií nákupů. Vyhledá uživatele Karel a Marie, kteří si oba v minulosti koupili hry A, B, D. Systém vyhodnotí, že si aktivní uživatel ještě nekoupil rozdílnou položku D a předpokládá, že by se o ni uživatel mohl zajímat, tak mu ji doporučí.



	Hra A	Hra B	Hra C	Hra D
Uživatel	x	x		
Karel	x	x		x
Petr		x	x	
Anna				x
Marie	x	x		x

Obrázek 12: Příklad kolaborativního doporučení

Kolaborativní doporučení je tedy proces, při kterém jsou informace filtrovány na základě daných kritérií. Obvykle se používá pro velmi rozsáhlé množiny dat a dále přispívá uživatelům k orientaci v tomto velkém množství dat. Principem tohoto filtrování je vytvoření filtrovacího vzorce na základě dat získaných od velkého množství uživatelů a následné vytvoření předpovědi použitím tohoto vzorce na množinu dat.[15]

Kolaborativní metody se mohou rozdělit do několika skupin a to na Paměťové, které se dále dělí na User-based a Item-based a na Modelové.

4.2.3.1 Paměťové kolaborativní filtrování

Paměťové metody byly prvními kolaborativními metodami. Používá se zde předpověď možných vztahů, která je počítána na základě vztahů mezi objektem a subjektem. Výpočetní funkcí může být prostý průměr, nebo jiné opatření využívající relativní rozdíly v průměrném hodnocení, nebo v podobnosti.[10] Tyto metody využívají ke generování předpovědi všechny záznamy z databáze uživatel-položka. Systém pomocí využití jedné nebo více statistických metod nalezne skupinu uživatelů známých jako sousedé, kteří mají stejnou historii zájmových položek jako aktuální uživatel (jejich hodnocení položek je stejné, nebo tíhnou k nakupování stejných skupin položek). Jakmile je sousedství aktuálního uživatele zformováno, systém použije různé algoritmy, aby zanalyzoval preference sousedů a vytvořil tak předpověď nebo top-N doporučení pro uživatele. [15]

Paměťové kolaborativní filtrování se rozdělujeme do dvou skupin podle techniky použité pro doporučení: Doporučení založené na podobnosti uživatelů (User-based) nebo Doporučení založené na podobnosti položek (Item-based).

4.2.3.1.1 User-based Collaborative filtering

User-based Collaborative filtering neboli doporučení založené na podobnosti uživatelů je založeno na následujícím: Je dána tabulka hodnocení položek v databázi uživateli a ID aktuálního uživatele jako vstup. Z tabulky se identifikují takoví uživatelé (sousedé), kteří mají podobné preference jako aktuální uživatel v minulosti. Pak pro každý produkt p, který uživatel ještě nenalezl, je vypočtena předpověď založena na hodnocení produktu p sousedy. Tato metoda vychází z následujících předpokladů:[15]

a) Uživatelé, kteří měli stejné nebo podobné zájmy v minulosti, budou mít stejné zájmy i v budoucnosti.
b) Preference zůstane v čase stabilní a konzistentní.

Postup doporučení můžeme rozdělit na 3 základní kroky[8]:

1. Určení váhy všem uživatelům s ohledem na podobnost s aktivním uživatelem.

2. Výběr určitého počtu *k* uživatelů (sousedů), kteří mají největší podobnost s aktivním uživatelem.

3. Vypočtení předpovědi z vážené kombinace vybraných uživatelských hodnocení.

Příklad doporučení (převzat z [15]):

Máme následující tabulku hodnocení a aktuálního Uživatele Alice

	Položka 1	*Položka 2*	*Položka 3*	*Položka 4*	*Položka 5*
Alice	5	3	4	4	?
Uživatel1	3	1	2	3	3
Uživatel2	4	3	4	3	5
Uživatel3	3	3	1	5	4
Uživatel4	1	5	5	2	1

Tabulka 5: Příklad User-based Collaborative filtering

Prvním krokem je vypočtení podobnosti aktuálního uživatele s ostatními uživateli. Podobnost mezi dvěma uživateli je realizována pomocí Pearsonova korelačního koeficientu:

$$sim(a,b) = \frac{\sum_{p \in P}(r_{a,p} - \overline{r_a})(r_{b,p} - \overline{r_b})}{\sqrt{\sum_{p \in P}(r_{a,p} - \overline{r_a})^2}\sqrt{\sum_{p \in P}(r_{b,p} - \overline{r_b})^2}},$$

kde *sim(a, b)* je podobnost mezi uživateli *a* a *b*, $U = \{u_1, \ldots, u_n\}$ je množina všech uživatelů v tabulce, kde *a* značí uživatele aktuálního uživatele a *b* značí uživatele z tabulky hodnocení. $P = \{p_1, \ldots, p_n\}$ je množina ohodnocených položek oběma uživateli. R je matice *M x N* hodnocení r_{ij}. $R_{b,p}$ je hodnocení položky *p* uživatelem *b* a r_b je průměrné hodnocení uživatele *b*. Posledními proměnnými jsou $(\overline{r_a})$ a $(\overline{r_b})$ které značí průměrné hodnocení uživatelů.

Konkrétně pro výpočet podobnosti mezi Alicí a Uživatelem1 je vzorec následující:

$$\frac{(5 - \overline{r_a}) * (3 - \overline{r_b}) + (3 - \overline{r_a}) * (1 - \overline{r_b}) + \cdots + (4 - \overline{r_a}) * (3 - \overline{r_b})}{\sqrt{(5 - \overline{r_a})^2 + (3 - \overline{r_b})^2 + \cdots} * \sqrt{(3 - \overline{r_a})^2 + (1 - \overline{r_b})^2 + \cdots}} = 0{,}85,$$

kde $(\overline{r_a}) = 4$ *a* $(\overline{r_b}) = 2{,}4$.

Podobnost sim(Alice,Uživatel1) je tedy 0,85. Podobnost Alice s ostatními uživateli je:

sim(Alice,Uživatel2) = 0,70

sim(Alice,Uživatel3) = 0,00

sim(Alice,Uživatel4) = -0,79

Pearsonův korelační koeficient nabývá hodnot od +1 (velmi pozitivní) do -1 (velmi negativí).

Druhým krokem je výběr uživatelů, kteří mají podobnost a aktivním uživatelem nejvyšší. Vybereme tedy Uživatele1 a Uživatele2.

Posledním krokem je vypočtení předpovědi, zda aktuální uživatel položku preferuje (jak by ji ohodnotil), kterou ovlivňuje jak Uživatel1 tak Uživatel2, pomocí vzorce:

$$pred(a,p) = \overline{r_a} + \frac{\sum_{u \in N} sim(a,b) * (r_{b,p} - \overline{r_b})}{\sum_{u \in N} sim(a,b)},$$

kde predikce hodnocení, položky *p* aktuálním uživatelem *a*, je *pred(a, p)* a kde *N* je množina vybraných nejpodobnějších sousedů. Vypočtení predikce hodnocení Alice u Položky5 následující:

$$pred(Alice, Položka5) = 4 + \frac{1}{(0{,}85 + 0{,}7) * (0{,}85 * (3 - 2{,}4) + 0{,}70 * (5 - 3{,}8))}$$
$$= 4{,}87$$

Predikce hodnocení u Alice vyšla těsně pod horní hranicí, z toho vyplývá, že položka bude Alici doporučena.

Nevýhodou tohoto řešení je, že při použití na velkých komerčních serverech s miliony uživateli a položkami je pro systém nemožné efektivně a rychle vypočítat doporučení. V těchto případech nastává to, že systém musí analyzovat rozsáhlý počet sousedů, čímž se úloha stává nesplnitelnou v reálném čase. Proto velké e-commerce používají odlišnou metodu Item-based.[15]

4.2.3.1.2 Item-based Collaborative filtering

Nebo-li doporučení založené na podobnosti položek je doporučovací metoda vhodná pro offline předzpracování a využití velkých hodnotících matic velkých systému e-commerce, což v reálném použití se projeví rychlejším a přesnějším doporučením. Dalším rozdílem oproti předchozí metodě je to, že Item-based algoritmy nepredikují doporučení na základě podobnosti uživatelů, ale na základě podobnosti položek samotných.[15] Principem je domněnka, že když si uživatel koupí určitou položku z nabídky odchodu, tak jej v budoucnu bude zajímat taková položka, která je podobná koupené.[9]

Předpověď uživateli k položce *p* probíhá tak, že nejprve jsou pomocí Kosínovi podobnosti porovnány vektory hodnocení všech položek s vektorem hodnocení položky p. Jsou vybrány takové, které se nejvíce přibližují zmíněnému vektoru. Poté systém zjistí, jak uživatel tyto položky hodnotil a provede vážený průměr těchto hodnocení[15].

Příklad převzat z[15] :

Máme tedy identickou tabulku a uživatele Alice, jako v předchozím případě. A chceme vypočítat predikci Položky 5.

	Položka 1	*Položka 2*	*Položka 3*	*Položka 4*	*Položka 5*
Alice	5	3	4	4	?
Uživatel1	3	1	2	3	3
Uživatel2	4	3	4	3	5
Uživatel3	3	3	1	5	4
Uživatel4	1	5	5	2	1

Tabulka 6: Příklad Item-based Collaborative filtering

Prvním krokem je vypočtení podobnosti položek podle Kosínovi podobnosti, která byla prohlášena za standardní metriku. Tato metrika udává podobnost mezi dvěma n-dimenzionálními vektory, která je založena na úhlu mezi nimi. Podobnost nabývá hodnot mezi 0 a 1, kde 1 udává silnou podobnost. Podobnost je tedy definována vzorcem:

$$sim(\vec{a}, \vec{b}) = \frac{\vec{a} \cdot \vec{b}}{|\vec{a}| * |\vec{b}|},$$

kde *a*,*b* jsou položky a ohodnocení těchto položek uživateli zastupují vekrory $|\vec{a}|$ a $|\vec{b}|$. Vzorec tedy představuje skalární součin vektorů vydělený součinem jejich velikostí.

Konkrétně tedy pro vypočtení podobnosti mezi potencionální Položkou 5 a Položkou 1:

$\overrightarrow{P5} = (3, 5, 4, 1)$

$\overrightarrow{P1} = (3, 4, 3, 1)$

$$sim(Položka5, Položka1) = \frac{3 * 3 + 5 * 4 + 4 * 3 + 1 * 1}{\sqrt{3^2 + 5^2 + 4^2 + 1^2} * \sqrt{3^2 + 4^2 + 3^2 + 1^2}} = 0{,}99$$

A zde nastává problém toho, že základní vzorec pro toto nestačí. Systém sice vypočítá podobnost vektorů, ale není zde započten rozdíl v chování uživatele v hodnocení. Tento problém je vyřešen použitím upravené podoby základního vzorce Kosínovi podobnosti a to:

$$sim(a,b) = \frac{\sum_{u \in U}(r_{u,a} - \overline{r_u})(r_{u,b} - \overline{r_{bu}})}{\sqrt{\sum_{u \in U}(r_{u,a} - \overline{r_u})^2}\sqrt{\sum_{u \in U}(r_{u,b} - \overline{r_u})^2}},$$

Kde *U* je množina uživatelů, kteří hodnotili jak položku *a*, tak položku *b*. $R_{u,a}$ je hodnocení uživatele *u* položky *a* a $r_{u,b}$ značí ohodnocení položky *b* uživatelem u a kde značí r_a a r_b průměrné hodnocení položek *a* a *b*.

V praxi to znamená, že můžeme upravit tabulku hodnocení tak, že každé ohodnocení je nahrazeno jeho odchylkou ad průměrné hodnoty ohodnocení položky od všech uživatelů. Tabulka bude tedy vypadat takto:

	Položka 1	*Položka 2*	*Položka 3*	*Položka 4*	*Položka 5*
Alice	1	-1,00	0	0	?
Uživatel1	0,60	-1,40	-0,40	0,60	0,60
Uživatel2	0,20	-0,80	0,20	-0,80	1,20
Uživatel3	-0,20	-0,20	-2,20	2,80	0,80
Uživatel4	-1,80	2,20	2,20	-0,80	-1,80

Tabulka 7:Příklad Item-based Collaborative filtering - upravená tabulka

Nyní můžeme vypočíst upravenou Kosínovu podobnost:

$$sim(P5, P1) = \frac{0{,}6 * 0{,}6 + 0{,}2 * 1{,}2 + (-0{,}2) * 0{,}8 + (-1{,}8) * (-1{,}8)}{\sqrt{0{,}6^2 + 0{,}2^2 + (-0{,}2)^2 + (-1{,}8)^2} * \sqrt{0{,}6^2 + 1{,}2^2 + 0{,}8^2 + (-1{,}8)^2}} = 0{,}80$$

Potom co jsou všechny podobnosti pro všechny položky vypočteny. Můžeme předpovědět ohodnocení Položky5 Alicí vypočtením vážené sumy ohodnoceních, které Alice provedla u položek, které byly vybrány jako nejvíce podobné, pomocí vzorce:

$$pred(u, p) = \frac{\sum_{i \in ohodnocenePoložky(u)} r_{u,i} * sim(i, p)}{\sum_{i \in ohodnocenePoložky(u)} sim(i, p)},$$

kde *u* značí aktuálního uživatele a *p* značí potencionální položku.

Příkladem tohoto využití tohoto doporučení v reálném světě je internetový obchod Amazon.com, kde by bylo, pro miliony uživatelů, kteří tento web navštěvují, User-based doporučení neefektivní, pomalé a zdrojově příliš náročné. Algoritmus zde funguje tak, že po přiřazení uživatelovým nákupům a hodnocením podobných položek jsou tyto položky zkombinovány do doporučení. Kvůli lepší škálovatelnosti a výkonnosti byl proces doporučení rozdělen do dvou komponent: V režimu offline je vytvořena tabulka podobnosti položek, která je pak druhou komponentou, již v online režimu, prohledávána a tím se produkuje doporučení.[9]

4.2.3.2 Modelové kolaborativní filtrování

Modelové kolaborativní filtrování, nebo-li Model based colaborative filtering, provádí predikci potencionálních položek sestavením modelu uživatelských ohodnocení. Algoritmy z této kategorie používají pravděpodobnostní přístup

a představu kolaborativního procesu k výpočtu očekávané hodnoty uživatelské predikce založené na jeho hodnocení ostatních položek. [19]

Návrh a vývoj modelů (např.: strojového učení, dolování dat algoritmy) může systému umožnit se naučit rozpoznávat složité vzory na základě trénovacích dat, a pak vytvářet inteligentní předpovědi pro úlohy kolaborativního filtrování pro testovací data, nebo reálná data na základě naučených modelů. Model based colaborative filtering algoritmy, jako jsou bayesovské modely, clustering modely, byly zkoumány k řešení nedostatků, které má Paměťové kolaborativní filtrování.[20]

Latentní sémantické modely používají vektory k zastoupení uživatelů a položek. Pro položky platí, že hodnoty vektorů popisují některé její charakteristiky. Tento přístup je podobný přístupu, který se používá v doporučení založeném na obsahu. Rozdíl je v získávání vektorů: hodnoty ve vektorech položek nejsou posílány uživateli, ale oba, jak vektory uživatelské, tak vektory položkové, jsou získány ze známých dat za použití různých technik. Proto je tato technika závislá na doméně.[10]

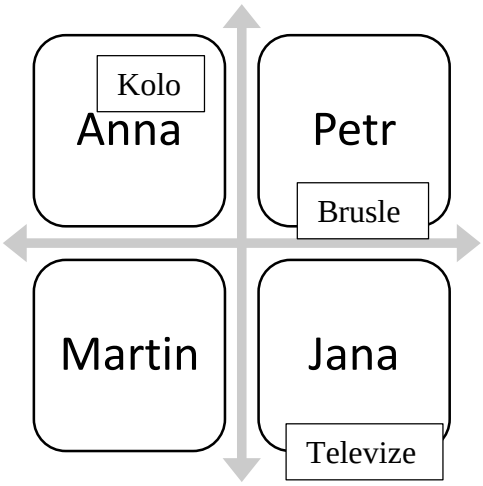
Uživatelské a položkové vektory dovolují promítat uživatele a položky do multidimenzionálního prostoru. Doporučené položky jsou takové, které jsou blízko aktivního uživatele v tomto prostoru. Některé dimenze mohou být párovány s nějakou známou charakteristikou položky, jako je žánr u filmů. Význam většiny dimenzí může být jen těžko objeven, protože nejsou vytvořeny člověkem, ale technikou strojového učení.[10]

Více formálně, latentní faktor doporučení odhaduje užitečnost díky funkci:

$$\hat{u}_R^{LF}(s,o) = fv_R^S(s)^T\, fv_R^o(o),$$

kde $fv_R^S : S \to R^f, fv_R^{So} : O \to R^f$ jsou funkce přiřazené každému uživateli/položce daného faktoru latentní prostorové dimenze.[10]

Příklad:



Obrázek 13: Dimenze k příkladu

Z obrázku, který ukazuje dvojdimenzionální prostor uživatelů a položek, je zřejmé, že v prostoru Anny se vyskytuje položka Kolo, tak ji Anně doporučíme. To samé se týká Jany a televize. Položka Brusle zasahuje jak do prostoru Petra a Jany tak bude doporučena oběma uživatelům.

Doporučení	
Anna	kolo
Petr	brusle
Martin	x
Jana	televize brusle

Tabulka 8: Doporučení položek

4.2.3.2.1 SVD - Singular Value Decomposition

Slabiny Pearsonova algoritmu pro vyhledání nejbližších sousedů ve velmi objemných databázích vedly k hledání nového alternativního algoritmu pro doporučení. Jedním z alternativních algoritmů byl algoritmus Singular Value Decomposition.[16] Singulární rozklad ihned odhaluje několik maticových vlastností a můžeme na něj pohlížet z 3 hledisek. Z prvního můžeme tento algoritmus vidět jako metodu pro transformaci korelačních proměnných na množinu nekorelačních, u kterých se lépe odhalí možné vztahy mezi původními položkami. Stejně tak SVD je metoda pro identifikaci rozměru podél datových bodů, které vykazují největší změnu. To vede ke třetímu hledisku a tím je, že, v případě jestliže byla provedena identifikace největší změny, je možné najít nejlepší sousedy použitím menšího množství rozměrů.[17]

Celý proces se skládá s následujících kroků:

1. Rozložení hlavní matice a vytvoření podmatic $U \Sigma V^T$.
2. Výběr nejdůležitějších sloupců matic.
3. Použití strategie pro generování doporučení.

Příklad převzat z [15]: Máme tedy znovu matici hodnocení uživatelů:

	Položka 1	*Položka 2*	*Položka 3*	*Položka 4*	*Položka 5*
Uživatel1	3	1	2	3	3
Uživatel2	4	3	4	3	5
Uživatel3	3	3	1	5	4
Uživatel4	1	5	5	2	1

Tabulka 9: Příklad SVD

SVD říká, že jakákoli m × n matice A s m ≥ n může být rozložena na:

$$M = U \Sigma V^T,$$

kde U ∈ $R^{m\times m}$ a V ∈ $R^{n\times n}$ jsou ortogonální a Σ ∈ $R^{n\times n}$ je diagonální Σ = diag(σ1, σ2, . . . , σn), σ1 ≥ σ2 ≥ · · · ≥ σn ≥ 0.

Protože 4 x 4 matice je kvadratická, U, Σ, V budou také kvadratické matice o rozměru 4 x 4.

Hlavním smyslem tohoto rozkladu je, že můžeme aproximovat celou matici pozorováním jen nejdůležitějších rys, těch s největší singulární hodnotou. V příkladu jsou vypočteny U, V a Σ (pomocí lineárně algebraického softwaru) a pro další krok jsou ponechané jen nejdůležitější hodnoty, což jsou hodnoty prvních dvou sloupců matic U a V:

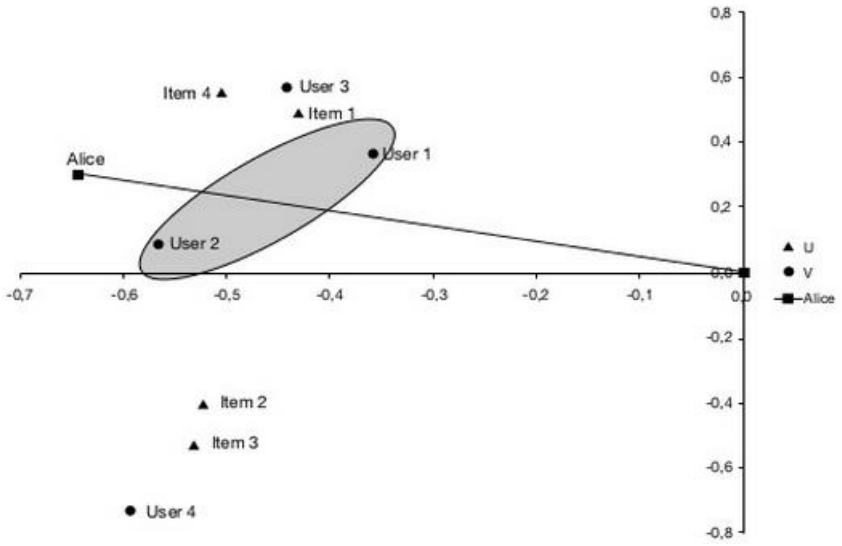
V_2	
-0,3593320	0,3676765
-0,567507	0,0879975
-0,442852	0,5686249
-0,593882	-0,7305724

U_2	
-0,431245	0,493150
-0,532737	-0,530525
-0,523745	-0,405200
-0,505874	0,557815

Σ_2	
12,2215	0
0	4,9282

Tabulka 10: Hodnoty promněných U, V a Σ

Při projekci těchto hodnot do dvou dimenzionálního prostoru matice U koresponduje s katalogem položek a matice V koresponduje s množinou uživatelů. Ačkoliv v tomto příkladu není možné pozorovat žádné shluky uživatelů, můžeme zde vidět dva shluky položek z matice U.[15]



Obrázek 14: Dimenzionální prostor matice U převzato z [15]

Při pohledu na originální ohodnocení je jasné, že Položka 1 a Položka 4 mají nějakým způsobem podobné hodnocení. To samé platí pro Položku 2 a Položku 3 z druhého shluku, který se nachází pod osou x. Co se týče uživatelů, jediné co je zřejmé, že Uživatel 4 je poměrně vzdálený od ostatních.[15]

Cílem tohoto procesu je ale získat doporučení pro aktuálního uživatele Alice. Nejprve je nutné určit, kde se bude Alice v prostoru nacházet. K nalezení tohoto bodu se použije vektor ohodnocení Alice [5, 3, 4, 4], který se vynásobí s dvousloupcovou podmaticí U_2 a invertovanou dvousloupcovou singulárně hodnotovou maticí Σ_2.

$$Alice_{2D} = Alice \times U^2 \times \Sigma_2^{-1} = [-0{,}64;\ 0{,}30]$$

Nyní nohou být použity různé strategie pro generování předpovědi. Jednou možností je, je najít v prostoru sousedy a použít jejich hodnocení k vytvoření predikce. [15]

4.2.3.2.2 Slope One

Jedním z kolaborativní doporučení je doporučení, které predikuje potencionální ohodnocení položky aktivním uživatelem na základě ohodnocení ostatních uživatelů. Algoritmus Slope One je založen na tak zvaném „poularity defferential“, nebo-li na rozdílu popularity mezi ohodnocenými položkami uživatelů. Jedním způsobem, jak tento rozdíl u dvou položek určit je jednoduše od sebe odečíst jejich průměrné ohodnocení. Tato vypočtená hodnota je dále použita na předpověď ohodnocení jedné z těchto položek dalším uživatelem. [18]

Přiklad převzat z [15]:

Máme tedy tabulku ohodnocení a úkolem je najít předpověď ohodnocení Položky 3 Alicí.

	Položka 1	*Položka 2*	*Položka 3*
Alice	2	5	?
Uživatel2	3	2	5
Uživatel3	4		3

Tabulka 11: Příklad Slope One

V tabulce jsou obsaženy dvě další blízké ohodnocení pro Položku 1 a Položku 3. V jednom případě je Položka 3 ohodnocena o 2 více (5 – 3 = 2) a v dalším o 1 méně (3 – 4), než Položka 1. Průměrná vzdálenost mezi těmito položkami je tedy (2 + (-1))/2 = 0,5. Společný rating Položky 3 a Položky 2 je jen jeden a jeho vzdálenost je (5 – 2) = 3. Předpověď ohodnocení Položky 3 na základě Položky 1 a Alčina ohodnocení 2 je 2 + 0,5 = 2,5 a na základě Položky 2 a Alčina ohodnocení 5 je 5 + 3 =

8. Celková predikce vznikne zprůměrováním vypočtených hodnot, do kterých se započítají i dříve vypočtené vzdálenosti:

$$pred(Alice, Položka3) = \frac{2 \times 2{,}5 + 1 \times 8}{2+1} = 4{,}33,$$

Obecně tedy platí, že máme databázi R, dále ohodnocení aktuálního uživatele je obsaženo v nekompletním poli *u* a u_i je ohodnocení položky *i* uživatelem. Dále též platí pro dvě položky *i* a *j*, že $S_{j,i}(R)$ je množina ohodnocení, která je zastoupena v obou ohodnocení pro položku i *i* položku *j*. Pak průměrná odchylka dvou výše zmíněných položek je vypočtena následovně:[15]

$$odchylka_{j,i} = \sum_{(u_j,u_i)\in S_{j,i}(R)} \frac{u_j - u_i}{|S_{j,i}(R)|}.$$

Jak již bylo zmíněno, pro každou položku *i* můžeme vytvořit předpověď pro položku *j* a uživatele *u* jako $odchylka_{j,i} + u_i$. Jednoduchou kombinací této individuální předpovědi nad všemi položkami je:

$$pred(u,j) = \frac{\sum_{i \in Relevant(u,j)} (odchylka_{j,i} + u_i)}{|Relevant(u,j)|},$$

kde funkce *Relevant(u, j)* vrací množinu relevantních položek, těch, které splňují podmínku, že kromě uživatele *u* existuje alespoň jeden další uživatel, který položku *j* ohodnotil (dále co-hodnocení). Jinými slovy $Relevant(u,j) = \{i | i \in S(u), i \neq j, |S_{i,j}(R)| > 0\}$, kde *S(u)* značí množinu vstupů uživatele *u*. Tento vzorec zjednodušen na $Relevant(u,j) = S(u) - \{j\}$, když $j \in S(u)$.

Intuitivní problém tohoto základního předpovědního schématu je ten, že nepočítá s počtem co-hodnocení položek. Tato metoda předpovědi by byla logicky přesnější a lepší, kdyby zde byl počet co-hodnocených položek. Z tohoto důvodu je schéma rozšířeno o váhy odchylek založených na počtu co-hodnocení:

$$pred(u,j) = \frac{\sum_{i \in S(u)-\{j\}} (odchylka_{j,i} + u_i) * \left|S_{i,j}(R)\right|}{\sum_{i \in S(u)-\{j\}} * \left|S_{i,j}(R)\right|}.$$

4.2.3.3 Nevýhody kolaborativního filtrování

Problém studeného startu

Jako u Doporučení založeného na obsahu se i u CF metod může stát, že systém neví o novém uživateli, co má rád a co preferuje, a proto mu není schopen vytvořit doporučení. Jedná se například o nového uživatele, který si žádnou položku ještě nekoupil, ani žádnou neohodnotil. Tento problém se řeší metodou, kdy se při registraci uživatele systém zeptá na jeho preference, nebo mu zobrazí určitý počet vybraných položek, které musí nový uživatel ohodnotit. Dalším řešením je využití hybridních doporučovacích přístupů, které mají v sobě zakomponovány i jednodušší doporučení (top 10 položek, nejpopulárnější položky, nejkupovanější položky, atd.), které jsou použity v případě nedostupnosti předpovědi z kolaborativního filtrování. Podobným problémem je i taková situace, když je nová položka přidána do systému. Tento problém se řeší také hybridním doporučením.[10,15]

Problém šedých ovcí (Grey sheep problem)

Šedými ovcemi rozumíme takové uživatele, které je, v porovnání se zbytkem komunity, obtížné zařadit do nějaké určité skupiny založené na potřebách uživatele. Tito uživatelé v porovnání s ostatními mají nízké korelační koeficienty, z toho důvodu, že se jejich potřeby překrývají jen z části. Tito uživatelé představují hrozbu hlavně pro malé a střední uživatele tím, že díky těmto šedým ovcím uživatel nedostane přesná a správná doporučení z důvodu ovlivnění výsledku. Dalším problémem, který

představují, je ten, že mohou v určitých případech ovlivnit doporučení celé komunity.[20] Tento problém je řešen spojením doporučení založeného na obsahu a kolaborativního filtrování. V tomto řešení váhy na obsahu založených a CF předpovědí jsou určeny (determinovány) na úrovni uživatele. Což dovoluje systému určit optimální mix z na Obsahu založených a CF doporučení pro každého uživatele, což pomáhá vyřešit problém šedých ovcí.[20,21]

Řídkost dat

V teoretické rovině je předpokládáno, že doporučovací systém bude mít dostatek informací a dat k vytvoření doporučení. V praxi to ale neplatí. V reálných aplikacích, kde je přítomen velký soubor položek, mají matice ohodnocení tendenci být velmi řídké, což je způsobeno tím, že uživatelé mají tendenci hodnotit jen malou část položek, které jsou v systému. Tento stav má za následek obtížnější a méně přesné doporučení potencionálně zajímavých položek pro aktuálního uživatele. Jedním z řešení situace, kdy je k dispozici jen málo uživatelských ohodnocení, je využití dodatečných informací, které se týkají uživatelů, jako je věk, pohlaví, vzdělání a další podobné informace, které by pomohly klasifikovat uživatele. Množina sousedů aktivního uživatele je pak založena nejen na analýze explicitní nebo implicitní zpětné vazby, ale i na informacích mimo matici ohodnocení.[15]

Synonymie

Synonymií se rozumí případ, kdy určitý počet stejných nebo velmi podobných položek má různá jména nebo vstupy. Doporučovací systémy jsou neschopny odhalit tuto skrytou spojitost a pracovat s těmito produkty jako s každým zvlášť. Například zdánlivě věci jako 'dětský film' a 'pohádka' jsou v jádru vlastně jedna a tatáž věc, ale na Paměti založené CF systémy by nenašly přes výpočet podobnosti mezi nimi

žádnou shodu. Ve skutečnosti stupně rozdílnosti/variability v popisných termínech které se používají, jsou větší, než se předpokládalo. [20,22]

Předchozí pokusy o vyřešení problémů se synonymií závisely na intelektuální nebo automatických rozšiřujících termínech, nebo na sestrojení Thesauru (program pro hledání synonym). Nevýhoda pro plně automatické metody je v tom, že některé připojené termíny se mohou lišit významem od záměru. Což vede k rapidnímu snížení výkonu vytváření doporučení.[20]

Problém škálovatelnosti

V situaci, kdy počty existujících uživatelů a položek nesmírně rostou, tradiční CF algoritmy budou trpět vážnými problémy se škálovatelností s výpočetními zdroji přesahující rámec praktické, nebo přijatelné úrovně. Doporučovací systém musí reagovat okamžitě na online požadavky a doporučovat pro všechny uživatele nehledě na množství jejich nákupů a historii hodnocení, což požaduje vysokou škálovatelnost CF systému.[20]

Shilling attacks

V případech, kde každý může poskytnout doporučení, se mohou lidi uchýlit k tomu, že budou dávat velké množství pozitivních ohodnocení svým vlastním materiálům a negativní doporučení konkurentům. Je proto pro CF systémy vhodné použít taková opatření, která odradí uživatele od tohoto fenoménu. Při eliminaci tohoto problému bylo zjištěno, že Item-based CF algoritmus byl výrazně méně postižen těmito útoky než algoritmus User-based. Dále je doporučeno hledat a využívat nové přístupy pro vyhodnocování a detekci falešných útoků na doporučovací systémy.

4.2.4 Doporučení založené na obsahu

Pro aplikování technik kolaborativního doporučení nemusí být, kromě uživatelských hodnocení, k doporučení známy charakteristiky nebo vlastnosti položek. Hlavni výhodou je, že při takovém řešení není potřeba nákladných úloh k zajištění detailních a aktuálních informací o položkách. Na druhou stranu s čistým kolaborativním filtrováním není možné intuitivně vybrat takové položky k doporučení v závislosti na jejich charakteristikách a specifických preferencích uživatele. Například v reálné situaci by bylo logické doporučit uživateli novou knihu Pán Prstenů, pokud o něm víme, že vždy preferoval fantazy knihy a toho tento přístup není schopen. Elektronický doporučovací systém může tohoto dosáhnout jedině tehdy, když jsou známy dva druhy informací: vlastnosti položky a uživatelský profil, který nějakým způsobem popisuje uživatelovi minulé zájmy. Doporučovací úloha pak spočívá v určení takových položek, které nejvíce odpovídají preferencím aktuálního uživatele. Tento proces nazývá Doporučení založené na obsahu. Ačkoliv tento přístup musí spoléhat na dodatečné informace o preferencích uživatele a položkách, není zde potřeba existence velké uživatelské komunity nebo historie uživatelských ohodnocení položek, což znamená, že doporučení potenciálních položek může být vygenerováno i když v systému bude jediný uživatel.[15]

V praxi technické popisy funkcí a charakteristik položek jsou často k dispozici v elektronické podobě tak, jak jsou z části již poskytovány poskytovatelem nebo výrobcem zboží. Náročné ale zůstává získání kvalitativních a subjektivních vlastností. Například v oblastech co se týče kvality a vkusu důvody, proč někdo má něco rád, nejsou vždy spojeny s charakteristikami produktu a mohou být založeny na subjektivním dojmu vnějšího dizajnu položky. Jedním příkladem řešení je „Music Gnome Project“, kde data jsou používána k doporučení muziky v populárním internetovém radiu. V tomto projektu jsou písně manuálně okomentovány hudebníky, co se týče vlastností, jako je instrumentace, ovlivnění nebo použité nástroje.[9,15]

Doporučovací proces doporučení je složen ze tří kroků, kde každý krok zastupuje jiná komponenta[9]:

- Analyzátor obsahu – Hlavním úkolem této komponenty je vytvořit profil položky z informací o vlastnostech položky, nebo jejím obsahu.

- Profilová komponenta – Shromažďuje data, která reprezentují uživatelovy preference a vytváří tak uživatelův profil.

- Filtrovací komponenta – Tato komponenta vytváří seznam potencionálních položek, které se doporučí uživateli na základě jeho profilu.

Nejjednodušší způsob pro popis katalogových položek je tedy výslovný seznam vlastností každé položky. Pro doporučení knih může být například použity vlastnosti jako žánr, jméno autora, vydavatel nebo cokoliv co popisuje položku a je uloženo v databázovém systému. Jakmile jsou uživatelovi preference popsány formou jeho zájmů použitím této množiny vlastností, doporučovací úloha porovná charakteristiky položky a uživatelovi preference.[15]

Příklad převzat z [15]:

Máme tedy tabulku, kde jsou knihy popsány charakteristikami, jako je název, žánr, autor, typ, cena a klíčová slova.

Název	**Žánr**	**Autor**	**Typ**	**Cena**	**Kličová slova**
The Night of the gun	Monografie	David Carr	Brožovaný	29,90	Osobní vzpomínky, New York

The Lace Reader	Fikce, Misteriózní	Brunonia Barry	Pevná vazba	49,90	Fikce, detektivní, historicky
Into the Fire	Romantický, Napětí	Suzzane Brockmann	Pevná vazba	45,90	Fikce, vražda
...	...	...	...	...	...

Tabulka 12: Příklad Doporučení založené na obsahu

Dále máme uživatelský profil Uživatele Alice, které jsou uloženy v přesně v té samé podobě.

Název	**Žánr**	**Autor**	**Typ**	**Cena**	**Kličová slova**
...	Fikce, Napětí	Brunonia Barry, Ken Follet	Brožovaný	25,65	detektivní, New York

Tabulka 13: Příklad Doporučení založené na obsahu - uživatelský profil

Systém může sestavit uživatelský profil mnoha způsoby. Nejjednodušší způsob je, se přímo Alice zeptat například na požadovanou cenu nebo na množinu preferovaných žánrů. Druhým způsobem je požádat uživatele o ohodnocení množiny položek buď jako celek nebo po různých skupinách. V neposlední řadě může být také použito generování uživatelského profilu na základě historie procházení či nákupů položek. V tomto příkladě je předpokládáno, že žánr a autor a typ vyplnila sama Alice a zbytek atributů byl vygenerován systémem.[15]

Jak již bylo řečeno, pro generování doporučení doporučovací systémy založené na obsahu typicky posuzují, jak silně je ještě neviděná položka podobná těm, které aktuální uživatel preferuje. Je dána tedy ještě nenavštívená knihu *B*. Systém by mohl jednoduše zkontrolovat, zda žánr této knihy je v uživatelském profilu Alice, kde podobnost je v tomto případě 1 nebo 0. Funkce sim je aplikována na knihu *B*, která je v nějakém vztahu *R* s Alicí a je následující:[10,15]

$$\hat{u}_R^{CB}(Alice, B_{alice}) = \sum_{(Alice, B_{alice}) \in R} sim(B_{alice}, B),$$

kde B je nenavštívená kniha, B_{alice} zastupuje uživatelský profil Alice. Suma může byt nahrazena jakoukoliv více sofistikovanou funkcí.[10]

Další možností je vypočíst podobnost nebo překrytí klíčových slov. Jako typickou metrikou podobnosti, která je vhodná pro multi-hodnotové charakteristiky je Diceuv koeficient, který je dán následovně:[15]

$$\frac{2 \times \left|klicovaSlova(b_i) \cap klicovaSlova(b_j)\right|}{|klicovaSlova(b_i)| + \left|klicovaSlova(b_j)\right|},$$

kde b_i a b_j jsou porovnávané knihy a množina *klicovaSlova* obsahuje klíčová slova všech knih. V našem případě se nejvíce podobá Aličině profilu kniha *The Lace Reader* tak ji systém doporučí.

Jako každý přístup má i tento nedostatky:

Limitovaná analýza obsahu

Soubor vlastností přiřazených ke každému objektu je vždy omezený a nikdy není schopen plně charakterizovat objekt. Kromě toho existují hodnoty charakteristik, které

musí být určeny buď ručně, což je časově náročné, nebo automaticky kde tato metoda v současné době dobře funguje pouze pro textové dokumenty.[10]

Přespecializace

Tato metoda vždy doporučuje objekty, které jsou podobné objektů z minulosti, jenž subjekt oblíbil v minulosti. Proto doporučující nikdy rozšiřuje obzory doporučením různých objektů.[10]

Problém nového uživatele

Když se objeví nový subjekt v systému a nemá žádný vztah k objektům, systém není schopen vytvořit doporučení pro tento subjekt.[10]

4.2.5 Doporučení založené na vědomostech

Výše zmíněné doporučovací techniky jako kolaborativní filtrování a doporučení založené na obsahu jsou hojně využívány ve většině online obchodních webů nebo systémů. Tyto dva přístupy mají své silné stránky jako je například rychlé osobní, na míru vytvořené, doporučení. Existují ale situace, kde tyto algoritmy nejsou vhodnou volbou a tyto přístupy selhávají. Tyká se to například položek, které uživatel nekupuje tak často jako je dům, auto, atd. Systém sice může mít k dispozici data o ohodnocení, se kterými může hledat potencionální položky, ale těchto dat je málo (problém řídkosti dat – nepoužitelná pro kolaborativní filtrování) a zároveň ohodnocení budou již několik let stará a ty jsou, z důvodu uživatelovi měnící se preference v čase, nepoužitelná pro Doporučení založené na obsahu. V tomto případě nastává problém, který právě řeší Knowledge-based recommendation nebo-li Doporučení založené na vědomostech.[15]

Doporučení založené na vědomostech tedy pomáhá řešit výše zmíněné situace. Výhodou těchto systémů je, že zde není riziko výskytu žádného dalšího, protože nejsou

zde potřeba žádná data o implicitních, nebo explicitních zpětných vazeb od všech uživatelů, na základě které by byl prováděn doporučovací proces. Doporučení je generováno nezávisle a jen na základě individuálních zájmů a preferencí aktuálního uživatele, buď formou podobnosti mezi jeho požadavky a položkami, nebo formou explicitních doporučovacích pravidel. Tradiční interpretací tohoto doporučovacího přístupu je soustředit se na „information filtering“ aspekt, ve kterém položky, které by pravděpodobně uživatel preferoval, jsou vyfiltrovány a zobrazeny.[15]

Obecně tyto systémy spoléhají na detailních vědomostech o položce a jejích vlastnostech, které jsou uloženy v databázi. Doporučovací proces spočívá ve výběru položek z databáze, které se shodují s potřebami, preferencemi, nebo požadavky uživatele. Uživatelův požadavek může například být, že chce notebook do 10 000 Kč a kapacitu disku 1Tb.[15]

Jedním z problémů tohoto přístupu jsou neosobní doporučení. Vzhledem k tomu, že si neuchovává žádné informace a neexistují zde uživatelské profily o zákaznících, jsou doporučení založena jen na informacích poskytnutých zákazníkem během hledání. Dalším problémem je „obtěžování“ zákazníka, který, aby dostal vhodná a přesná doporučení, musí plně a detailně specifikovat jeho preference a zájmy.[10]

Máme dva základní typy Doporučení založené na vědomostech: doporučení na základě omezení (constraint-based) nebo na základě požadavků (case-based). Oba přístupy jsou si podobné v tom, že uživatel musí popsat své preference a v dalším kroku systém nalezne optimální doporučení. Když není nalezena žádná potencionální položka, je uživatel vyzván ke změně jím specifikovaných vlastností. Naopak tyto dva typy jsou rozdílné ve způsobu nalézání položek. Case-based se soustřeďuje na získání podobných položek na základě různých měřítek podobnosti, zatímco v Constraint-based spoléhá na množinu explicitně definovaných doporučovacích pravidel.[15]

4.2.5.1 Constraint-based

Klasický „constraint satisfaction problem“(CSP) může být popsán n-ticí *(V, D, C),* kde

- V je množina proměnných,
- D je množina konečných domén pro tyto proměnné a
- C je množina omezení, které popisují hodnoty, které může proměnná obsahovat.

Výsledek CSP pak odpovídá zadání hodnoty každé proměnné *V* tak, aby všechna omezení byla akceptována. Constraint-based doporučovací systémy jsou založeny na této formalismu využitím vědomostní báze systému, která typicky zahrnuje dvě různé množiny proměnných ($V = V_C, V_{PROD}$), kde jedna popisuje potencionální uživatelovi požadavky a další popisuje položku a její vlastnosti. Dále zde jsou 3 rozdílné množiny omezení ($C = C_R, C_F, C_{PROD}$) definující jaké potencionální položky a za jakých okolností by měly byt uživateli doporučeny. Úloha identifikování množiny položek, které splňují požadavky zákazníka, je nazývána doporučovací úloha.

Příklad doporučovací úlohy ($V_C, V_{PROD}, C_R, C_F, REQ$) a odpovídající odpovědi systému:[15]

V_c {maxcena(0...20000, procesor(i3, i5, i7), disk(512,1024,2048), ram(4,8,16)}

V_{prod} {maxcena(0...10000), lcd(15), disk(1024), vydrzbaterie(10h}

C_f {ram = 16 → maxcena > 5000}

C_r {disk = 2048 → maxcena >10000}

C_{prod} {(id = 1 & cena = 5000 & procesor = i5 & disk = 512 & ram=4),(id = 1 & cena = 8520 & procesor = i5 & disk = 1024 & ram=8)..... (id = 258 & cena = 10500 & procesor = i5 & disk = 512 & ram=16)}

REQ {maxcena = 5000, procesor = i5, ram = 8}

RES { maxcena = 5000, procesor = i5, ram = 8, id = 5, cena = 4500, procesor = i5, ram = 8}

Kde:

$\boldsymbol{V_C}$ – Množina popisuje všechny možné požadavky zákazníka, kde maxcena je maximální cena notebooku, procesor znamená typ procesoru, ram velikost paměti, atd.

$\boldsymbol{V_{PROD}}$ – Tato množina popisuje vlastnosti položky ve výběru, například lcd značí úhlopříčku displeje.

$\boldsymbol{C_R}$ – Povolené výběry požadavků uživatele, například notebook s velikostí ram = 16 nemůže mít maximální cenu menší nebo rovno 500 Kč.

$\boldsymbol{C_F}$ – Určuje, podle jakých podmínek bude jaký produkt vybrán. Filtrovací podmínky definují vztahy mezi vlastnostmi produktu a požadavky zákazníka.

$\boldsymbol{C_{PROD}}$ – Množina všech dostupných položek.

Požadavky zákazníka REQ mohou být kódovány jakou unární omezení nad proměnnými v $\boldsymbol{V_C}$ a $\boldsymbol{V_{prod}}$ například maxcena = 2000. Formálně tedy každé řešení CSP ($V = V_C \cup V_{PROD}, D, C = C_R \cup C_F \cup REQ$) odpovídá konzistentnímu doporučení položek. V mnoha praktických nastaveních proměnné v množině V_c nemusí být konkretizovány. Jako relevantní proměnné jsou brány ty, které jsou spojeny s požadavkem REQ.[15]

Konjunktivní dotazy

Konjunktivní dotazy jsou dalším způsobem získávání položek při Constraint-based doporučování. Způsob získávání pro uživatele potencionálně zajímavých položek je zde převeden z CSP na úlohu filtrování dat. Hlavním úkolem není nalézt řešení CSP

ale vytvořit konjunktivní databázový dotaz, který je proveden nad databází položek. Konjunktivní databázový dotaz je takový, u něhož jsou hodnoty výběrových kritérií spojeny konjunktivně.

Například $\sigma_{[procesor = i5,cena<6000]}(P)$ je konjunktivní dotaz na databázovou tabulku *P*, kde σ reprezentuje volbu operátoru a *[procesor = i5,cena<6000]*koresponduje s výběrovými kritérii.

4.2.5.2 Case-based

V Case-based přístupu jsou potencionální oblíbené položky pro aktivního uživatele získány využitím podobnostních meřítek, které popisují, jské vlastnosti položky odpovídají uživatelem zadaným požadavkům. Je zde použita takzvaná „distance similarity“, nebo-li podobnost ve vzdálenosti položky *p* s požadavky $r \in REQ$, která je definována jako funkce *sim(p, r)* vyjadřující pro každou hodnotu atributu položky $\Phi_r(p)$ její vzdálenost od požadavku aktuálního uživatele $r \in REQ$:

$$similarity(p, REQ) = \frac{\sum_{r \in REQ} w_r * sim(p, r)}{\sum_{r \in REQ} w_r},$$

kde w_r je váha důležitosti požadavku *r*.

Ve skutečném světě existují i takové části požadavku, na které by chtěl zákazník přidat důraz, nebo naopak u některých zase ubrat. Například, když chce počítač na hraní her a nezáleží mu na ceně, na vlastnost typ grafické karty bude klást největší důraz a na cenu nejmenší. V tomto případě mluvíme tzv. MIB (more-is-better) a LIB (less-is-better) částech. Vzorec pro výpočet podobnosti s ohledem MIB a LIB získáme následovně:

Nejprve vzorec pro podobnost mezi položkou *p* a požadavkem aktuálního uživatele *r* zahrnující MIB vypadá takto:

$$sim(p,r) = \frac{\Phi_{\mathrm{r}}(\mathrm{p}) - \min(\mathrm{r})}{\max(r) - \min(r)}.$$

Podobnost mezi položkou *p* a požadavkem aktuálního uživatele *r* s LIB je následující:

$$sim(p,r) = \frac{\max(r) - \Phi_{\mathrm{r}}(\mathrm{p})}{\max(r) - \min(r)}$$

Existují také situace, ve kterých by podobnost měla být výhradně založena na originálně definovaných požadavcích. Například, když uživatel požaduje určitou velikost monitoru a také největší výdrž, pak monitor o největší úhlopříčce není optimální řešení. Pro tyto případy je dán vzorec:

$$sim(p,r) = 1 - \frac{|\Phi_{\mathrm{r}}(\mathrm{p}) - \mathrm{r}|}{\max(r) - \min(r)}$$

4.2.6 **Hybridní přístup**

Budováním hybridního systému je používáno pro kombinaci silných stránek různých algoritmů a modelů k překonání slabin. Z lingvistického hlediska je termín *hybrid* odvozen z latinského slova *hybrida* (smíšený původ) a značí objekt, který je tvořen kombinací dvou rozdílných elementů. Analogicky hybridní doporučovací systémy jsou technické přístupy, které kombinují dvě nebo více algoritmických implementací doporučovacích komponent.[15]

Existuje mnoho rozdílných přístupů kombinování jednotlivých doporučovacích technik. Většinou se jedná o kombinaci kolaborativního filtrování jinou technikou a tím se vypořádat se slabými stránkami tohoto přístupu. Nejpoužívanější kombinace jsou: [23]

Implementace kolaborativních metod a content-based metod zvlášť a odděleně s následnou kombinací jejich předpovědí potencionálních položek.

Zde máme dva možné scénáře. Je možné buď zkombinovat výstup z individuálních doporučovacích systémů do jedné velké předpovědi, nebo při procesu individuálního doporučení v jakýkoliv moment, na základě kvalitativních metrik, vybrat, jaký přistup je v dané situaci lepší a použít jej.[23]

Vložení vlastností a algoritmů content-based přístupu do kolaborativních metod.

Tato kombinace umožnuje předejít problému řídkosti dat v situaci, kdy v čistém kolaborativním přístupu je malý počet uživatelských párů, kteří mají značné množství stejných ohodnocených položek. Další výhodou této kombinace je možnost doporučení takových položek uživatelům, které nejsou ostatními uživateli vysoce ohodnoceny a jsou hodnoceny vysoce oproti uživatelskému profilu.[23]

Vložení vlastností a algoritmů kolaborativních metod do content-based metod.

Jedna z nepopulárnějších kombinací, kde jsou použity techniky na redukci dimenzí na množinu uživatelských profilů. [23]

Vytvoření takového jednotného modelu, kde jsou zastoupeny charakteristiky obou přístupů a dokáží spolu spolupracovat.[23]

Druhy hybridních návrhů podrobněji

4.2.6.1 Monolitický hybridní doporučovací systém

Monolitický hybridní doporučovací systém se skládá z jedné doporučovací komponenty, ve které je začleněno více přístupů předzpracováním a kombinováním několika znalostních zdrojů. Hybridizace je zde dosažena modifikací chování vestavěných algoritmů pro využití různých typů vstupních dat. Typicky kroky k předzpracování určitých dat jsou použity k transformování vstupních dat do takové reprezentace, která může být využita určitým algoritmickým vzorem. Do této kategorie patří strategie zvané „feature combination“ (kombinace funkcí) a „feature augmentation“ (rozšíření funkcí).[15]

Kombinace funkcí

Je monolitická hybridní doporučovací komponenta, která využívá pestrou škálu vstupních dat. Příkladem komponenty může být kombinace kolaborativních vlastností, co se uživatelovi líbí, nelíbí, s obsahovými vlastnostmi katalogových položek v oblasti knih.[15]

Máme tabulku, která představuje uživatelovo unární ohodnocení produktu v katalogu.

	Položka 1	*Položka 2*	*Položka 3*	*Položka 4*	*Položka 5*
Alice	1			1	

Uživatel1		1	1	1
Uživatel2	1	1		1
Uživatel3	1		1	
Uživatel4				1

Tabulka 14: Příklad Kombinace funkcí

Ohodnocení představuje implicitně zaznamenanou zpětnou vazbu jako je nákup. Informace o produktu je zde v tomto příkladu omezena na žánr položky (knihy). Tabulka položek a jejich žánru vypadá následovně:

	Žánr
Položka 1	Romantika
Položka 2	Mysteriózní
Položka 3	Mysteriózní
Položka 4	Mysteriózní
Položka 5	Fikce

Tabulka 15: Příklad Kombinace funkcí - tabulka položek a jejich žánru

V této situaci je jasné, že samotný kolaborativní přístup, bez započítání žánru položky, oba Uživatele 1 a Uživatele 2 by usuzoval jako takové sousedy, které jsou stejnou mírou podobné Alici. To by zapříčinilo neobjektivní a špatnou predikci potencionálních položek pro Alici. Řešením je využití hybridního přístupu kombinací. Následující tabulka poskytuje dekódování informací nacházející se v předchozích tabulkách příkladu. [15]

Vlastnost	*Alice*	*Uživate1*	*Uživate2*	*Uživate3*	*Uživate4*
Uživatel má rád mnoho mysteriózních knih	true	true			
Uživatel má rád několik mysteriózních knih			true	true	
Uživatel má rád mnoho romantických knih					
Uživatel má rád několik romantických knih			true	true	
Uživatel má rád mnoho knih fikce					
Uživatel má rád mnoho knih fikce		true	true		true

Tabulka 16: Příklad Kombinace funkcí - zpracovaná tabulka

Tyto vlastnosti jsou odvozeny podle následujících pravidel. Když si uživatel kupuje hlavně knihy žánru X (dvě třetiny všech nákupů a nejméně dvě knihy zakoupeny) je mu nastavena charakteristika *Uživatel má rád mnoho X knih* na true. Analogicky nastavení charakteristiky *Uživatel má rád několik X knih* na true se provede v případě, když žánr zakoupené knihy se objevuje ve třetině jeho historii nákupů. Transformace se zá na první pohled triviální, ale to nicméně reflektuje, že nějaká znalost, jako žánr položky může vést ke značným vylepšením. Z počátku se zdálo, že Alice má podobné zájmy jako Uživatel1 a Uživatel2, ale obraz se po transformování matice změnil. Proces ukázal, že chování Uživatele2 je rozdílné tím, že rád nakupuje několik knih od všech 3 žánrů na rozdíl od Alice, která preferuje nakupování jen z jediného žánru. Proto Uzivatele2 z výpočtu predikce vynecháme.[15]

Rozšíření funkcí

Rozšíření funkcí je dalším z hybridních monolitických návrhů, který muže být použit k integraci několika doporučovacích algoritmů. V porovnání s hybridním návrhem kombinace funkcí, tento hybrid nekombinuje a nepředzpracovává více typů vstupu, ale spíše aplikuje více komplexní transformační kroky. Ve skutečnosti výstup z přispívajících doporučovacích systému vylepšuje aktuální doporučení předzpracováním jejich znalostních zdrojů. Příkladem této varianty je tzv. Content-boosted collaborative filtering, které predikuje uživatelovo předpokládané ohodnocení na základě kolaborativního mechanismu a content-based predikce.[15]

Přiklad (převzat z [15]):

Máme tabulku, která reprezentuje matici uživatel/položka společně s hodnotami ohodnocení položky pro Položku5 ($\upsilon_{Užiatel,Položka4}$), hodnotami Pearsonova korelačního koeficientu $P_{Alice,\ Užiatel}$ určující podobnost mezi Alicí a příslušnými uživateli a počet překrývajících se ohodnocení mezi Alicí a ostatními uživateli ($\upsilon_{Alice,Užiatel}$).

Uživatel	$\upsilon_{Užiatel,Položka4}$	$P_{Alice,Užiatel}$	$\upsilon_{Užiatel}$	$\upsilon_{Alice,Užiatel}$
Alice	?		40	
Uživatel1	4	0,8	14	6
Uživatel2	2,2	0,7	55	28

Tabulka 17: Příklad Rozšíření funkcí

Prvním krokem k vytvoření predikce pro Alici je vytvoření pseudo vektoru uživatelských ohodnocení $\upsilon_{u,p}$:

$$\upsilon_{u,p} = \begin{cases} r_{u,p} : když\ uživatel\ u\ ohodnotil\ položku\ p \\ c_{u,p} : jinak\ predikce\ Content-based \end{cases}$$

Ve druhém kroku algoritmus na základě těchto pseudo hodnocení vypočte předpověď. V závislosti na počtu ohodnocených položek a počtu co-hodnocených položek mezi dvěma uživateli mohou být použity váhoví činitelé pro úpravu hodnoty předpovědi pro specifický pár uživatel položka *a*, *p* následovně:[15]

$$rec_{cbcf}(a,p) = \left(s\omega_a c_{a,p} + \sum_{\substack{u=1 \\ u \neq a}}^{n} h\omega_{a,u} P_{a,u} \upsilon_{u,p} \right) / \left(s\omega_a + \sum_{\substack{u=1 \\ u \neq a}}^{n} h\omega_{a,u} P_{a,u} \right),$$

kde $h\omega_{a,u}$ je hybridní korelační váha definována jako:

$$h\omega_{a,u} = sg_{a,u} + hm_{a,u}\ ,\text{kde}$$

$$sg_{a,u} = \begin{cases} \frac{n_{a,u}}{50} : když\ n_{a,u} < 50 \\ 1 \end{cases} \text{ a } hm_{a,u} = \frac{2m_a m_u}{m_a + m_u}.$$

Hybridní korelační váha upravuje vypočítanou Pearsonovu korelaci na základě významového váhového vektoru $sg_{a,u}$, který upřednostňuje sousedy s více co-ohodnocenýni položkami.[15]

4.2.6.2 Paralelní hybridní doporučovací systém.

Paralelní hybridní doporučovací systém využívá několika doporučovacích přístupů, které pracují společně bok po boku, a používá specifický hybridizační mechanismus k agregaci jejich výstupů.[15]

Vážený hybridní doporučovací systém

Vážený hybridní doporučovací systém kombinuje doporučení dvou nebo více doporučovacích modulů v systému. Celkové doporučení je vypočteno pomocí vážené sumy výstupů z těchto modulů. Výhodou tohoto systému je využití potenciálu všech doporučovacích modulů k předpovězení doporučení a jeho relativní jednoduchost zavedení tohoto hybridní ho systému. Problémem může být, že ne všechny techniky jsou si ve všech situacích rovny, ale v tomto přístupu se často hodnotí ekvivalentně. Například kolaborativní doporučení bude mít slabší váhu při doporučování položek s malým počtem hodnocení.[9,24]

Pro n různých doporučovacích funkcí rec_k se spojenými relativními váhami β_k se vypočte doporučení následovně:

$$rec_{weighted}(u,p) = \sum_{k=1}^{n} \beta_k * rec_k(u,p),$$

kde skóre položek musí být omezeno do stejného rozsahu pro všechny doporučovací moduly a $\sum_{k=1}^{n} \beta_k = 1$.

Smíšený hybridní doporučovací systém

Strategie smíšených hybridních doporučovacích systémů kombinuje výsledky různých doporučovacích modulů na úrovní uživatelského rozhraní, ve kterém výsledky jednotlivých technik jsou prezentovány dohromady. Například při použití kolaborativní doporučení a zároveň i doporučení založeného na obsahu je možné se vyhnout problému studeného startu tím, že první technika neposkytne výsledek, ale druhá ano.[9,15]

Celkové doporučení pro uživatele *u* a položku *i* je množina n-tic *⟨score,k⟩*, každé *n* doporučovacích funkce rec_k [15]:

$$rec_{mixed}(u,i) = \bigcup_{k=1}^{n} \langle rec_k(u,i), k \rangle$$

Položky s největším skóre z každého doporučovacího modelu jsou poté zobrazeny uživateli vedle sebe. Ačkoliv je zde nutné mít zavedenou formu řešení konfliktů pro případ různých výsledku pro jednu a tu samou entitu.[15]

Přepínací hybridní doporučovací systém

V tomto přístupu systém použije nějaké kritérium k přepínání mezi doporučovacími moduly. Například, když hlavní přístup nedokáže vyprodukovat predikci pro aktuálního uživatele, použije se jiný přístup, který doporučí potencionální položky. Jinými slovy přepínací hybridní doporučovací systém vyžaduje autoritu, která rozhodne, jaký doporučovací přístup a v jaké situaci by měl být použit v závislosti na uživatelském profilu, nebo kvalitě výsledků doporučení. Posouzení, jaké výsledky použít je provedeno následovně:[9,15]

$$\exists_1 k : 1 \dots n \; rec_{switching}(u,i) = \; rec_k(u,i),$$

kde *k* je stanovené přepínací kritérium. Například máme hybridní doporučovací systém skládající se z kolaborativního modulu a modulu založeného na obsahu. V situaci problému studeného startu, kdy modul kolaborativního doporučení není schopen za těchto podmínek vytvořit předpověď potenciálních položek je okamžitě přepnuto na modul druhý, který vytvoří doporučení.[15]

4.2.6.3 Pipelined hybridní doporučovací systém

Pipelined (v překladu trubicový) hybridní systém implementuje postupný proces, ve kterém jednotlivé doporučovací moduly postupně pracují za sebou a až poslední vytvoří finální doporučení pro aktuálního uživatele. Jinými slovy předchozí modul předzpracovává vstupní data k vytvoření modelu, který je následně využit pro další etapu nebo pro dodání doporučení. Existují dva druhy tohoto systému, které sami sebe odlišují hlavně podle typu a druhu výstupu, který produkují pro další etapu, Kaskádové hybridy (Cascade hybrids) a Meta-úrovňové hybridy (Meta-level hybrids).

Kaskádový hybridní doporučovací systém

Tento doporučovací systém je založen na sekvenčním seřazení doporučovacích technik, ve kterém každý následující doporučovací modul pouze upřesňuje potencionální doporučení svého předchůdce. Seznam potencionálních položek k doporučení vygenerovaný následujícím technikou je pak omezen množinou položek, které vygeneroval předcházející modul. Jinými slovy první modul v pořadí vyprodukuje hrubé položky na doporučení a tyto položky a jejich pořadí jsou následujícími moduly pouze upřesňovány.[15]

Formálně je předpokládána sekvence *n* modulů, kde rec_1 reprezentuje doporučovací funkci první techniky a rec_n poslední. Z toho tedy vyplívá, že potenciální položka je vygenerována *n*-tou technikou. Nicméně tato potencionální položka k doporučení bude navrhnuta *k*-tou technikou pouze za takové situace byla-li splněna podmínka, že technika *(k - 1)* ji také doporučila. Pak pro všechny $k \geq 2$ indukcí platí následující:[15]

$$rec_{cascade}(u, i) = rec_n(u, i),$$

kde $\forall k \geq 2$ musí platit:

$$rec_k(u,i) = \begin{cases} rec_k(u,i) & : \quad rec_{k-1}(u,i) \neq 0 \\ 0 & : \quad else \end{cases}.$$

Tudíž v kaskádových hybridech všechny techniky (moduly) kromě té první mohou jen upravit uspořádání seznamu potencionálních položek, které dostanou od svého předchůdce, nebo určitou položku vyloučit nastavením jejich „užitečnosti“ na 0. Do tohoto seznamu se již nezavádí nové položky, protože již byly vyloučený technikou s vyšší prioritou. Z toho vyplívá, že jednou kaskádové strategie mají jednu nepříznivou vlastnost a tou je potencionální snížení množiny pro doporučení pro každou následující techniku, která je aplikována. V důsledku tohoto může nastat situace, kdy kaskádový přístup není schopen dodat požadovaný počet návrhů, čímž se snižuje užitečnost systému. Řešením tohoto problému je kombinace kaskádových hybridů s přepínací strategií, která je zmíněna výše.[15]

Meta-úrovňový hybridní doporučovací systém

Tento design hybridního doporučovacího systému se zakládá na principu, kde jeden doporučovací modul vytvoří model, který je dále využíván hlavním modulem, který vytvoří doporučení. Následující vzorec představuje chování tohoto hybridního doporučení, kde n-tá doporučovací technika (ve většině doporučovacích systémů n = 2) využívá předchůdcem vytvořený model Δ.:

$$rec_{meta-level}(u,i) = rec_n(u,i,\Delta_{rec_{n-1}})$$

Příkladem meta-úrovňového hybridu je systém Fab. Zde je použito spojení kolaborativního přístupu, který pracuje s uživatelským modelem, vytvořeným content-based přístupem, založeného na vektoru pojmů kategorií a na uživatelských stupních zájmu o ně. Doporučovací část nenavrhuje položky, které jsou podobné uživatelskému modelu, ale využívá kolaborativní techniku. Ta určuje uživatelovi nejbližší sousedy

na základě modelů položek a generuje takové potencionálně zajímavé položky, které stejní „vrstevníci“ preferují.

Dalším příkladem může být hybrid kombinující kolaborativní přístup s doporučením založeném na znalostech. Tento hybridní přístup generuje binární uživatelskou preferenci ve tvaru $a \rightarrow b$, kde a reprezentuje uživatelův požadavek a b vlastnost produktu. Když například zákazník internetového obchodu s počítači chce koupit jako dárek notebook a nakonec si koupí notebook značky Asus, pak omezení *for_whom = „gift“* → *brand = „Asus“* se uloží do jeho uživatelského profilu a stane se jeho součástí. Při získávání doporučení kolaborativní část získá všechna omezení od všech uživatelových „vrstevníků“. Doporučení založené na znalostech pak aplikuje tyto omezení na databázi produktů a odvozuje návrhy položek.

4.3 Využívané metody

4.3.1 Podobnost uživatelů

Podobnost je hodnota, která udává vztah mezi dvěma body (vektory) v prostoru, které jsou stanoveny podle určitého specifického předpisu funkce. Při menší vzdálenosti jsou si body podobné více a naopak při velké vzdálenosti od sebe méně. Platí tedy, že podobnost je duální mírou – čím je podobnost dvou objektů větší, tím bližší si tyto objekty jsou. A tohoto je využíváno v doporučovacích systémech, které využívají podobnosti uživatelů.

4.3.1.1 Euklidovská podobnost

Euklidovská podobnost je nejvíce používanou metodou pro měření vzdálenosti mezi dvěma objekty. Vzhledem k tomu, že představuje měření nejkratší vzdálenosti

dvou bodů tak, jako by byla změřena pravítkem, můžeme říct, že jde tedy o přímou vzdálenost mezi dvěma body. Tato vzdálenost je určena jako druhá odmocnina sumy čtvercových vzdáleností mezi souřadnicemi objektů. Jinými slovy vzdálenosti mezi jednotlivými objekty jsou počítány podle Pythagorovy věty. Euklidovská vzdálenost je speciálním případem Minkowského vzdálenosti s proměnnou λ = 2. Dále pro ni platí, že výsledkem je hodnota, která je vždy větší (vyšší hodnoty značí menší podobnost), nebo - pro stejné body - rovna nule. Tedy pro každé 2 objekty se euklidovská vzdálenost spočítá dle následujícího vztahu: [25,26]

$$|AB| = \sqrt{\sum_{i=1}^{n}(a_i - b_i)^2},$$

kde a_i a b_i jsou i-té souřadnice bodů A a B.

4.3.1.2 Kosinova podobnost

Kosinova podobnost je nejpoužívanější metrikou podobnosti mezi vektorovými reprezentacemi. Kosinová podobnost udává míru podobnosti dvou vektorů, která se získá pomocí kosinu úhlu těchto vektorů a předpokládá, že oba vektory jsou normovány a mají tedy stejnou délku. Tato metoda je užitečná například pro hledání podobnosti mezi dvěma textovými dokumenty, jejichž atributy jsou frekvence slov. Hodnota podobnosti nabývá hodnot mezi 0 a 1, kde 1 udává silnou podobnost. Kosinova podobnost dvou vektorů a a b je definována jako:[15]

$$sim(\vec{a}, \vec{b}) = \frac{\vec{a} \cdot \vec{b}}{|\vec{a}| * |\vec{b}|},$$

kde operátor · značí vektorový součin.

4.3.1.3 Pearsonova podobnost

Pearsonův koeficient je více komplexní a sofistikovanější přístup k nalezení podobnosti. Je to korelace míra síly lineárního vztahu mezi dvěma proměnnými a je označován *r*. Korelační koeficient se pohybuje mezi hodnotami 1 a -1 včetně. Hodnota 0 znamená, že neexistuje žádný lineární vztah mezi dvěma proměnnými. Hodnota menší než 0 indikuje negativní asociaci, což znamená, že hodnota jedné proměnné je stoupající a druhé naopak je klesající. Hodnota větší než 0 udává, že obě hodnoty jsou stoupající a v případě r = 1 znamená, že lineární rovnice popisuje vztah mezi X a Y dokonale, všechny body leží na přímce a přímka je stoupající. Hodnotu tohoto koeficientu lze určit jako podíl kovariance sledovaných proměnných a jejich směrodatných odchylek. Kovariance je definována jako střední hodnota součinu rozdílu sledovaných proměnných od jejich středních hodnot: [28,29]

$$r = \frac{cov(X,\ Y)}{s_x s_y} = \frac{E\left[\left(X_i - E(X)\right)\left(Y_i - E(Y)\right)\right]}{s_x s_y} = \frac{1}{n-1}\sum_{i-1}^{n}\left(\frac{X_i - \bar{X}}{s_x}\right)\left(\frac{Y_i - \bar{Y}}{s_y}\right),$$

kde *n* udává počet hodnot použitých při výpočtu, X_i a Y_i jsou konkrétní hodnoty, $\bar{X}$ a $\bar{Y}$ jsou průměrné hodnoty a s_x a s_y jsou směrodatné odchylky.[29]

4.3.1.4 Tanimotova podobnost

Tanimotova metrika podobnosti je dalším metrika určující podobnost vektorů, která je definována následovně[30]:

$$s_T(x, y) = \frac{x^T y}{\|x\|^2 + \|y\|^2 - x^T y}$$

Přidáním a odečtením výrazu $x^T y$ ve jmenovateli a použitím několika algebraických manipulací dostaneme:

$$s_T(x, y) = \frac{1}{1 + \frac{(x - y)^T (x - y)}{x^T y}}$$

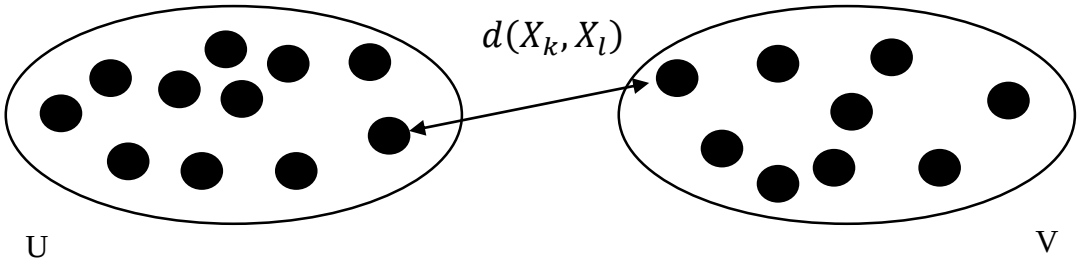
Dostáváme důkaz o tom, že Tamiotova míra podobnosti mezi vektory *x* a *y* je nepřímo úměrná kvadrátu euklidovské vzdálenosti mezi nimi dělených jejich skalárním součinem. Intuitivně řečeno, jelikož skalární součin může být považován jako míra korelace mezi *x* a *y,* pak *ST(X, y)* je nepřímo úměrná kvadrátu euklidovské vzdálenosti mezi x a y, děleno jejich korelací.[30]

4.3.2 **Shluková analýza**

Shluková analýza se zabývá takovými metodami, pomocí kterých sdružuje objekty s podobnými vlastnostmi do shluku. Zabývá se tedy analýzou podobnosti vícerozměrných objektů a jejich klasifikací do tzv. shluků, že podobnost dvou objektů, které patří do stejného shluku, je maximální, zatímco podobnost s objekty mimo tento shluk je minimální. Dále poskytuje také zjednodušený pohled na objekty a odhaluje vztahy mezi nimi. Nevýhodou je, že tato analýza je velmi citlivá na přítomnost odlehlých objektů, které se liší od ostatních a mohou tak zapříčinit zničení struktury dat a nemožnost nalezeným shlukům popisovat přesnou strukturu. Tyto objekty je nutné identifikovat a eliminovat. Podobnost objektů může být měřena různými způsoby, mezi nejpoužívanější však patří míry vzdálenosti.[31,32]

4.3.2.1 Metoda nejbližšího souseda

Tato metoda spadá do skupiny hierarchických metod tedy do metod, které jsou založeny na hierarchickém uspořádání objektů a jejich shluků. Tato metoda spojuje vždy objekty, které jsou si nejbližší. V případě shluků pak spojí vždy takové, jejichž, byť jen jediné dva objekty, jsou si nejbližší. Vzdálenost dvou shluků je tedy dána vzdáleností dvou nejbližších objektů v nich. Nevýhodou této metody je řetězový efekt, kdy se spojují shluky, pro které sice platí, že jejichž dva objekty jsou nejbližší, ale v porovnání s ostatními se nejedná o nejbližší shluky. Oproti ostatním metodám je ale tato invariantní k monotónním transformacím matice podobnosti, což znamená, že tato metoda není ovlivněna žádnou transformací dat. Další výhodou je, že metoda nejbližšího souseda není ovlivněna vazbami v datech.[31,32]



Obrázek 15: Metoda nejbližšího souseda převzato z [33]

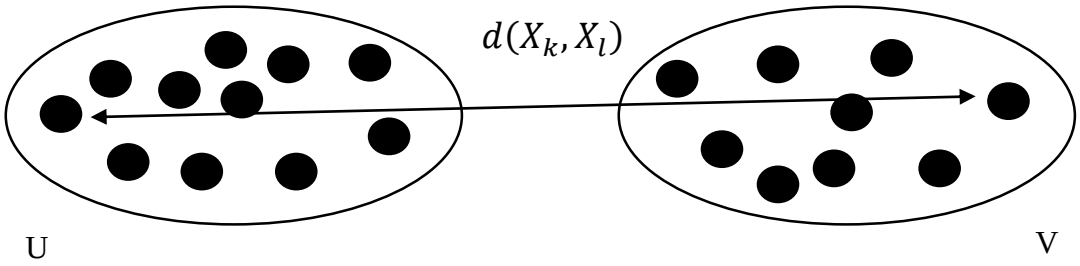
Vzdálenost mezi dvěma shluky získáme vybráním nejmenší vzdálenosti mezi všemi objekty z obou shluků. Vzdálenost shluků je dána minimem ze vzdálenosti mezi jejich objekty:[33]

$$D(U,V) = \min d(X_k, X_l),\ X_k \in U, X_l \in V,$$

kde U, V značí shluky, X_k je objekt patřící do U a X_l objekt patřící do V.

4.3.2.2 Metoda nejvzdálenějšího souseda

Tato metoda počítá vzdálenost dvou shluků jako maximum z možných shlukových vzdáleností objektů. Probíhá podobně jako předchozí metoda s rozdílem, že vzdálenost (podobnost) mezi shluky je určována vzdáleností mezi dvěma nejvzdálenějšími objekty z rozdílných shluků, proto jsou všechny objekty ve shluku klasifikovány na základě maximální vzdálenosti či minimální podobnosti vůči objektům ve druhém shluku. Nežádoucí řetězový efekt zde odpadá, naopak je tu tendence ke tvorbě kompaktních shluků, které nebývají velké.[31,34] Metoda se také nazývá metodou úplného propojení, protože všechny objekty ve shluku jsou propojeny každý s každým při maximální vzdálenosti čili minimální podobnosti.[35]



Obrázek 16: Metoda nejvzdálenějšího souseda převzato z [33]

Vzdálenost mezi dvěma shluky získáme naopak vybráním největší vzdálenosti mezi všemi objekty z obou shluků. Vzdálenost shluků je dána minimem ze vzdálenosti mezi jejich objekty:[33]

$$D(U,V) = \max d(X_k, X_l),\ X_k \in U, X_l \in V,$$

kde U, V značí shluky, X_k je objekt patřící do U a X_l objekt patřící do V.

4.3.2.3 Sekvenční práh

Tato metoda začíná volbou jednoho zárodku shluku a zahrnuje všechny objekty uvnitř předem specifikované vzdálenosti zvané práh. Když jsou všechny objekty uvnitř této vzdálenosti zahrnuty do shluku, je vybrán zárodek druhého shluku a všechny objekty uvnitř přespecifikované vzdálenosti jsou zahrnuty do shluku. Pak je vybrán třetí zárodek shluku a proces se opakuje. Když je jednou objekt shlukován se zárodkem, není již dále započítáván do některého jiného shluku.[35]

4.3.2.4 Metoda K-means

Je metoda používající jako vzorové body těžiště shluků. Vyžaduje spojité proměnné a především bez odlehlých hodnot. Diskrétní data mohou být také analyzována touto metodou, ale mohou způsobit problémy. Jedná se nehierarchický algoritmus, který má za úkol třídit data do *k* shluků na základě jejich vlastností. Počet shluků *k* je definován na začátku algoritmu a je menší než počet objektů. Tento algoritmus pracuje tak, že přiřadí každý bod do shluku, jemuž středu je nejblíže. Středy shluků se při každém běhu algoritmu znovu spočítají jako aritmetické průměry všech bodů shluku. Cílem je dosáhnout co nejmenších rozdílů uvnitř shluků.[36]

Jednotlivé kroky algoritmu jsou:[31]

1. Vybrání počet shluků *k*, vygenerování, nebo zadání těžiště shluků.
2. Vybrání shluku pro každý objekt, jehož těžiště je nejblíže, a pokud se vybraný shluk nerovná původnímu shluku, přemístění objektu do něj a následné přepočítání těžiště.
3. Pokud nedošlo ke změně žádného shluku, ukončení algoritmu, jinak opakování kroku 2.

Výhodou algoritmu je jednoduchost a rychlost a také to, že dá se použít na velké množství dat. V konečném počtu kroků konverguje k nějakému řešení, těch ovšem může být více. Nevýhodou naopak je ovlivnění výsledku výběrem počátečních shluků.[31]

4.3.2.5 Metoda Fuzzy C-means

V této metodě se pro každý bod množiny počítá, s jakou pravděpodobností patří do jakého shluku. Z toho vyplívá, že krajní body shluku mají menší stupeň příslušnosti než body, které se nacházejí v centru tohoto shluku. Vzhledem k tomu, že každý bod má stupeň příslušnosti, můžeme velice dobře a přesně popsat rozložení objektů ve shlucích, díky čemuž může objekt patřit do více shluků zároveň. Touto metodou se dají i lépe identifikovat ty objekty, které se nedají přiřadit do žádného shluku. Vytvoření shluků na základě pravděpodobnosti příslušnosti objektů je také jednodušší. V případě, že každý objekt má pravděpodobnost příslušnosti k nějakému shluku rovnou jedné a k ostatním nulovou, pak je výsledkem pevné shlukování. Naopak jednotlivé shluky jsou neurčitelné, pokud se stupeň příslušnosti každého objektu k libovolnému shluku rovná převrácené hodnotě počtu shluků. Součet koeficientů příslušnosti jednotlivých objektů ke všem shlukům musí být roven 1.[31]

Algoritmus se skládá s následujících kroků: [31]

1. Vybrání počtu shluků.
2. Určení prahu.
3. Náhodné přiřazení koeficientu příslušnosti všem bodům.
4. Opakování následujících kroků změna koeficientu příslušnosti není menší, než daný práh citlivosti:
    a. Vypočítání středu každého shluku.

b. Pro každý bod spočtení pravděpodobnost příslušnosti k určitému shluku.

Algoritmus v průběhu procesu minimalizuje rozdíly uvnitř shluku. Výsledek závisí na zvolených mírách příslušnosti.[31]

5 Mahout

Mahout je open source knihovna pro strojové učení od společnosti Apache. Tato knihovna je hlavně využívána pro oblast klasifikace, shlukování a v neposlední řadě také využívána pro doporučovací systémy pro svou část kolaborativního filtrování.

Jméno Mahout pochází z hindského slova označující jezdce na slonovi. Název se odvíjí od faktu, že Mahout je brán jako nadstavba Hadoopu, který má ve znaku žlutého slona. Mahout byl vytvořen v roce 2008 jako pod projekt k projektu Apache Lucene, který poskytuje známý open source vyhledávací engine téhož jména. Lucene poskytuje pokročilé implementace vyhledávání, dolování textu a technik pro získávání informaci. Ve světě počítačové vědy jsou tyto koncepty spojeny s technikami strojového učení, jako je shlukování a klasifikace. V důsledku toho některá práce vývojářů Lucene klesla do oblasti tohoto strojového učení a vznikl samostatný podprojekt. V roce 2010 se Mahout stal jedním z projektů nejvyšší úrovně a dostal logo jezdce na slonovi.[37]

Co se týče vlastností, tak má jednu nepopiratelnou výhodu, a to škálovatelnost. Mahout si klade za cíl být nástrojem strojového učení v případech, kdy kolekce dat, která má být zpracována, je velmi velká, tak, že jeden stroj již nestačí. Ve své současné inkarnaci jsou tyto škálovatelné realizace strojového učení v Mahutu napsány v Javě a některé části jsou postaveny na distribuovaném výpočetním projektu Apache Hadoop.[37] Apache Hadoop je softwarový framework, který poskytuje možnost distribuovaného zpracování velkých dat pomocí skupiny počítačů, která může čítat až tisíce strojů, s využitím jednoduchých datových modelů.[42]

Jak již bylo zmíněno Mahout je Java knihovna. Což znamená, že neposkytuje uživatelské rozhraní, server ani instalátor. Je to framework nástrojů určený pro použití vývojáři, kteří si jej mohou přizpůsobit podle svého.[37]

6 Návrh modulu implementující doporučovací systém

6.1 Použité nástroje a technologie

6.1.1 Java EE

Java Enterprise Edition (neboli Java EE v minulosti označovaná také jako Java 2 Enterprise Edition) je součást platformy Java určená pro vývoj a provoz podnikových aplikací a informačních systémů. Rozšiřuje platformu Java Standard Edition (JSE) o podporu tvorby webových aplikací, webových služeb a distribuovaných vícevrstvých aplikací. Jejím cílem je poskytnout vývojáři infrastrukturu usnadňující jejich vývoj. Základem těchto Java EE aplikací je vícevrstvá architektura, kde jsou komponenty aplikace rozděleny podle svého typu do oblastí zvaných vrstvy, jež často bývají instalovány na různých počítačích. Java EE používá tzv. čtyřvrstvou architekturu:[38]

- Klientská vrstva - představuje internetový prohlížeč, nebo jinou klientskou aplikaci, jejímž úkolem jakožto klienta je zasílat požadavky spodní vrstvě, přijímat od ní odpovědi a prezentovat je.

- Webová vrstva - je tvořena především tzv. Java Servlety, tedy komponentami, které zpracovávají klientské požadavky a vrací odpověď, která je v dalším kroku zaslána zpět do klientské vrstvy.

- Aplikační business vrstva - zde je by měla být umístěna všechna logika a funkcionalita, konkrétně se zde používají EJB komponenty. Tyto komponenty přijímají požadavky od klientské a webové vrstvy a na jejich základě pracují se zdroji z vrstvy pod nimi. Následně zasílají odpověď zpět klientské nebo webové vrstvě.

- Enterprise Information System vrstva- tato vrstva představuje veškeré externí systémy, jejichž funkcionalitu, nebo data Enterprise aplikace využívá. Může se jednat například o ERP systém, databázový systém atp. Komunikace s EIS je především zajišťována přes aplikační vrstvu.

Platforma Java EE definuje množství služeb nutných pro vlastnosti jako je škálovatelnost, robustnost, bezpečnost a hlavně udržitelnost. Služba pro bezpečnost se stará o to, aby uživatelé v systému byli těmi, za které se vydávají, a aby měli platný přístup pouze tam, kam mají povoleno. Další službou je přístup do databáze, odkud a kam můžou aplikace získávat a ukládat svá data. Právě k této funkci je nezbytná služba transakcí napomáhající k udržení konzistence dat a k jejich aktualizaci v logickém pořadí, čímž zaručí správný a zapsaný výsledek. Tyto služby jsou používané stejným způsobem, jako jsou například ve standardní edici jazyka Java užívány kolekce.[38]

6.1.2 Glassfish

GlassFish je název pro developerský open source projekt pro budování aplikačního serveru pro Java EE 5. Je založen na zdrojovém kódu Sun Java System Application Server PE 9, který byl darován Sun Microsystems, a TopLink persistentním kódu darovaném společností Oracle. Tento projekt poskytuje strukturovaný proces pro vývoj vysoce kvalitního aplikačního serveru, který umožňuje novým funkcím být rychleji k dispozici. Je to reakce na Java vývojáře, kteří chtějí přístup ke zdrojovému kódu a možnosti přispívat k rozvoji příští generace aplikačního serveru. Tento projekt je určen na podporu komunikace mezi inženýry Sun a Oracle a komunitou, a umožní všem vývojářům účast na procesu vývoje tohoto aplikačního serveru.[39]

Každodenní buildy a zdrojový kód aplikačního serveru jsou k dispozici na stránce http://glassfish.dev.java.net. Stejně jako u jiných vývojářských komunitních stránek je zde také možné najít seznam e-mailů, diskuzní fóra, novinky, licenční informace a rozsáhlé zdroje nápovědy. Rozvoj komunity okolo GalssFish je teprve v počátcích. Dnes již existuje více než 100 zaregistrovaných přispěvatelů a počet inženýrů Sun a Oracle pracujících na tomto produktu je dnes srovnatelná s velikostí jiných komunit pro vývoj aplikačních serverů. S vydáním projektu GlassFish pro komunitu se očekává výrazné zvýšení počtu vývojářů pracujících na kódu.[39]

6.1.3 MySQL

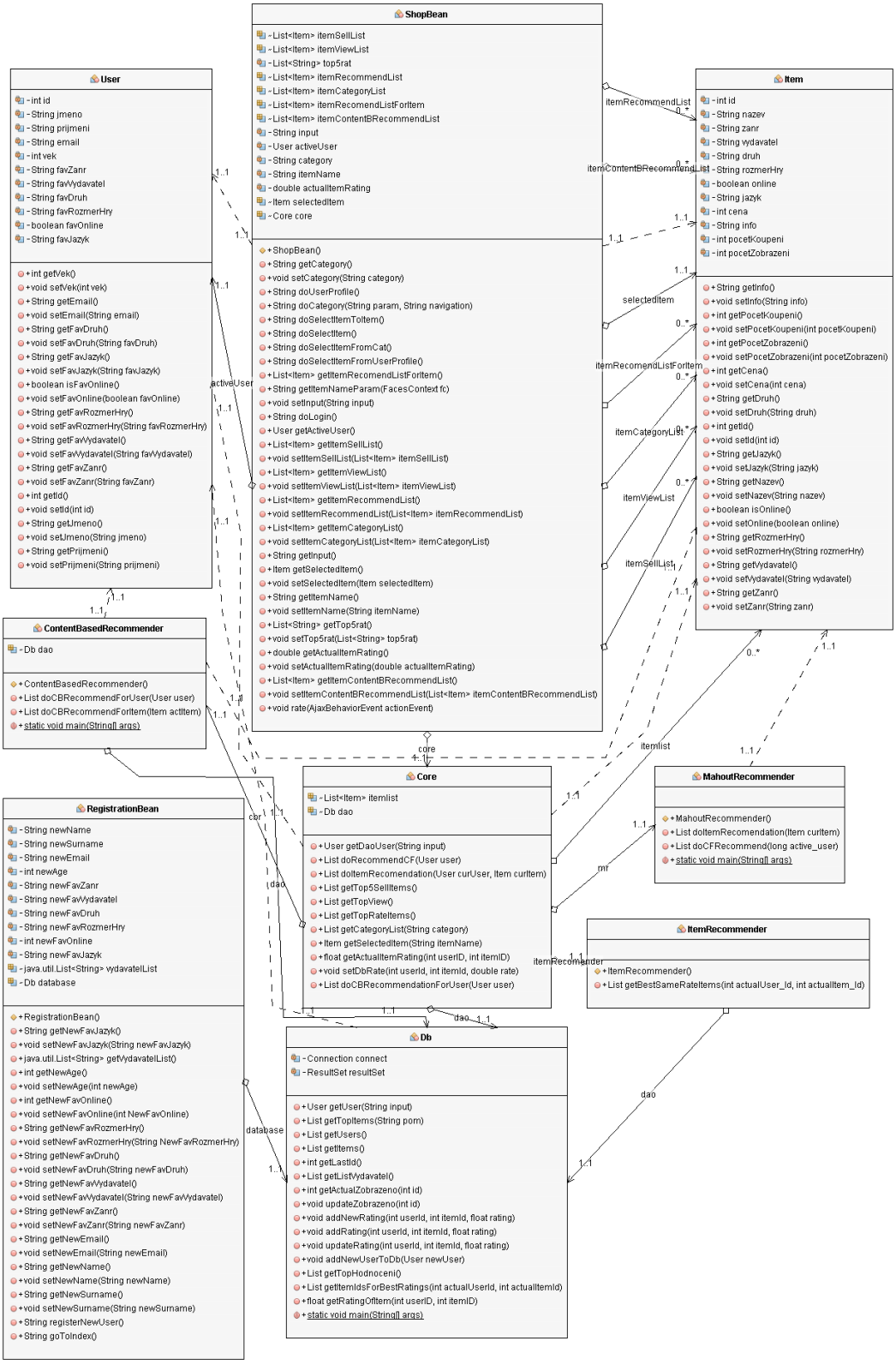
MySQL je jedním z nejpopulárnějších open source (každý si jej může bezplatně stáhnout a používat) databázových systémů, který je vyvíjen, distribuován a podporován společností Oracle.

Tento databázový systém poskytuje možnost uložení, přístupu a zpracování dat uložených v počítačové databázi. Data jsou ukládána v oddělených tabulkách, kde tento přístup organizovaných databázových struktur v porovnání s uložením dat dohromady v jednom velkém skladu zajišťuje vyšší rychlost.

MySQL Server lze spustit pohodlně na desktopu nebo notebooku společně s dalšími jinými aplikacemi, webových servery atd. Při situaci, kdy je MySQL vyčleněn celý, je možné upravit nastavení tak, aby byly využity všechny paměti, výkon CPU a I/O kapacity, která je dispozici. MySQL lze také rozšířit až na clustery strojů, které jsou zesíťované dohromady.

Součástí je také strukturovaný dotazovací jazyk SQL, který je nejběžněji používaný standardizovaný jazyk k přístupu k databázím. V závislosti na programovacím prostředí, může být SQL dotaz zadán přímo vložením SQL příkazu do kódu napsaného v jiném jazyce, nebo použitím rozhraní, které skrývá syntaxi SQL.

6.2 Diagram tříd a základní popis struktury



Obrázek 17: Diagram tříd navržené aplikace

Navržený modul je součástí webové MVC Java aplikace simulující internetový obchod, která s ním pracuje a poskytuje zobrazení výstupů tohoto modulu. Jak je vidět na Obrázku 17, aplikace obsahuje 9 tříd, kde každá má svou určitou funkci. User.java a Item.java zastupují objekty daného typu. Mezi hlavní patří třídy ShopBean.java, Core.java a Db.java. ShopBean.java je kontroler k zobrazovaným *xhtml souborům a dále se využívá jako prostředník mezi aplikací a zobrazením. Core.java je třída, která zajišťuje zpracování dat a doporučení. Zde jsou všechny důležité metody volané v aplikaci. Poslední základní třída je Db.java, která, jak je dáno z názvu, se stará o práci s databází jako je čtení, zápis nebo aktualizace. Dále jsou v aplikaci třídy, které se starají o generování doporučení, jako je ContentBasedRecommender.java, MahoutRecommender.java a ItemRecommender.java. Tyto třídy jsou popsány detailně dále. Poslední třídou je třída RegisterBean.java, která je kontrolerem pro registraci nového uživatele a zajišťuje vytvoření nového uživatele v systému.

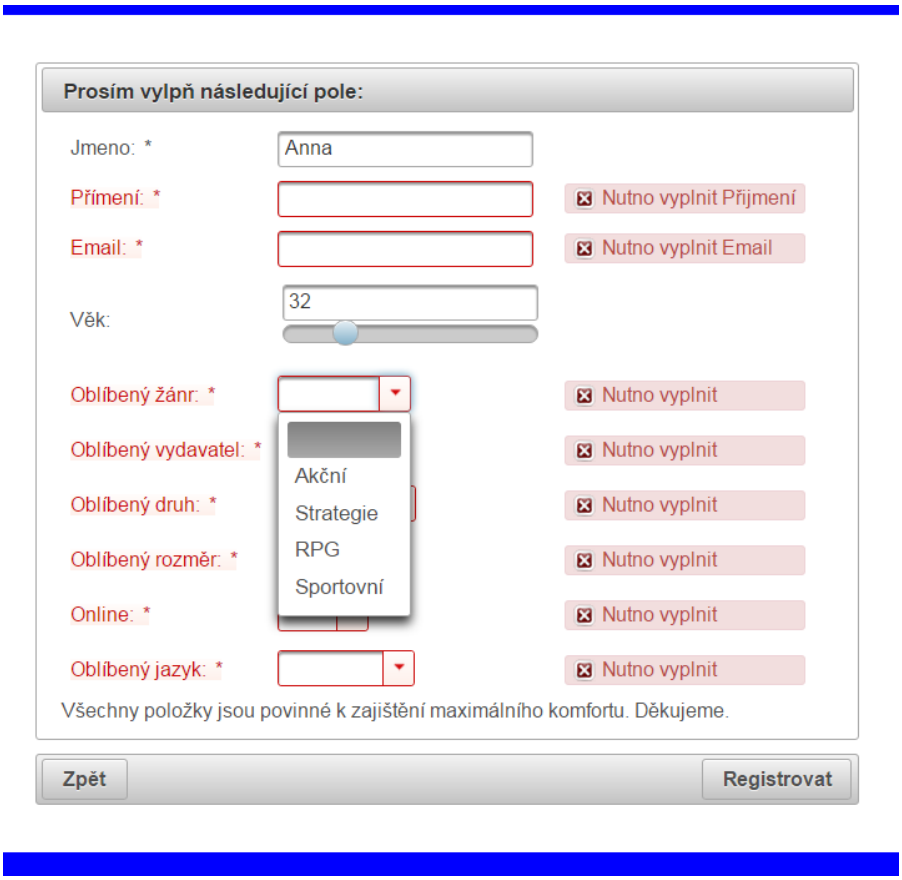
6.3 Funkce naprogramovaného projektu

Webová aplikace je napsaná v jazyce Java, kde byla konkrétně využita technologie JSF. Databáze, kterou zajišťuje již zmíněný MySQL server, obsahuje 3 hlavní tabulky: *user*, kde jsou uloženy uživatelské profily uživatelů, *item* obsahující informace o položkách v obchodě, a tabulku hodnocení položek uživateli *preferences*. Hlavní funkce tykající se doporučení jsou popsány dále.

6.3.1 Vytvoření uživatelského profilu

Jako v každém systému e-commerce i zde je potřeba přihlášení uživatele do systému. Pro registraci uživatel na úvodní stránce *index.xhtml* zvolí tlačítko registrovat, které ho přesměruje na stránku registrace *registerPage.xhtml*, jejímž kontrolerem je *RegistrationBean.java*. Tato stránka obsahuje povinná pole, která uživatel musí vyplnit. V první části uživatel zadává své osobní informace jako je jméno,

příjmení, email a věk. Tím sám sebe odlišuje od ostatních uživatelů. V druhé části zadává své záliby a preference, které později slouží k doporučení položek.



Obrázek 18: Registrace uživatele

Po vyplnění všech polí a kliknutí na tlačítko registrovat je zavolána metoda *registerNewUser()* z kontroleru, která vytvoří objekt uživatele se zadanými vlastnostmi a zavolá metodu *addNewUserToDb()* instanci třídy Db.java ukládající uživatelský profil do databáze. Uživatel je následné přesměrován zpět na *index.xhtml*, kde je mu zobrazena informace o úspěšné registraci. Následně se již může nový uživatel přihlásit. Tento uživatelský profil je potřeba ke generování doporučení pro použitý postup doporučení při doporučení založeném na obsahu.

6.3.2 Doporučení nejprodávanějších položek

Prvním z neosobních doporučení, které analyzuje data bez ohledu na uživatelovo chování je doporučení nejprodávanějších 5 položek. Tyto položky můžou uživatele zlákat, aby si kopil takové zboží, které je zrovna populární a v módě. Nejprodávanější položky se uživatelům zobrazují v levém panelu pod panelem menu na každé stránce, kterou navštíví.

Obrázek 19: Doporučení nejprodávanějších položek

Při přihlášení je v kontroleru ShopBean.java je zavolána metoda *getTop5SellItems()* ze třídy Core.java. Tato metoda vrací seznam o 5 objektech typu *Item*, který je generován pomocí metody *getTopItems("topProdavane"*) v Db.java s parametrem, který specifikuje vyhledávané top položky. Vyhledání položek je vykonáno SQL dotazem na tabulku *Item*, která, jak již bylo zmíněno, obsahuje informace o položkách včetně sloupce „koupeno“, který udává počet koupení dané položky. Dotaz je tedy následující: *SELECT * from item ORDER BY Koupeno DESC LIMIT 5*. Po vyhledání se vytvoří požadovaný seznam objektů typu *Item* podle vrácené odpovědi z databáze.

Id	Nazev	Zanr	Vydavatel	Druh	Rozmer	Online	Jazyk	Cena	Info	Koupeno
0	Torchlight2	RPG	Runic games	Digitalni	3D	1	Cestina	900	Multiplayer: hrajte se svými přáteli zcela zdarma ...	38
1	Borderlands	Akcni	2K Games	Krab	3D	1	Cestina	1200	Bordelands 2 tentokrát představí zlepšený zbrojní ...	58
2	Far Cry3	Akcni	Ubisoft	Digitalni	3D	0	Cestina	1600	Ve hře Far Cry 3 si obujete boty Jasona Brodyho, m...	39
3	Settlers3	Strategie	Ubisoft	Digitalni	3D	1	Anglictina	600	The Settlers 7 se vrací v plné síle. Nový díl hry ...	20
4	Anno2070	Strategie	Ubisoft	Krab	3D	1	Cestina	700	Dalsi dil z rady Anno se presouvá do blizke budouc...	28
5	NBA 2013	Sportovni	2K Games	Digitalni	3D	1	Anglictina	500	S více než 5 miliony prodaných kopií po celém svět...	19

Obrázek 20: Databázová tabulka

6.3.3 Doporučení nejvíce zobrazovaných položek

Dalším doporučením, které je založeno na chování všech zákazníků internetového obchodu je Doporučení nejvíce navštěvovaných položek, nebo-li nejvíce zobrazovaných položek.

Obrázek 21: Doporučení nejvíce zobrazovaných položek

Systém zachytává přístupy na stránky jednotlivých položek, jejíž počet je aktualizován kdykoliv jakýkoliv uživatel zobrazí stránku itemPage.xhtml. Aktualizace je provedena v metodě *getSelectedItem(String itemName)*, která vrací aktuálně prohlíženou položku pro itemPage.xhtml, konkrétně zavoláním metody *updateZobrazeno(int id)* ze třídy *Db.java* s parametrem ID aktuální položky, která aktualizuje buňku Zobrazeno v tabulce *Item*. Metoda je následující:

```
public void updateZobrazeno(int id) throws Exception
    {
        String pz = Integer.toString(getActualZobrazeno(id)+1);
        connect =
DriverManager.getConnection("jdbc:mysql://localhost:3306/eshop","root","");
        String ids = Integer.toString(id);
        PreparedStatement statement =
connect.prepareStatement("UPDATE item SET Zobrazeno = ? where id = ?");
        statement.setString(1, pz);
        statement.setString(2, ids);
        statement.executeUpdate();

        connect.close();
        resultSet.close();
        statement.close();

    }
```

Samotné doporučení se nachází ve stejném panelu jako předchozí doporučení nejprodávanějších položek a je generováno stejnými metodami s rozdílným parametrem u metody *getTopItems("topZobrazeno")* v Db.java, kde parametr nyní udává vyhledávání Top zobrazených položek. Dotaz na databázi je v tomto případě následující: *SELECT * from item ORDER BY Zobrazeno DESC LIMIT 5*.

Id	Nazev	Zanr	Vydavatel	Druh	Rozmer	Online	Jazyk	Cena	Info	Kc	Zobrazeno
0	Torchlight2	RPG	Runic games	Digitalni	3D	1	Cestina	900	Multiplayer: hrajte se svými přáteli zcela zdarma ...	38	101
1	Borderlands	Akcni	2K Games	Krab	3D	1	Cestina	1200	Bordelands 2 tentokrát představí zlepšený zbrojní ...	58	148
2	Far Cry3	Akcni	Ubisoft	Digitalni	3D	0	Cestina	1600	Ve hře Far Cry 3 si obujete boty Jasona Brodyho, m...	39	82
3	Settlers3	Strategie	Ubisoft	Digitalni	3D	1	Anglictina	600	The Settlers 7 se vrací v plné síle. Nový díl hry ...	20	47
4	Anno2070	Strategie	Ubisoft	Krab	3D	1	Cestina	700	Dalsi dil z rady Anno se presouvá do blizke budouc...	28	153
5	NBA 2013	Sportovni	2K Games	Digitalni	3D	1	Anglictina	500	S více než 5 miliony prodaných kopií po celém svět...	19	101

Obrázek 22: Databázová tabulka

6.3.4 Doporučení nejlépe hodnocených položek

Třetím doporučením tohoto druhu je doporučení nejlépe hodnocených položek. Toto doporučení dává uživatelům možnost koupit si takové zboží, které se nejvíce líbilo ostatním uživatelům obchodu. Toto doporučení je společně s předchozími také umístěno v levém panelu a zobrazováno na každé stránce obchodu.

Obrázek 23: Doporučení nejlépe hodnocených položek

Po přihlášení uživatele je zavolána z Core.java metoda *getTopRateItems()*, která vrací seznam 5 nejlépe hodnocených položek pomocí metody *getTopHodnoceni()* ze třídy Db.java. Metoda *getTopHodnoceni()* pomocí SQL dotazu vrátí seznam s názvy nejlepších položek. SQL dotaz na tabulku *preferences* je následující:

SELECT item.Nazev, sum(`preference`)/ COUNT(*) as avgpref FROM preferences LEFT JOIN item on preferences.item_id = item.Id GROUP BY item_id ORDER BY avgpref DESC LIMIT 5 ,

Kde pro každou položku je zjištěno kolikrát byla hodnocena, sečteno hodnocení a vypočten aritmetický průměr, který udává její populárnost. Ze všech položek je pak vybíráno 5 s nejvyšším průměrem.

6.3.5 Doporučení položek na základě hodnocení

Toto doporučení je jedním z nejdůležitějších a nejzásadnějších v aplikaci. Hlavním jeho úkolem je predikovat položky, které by aktuální uživatel mohl preferovat a potencionálně koupit na základě jeho chování v minulosti, co se týče hodnocení položek, se kterými se dostal do kontaktu. Jak je tedy zřejmé, jedná se o user-based kolaborativní doporučení (Collaborative filtering).

6.3.5.1 Řešení hodnocení položek

Hodnocení je umístěno na stránce položky tedy v *itemPage.xhtml*. Pokud je položka aktuálním uživatelem neohodnocena, je zobrazen text „nehodnoceno“ a uživatel ji může ohodnotit. Položka může být hodnocena i v případě, že již v minulosti uživatel zadal svou zpětnou vazbu a je nalezeno hodnocení. V tomto případě se toto staré hodnocení nahradí novým a aktuálním k zajištění aktuálnosti uživatelova chování. Škála možného hodnocení je po 0.5 od 1 (špatný) až 5 (nejlepší) zobrazená v podobě interaktivních hvězd. Pro generování doporučení byl použit výše zmíněný Mahout.

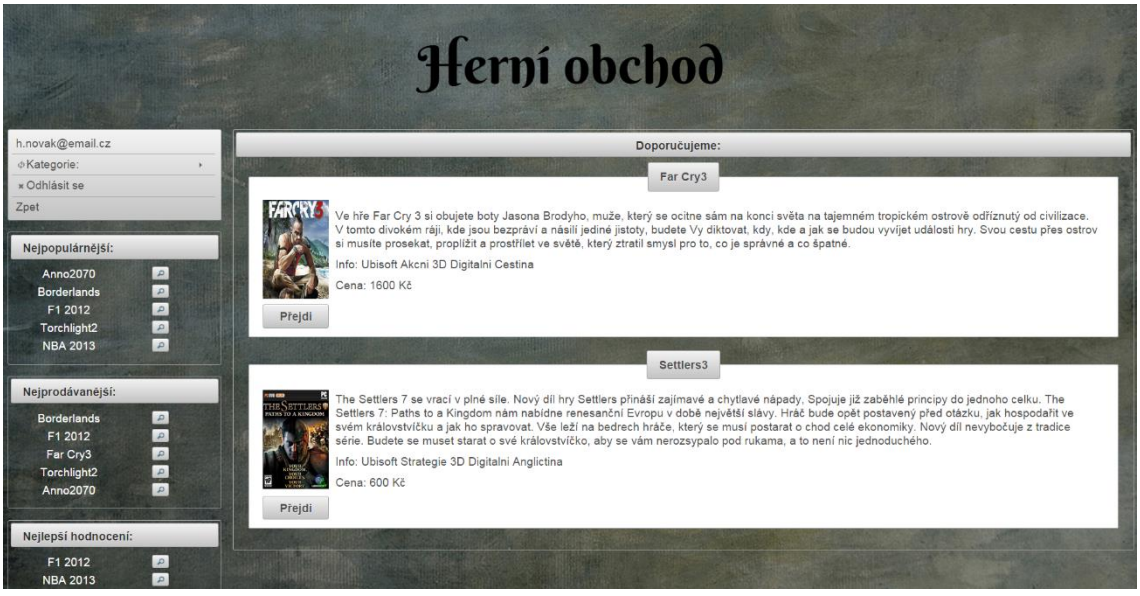
Obrázek 24: Hodnocení položky

V aplikaci je zobrazení hodnocení řešeno tak, že před přesměrováním uživatele na stránku položky se zavolá metoda z *Core.java getActualItemRating(int userID,int itemID)* s parametry ID aktuálního uživatele a ID prohlížené položky, která pomocí metody se stejnými parametry *getRatingOfItem(userID, itemID)* z *Db.java* vrací rating typu float. Metoda *getRatingOfItem(userID, itemID)* vyhledá pomocí SQL dotazu na tabulku preferences *SELECT `preference` FROM `preferences` WHERE `user_id` = userID and `item_id` = itemID* hodnocení aktuální položky aktuálním uživatelem. Pokud je hodnocení nalezeno, je zobrazeno. V případě nenalezení je zobrazen již zmíněný text „nehodnoceno“.

V situaci, kdy chce uživatel položku ohodnotit, klikne na určitou hvězdu v pořadí, tím se provede Ajaxové volání metody *rate(AjaxBehaviorEvent actionEvent)* z třídy *ShopBean.java*, kde se skrz Core.java zavolá metoda *addNewRating(int userId,int itemId, float rating)* z Db.java. V této poslední metodě se nejprve SQL dotazem zjistí, zda hodnocení již existuje, pokud ano je existující hodnocení aktualizováno dotazem *UPDATE preferences SET preference = reting WHERE user_id = userId and item_id = itemId,* v opačném případě je vytvořen v tabulce preferences nový záznam dotazem *INSERT INTO preferences(user_id,item_id,preference) VALUES (userId, itemId, rating)*. Poté je zobrazení na stránce ihned aktualizováno.

6.3.5.2 Generování doporučení

Generování dvou potencionálních položek pomocí kolaborativního doporučení na základě hodnocení je prováděno při přihlášení uživatele třídou *MahoutRecommender.java* a je k nalezení na hlavní stránce obchodu.



Obrázek 25: Kolaborativní doporučení položek

Po přihlášení je přes Core.java zavolána metoda *doCFRecommend(long active_user)* v již zmíněné třídě *MahoutRecommender.java* s parametrem Id aktivního uživatele, která vrací 2 doporučené položky. V této metodě je implementován Mahout.

```
public List doCFRecommend(long active_user)
{

  List<RecommendedItem> recommendations = new
ArrayList<RecommendedItem>();

try {
      MysqlDataSource dataSource = new MysqlDataSource();
      dataSource.setServerName("localhost");
      dataSource.setUser("root");
      dataSource.setPassword("");
      dataSource.setDatabaseName("eshop");

      JDBCDataModel model = new MySQLJDBCDataModel(
                  dataSource, "preferences", "user_id",
                  "item_id", "preference", "");

      UserSimilarity similarity = new PearsonCorrelationSimilarity(
                                    model);


      UserNeighborhood neighborhood = new NearestNUserNeighborhood(
                                    2, similarity, model);

      Recommender recommender = new GenericUserBasedRecommender(
                              model, neighborhood, similarity);


      recommendations = recommender.recommend(active_user, 2);

      System.out.println("Generuji CF doporučení pro Uzivatele");
      System.out.println("-----------------------------------");

        }
 catch (TasteException ex) {

Logger.getLogger(MahoutRecommender.class.getName()).log(Level.SEVERE,
null, ex);
      }
        return recommendations;
}
```

Prvním krokem je vytvoření modelu. Datový model s názvem model bude používán k výpočtům. JDBCDataModel je vytvořen načtením tabulky hodnocení

preferences z databáze příkazem, kde jako parametry jsou definovány jména sloupců v databázi:

```
JDBCDataModel model = new MySQLJDBCDataModel(
                    dataSource, "preferences", "user_id",
                          "item_id", "preference", "");
```

K tomu, aby se model bezchybně vytvořil, musí v databázi tabulka *preferences* splňovat následující podmínky:

- Musí obsahovat sloupec Id uživatele, který reprezentuje toho, kdo položku hodnotil, dále sloupec Id hodnocené položky a sloupec hodnocení. Volitelně může tabulka obsahovat ještě čtvrtý sloupec s časovým údajem hodnocení.
- Id uživatele a Id položky musí být nenulové a indexované.
- Primární klíč se musí skládat z kombinace Id uživatele a Id položky.
- Datové typy musí korespondovat s datovými typy jazyku Java.

Pro vytvoření tabulky s těmito parametry byl použit následující SQL dotaz:

```
CREATE TABLE taste_preferences (
      user_id BIGINT NOT NULL,
      item_id BIGINT NOT NULL,
      preference FLOAT NOT NULL,
PRIMARY KEY (user_id, item_id),
INDEX (user_id),
INDEX (item_id)
)
```

Dalším krokem je zvolení způsobu výpočtu podobností mezi uživateli. K výběru je k dispozici několik metod, kde každá má své výhody i nevýhody. Pro navrženou aplikaci byl zvolen přístup Pearsonovi podobnosti. Nastavení je v kódu následující:

```
UserSimilarity similarity = new PearsonCorrelationSimilarity(
model);,
```

kde parametrem jsou data, ze kterých bude podobnost vypočtena, v případě navržené aplikace je parametrem vytvořený model.

Předposledním krokem je nastavení hledání zpracovávaných sousedů. Mahout zahrnuje 2 třídy implementující rozdílnou metodu hledání a to metodu Nejbližších N sousedů a metodu Thresholdu. Z těchto dvou byla zvolena první varianta z důvodu malé datové základny aplikace. Kód je následující:

```
UserNeighborhood neighborhood = new NearestNUserNeighborhood(
      2, similarity, model);,
```

kde první parametr udává počet sousedů ze kterých je vypočtena predikce, druhý udává metodu použitou pro nalezení koeficientu podobnosti a posledním parametrem je datový model.

Posledním krokem je inicializace a nastavení Recommenderu, který zajistí doporučení. Nastavení a inicializace se zajistí následujícím příkazem:

```
Recommender recommender = new GenericUserBasedRecommender(
model, neighborhood, similarity);
```

Nejprve se specifikuje, o jaké doporučení se jedná, v případě navrhnuté aplikace se jedna o user-based doporučení a to se také nastaví příslušnou třídou. Nastavení má 3 parametry. První parametr určuje používaný datový model, druhý použitou metodu hledání sousedu a třetí způsob vypočtení podobnostních koeficientů.

Pro vygenerování doporučení pro příslušného uživatele se v programu zaručí provedením tohoto příkazu, který uloží výsledek do seznamu *recommendations*:

```
recommendations = recommender.recommend(active_user, 2);,
```

kde prvním parametrem je Id uživatele, pro kterého je hledané doporučení, a druhým parametrem je počet doporučených položek.

Seznam *recommendations* je po provedení výpočtu doporučení následně vrácen metodou do *Core.java*, která výsledek pošle přes kontroler až do *view*. Více o vypočtení predikce pro uživatele potenciálně atraktivních položek je možné nalézt v kapitole Kolaborativního doporučení.

6.3.5.3 Řešení problému studeného startu

Jak již bylo zmíněno, navržené doporučení položek na základě hodnocení patří do kategorie kolaborativního doporučení. Použitý kolaborativní přístup má i své nevýhody a mohou nastat situace, kdy není schopen vygenerovat žádná doporučení. Například když se do systému přihlásí nový uživatel bez předchozích interakcí. Tento uživatel nemá v databázi žádná hodnocení, podle kterých by mohla být vypočítána podobnost s ostatními uživateli a logicky není ani možné najít potencionální položky, které by tohoto uživatele mohli zajímat. Tento problém se nazývá Problém studeného startu.

Pro řešení tohoto problému bylo navrženo nezávislé doporučení netrpící problémem studeného startu, které buď přebírá funkci hlavního doporučovacího systému v případech, kdy kolaborativní přístup není schopen generování doporučení, nebo doplňuje neúplný výsledek (méně než 2 doporučení) jednou položkou jím vygenerovanou. Byl tedy vytvořen přepínací hybridní doporučovací systém, který kombinuje kolaborativní doporučení a doporučení založené na obsahu.

V aplikaci konkrétně v *Core.java* v metodě *doRecommendCF(User user)* po vrácení seznamu dvou doporučených položek ze třídy *MahoutRecommender.java* je tento seznam otestován zda je prázdný a zda obsahuje daný počet doporučených položek. Pokud není jedna z těchto podmínek splněna je zavolána třída

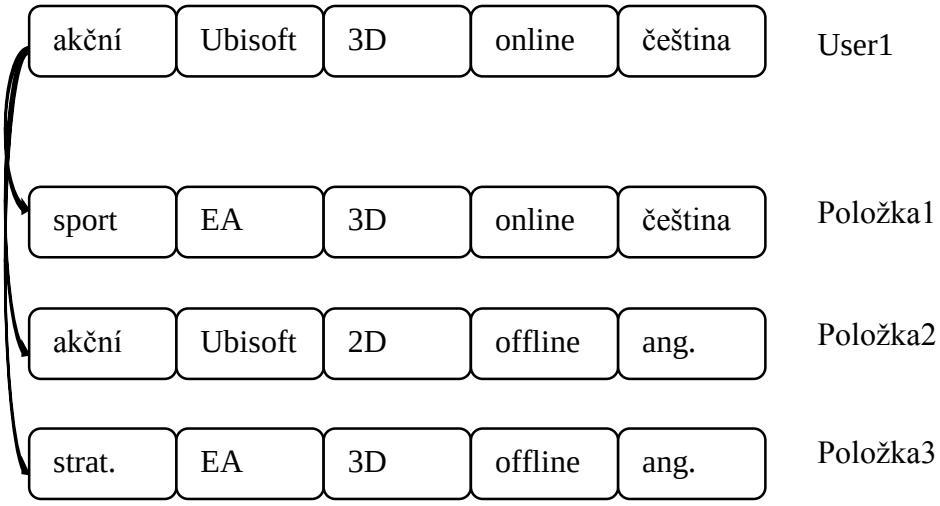
ContentBasedrecommender.java, která vygeneruje doporučení na základě doporučovacího přístupu založeného na obsahu (content based), jak ukazuje následující část kódu:

```
recommendationsCF=mr.doCFRecommend(user.getId());

if (recommendatiosCF.isEmpty()) {
        recommendList = cbr.doCBRecommendForUser(user);

}
else {
      if (recommendationsCF.size() == 1) {
      recommendList.add(itemlist.get(
                  (int)recommendationsCF.get(0).getItemID())
                  );
      pomRecommendList = cbr.doCBRecommendForUser(user);


            if(!pomRecommendList.get(0).getNazev().equals(itemlist.get(
            (int)recommendationsCF.get(0).getItemID()).getNazev()))
            {
                recommendList.add(pomRecommendList.get(0));
            }
            else
            {
                recommendList.add(pomRecommendList.get(1));
            }

        } else {
            for (RecommendedItem recommendation : recommendationsCF) {
                  recommendList.add(itemlist.get(
                                    (int)recommendation.getItemID())
                              );
                }
            }
    }
return recommendList;
```

Samotné generování doporučení je prováděno tedy ve třídě *MahoutRecommender.java* a to konkrétně v metodě *doCBRecommendForUser(User user)* s parametrem objektu aktuálního uživatele, pro kterého je generováno doporučení. V objektu aktuálního uživatele je uložen jeho uživatelský profil, který byl vytvořen při registraci, jak je popsáno v kapitole 6.3.1.

Content based doporučovací systémy fungují na principu srovnávání vlastností položky s preferovanými vlastnostmi uživatele, které jsou uloženy v jeho uživatelském

profilu. Máme například nového uživatele User1, který preferuje akční 3D online hry s českým dabingem od společnosti Ubisoft v digitálním provedení a 3 položky (hry).



Obrázek 26: Příklad generování doporučení

Nejprve se pro každou položku spočítá skóre, které udává míru podobnosti s uživatelovým profilem. Skóre (začínající pro každou položku na hodnotě 0) je vypočteno porovnáním jednotlivých atributů (vlastností) položky a uživatelského profilu a v případě, že se jednotlivé atributy rovnají, připočte se k hodnotě skóre hodnota 1. V případě příkladu jsou tedy skóre položek následující:

Skóre(Položka1)= 0 + 0 + 1 + 1 + 1 = 2
Skóre(Položka2)= 1 + 1 + 0 + 0 + 0 = 3
Skóre(Položka3)= 0 + 0 + 1 + 0 + 0 = 1

Uživateli tedy bude doporučena Položka2, která se nejvíce podobá jeho profilu.

V reálném světě ale platí, že všechny vlastnosti položky nemají pro uživatele stejnou váhu. Některé jsou pro uživatele důležitější a jiné zase méně. Uživatel raději oželí český dabing, pokud hra splňuje jeho preferovaný žánr a koupí si ji. Proto v navrženém doporučení je aplikována váha u jednotlivých atributů připočtením jiné hodnoty ke skóre, než je hodnota 1. Kód celé doporučovací metody je následující:

```
public List doCBRecommendForUser(User user) {
        List<Item> recommendList = new ArrayList<Item>();
        try {
            List<Item> allItemList = dao.getItems();
            List<Double> rateList = new ArrayList<Double>();



            for (Item item : allItemList) {
                double rating = 0;
                if (user.getFavZanr().equals(item.getZanr())) {
                    rating = rating + 1;
                }
                else
                {
                    rating = rating - 0.5;
                }

                if(user.getFavVydavatel()
                  .equals(item.getVydavatel())) {
                    rating = rating + 1;
                }

                if (user.getFavDruh().equals(item.getDruh())) {
                    rating = rating + 0.3;
                }

                if(user.getFavRozmerHry()
                  .equals(item.getRozmerHry())) {
                    rating = rating + 0.5;
                }

                if (user.isFavOnline() && item.isOnline()) {
                    rating = rating + 1;
                }

                if (user.getFavJazyk().equals(item.getJazyk())) {
                    rating = rating + 0.5;
                }

                rateList.add(rating);



            }

            double high1 = Double.MIN_VALUE;
            double high2 = Double.MIN_VALUE;
            int indexh1 = Integer.MIN_VALUE;
            int indexh2 = Integer.MIN_VALUE;

            int index = 0;

            for (double rat : rateList) {
                if (rat > high1) {
```

```
                    high2 = high1;
                    indexh2 = indexh1;

                    high1 = rat;
                    indexh1 = index;
                } else if (rat > high2) {
                    high2 = rat;
                    indexh2 = index;
                }
                index++;
            }

            recommendList.add(allItemList.get(indexh1));
            recommendList.add(allItemList.get(indexh2));

        } catch (Exception ex) {

Logger.getLogger(ContentBasedRecommender.class.getName()).log(Level.SE
VERE, null, ex);
        }

         return recommendList;
    }
```

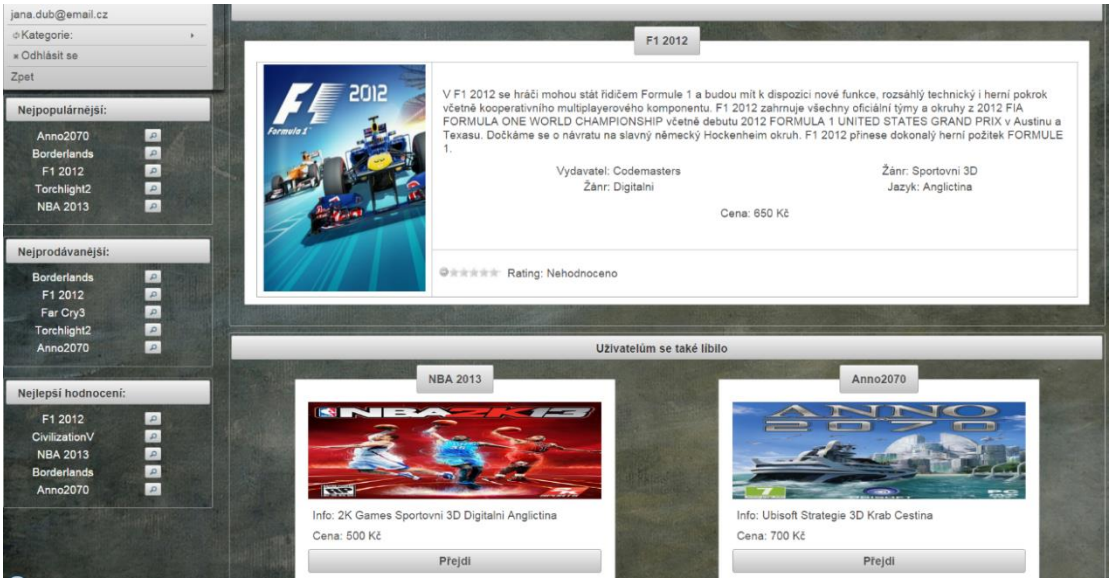
kde za atribut s největší vahou je stanoven atribut žánr hry, připočítaná hodnota k celkovému skóre při situaci, kdy se hodnoty atributu rovnají je 1 a v případě, že jsou hodnoty rozdílné, se odečítá 0,5. Mezi atributy s vysokými váhami patří dále možnost hraní online a atribut vydavatele. Naopak mezi atributy s menší váhou patří rozměr hry (0,5), jazyk hry (0,5) a atribut s nejmenší váhou 0,3 druh. Začlenění vah dává vetší pravděpodobnost položkám, které se liší v bezvýznamných atributech být doporučeny a tím se více přiblížit uživatelově potřebě a prioritám.

Po provedení výpočtu skóre pro všechny položky jsou v druhé části metody vybrány 2 položky s nejvyšším skóre, které jsou následně vráceny. Tímto způsobem bylo zamezeno i problému řídkosti dat.

6.3.6 Doporučení položek k prohlížené položce

Další doporučení zakomponované v navrženém modulu je doporučení dvou položek k prohlížené položce. Toto doporučení funguje na principu „Uživatelé, kteří

vysoce hodnotili tuto položku, také vysoce hodnotili tyto položky“. Toto doporučení se nalézá v panelu pod informacemi o aktuální položce.



Obrázek 27: Doporučení k aktuálně prohlížené položce

Toto doporučení spoléhá na myšlenku, co se líbilo ostatním společně s touto položkou, může preferovat i aktuální uživatel. V aplikaci se před zobrazením položky zavolá přes *Core.java* třída *ItemRecommender.java*, která má za úkol vrátit seznam potencionálních položek. V této třídě se o vrácení stará metoda *getBestSameRateItems(int actualUser_Id, int actualItem_Id)* s parametry ID aktuálního uživatele a ID aktuálně prohlížené položky. Seznam doporučení je generován metodou getItemIdsForBestRatings(int actualUserId, int actualItemId) v *Db.java* pomocí složeného dotazu na tabulku preferences, který je popsán v následujícím příkladu.

Příklad:

Máme tabulku hodnocení, uživatele s ID 2, položku s ID 1 úkolem je pro ni vygenerovat jedno doporučení.

User_ID	Item_ID	Preference
1	1	5
1	2	3
1	3	2.5
1	4	2.5
2	1	2
2	2	2.5
2	3	5
2	4	2
3	1	2.5
3	4	4
3	5	4.5
3	7	5
4	1	5
4	3	3
4	4	4.5
4	6	4
5	1	4
5	2	3
5	3	2
5	4	4
5	5	3.5
5	6	4

Tabulka 18: Tabulka hodnocení uživatelů

Prvním krokem je vybrání všech záznamů hodnocení položky 1, které neudělil uživatel s ID 2 a hodnocením s vyšším než 4. Tomu odpovídá následující část dotazu:

```
SELECT `user_id` as uziv
FROM `preferences`
WHERE `item_id`=1 and `user_id`!=2 and preference>=4
```

Výsledkem je tedy vybrání všech uživatelů, kteří hodnotili aktuální položku pozitivně (4 a více), v našem příkladu jsou to Uživatelé s ID 1, ID 4 a ID 5.

Druhým krokem je získání všech hodnocení od těchto uživatelů, která jsou větší nebo rovno 4. V tomto kroku je nutné také odfiltrovat hodnocení aktuální položky. Použitím předchozího dotazu jako poddotazu dostáváme následující dotaz:

```
SELECT item_id,preference
FROM preferences
WHERE (user_id IN(SELECT `user_id` as uziv
                  FROM `preferences`
                  WHERE `item_id`=1 and `user_id`!=2 and
                  preference>=4)and preference >=4 and item_id!=1)
```

Výsledkem jsou hodnocení položek 4 (rat. 4,5) a 6 (rat. 4) uživatelem s ID 4 a položek 4 (rat. 4), 6 (rat. 4) uživatele s ID5. Z výsledků vypadl uživatel s ID 1 z důvodu, že nebylo u něj nalezeno hodnocení položek větší nebo rovno 4.

Posledním krokem je vypočtení průměru hodnocení jednotlivých položek, kde v případě navržené aplikace jsou dvě položky s nejlepším průměrem vráceny jako výsledek SQL dotazu. Konečný SQL dotaz je následující:

```
SELECT item_id,sum(`preference`)/ COUNT(*) as avgpref
FROM (Select item_id,preference
      FROM preferences
      WHERE (user_id IN (SELECT `user_id` as uziv
                         FROM `preferences`
                         WHERE `item_id`=1 and `user_id`!=2
                         and preference>=4)
            and preference >=4 and item_id!=1)) as samepref
GROUP BY item_id
ORDER BY avgpref desc limit 2
```

Průměr je vypočten podílem funkce *Sum()*, která součet všech hodnocení položky a funkce *Count()*, která vrací počet záznamů. Výsledným doporučením pro daný příklad je doporučení položky s ID 4 s průměrným vysokým hodnocením 4,25 od ostatních uživatelů, kteří ji hodnotili.

6.3.6.1 Řešení problému studeného startu a řídkosti dat

Jako v případě hodnocení položek i tento návrh má nevýhodu v podobě problému studeného startu a řídkosti dat. Problém nastává v případě, kdy se objeví v katalogu nová ještě nikým nehodnocená položka, nebo pokud není nikým ohodnocena nad požadovaný práh nebo nebude mít dostatečný počet hodnocení od uživatelů. Ve všech těchto případech je jisté, že navržený algoritmus nebude schopen vygenerovat adekvátní doporučení, nebo vůbec žádná.

Řešením je opět použití doporučení založeném na obsahu. O analýzu seznamu doporučení se stará v aplikaci ve třídě *Core.java* již zmíněná metoda *doItemRecomendation(User curUser,Item curItem).*

```
public List doItemRecomendation(User curUser,Item curItem)
{

ContentBasedRecommender cbRecommender = new ContentBasedRecommender();
ItemRecommender itemRec = new ItemRecommender();


List<Integer> topRatItemId =
itemRec.getBestSameRateItems(curUser.getId(), curItem.getId());


      if (topRatItemId.isEmpty()) {

itemRecommendList=cbRecommender.doCBRecommendForItem(curItem);

      }
      else {
            if (topRatItemId.size() == 1)
            {
            pomList = cbRecommender.doCBRecommendForItem(curItem);

            itemRecommendList.add(itemlist.get(topRatItemId.get(0)));

                if
(!itemlist.get(topRatItemId.get(0)).getNazev().equals(pomList.get(0).g
etNazev())) {
                    itemRecommendList.add(pomList.get(0));
                } else {
                    itemRecommendList.add(pomList.get(1));
                }


            } else {
```

```
                for (int id : topRatItemId) {
                    itemRecommendList.add(itemlist.get(id));


                }

            }
        }


        return itemRecommendList;
    }
```

Tato metoda v případě, že algoritmus nebude schopen generovat doporučení k položce, zavolá metodu ze třídy *ContentBasedRecommender.java* konkrétně metodu *doCBRecommendForItem(Item actItem)*, která je podobná přechozímu řešení studeného startu. Jako parametrem je zde aktuální prohlížená položka.

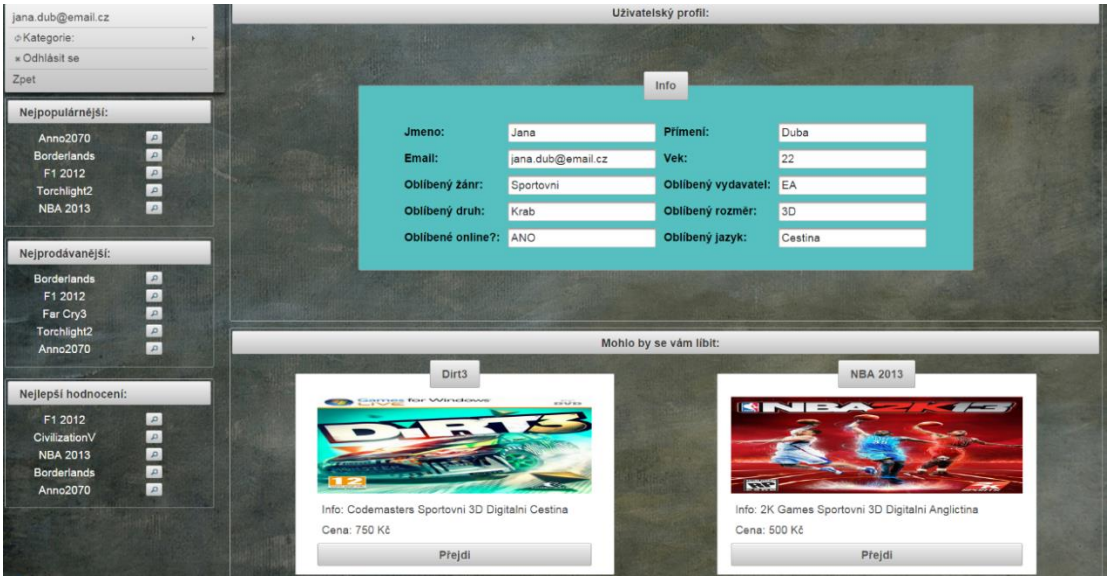
```
public List doCBRecommendForItem(Item actItem) {
        List<Item> recommendList = new ArrayList<Item>();
        try {
            List<Item> allItemList = dao.getItems();


            List<Integer> rateList = new ArrayList<Integer>();

            for (Item item : allItemList) {
                int itemRating = 0;
                if (actItem.getId() != item.getId()) {
                    if (actItem.getZanr().equals(item.getZanr())) {
                        itemRating = itemRating + 1;
                    }

                    if (actItem.getVydavatel()
                        .equals(item.getVydavatel())) {
                        itemRating = itemRating + 1;
                    }

                    if (actItem.getDruh().equals(item.getDruh())) {
                        itemRating = itemRating + 1;
                    }

                    if (actItem.getRozmerHry()
                        .equals(item.getRozmerHry())) {
                        itemRating = itemRating + 1;
                    }

                    if (actItem.isOnline() && item.isOnline()) {
                        itemRating = itemRating + 1;
```

```
                    }

                    if (actItem.getJazyk().equals(item.getJazyk())) {
                        itemRating = itemRating + 1;
                    }
                }
                else
                {
                    itemRating = 0;
                }
                    rateList.add(itemRating);

                }



                int high1 = 0;
                int high2 = 0;
                int indexh1 = Integer.MIN_VALUE;
                int indexh2 = Integer.MIN_VALUE;
                for (int rat : rateList)
                {

                        }
                int index = 0;
                for (int rat : rateList) {

                    if (rat > high1) {
                        high2 = high1;
                        indexh2 = indexh1;


                        high1 = rat;
                        indexh1 = index;
                    }
                    else{
                        if (rat >= high2) {
                        high2 = rat;
                        indexh2 = index;
                        }
                    }
                    index++;
                }

                recommendList.add(allItemList.get(indexh1));
                recommendList.add(allItemList.get(indexh2));


        } catch (Exception ex) {

Logger.getLogger(ContentBasedRecommender.class.getName()).log(Level.SE
VERE, null, ex);
        }

        return recommendList;
    }
```

Metoda porovná atributy jednotlivých položek s atributy aktuální položky a vypočte skóre. Při tomto řešení nebyly použité žádné váhy, protože se nejedná o osobní doporučení, ale o doporučení k položce. Po vygenerování skóre u všech dostupných položek jsou dvě, které jej mají nejvyšší, tudíž jsou nejvíce podobné aktuální položce, vráceny v podobě seznamu, podle kterého jsou doplněna doporučení.

6.3.7 Doporučení na základě uživatelského profilu

Jak bylo uvedeno, doporučení založené na obsahu se v modulu používá jako záložní doporučovací přístup v takových situacích, kdy hlavní kolaborativní přístup selže a není schopen dodat doporučení. Může se tedy stát, že za celou dobu, kterou uživatel stráví v internetovém obchodě, není využita. Proto bylo navrženo další doporučení, které uživateli doporučuje položky na stránce jeho uživatelského profilu. Toto doporučení se nalézá ihned pod panelem tohoto profilu a je založeno čistě na přístupu založeném na obsahu.



Obrázek 28: Doporučení na základě uživatelského profilu

V aplikaci po kliknutí na tlačítko, které přesměruje uživatele na stránku jeho profilu je kontrolerem zavolána metoda v *Core.java* metoda *doCBRecommendationForUser(User user)*, kde parametrem je objekt aktivního uživatele, který obsahuje i atributy jeho profilu. Tato metoda vrací seznam 2 doporučených položek, které jsou získány zavoláním stejné metody jako při selhání kolaborativního doporučení tedy *doCBRecommendForUser(User user)* ze třídy *ContentBasedRecommender.java*.

Výsledný seznam doporučení je vygenerován také stejně i se stejnými váhami atributů. Pro každou položku se vypočte skóre, které závisí na podobnosti jednotlivých vlastností položky a atributů profilu. Když se atributy rovnají, ke skóre se připočte určitá hodnota (váha). V druhé části algoritmu se vyberou dvě položky s nejvyšším skóre a jsou v podobě seznamu vráceny.

Po vygenerování doporučení je uživatel přesměrován na stránku svého profilu, kde jsou mu zobrazeny informace o jeho profilu spolu s doporučením dvou položek, které se nejvíce podobají již zmíněnému profilu.

6.4 Testování optimální kombinace metod pro CF

Jak již bylo zmíněno, Mahout při generování kolaborativního doporučení podporuje několik metod pro výpočet podobností a hledání sousedů. Každá metoda pracuje jinak a má své výhody a nevýhody. V této kapitole bude nejprve provedeno testování jednotlivých metod výpočtu podobností, kde kritériem bude průměrná kvadratická odchylka, střední absolutní odchylka, průměrný čas na jedno doporučení a počet uživatelů, kterým nebylo možné vygenerovat doporučení. V druhé části budou vybrány dvě nejlepší metody výpočtu podobnosti a bude testován způsob výběru sousedů a jejich optimální počet.

6.4.1 RMSE (průměrná kvadratická odchylka)

Metrika RMSE (Root mean square error) udává rozdíl mezi datovým modelem predikovanými hodnotami (predikcemi) a skutečnými hodnotami. Tyto rozdíly jsou označovány jako rezidua neboli chyby predikce. Malé hodnoty RMSE udávají lepší predikcí model a značí, že se predikce blížící se realitě. V případech, kdy RMSE rovná nule, jsou vygenerované predikce absolutně přesné. Výhodou této metriky je znevýhodňování predikcí, které jsou svou hodnotou nejvíce vzdálené skutečnosti. Metrika RMSE patří k častým metrikám používaným při hodnocení kvality predikčního modelu.[40]

RMSE je definována jako:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}\left(X_s - X_p\right)^2}{n}},$$

kde n je počet vzorků, X_s je skutečná hodnota a X_p je hodnota predikce.

6.4.2 MAE (střední absolutní odchylka)

Metrika MAE měří průměrnou velikost chyb v sadě prognóz, bez ohledu na jejich směr a měří přesnost spojitých proměnných. Jinými slovy MAE je průměr absolutních hodnot rozdílů mezi prognózou a odpovídající realitou. MAE je lineární skóre, což znamená, že všechny individuální rozdíly mají v průměru stejnou váhu.

MAE je definováno následovně:

$$MAE = \frac{\sum_{i=1}^{n}(X_s - X_p)}{n},$$

kde n je počet vzorků, X_s je skutečná hodnota a X_p je hodnota predikce.

6.4.3 Testovací dataset

Pro testování byl použit dataset od GroupLens Research, kteří shromažďují a zpřístupňují datasety hodnocení z webových stránek MovieLens (http://movielens.org). Datové soubory byly shromážděny v průběhu různých časových úseků v závislosti na velikosti souboru.[41]

Použitý dataset obsahuje 100 000 hodnocení 1682 filmů (1-5) od 943 uživatelů, kde každý uživatel ohodnotil nejméně 20 filmů a vypadá následovně:[41]

1	1	5	874965758
1	2	3	876893171
1	3	4	878542960
1	4	3	876893119
1	5	3	889751712
1	6	5	887431973
1	7	4	875071561
1	8	1	875072484
1	9	5	878543541
1	10	3	875693118
1	11	2	875072262
1	12	5	878542960
1	13	5	875071805
1	14	5	874965706
1	15	5	875071608
1	16	5	878543541
1	17	3	875073198
1	18	4	887432020
1	19	5	875071515
1	21	1	878542772
1	22	4	875072404
1	23	4	875072895
1	24	3	875071713
1	25	4	875071805

Obrázek 29: Ukázka datasetu

Kde první sloupec udává ID uživatele, který hodnotil položku, druhy sloupec udává ID hodnocené položky a třetí hodnocení. Ve čtvrtém sloupci je uložena časová známka.

6.4.4 Návrh evaluátoru

Mahout poskytuje možnost zhodnocení vytvořeného řešení doporučení oproti skutečné hodnotě preference a to pomocí evaluátoru. Při hodnocení jsou data náhodně rozdělena na testovací množinu a množinu (poměr je dán v evaluátoru parametrem), pomocí které jsou vypočítána doporučení pro všechny testovací uživatele. Vypočítané doporučení je poté porovnáno s testovací množinou a vzápětí je vypočtena odchylka. Evaluátor je napsán v jazyce Java a to následovně:

```
public class RSEvaluate {

public static void main(String[] args) throws IOException,
TasteException {

  BasicConfigurator.configure();

  RandomUtils.useTestSeed();

  DataModel model = new FileDataModel(new File("ua.base"));

  RecommenderEvaluator evaluator = new
                AverageAbsoluteDifferenceRecommenderEvaluator ();

  RecommenderEvaluator evaluator2 = new RMSRecommenderEvaluator();

  RecommenderBuilder builder = new RecommenderBuilder() {
      @Override
      public Recommender buildRecommender(DataModel model) throws
TasteException {

         UserSimilarity similarity =
                            new PearsonCorrelationSimilarity(model);
       //UserSimilarity similarity =
                            new TanimotoCoefficientSimilarity(model);
       //UserSimilarity similarity =
                            new SpearmanCorrelationSimilarity(model);
       //UserSimilarity similarity =
                            new EuclideanDistanceSimilarity(model);



      //UserNeighborhood neighborhood =
                  new NearestNUserNeighborhood(2, similarity, model);
        UserNeighborhood neighborhood =
               new ThresholdUserNeighborhood(0.9, similarity, model);

      return new GenericUserBasedRecommender(model, neighborhood,
similarity);
      }
  };

  double score = evaluator.evaluate(builder, null, model, 0.8, 1.0);
  double score2 = evaluator2.evaluate(builder, null, model, 0.8, 1.0);

  System.out.println("Průměrná odchylka: "+ score);
  System.out.println("Průměrná kvadratická odchylka: "+ score2);


  }

}
```

kde kroky algoritmu jsou následující:

1. *BasicConfigurator.configure()* – inicializace logování

2. *RandomUtils.useTestSeed()* – generování opakovatelných výsledků

3. *DataModel model = new FileDataModel(new File("ua.base"))* – vytvoření datamodelu a načtení dat do něj

4. *RecommenderEvaluator evaluator = new AverageAbsoluteDifferenceRecommenderEvaluator()* – inicializace výpočtu MAE

5. *RecommenderEvaluator evaluator2 = new RMSRecommenderEvaluator()* – inicializace výpočtu RMSE

6. *RecommenderBuilder builder = new RecommenderBuilder() ...* – inicializace výpočtu doporučení, kde v těle jsou nadefinovány použité metody podobnosti a metody sousedu, které při testování budou měněny

7. *Double score = evaluator.evaluate(builder, null, model, 0.8, 1.0)* – výpočet MAE, kde parametrem je builder, model, rozdělení datasetu (0.8 = 80% trénovací množina, 20% testovací množina) a posledním parametrem je procentuální využití datasetu (1.0 = 100%)

8. *Double score2 = evaluator2.evaluate(builder, null, model, 0.8, 1.0)* – výpočet RMSE se stejnými parametry jako v předchozím případě

9. Výpis výsledků, kde ke každému výsledku odchylky jsou vypsány i další informace, jako je počet uživatelů pro které nebylo možné vygenerovat doporučení, nebo čas nutný k vypočtení jednoho doporučení.

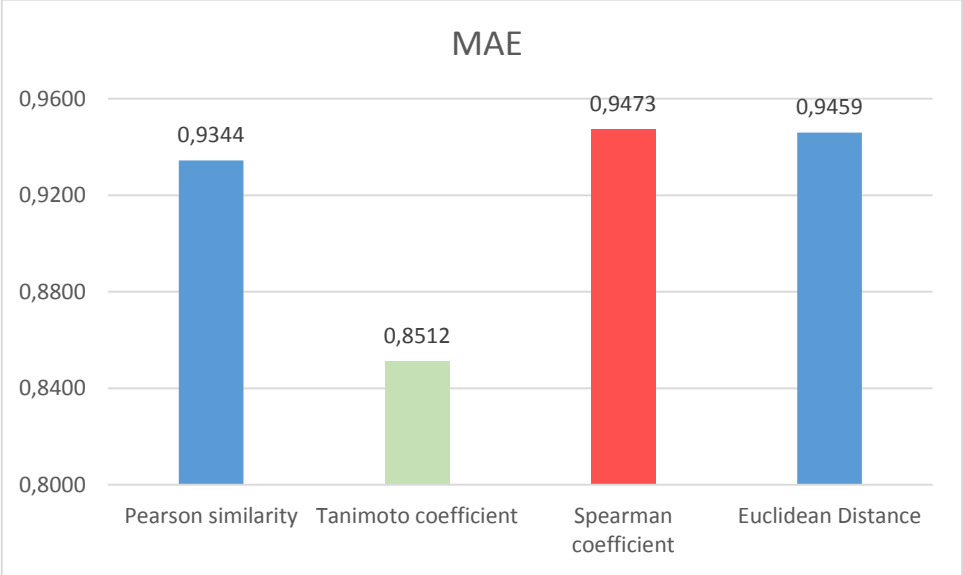
6.4.5 Testování metod pro výpočet podobností

Nejprve bylo provedeno testování metod výpočtu podobnosti pomocí evaluátoru. Metody, které byly testovány, jsou: Pearsonova podobnost, Taminotova podobnost, Spearmanova podobnost a Euklidovskou vzdálenost. Evaluace byla provedena 40x, kde každá metoda byla při evaluaci doporučení použita 10x (viz příloha 2), a výsledky byly zprůměrovány. Výsledkem je následující tabulka, kde nejlepší výsledky jsou označeny zelenou barvou a naopak nejhorší červenou:

	RMSE	MAE	ATPR (ms)	UTR(z 943)
Pearson similarity	1,1766	0,9344	83,60	210,80
Tanimoto coefficient	1,1089	0,8512	321,30	172,75
Spearman coefficient	1,2067	0,9473	840,45	238,35
Euclidean Distance	1,2164	0,9459	96,10	210,15

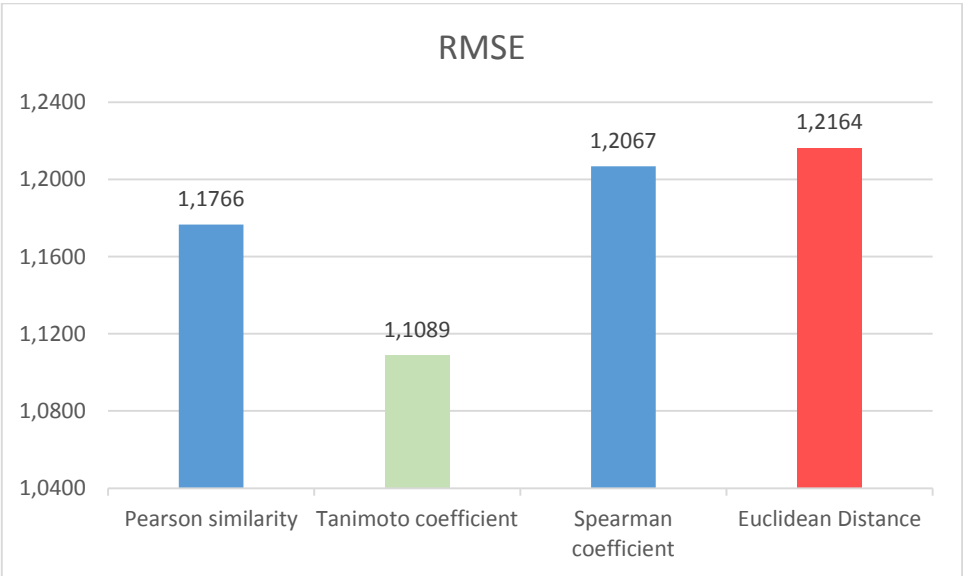
Tabulka 19: Výsledek testování metod podobnosti

Při pohledu na pohled na sloupec MAE je zjevně jasné, že nejlepší výsledek s velkým dosáhla Taminotova podobnost s průměrnou odchylkou 0,8512. Na druhé straně nejhorší výsledek zaznamenala Spearmanova podobnost s výsledkem 0,9473 o 14 deseti tisícin za Euklidovskou podobností (0,9459). Pearsonova podobnost je zde na druhém místě s hodnotou 0,9344.



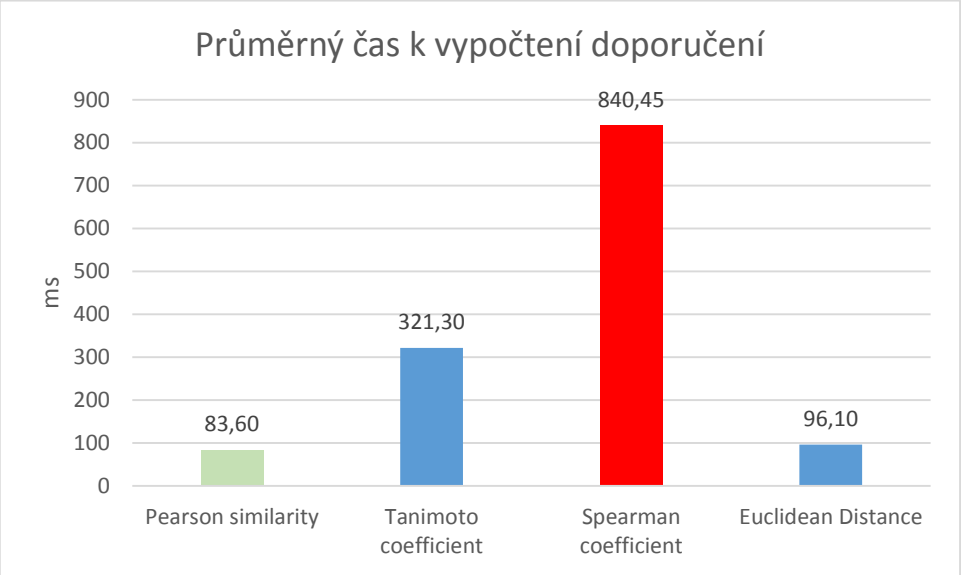
Graf 1: Porovnání metod podobnosti na základě MAE

Co se týče chyby RMSE nejlépe dopadla Taminotova podobnost s průměrnou odchylkou 1,1089. Se značným odstupem na druhém místě opět Pearsonova podobnost s hodnotou. Na druhém konci se pořadí změnilo díky znevýhodnění velkých odchylek v RMSE. Nejhoršího výsledku dosáhla Euklidovská podobnost s hodnotou průměrné chyby 1,2164 těsně za Spearmanovou podobností (1,2067).



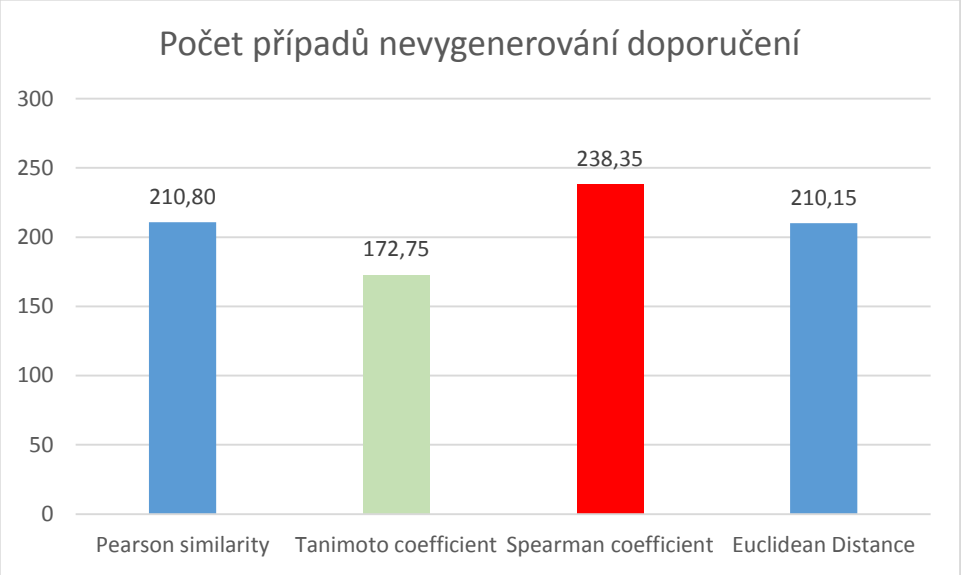
Graf 2: Porovnání metod podobnosti na základě RMSE

Třetí vlastností, která byla testována, je průměrná doba na jedno doporučení. Tedy doba, po které vygenerováno doporučení pro jednoho uživatele. Zde jsou rozdíly mezi metodami více signifikantní. Nejlepšího času dosáhla Pearsonova podobnost s průměrným časem 83,6 ms. Jako druhá nejrychlejší byla s odstupem jen 12,5 ms Euklidovská podobnost s průměrnou dobou 96,10 ms. Mezi těmito metodami je jen nepatrný rozdíl. Naopak nejdelšího času pro vygenerování doporučení dosáhla Spearmanova podobnost se skoro desetinásobkem času nejrychlejší metody 840,45 ms. Taminotova podobnost dosáhla průměrné doby 321,30 ms.



Graf 3: Porovnání metod podobnosti na základě průměrného času

Posledním testovacím aspektem byl průměrný počet uživatelů z 943, kterým nebylo možno vygenerovat doporučení. Zde nejlepšího výsledku dosáhla Taminotova podobnost s 172,75 uživateli. Na průměrnou hodnotu 210 uživatelů dosáhla Euklidovská podobnost spolu s Pearsonovou podobností. Nejvyšší průměrný počet uživatelů, pro které nebylo možné najít doporučení, bylo zaznamenáno u metody Spearmanovy podobnosti s hodnotou 238,5.



Graf 4: Porovnání metod podobnosti na základě počtu uživatelů, kterým nebylo možné vygenerovat doporučení

Testováním bylo potvrzeno, že každá metoda podobnosti má své silné a slabé stránky. Nejhůře dopadla Spearmanova podobnost, která nejen, že požaduje nejdelší dobu pro vygenerování doporučení, ale i její přesnost patří mezi úplně nejhorší. Euklidovská podobnost sice patří mezi rychlejší metody, ale na úkor správnosti vygenerovaného doporučení. Nejlépe dopadla metoda Taminotova koeficientu, která se umístila, krom testování rychlosti, vždy na prvním místě. Její slabostí ale je vysoká doba výpočtu doporučení, která se bude projevovat při růstu datové základny.

Taminotova a Pearsonova metoda výpočtu podobnosti byly zvoleny jako metody k testování výběru sousedů.

6.4.6 Testování způsobu a výběru sousedů a jejich optimálního počtu

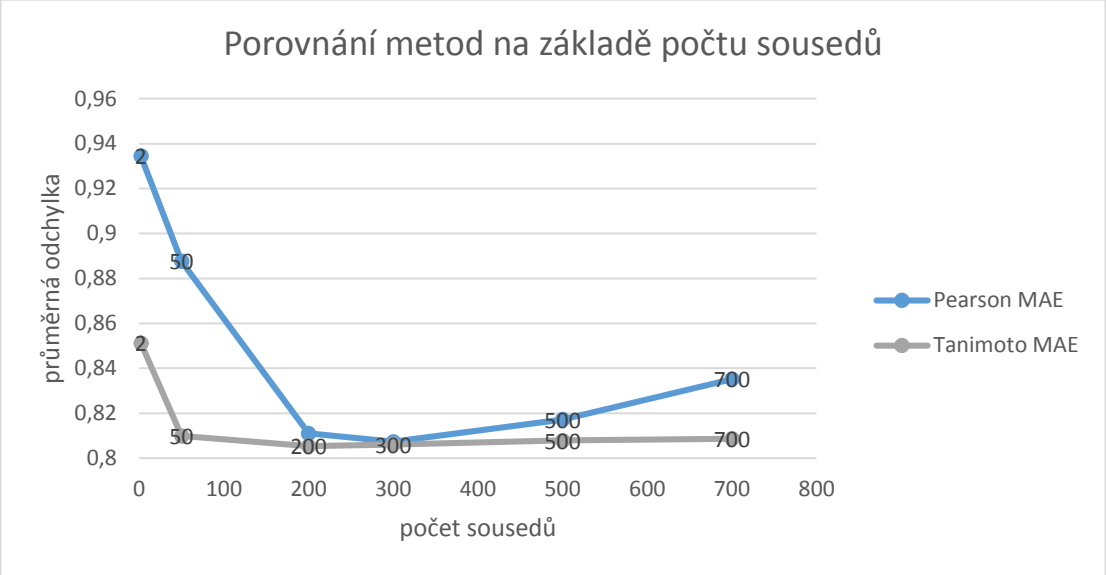
V této testovací části bylo hlavním úkolem je porovnat metody výběru a to metod Nejblížších N sousedů a Tresholdu a zjistit jejich ideální hodnotu parametrů při kombinaci s metodami výpočtu podobnosti, které vyšli nejlépe z předchozího testování. Hlavním porovnávacím hlediskem zde byly hodnoty MAE a RMSE v průběhu změny parametrů. Pro testování byl použit stejný datový model i evaluátor, jako v předchozím případě.

Prvním testovaným přístupem byl vybírání nejbližších N sousedů. Evaluace doporučovacího návrhu byla provedena pro každou metodu podobnosti pro hodnoty N = 2, 50, 200, 300, 500, 700, kde N značí počet vybíraných nejbližších sousedů a pro každou evaluaci byly zaznamenány hodnoty MAE a RMSE do následující tabulky:

NearestN		2	50	200	300	500	700
Pearson	MAE	0,93448	0,88764	0,81100	0,80729	0,81722	0,8352
	RMSE	1,17664	1,12946	1,02573	1,02188	1,03879	1,07129
Tanimoto	MAE	0,85124	0,80987	0,80541	0,80612	0,80789	0,80866
	RMSE	1,10896	1,02042	1,01062	1,01101	1,01315	1,01353

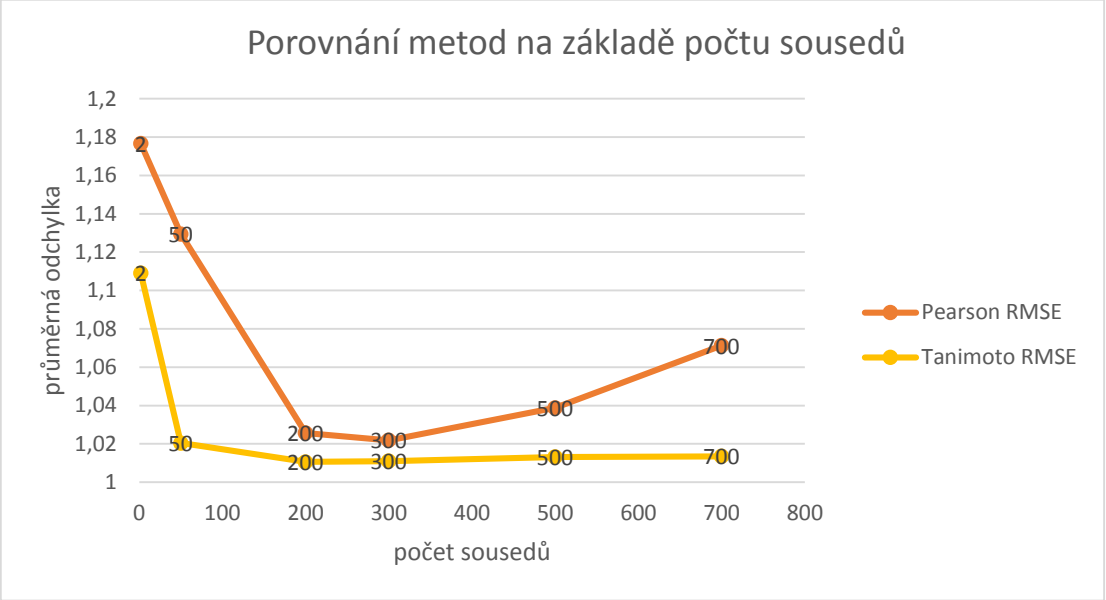
Tabulka 20: Výsledek testování nejbližších N sousedů

Poté bylo provedeno porovnání řádků MAE u Pearsonovy a Tanimotovy metody. Graf je následující:



Graf 5: Porovnání vývoje MAE

Z grafu je jasný nejhorší výsledek u obou při hodnotě sousedů 2, kde se průměrná odchylka pohybuje na hodnotě pro Pearsonovu metodu 0,93448 a 0,85124 pro Taminotovu metodu. Při zvýšení počtu sousedů na 50 je zřejmý výrazný pokles chybovosti na 0,88764 a 0,80987. Při hodnotě 200 sousedů je na grafu vidět prudký pokles hodnoty odchylky u Pearsonovy metody (0,81100). Co se týče Taminotovi metody, tak ta nabývá při této hodnotě svého minima 0,80541, kde od této doby se při zvýšení počtu sousedů nepatrně zvyšuje. Minimum druhé metody nastává při 300 sousedech, ale ani zde při hodnotě 0,80729 nedosahuje lepších výsledků než Taminotova metoda. Vzhledem k malému rozdílu chybovosti se mohou v tomto případě brát metody jako identické. Dále pak chybovost Pearsonovy metody při hodnotách 500 a 700 sousedů stoupá až k chybovosti 0,8352.



Graf 6: Porovnání vývoje RMSE

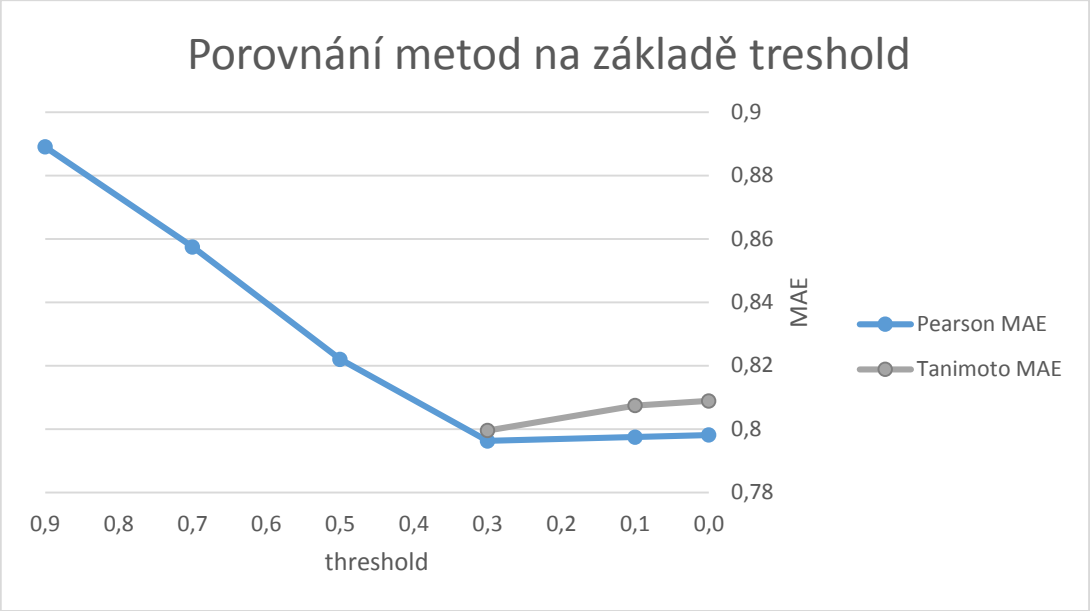
V případě odchylky RMSE je graf téměř identický s předchozím. I zde jsou zřejmé vysoké hodnoty při použití dvou sousedů, kde odchylka RMSE Pearsonovi metody se pohybuje na hodnotě 1,17664 a Taminotovi na 1,10896. V případě Pearsonovi metody nastává v dalším zvyšování počtu sousedů razantní pokles, který ustává v minimu při chybovosti 0,80729 a počtu sousedů 300. Od této hodnoty chybovost stoupá a nikdy nedosahuje lepšího výsledku než Taminotova metoda, která dosahuje svého minima při 200 sousedech při hodnotě 1,01062.

Druhým testovaným přístupem byl vybírání sousedů podle tresholdu, čili hranice po kterou jsou sousedé vybráni. Evaluace doporučovacího návrhu byla znovu provedena pro každou metodu podobnosti pro hodnoty tresholdu 0,9; 0,7; 0,5; 0,3; 0,1; 0, kde hodnota udává minimální podobnost. Pro každou evaluaci byly znovu zaznamenány hodnoty MAE a RMSE a výsledkem byla následující tabulka:

Treshold		0,9	0,7	0,5	0,3	0,1	0
Pearson	MAE	0,88904	0,85749	0,82202	0,79625	0,79748	0,79811
	RMSE	1,12948	1,08843	1,03948	1,00432	1,0026	1,0038
Tanimoto	MAE	NaN	NaN	NaN	0,79953	0,8074	0,80887
	RMSE	NaN	NaN	NaN	1,00983	1,0142	1,1371

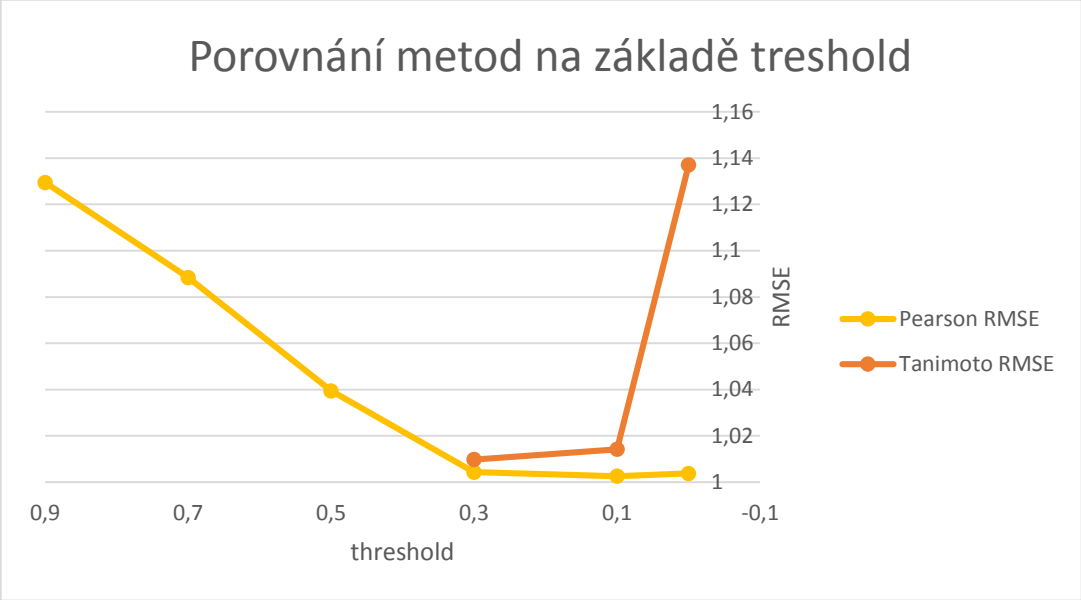
Graf 7: Výsledek testování Tresholdu

Poté byly porovnány jednotlivé metody, viz následující graf:



Graf 8: Porovnání průběhu MAE

Při pohledu na graf je jasné, že nejvyšších hodnoty a to 0,88904 nabývá Pearsonova podobnost při hodnotě tresholdu 0,9. Při snižování této hodnoty klesá i chybovost a to až do svého minima, kde hodnota průměrné odchylky je 0,79625 při tresholdu 0,3, což je i nejnižší hodnota MAE, která byla při testování naměřena. Při dalším snižování tresholdu se chybovost jen o několik tisícin zvyšuje. Co se týče Taminotovy metody, tak pomocí ní není možné vygenerovat při hodnotách tresholdu 0,9-0,5 žádná doporučení proto MAE je rovno NaN. První doporučení je vygenerováno při hodnotě 0,3, kde hodnota odchylky je rovna 0,79953 a zároveň představuje i minimum Taminotovi metody. Při dalších změnách tresholdu se odchylka pohybuje na hodnotě 0,8074 respektive 0,80887. Při této kombinaci metod a tresholdu poprvé se ukazuje lepší metoda Pearsonovi podobnosti nad Taminotovou podobností.



Graf 9: Porovnání průběhu RMSE

Druhým grafem je opět porovnání RMSE. Pearsonova podobnost opět rychle klesá ze svého maxima při tresholdu 0,9 a hodnotě 1,12948 do hodnoty 1,00432 při tresholdu 0,3, která na rozdíl od MAE není minimem. Minimem je v tomto případě hodnota RMSE 1,0026, kde nastavený treshold je roven 0,1. Pro Taminotovu podobnost opět není možné nalézt doporučení a to až do hodnoty tresholdu 0,3, kde se odchylka pohybuje na hodnotě 1,00983. Při dalším snižování tresholdu se RMSE zvyšuje a to až do maxima a zároveň i maximální hodnoty celého grafu 1,1371 při tresholdu 0. Je tedy jasné, že i z pohledu RMSE dopadla výrazně lépe Pearsonova podobnost.

6.4.7 Diskuse výsledků testování

Při testování bylo zjištěno chování různých metod výpočtu podobností, kde dvě nejlepší metody byly vybrány pro další část testování. V této druhé části byly testovány způsoby výběru sousedů v kombinaci s vybranými metodami. Úplně nejlepšího výsledku dosáhla kombinace Pearsonovi podobnosti s výběrem sousedů pomocí tresholdu s hodnotou 0,3 a to s průměrnou odchylkou 0,79625. Tímto byla vybrána ideální kombinace pro velká budoucí data. Pro navrženou aplikaci ale z důvodu malé

datové základny byla použita kombinace Pearsonovy podobnosti (z duvodu nejlepších výsledků, co se týče rychlosti výpočtu) a výběru sousedů pomocí dvou nejbližších sousedů.

7 Závěr

Byly splněny všechny vymezené cíle této diplomové práce. Nejprve byly prozkoumány druhy uživatelské preference a způsoby jejího získávání, poté byly představeny doporučovací systémy, analyzovány a vysvětleny jednotlivé druhy doporučovacích systémů společně s ukázkami generování doporučení. V poslední části teoretického oddílu této práce byl představen Mahout.

V praktické části této práce byl navrhnut doporučovací modul, který byl zakomponován do webové aplikace simulující internetový obchod. V tomto modulu byl navržen způsob vytváření uživatelského profilu a dále různé druhy doporučení. Jedná se o doporučení top 10 nejvíce prodávaných, nejprohlíženějších a položek nejlépe hodnocených položek, dále pak o implementaci kolaborativního filtrování pomocí představeného Mahoutu při generování potenciálně preferovaných položek na základě podobnosti hodnocení, o doporučení položek k aktuálně prohlížené položce, které byly vygenerovány na základě „uživatelé, kteří vysoce hodnotili tuto položku, také vysoce hodnotili následující položky“ a o aplikaci doporučení založeného na obsahu v podobě doporučení položek na základě vlastností uživatelského profilu. Dalším krokem bylo vyřešení problému studeného startu. Jako řešení bylo aplikování přepínacího hybridního doporučovacího systému, který v případě, že hlavní část selže, je doporučení vygenerováno nebo doplněno doporučením založeném na obsahu. Tento návrh zároveň vyřešil také problém řídkosti dat.

V poslední části byla navržena optimální kombinace metod výpočtu podobnosti a způsobu výběru sousedů při generování doporučení. Tato kombinace byla výsledkem testování a analýzy výsledků pomocí naprogramovaného analyzátoru, který využívá Mahoutu. Jako nejpřesnější metoda byla vyhodnocena Taminotova metoda, a jako druhá, jejíž výhodou je hlavně rychlost, se umístila Pearsonova metoda. V další části při testování a hledání ideální kombinace těchto metod s metodami výběru sousedů se prokázal rozdílný výsledek a jako nejefektivnější byla nalezena kombinace Pearsonovi

metody a vybíraní sousedů pomocí Treshold o hodnotě 0,3 s průměrnou odchylkou 0,79953.

Seznam tabulek

Seznam obrázků

Seznam grafů

Literatura

[1] PEŠKA, Ladislav. *Uživatelské preference v prostředí prodejních webů* [online]. Praha, 2010 [cit. 2015-10-14]. Dostupné z:

<up-comp.googlecode.com/files/diplomka-text_final.pdf>. Diplomová práce. Univerzita Karlova v Praze, Matematicko-fyzikální fakulta, Katedra softwarového inženýrství. Vedoucí práce prof. RNDr. Peter Vojtáš, DrSC.

[2] VOJTÁŠ, Peter. *Modely uživatelských preferencí* [online]. [cit. 2015-10-14]. Dostupné z: http://www.ksi.mff.cuni.cz/~vojtas/vyuka/NDBI021PrincipyUzivatelskychPreferenci/1112_NSWI021_DotazovaniSPreferencemi/DBI021modelyUzivatele.ppt

[3] KUKHAR, Maria. *Content-based doporučovací systémy*. Praha, 2015. Dostupné také z: <https://is.cuni.cz/webapps/zzp/download/120175400>. Diplomová práce. Univerzita Karlova v Praze Matematicko-fyzikální fakulta, Studijní program: Informatika, Studijní obor: softwarové systémy. Vedoucí práce Mgr. Peška Ladislav.

[4] Diane Kelly, Jaime Teevan: Implicit feedback for inferring user preference: a bibliography. ACM SIGIR Forum, Volume 37 Issue 2, Fall 2003, 18-28. ACM, 2003. URL: http://portal.acm.org/citation.cfm?id=959258.959260

[5] Gawesh Jawaheer, Martin Szomszor,Patty Kostkova: Comparison of implicit and explicit feedback from an online music recommendation service. HetRec '10, 47-51. ACM, 2010. URL :http://portal.acm.org/citation.cfm?id=1869446.1869453. Dostupné také z: <http://ir.ii.uam.es/hetrec2010/res/papers/hetrec2010_paper_07.pdf>

[6] HU, Yifan, Yehuda KOREN a Chris VOLINSKY. *Collaborative Filtering for Implicit Feedback Datasets* [online]. : 1-10 [cit. 2015-10-14]. Dostupné z:

<http://www.researchgate.net/publication/220765111_Collaborative_Filtering_for_Implicit_Feedback_Datasets>

[7] HOLUB, Michal a Maria BIELIKOVA. *Estimation of user interest in visited web page.* [online]. [cit. 2015-10-14]. Dostupné z: <http://www.researchgate.net/publication/221022929_Estimation_of_user_interest_in_visited_web_page>

[8] MELVILLE P., SINDHWANI, V., Recommender Systems [online]. In Encyclopedia of Machine Learning, 2010. [cit. 2015-10-14]. Dostupné z: <http://www.prem-melville.com/publications/recommender-systems-eml2010.pdf>.

[9] VALA, Martin. *E-learning – doporučovací systémy* [online].Brno, 2012 [cit. 2015-10-14]. Dostupné z: <http://is.muni.cz/th/359917/fi_b/bp_final_vala.pdf>. Bakalářská práce. Masarykova univerzita, Fakulta informatiky. Vedoucí práce Mgr. Jan Géryk.

[10]CVENGROŠ, Petr. *Universal Recommender System* [online]. Prague, 2011 [cit. 2015-10-14]. Dostupné z: <https://unresyst.googlecode.com/files/dp_final.pdf>. Master thesis. Charles University in Prague, Faculty of Mathematics and Physics, Department of Software Engineering. Vedoucí práce prof. RNDr. Peter Vojtáš, DrSc.

[11] SCHAFER, J. Ben, Joseph KONSTAN a John Riedl. *Recommender Systems in E-Commerce* [online]. : 1-9 [cit. 2015-10-14]. Dostupné z: <http://files.grouplens.org/papers/ec-99.pdf>

[12] Historie a zajímavosti fenoménu eBay. *Icons.cz* [online]. [cit. 2015-10-14]. Dostupné z: <http://icons.cz/92-historie-a-zajimavosti-fenomenu-ebay.html>

[13] SMITH, Matt. The Amazon Shopping Guide. *Makeuseof* [online]. [cit. 2015-10-14]. Dostupné z: <http://www.makeuseof.com/tag/your-unofficial-amazon-trail-guide/>

[14] JANNACH, D., ZANKER, M., FELFERNIG, A., FRIEDRICH, G., *Recommender Systems: An Introduc-tion*. Cambridge University Press, 2010. ISBN: 978-0521493369.

[15] HOŘEJŠ, Martin. Analýza a návrh doporučovacího systému [online]. České Budějovice, 2015 [cit. 2015-10-14]. Bakalářská práce. Jihočeská univerzita v Českých Budějovicích, Ekonomická fakulta. Vedoucí práce doc. Ing. Ladislav Beránek, CSc.. Dostupné z: <http://theses.cz/id/3gwk6g/>.

[16] BADRUL M. SARWARL, Badrul, George KARYPIS, Joseph A. KONSTAN a John T. RIED. *Application of Dimensionality Reduction in Recommender System -- A Case Study* [online]. [cit. 2015-10-14].
Dostupné z: <http://files.grouplens.org/papers/webKDD00.pdf>

[17] BAKER, Kirk. *Singular Value Decomposition Tutorial* [Online]. 29.3.2005. [cit. 2015-10-14]. Dostupné z: <https://www.ling.ohio-state.edu/~kbaker/pubs/Singular_Value_Decomposition_Tutorial.pdf>

[18] LEMIRE, Daniel a Anna MACLACHTAN. *Slope One Predictors for Online Rating-Based Collaborative Filtering* [online]. [cit. 2015-10-14]. Dostupné z: <http://lemire.me/fr/documents/publications/lemiremaclachlan_sdm05.pdf>

[19] SARWAR, Badrul, George KARYPIS, Joseph KONSTAN a John REIDL. *Item-Based Collaborative Filtering Recommendation Algorithms* [online]. [cit. 2015-10-14]. Dostupné z: <http://www.ra.ethz.ch/cdstore/www10/papers/pdf/p519.pdf>

[20] SU, Xiaoyuan a Taghi M. KHOSHGOFTAAR. *A Survey of Collaborative Filtering Techniques* [online]. 2009, 19 s. [cit. 2015-10-14]. Dostupné z: <http://www.hindawi.com/journals/aai/2009/421425/#sec4>

[21] CLAYPOOL, Mark, Anuja GOKHALE, Tim MIRANDA, Pavel MURNIKOV, Dmitry NETES a Matthew SARTIN. *Combining Content-Based and Collaborative Filters in an Online Newspaper* [online]. 1999 [cit. 2015-10-14]. Dostupné z: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.45.5230&rep=rep1&type=pdf>

[22] KUMAR, B. Raja Sarath a B. John RATNAM. *Evaluation of Collaborative Filtering Personalized Recommendation algorithms* [online]. [cit. 2015-10-14]. Dostupné z: <http://worldcomp-proceedings.com/proc/p2012/ICA4012.pdf>

[23] ADOMAVICIUS, Gediminas a Alexander TUZHILIN. *Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions*. 2005[cit. 2014-10-14]. Dostupné z: <http://citeseer.ist.psu.edu/viewdoc/download?doi=10.1.1.107.2790&rep=rep1&type=pdf>

[24] BURKE, Robin. *Hybrid Recommender Systems: Survey and Experiments* [online]. 2002, 29 s. [cit. 2015-10-14]. Dostupné z: <http://dl.acm.org/citation.cfm?id=586352>

[25] KARÁSEK, J. Citlivost metod pro měření podobnosti kvantitativních proměnných. Acces server [online]. 2012, roč. 12 [cit. 2015-6-25]. Dostupné z: <http://access.feld.cvut.cz/view.php?cisloclanku=2012090003>.

[26] CHROBÁK, Martin. *SHLUKOVÁ ANALÝZA* [online]. Brno, 2012 [cit. 2015-10-14]. Dostupné z: <https://dspace.vutbr.cz/bitstream/handle/11012/12032/Martin_Chrobak_DP.pdf>. Diplomová práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav biomedicínského inženýrství. Vedoucí práce Doc. Ing. JIŘÍ KOZUMPLÍK, CSc.

[27] BAKOS, Yong Joseph. *Mining Similarity Using Euclidean Distance, Pearson Correlation, and Filtering* [online]. [cit. 2015-10-14]. Dostupné z: <http://mines.humanoriented.com/classes/2010/fall/csci568/portfolio_exports/mvoget/similarity/similarity.html>

[28] Pearson Product-Moment Correlation. *Laerd Statistics* [online]. [cit. 2015-10-14]. Dostupné z: <https://statistics.laerd.com/statistical-guides/pearson-correlation-coefficient-statistical-guide.php>

[29] STRUŽSKÝ, Martin. *Kolaborativní filtrování pro adaptivní web* [online]. Praha, 2009 [cit. 2015-10-14]. Bakalářská práce. České vysoké učení technické v Praze, Fakulta elektrotechnická, Katedra počítačů. Vedoucí práce Ing. Martin Balík. Dostupné z: <https://dip.felk.cvut.cz/browse/pdfcache/struzm1_2009bach.pdf>.

[30] SERGIOS THEODORIDIS, Konstantinos Koutroumbas. *Pattern Recognition*. 4th ed. Burlington: Elsevier, 2008. ISBN 978-008-0949-123.

[31] KUČERA, Jiří. Metody kategorizace dat [online]. Brno, 2008 [cit. 2015-08-28]. Bakalářská práce. Masarykova univerzita, Fakulta informatiky. Vedoucí práce Matěj Štefaník. Dostupné z: <http://is.muni.cz/th/172767/fi_b/>.

[32] PŘIBYLOVÁ, Alexandra. Průzkumová analýza vícerozměrných dat [online]. Brno, 2015 [cit. 2015-08-28]. Diplomová práce. Masarykova univerzita, Přírodovědecká fakulta. Vedoucí práce Marie Budíková. Dostupné z: <http://is.muni.cz/th/379569/prif_m/>.

[33] BERKA, Petr. *Dobývání znalostí z databází*. Vyd. 1. Praha: Academia, 2003, 366 s. ISBN 80-200-1062-9.

[34] MELOUN, M. Analýza shluků CLU. [online]. 2002 [cit. 2015-10-14]. Dostupné z: <http://meloun.upce.cz/docs/research/chemometrics/methodology/4gmetody.pdf>

[35] MELOUN, Milan a Jiří MILITKÝ. Přednosti analýzy shluků ve vícerozměrné statistické analýze. In: *Zajištění kvality analytických výsledků: Sborník přednášek z konference Zajištění kvality analytických výsledků* [online]. 2004 [cit. 2015-10-15]. ISBN 80-86380-22-X. Dostupné z: <http://meloun.upce.cz/docs/publication/152.pdf>

[36] Shluková analýza: Algoritmus k-means. *Informační systém MU* [online]. [cit. 2015-10-15]. Dostupné z: <http://is.muni.cz/th/172767/fi_b/5739129/web/web/kmeans.html>

[37] OWEN, Sean. *Mahout in action*. Shelter Island, New York: Manning, c2012, xxv, 387 p. ISBN 19-351-8268-4.

[38] KUBOVÝ, Tomáš. *Vaadin - přechod na novou technologii existujícího portfolia produktů* [online]. Plzeň, 2014 [cit. 2015-10-15]. Diplomová práce. Západočeská univerzita v Plzni, Fakulta aplikovaných věd. Vedoucí práce Ing. Jiří Kala. Dostupné z: <http://theses.cz/id/bet5af/>.

[39] *GlassFish Community - Frequently Asked Questions* [online]. [cit. 2015-10-15]. Dostupné z: <https://glassfish.java.net/public/faq/GF_FAQ_2.html>

[40] LIPTÁK, Maroš. *Predikce datových toků počítačových sítí* [online]. Brno, 2013 [cit. 2015-10-15]. Diplomová práce. Masarykova univerzita, Fakulta informatiky. Vedoucí práce Petr Holub. Dostupné z: <http://is.muni.cz/th/324676/fi_m/>.

[41] MovieLens. *Grouplens* [online]. [cit. 2015-10-15]. Dostupné z: <http://grouplens.org/datasets/movielens/>

[42]MATEJOVSKÝ, Vladimír. Podpora shlukování webových stránek pomocí link mining. [online]. Brno, 2013 [cit. 2015-10-28]. Bakalářská práce. Masarykova univerzita, Fakulta informatiky. Vedoucí práce Jaroslav Bayer. Dostupné z: <http://is.muni.cz/th/374563/fi_b/>.

Přílohy

Příloha č. 1

Obsah přiloženého CD

- text práce ve formátu PDF,
- exportovaná databáze ve formátu sql,
- zdrojový kód doporučovacíhosystému v projektu netbeans.

Příloha č. 2

Jednotlivá měření při hledání optimální kombinace metod pro kolaborativní filtrování.

	Pearson	943	
Měření:	RMSE	Average time per recommendation(ms)	Unable to recommend
1	1.1766404542682907	78	251
2	1.1766404542682907	81	220
3	1.1766404542682907	82	145
4	1.1766404542682904	71	158
5	1.1766404542682907	83	267
6	1.1766404542682907	91	156
7	1.1766404542682904	89	260
8	1.1766404542682907	109	130
9	1.1766404542682907	86	300
10	1.1766404542682907	74	225
Průměr		84,4	211,2

	MAE	Average time per recommendation(ms)	Unable to recommend
1	0.9344827586206896	69	176
2	0.9344827586206896	64	158
3	0.9344827586206896	78	291
4	0.9344827586206896	100	284
5	0.9344827586206896	89	154
6	0.9344827586206896	91	251
7	0.9344827586206896	92	173
8	0.9344827586206896	63	219
9	0.9344827586206896	90	160
10	0.9344827586206896	92	238

Průměr		82,8	210,4

	Spearman	943	
Měření	RMSE	Average time per recommendation(ms)	Unable to recommend
1	1.206706406246852	827	233
2	1.206706406246852	808	219
3	1.206706406246852	831	230
4	1.206706406246852	839	228
5	1.206706406246852	989	229
6	1.206706406246852	791	225
7	1.206706406246852	814	233
8	1.206706406246852	789	230
9	1.206706406246852	869	244
10	1.206706406246852	861	234
Průměr		841,8	230,5

	MAE	Average time per recommendation(ms)	Unable to recommend
1	0.9473684210526305	855	266
2	0.9473684210526305	839	240
3	0.9473684210526305	848	240
4	0.9473684210526305	806	253
5	0.9473684210526305	830	237
6	0.9473684210526305	827	230
7	0.9473684210526305	877	243
8	0.9473684210526305	828	277
9	0.9473684210526305	860	239
10	0.9473684210526305	821	237
Průměr		839,1	246,2

	Tanimoto	943	
	RMSE	Average time per recommendation(ms)	Unable to recommend
1	1.1089689753835041	302	179
2	1.1089689753835050	347	172
3	1.1089689753835028	311	151
4	1.1089689753835037	315	163
5	1.1089689753835017	300	160
6	1.1089689753835046	360	214

7	1.1089689753835046	317	193
8	1.1089689753835017	328	190
9	1.108968975383503	292	161
10	1.108968975383504	310	171
Průměr		318,2	175,4

	MAE	Average time per recommendation(ms)	Unable to recommend
1	0.8512498782231285	427	145
2	0.8512498782231246	334	181
3	0.8512498782231256	305	150
4	0.8512498782231237	234	192
5	0.8512498782231278	338	187
6	0.851249878223126	327	178
7	0.8512498782231268	293	150
8	0.8512498782231268	315	157
9	0.8512498782231251	355	180
10	0.8512498782231283	316	181
Průměr		324,4	170,1