Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií

FIIT-5220-46118

Adam Lieskovský

PERSONALIZOVANÉ ODPORÚČANIE VYUŽITÍM ŠKÁLOVATEĽNEJ ARCHITEKTÚRY

Diplomová práca

Študijný program: Softvérové inžinierstvo

Študijný odbor: 9.2.5 Softvérové inžinierstvo

Miesto vypracovania: Ústav informatiky a softvérového inžinierstva. FIIT STU

Vedúci práce: Ing. Michal Kompan, PhD.

Máj 2016

STU FIIT
SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA INFORMATIKY A INFORMAČNÝCH TECHNOLÓGIÍ

Zadanie diplomovej práce

Meno študenta: **Bc. Adam Lieskovský**

Študijný program: Softvérové inžinierstvo

Študijný odbor: Softvérové inžinierstvo

Názov práce: **Personalizované odporúčanie využitím škálovateľnej architektúry**

Samostatnou výskumnou a vývojovou činnosťou v rámci predmetov Diplomový projekt I, II, III vypracujte diplomovú prácu na tému, vyjadrenú vyššie uvedeným názvom tak, aby ste dosiahli tieto ciele:

Všeobecný cieľ:

Vypracovaním diplomovej práce preukážte, ako ste si osvojili metódy a postupy riešenia relatívne rozsiahlych projektov, schopnosť samostatne a tvorivo riešiť zložité úlohy aj výskumného charakteru v súlade so súčasnými metódami a postupmi študovaného odboru využívanými v príslušnej oblasti a schopnosť samostatne, tvorivo a kriticky pristupovať k analýze možných riešení a k tvorbe modelov.

Špecifický cieľ:

Vytvorte riešenie zodpovedajúce návrhu textu zadania, ktorý je prílohou tohto zadania. Návrh bližšie opisuje tému vyjadrenú názvom. Tento opis je záväzný, má však rámcový charakter, aby vznikol dostatočný priestor pre Vašu tvorivosť.

Riaďte sa pokynmi Vášho vedúceho.

Pokiaľ v priebehu riešenia, opierajúc sa o hlbšie poznanie súčasného stavu v príslušnej oblasti, alebo o priebežné výsledky Vášho riešenia, alebo o iné závažné skutočnosti, dospejete spoločne s Vaším vedúcim k presvedčeniu, že niečo v texte zadania a/alebo v názve by sa malo zmeniť, navrhnite zmenu. Zmena je spravidla možná len pri dosiahnutí kontrolného bodu.

Miesto vypracovania: Ústav informatiky a softvérového inžinierstva FIIT STU v Bratislave

Vedúci práce: **Ing. Michal Kompan, PhD.**

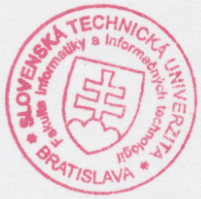
Termíny odovzdania:

Podľa harmonogramu štúdia platného pre semester, v ktorom máte príslušný predmet (Diplomový projekt I, II, III) absolvovať podľa Vášho študijného plánu

Predmety odovzdania:

V každom predmete dokument podľa pokynov na www.fiit.stuba.sk v časti: home > Informácie o > štúdiu > organizácia štúdia > diplomový projekt.

V Bratislave dňa 16. 2. 2015



prof. Ing. Pavol Návrat, PhD.
riaditeľ Ústavu informatiky a softvérového inžinierstva

STU FIIT
SLOVENSKÁ TECHNICKÁ
UNIVERZITA V BRATISLAVE
FAKULTA INFORMATIKY
A INFORMAČNÝCH TECHNOLÓGIÍ

Návrh zadania diplomovej práce

Finálna verzia do diplomovej práce [1]

Študent:

Meno, priezvisko, tituly:	Adam Lieskovský, Bc.
Študijný program:	Softvérové inžinierstvo
Kontakt:	adamliesko@gmail.com

Výskumník:

Meno, priezvisko, tituly:	Michal Kompan, Ing. PhD.

Projekt:

Názov:	Personalizované odporúčanie využitím škálovateľnej architektúry
Názov v angličtine:	Personalized scalable recommendation system
Miesto vypracovania:	Ústav informatiky a softvérového inžinierstva, FIIT STU, Bratislava
Oblasť problematiky:	Personalizované odporúčanie

Text návrhu zadania[2]

Personalizované odporúčanie je už neoddeliteľnou súčasťou webu. Na jednej strane neustále pomáha pri redukovaní informačného zahltenia, na strane druhej čoraz viac reprezentuje pridanú hodnotu poskytovanú jednotlivými stránkami, či službami využívanú v konkurenčnom prostredí pre získanie používateľa.

S rozmachom webu rastú aj požiadavky na samotný návrh metód pre odporúčanie s dôrazom na škálovateľnosť či znovupoužitelnosť odporúčacích systémov. Problémom pri realizovaní personalizovaného odporúčania v reálnom čase je jeho výpočtová náročnosť na systémové zdroje.

Analyzujte existujúce prístupy ku personalizovanému odporúčaniu jednotlivcov a/alebo skupiny. Navrhnite metódu odporúčania schopnú generovať personalizované odporúčania v reálnom čase, so zameraním na vhodný softvérový návrh, architektúru a škálovateľnosť navrhnutej metódy. Metódu experimentálne overte na netriviálnych dátových vzorkách prostredníctvom vytvorenia prototypu, pričom preskúmajte kvalitatívne a kvantitatívne vlastnosti metódy.

---

[1] Vytlačiť obojstranne na jeden list papiera

[2] 150-200 slov (1200-1700 znakov), ktoré opisujú výskumný problém v kontexte súčasného stavu vrátane motivácie a smerov riešenia

Literatúra[3]

- Ricci, F., Rokach, L., Shapira, B., & Kantor, P. B. (2011). Recommender Systems Handbook. (F. Ricci, L. Rokach, B. Shapira, & P. B. Kantor, Eds.). Boston, MA: Springer US. doi:10.1007/978-0-387-85820-3
- Amatriain, X. (2013). Big & Personal: data and models behind Netflix recommendations. Proceedings of the 2nd International Workshop on Big Data, 1–6.

Vyššie je uvedený návrh diplomového projektu, ktorý vypracoval(a) Bc. Adam Lieskovský, konzultoval(a) a osvojil(a) si ho Ing. Michal Kompan, PhD. a súhlasí, že bude takýto projekt viesť v prípade, že bude pridelený tomuto študentovi.

V Bratislave dňa 4.2.2015

Podpis študenta

Podpis výskumníka

Vyjadrenie garanta predmetov Diplomový projekt I, II, III

Návrh zadania schválený: áno / ~~nie~~[4]

Dňa: 16. 2. 2015

Podpis garanta predmetov

[3] 2 vedecké zdroje, každý v samostatnej rubrike a s údajmi zodpovedajúcimi bibliografickým odkazom podľa normy STN ISO 690, ktoré sa viažu k téme zadania a preukazujú výskumnú povahu problému a jeho aktuálnosť (uveďte všetky potrebné údaje na identifikáciu zdroja, pričom uprednostnite vedecké príspevky v časopisoch a medzinárodných konferenciách)

[4] Nehodiace sa prečiarknite

Anotácia

Slovenská technická univerzita

Fakulta informatiky a informačných technológií

Študijný program: Softvérové inžinierstvo

Autor: Bc. Adam Lieskovský

Diplomová práca: Personalizované odporúčanie využitím škálovateľnej architektúry

Vedúci diplomovej práce: Ing. Michal Kompan, PhD.

Máj 2016

V práci sa zaoberáme témou personalizovaného odporúčania využitím škálovateľnej architektúry. Zameriavame sa na doménu novinových článkov, ktorá ako aj iné domény na Internete, požaduje aktuálne odporúčania generované v reálnom čase s nízkou odozvou. Práve personalizované odporúčanie je jeden z prostriedkov ako eliminovať informačnú zahltenosť používateľov a poskytnúť im prívetivejší zážitok pri používaní informačných systémov. Vďaka personalizovanému odporúčaniu vieme poskytnúť používateľom nie len informácie, ktoré hľadajú efektívnejšie a rýchlejšie, ale aj informácie, ku ktorým sa často bežnou navigáciou nedopracovali a môžu byť pre nich príjemne prekvapivé.

Analyzujeme existujúce prístupy, modely používateľa a architektúry, ktoré sú využívané pri škálovateľnom personalizovanom odporúčaní. Navrhujeme vlastnú hybridnú metódu odporúčania, zohľadňujúcu obsah novinových článkov, kontext a správanie používateľov. Výrazný vplyv v navrhovanej metóde má aj popularita a aktuálnosť novinových článkov. V navrhovanej metóde využívame škálovateľné algoritmy a poskytujeme flexibilný návrh systému, schopného adaptácie podľa aktuálneho zaťaženia. Metódu a overenie sme realizovali na platforme Plista ORP.

Annotation

Slovak University of Technology

Faculty of Informatics and Information Technologies

Degree Course: Software Engineering

Author: Bc. Adam Lieskovský

Master's thesis: Personalized scalable recommender system

Supervisor: Ing. Michal Kompan, PhD.

2016, May

Our thesis deals with personalized recommender systems and their scalability. Thesis focuses on the domain of news articles, which as any other popular domains on the Internet has to deal with real-time recommendations on a large scale. Personalized recommendations are one of the ways how to reduce information space and eliminate users' information overload. Recommendations can help users find desired information easier and more efficiently, which improves the overall user experience. Additionally, recommendations can lead to surprising or serendipitous information discoveries for users, which they would otherwise miss and never encounter with the regular navigation.

We analyze existing approaches to recommendation, user models and architectures of scalable recommender systems. We propose our own method of scalable hybrid recommender system, incorporating the content of news articles, users' context and behavior. Popularity, recency and trendiness of articles are also significant aspects which the method takes into account. We use scalable algorithms and propose design of a flexible recommender system capable of adapting to it's current workload. We implemented and evaluated proposed method on Plista ORP platform.

Obsah

1 Úvod

Problém informačnej zahltenosti a neustále rastúceho objemu dát je prítomný takmer vo všetkých doménach – napr. multimédia, novinové články alebo sociálne siete. Zaužívaným spôsobom ako odbremeniť používateľov informačných systémov je využitie odporúčaní. Inšpirácia pochádza zo situácií z bežného života. Napríklad pri konfrontácii s novou doménou, v ktorej máme minimum znalostí, sa často spoľahneme na ľudí, ktorí sú v danej oblasti zorientovaní a majú o nej hlbšie znalosti. Tento jav môžeme preniesť aj do oblasti informačných technológií, kde používatelia dôverujú správnemu odhadu odporúčacích systémov a spoliehajú sa na generované odporúčania. Pomocou nich vieme redukovať objem prezentovaných informácií, a tak uľahčiť a zefektívniť prácu používateľa so systémom.

Odporúčacie systémy sa zvyčajne vyskytujú v systémoch, ku ktorým paralelne pristupuje veľa používateľov a objem prehliadaneho obsahu je tak veľký, že je nutné ho zúžiť a personalizovať. Problémom pri realizovaní personalizovaného odporúčania v reálnom čase je jeho výpočtová náročnosť na systémové zdroje. V on-line systémoch, kde človek interaguje so systémom frekventovane za pomerne krátku dobu, je nutné reagovať na akcie používateľa nanajvýš v stovkách ms tak, aby používateľ systém predčasne neopustil z dôvodu frustrácie a pomalej odozvy. V jednom okamihu je nutné v reálnom čase spracovať veľké množstvo dát a požiadaviek prichádzajúcich z viacerých zdrojov. Jedným z riešení, ako urýchliť spracovanie týchto dát, je ich predpočítavanie na pozadí alebo v čase, keď systémy nie sú až tak zaťažené. Toto riešenie ale nie je využiteľné vo všetkých prípadoch a doménach, napr. pri dynamickom rýchlom raste správ na sociálnej sieti, kde v najvyťaženejšom momente systémy nestíhajú predpočítavať a zároveň aj pridávať obsah vo forme nových dát. Pokročilejším riešením je využitie paralelizácie a distribuovaných systémov, ktoré sú schopné rozdeliť výpočty medzi viacero uzlov v sieti. Podľa oblastí záujmu tieto systémy využívajú napríklad GPU, siete s architektúrou vzájomnej komunikácie klientov (angl. peer to peer) alebo dávkové spracovania dát sú len niektorými z nich.

Práve z týchto dôvodov je veľmi dôležitým aspektom odporúčacích systémov ich softvérový návrh, architektúra a aspekt škálovateľnosti, ktorý obšírne analyzujeme v rámci celej práce. V kapitole 2 analyzujeme prístupy ku personalizovanému odporúčaniu a modelovaniu používateľa. Zameriavame sa najmä na kolaboratívne a hybridné odporúčacie systémy, obohatenie odporúčaní o kontext a využitie spätnej väzby. V kapitole 3 diskutujeme o rôznych spôsoboch, ktorými je možné dosiahnuť škálovateľnosť systému. V kapitole 4 opíšeme existujúce škálovateľné odporúčacie systémy z komerčnej aj akademickej sféry. V kapitole 5 špecifikujeme ciele a očakávania našej práce na škálovateľnom personalizovanom odporúčacom systéme. Obsahom kapitoly 6 je návrh metódy hybridného personalizovaného odporúčacieho systému, ktorého realizáciu opisujeme v kapitole 7 a overujeme v kapitole 8. Zhodnotenie našej práce uvádzame v kapitole 9.

2 Personalizované odporúčanie

Veľké rozšírenie Webu a jeho používateľov so sebou prinieslo aj nárast popularity odporúčacích systémov. Ľudia nemajú dostatok času, znalostí alebo síl na to, aby porozumeli všetkým oblastiam, o ktoré sa zaujímajú v bežnom živote. Personalizované odporúčania zmenšujú inak rozsiahly informačný priestor používateľov a šetria ich čas. Jednoduchou cestou skráteného výberu pomáhajú používateľom vybrať si z veľkého množstva údajov tie, ktoré ich skutočne zaujímajú a reflektujú ich preferencie. Na odporúčacie systémy sa abstraktnejšie môžeme pozrieť ako na filtre informačného priestoru. Vďaka vzájomnej prepojenosti webových služieb nie je dnes náročné obohatiť používateľské profily z externých služieb a vytvoriť tak modely, ktoré presnejšie kopírujú používateľove záujmy.

Primárnou funkciou odporúčacieho systému je poskytnutie odporúčaní používateľom. Kľúčovou zložkou odporúčacích systémov je predikcia hodnotenia jednotlivých objektov používateľom resp. predikcia preferencií používateľa. V doméne odporúčacích systémov figurujú 3 množiny – používatelia, odporúčané objekty a akcie resp. transakcie. Odporúčacie systémy obvykle predikujú hodnotenie R používateľa $User$, ktoré prislúcha objektu $Item$.

$$R: User \times Item \rightarrow Rating$$

Dominantnými sa stali tri prístupy k odporúčaniu: prístup založený na obsahu, kolaboratívny prístup a hybridný prístup. V kapitole si postupne predstavíme jednotlivé prístupy, analyzujeme ich problémy, výhody a nevýhody. Tiež ukážeme ako je možné modelovať preferencie používateľa, zachytávať spätnú väzbu alebo vylepšiť proces odporúčania pridaním kontextu používateľa.

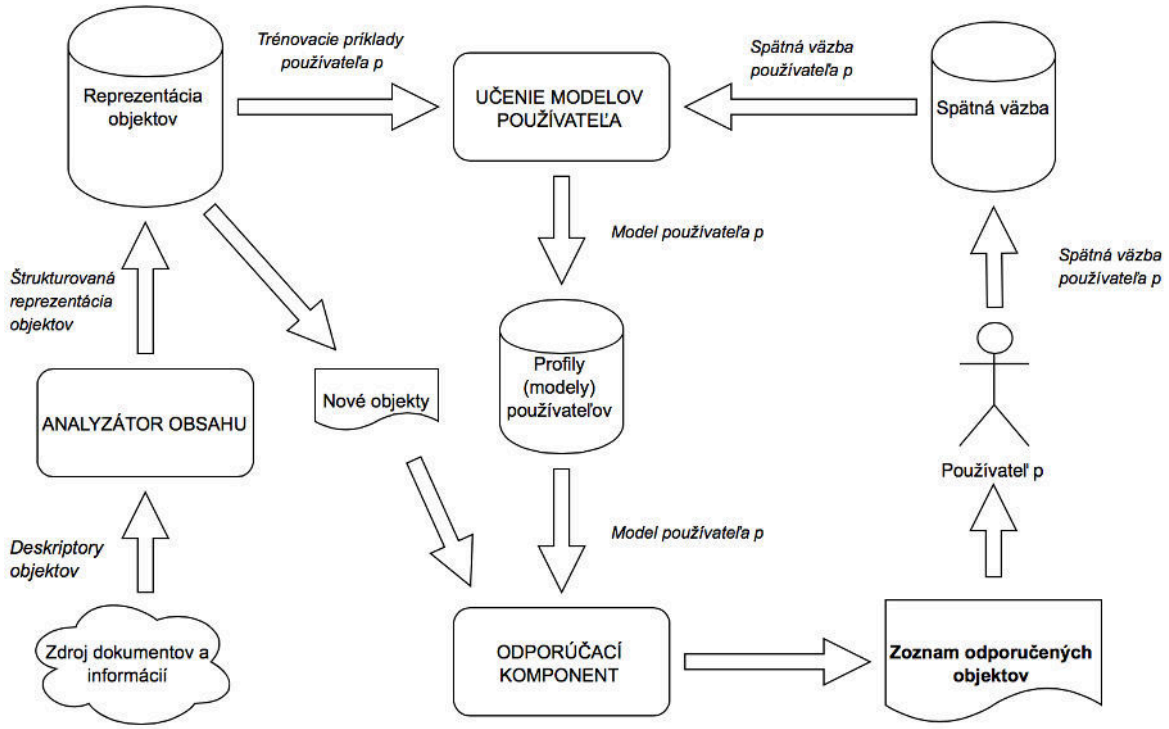
2.1 Odporúčanie založené na obsahu

Systém, ktorý odporúča objekty na základe podobnosti obsahu objektov, ktoré sa v minulosti používateľovi páčili, využíva odporúčanie založené na obsahu. Podobnosť medzi jednotlivými objektmi je určená na základe vlastností objektov. Tieto systémy analyzujú množinu dokumentov resp. objektov, za účelom vytvorenia vhodnej reprezentácie, modelujú preferencie používateľov na základe ich správania a hodnotenia. Následne hľadajú zhody v modeli používateľa s vlastnosťami odporúčaných objektov. Výskum v tejto oblasti využíva znalosti z oblastí vyhľadávania informácií a strojového učenia.

Tintarev et al. ponúkajú vysoko úrovňový pohľad na najčastejšiu architektúru odporúčacieho systému založeného na obsahu zobrazenú na Obr. 1 (Tintarev et al. 2011). V procese odporúčania opísaného vyššie identifikujú 3 hlavné komponenty:

- *Analyzátor obsahu* – dáta, ktoré zastupujú množinu potenciálne odporúčaných objektov, sú často uchovávané bez akejkoľvek štruktúry, a tak je nutné ich predspracovať a extrahovať relevantné informácie. Výstupom analyzátora je štruktúrovaná reprezentácia objektov, ktorá je získaná technikami extrahovania informácií, a s ktorou ďalej pracuje odporúčací komponent a komponent modelujúci používateľa.

- *Komponent modelujúci používateľa* – vstupom sú reprezentatívne dáta o používateľových preferenciách, ktoré sa snaží zovšeobecniť za účelom vytvorenia používateľského profilu. Využívajú sa tu techniky strojového učenia, ktoré na základe v minulosti ohodnotených objektov vytvárajú model používateľa.
- *Odporúčací komponent* – hlavný komponent generujúci odporúčania, na základe hľadania zhôd medzi modelom používateľa a množinou potenciálne odporúčaných objektov. Určením podobnosti medzi vektorovou reprezentáciou modelu používateľa a odporúčanými objektmi vytvára zoznam odporúčaných objektov.



Obr. 1 Architektúra odporúčacieho systému založeného na obsahu (Tintarev et al. 2011)

Odporúčané objekty sú reprezentované množinou vlastností, ktoré ich popisujú. Pri multimediálnom obsahu to môže byť autor, žáner alebo rok vydania. Výhodou je jednoduchá aplikácia techník strojového učenia vo fáze modelovania používateľa, ak majú objekty rovnakú reprezentáciu. Tieto objekty sú neskôr reprezentované formou termov resp. kľúčových slov, ktoré bez využitia sémantickej analýzy prinášajú problémy, pretože počítačové systémy ich porovnávajú iba ako textové reťazce. Najčastejšou reprezentáciou objektov je vektorový (angl. Vector Space Model) model s využitím TF-IDF váhovania, na ktorý nadväzuje kosínusová podobnosť pri výpočte podobnosti dokumentov resp. objektov. Modelovanie používateľa je realizované technikami strojového učenia, ktoré napr. pri textových dokumentoch vytvárajú klasifikátory objektov. Ďalej sú využívané pravdepodobnostné metódy, Bayesovské klasifikátory (Rendle et al. 2009; Zhang and Koren 2007), rozhodovacie stromy alebo algoritmy hľadania najbližších susedov (k-NN) (Tintarev et al. 2011; Pazzani and Billsus 2007).

Výhodou pri využití odporúčania založeného na obsahu je absolútna nezávislosť od hodnotení objektov inými používateľmi, pretože metódy využívané pri obsahovo založenom odporúčaní využívajú iba hodnotenia konkrétneho používateľa. Pri týchto systémoch sa

neobjavuje problém riedkych matíc hodnotení používateľov, ani problém studeného štartu nového objektu. Takisto systém dokáže odporučiť aj nepopulárne položky, alebo naplniť preferencie unikátnych a špecifických používateľov. Vďaka vhodnej reprezentácii vieme objasňovať dôvody, pre ktoré sme dané objekty odporučili. Naopak, problémom je studený štart nového používateľa, pre ktorého nemáme na začiatku dostatok hodnotení, aby sme poznali jeho preferencie. Nevýhodou pri odporúčaní založenom na obsahu je predpoklad možnosti extrakcie informácií a vlastností z objektov, čo nie je triviálna úloha, napr. v doméne multimédií. Príkladom môže byť popísanie obsahu videa na webe. Našťastie sú často k dispozícií metadáta (napr. autor, kategória), ktoré bližšie charakterizujú ich obsah. Okrem extrakcie informácií sa tento problém vyskytuje aj pri samotnej reprezentácií obsahu. Používateľov vkus je možné reprezentovať pomocou funkcie ceny (angl. cost function), ktorá pracuje s vlastnosti obsahu. Ak nevyužívame pokročilú previazanosť a sémantiku, pri tomto prístupe je náročné generovať prekvapivé odporúčania. Spomínaný problém je označovaný aj ako prílišná prispôsobivosť odporúčacích systémov (angl. overfitting), kedy systém odporúča objekty presne podobné preferenciám používateľa. Pri tvorbe odporúčacieho systému je často požadovaná aspoň minimálna doménová znalosť alebo kooperácia s doménovými expertmi.

2.2 Kolaboratívne filtrovanie

Úspešnejšou a rozšírenejšou technikou odporúčania je kolaboratívne filtrovanie, ktoré je založené na základnom predpoklade zdieľania záujmov a preferencií medzi používateľmi v budúcnosti, ak v minulosti vyjadril podobné preferencie (napr. hodnotením produktov v elektronickom obchode). Inak povedané predikcia hodnotenia a generovanie odporúčaní je založené na hodnoteniach a správaní používateľov v odporúčacom systéme. V kolaboratívnom prístupe generovania odporúčaní dominuje matica hodnotení – pozostávajúca z množiny používateľov, množiny hodnotených objektov a množinou hodnotení (aj predikovaných). Hodnotenia môžu odzrkadľovať okrem priameho hodnotenia na vymedzenej škále aj zachytenie určitej interakcie používateľa s objektom – kliknutia alebo otvorenia. Problémom je riedkosť matíc hodnotení, ktorú odporúčacie systémy prekonávajú redukciou vysokého počtu dimenzií využitím faktorizácie matíc alebo zhlukovania. Vyplýva z malého prekryvu spoločne hodnotených objektovo pri početných množinách používateľov a objektov.

Jedným z rozšírených delení kolaboratívnych odporúčacích systémov je ich rozdelenie na:

- *Odporúčacie systémy využívajúce model* - vytvárajú modely, ktoré sú zodpovedné za generovanie odporúčaní napr. formou klasifikátorov, faktorizácie matíc alebo neurónových sietí. Výhodou je ich škálovateľnosť (za cenu presnosti odporúčaní), jednoduchšia implementácia, jednoduché pridávanie nových objektov do systému a možnosť objasnenia odporúčaní. Hlavnou nevýhodou tohto prístupu je náročný proces tvorby modelu z pohľadu zdrojov, strata použiteľných informácií pri redukcií priestoru.
- *Odporúčacie systémy pracujúce s pamäťou* - pracujú s maticou hodnotení objektov používateľmi, využívajú všetky hodnotenia zaznamenané pred procesom generovania odporúčania (odporúčania sú vždy aktuálne). Pri odporúčaní sa určuje podobnosť medzi objektmi alebo používateľmi. Výhodou je ich relatívne vyššia presnosť a ľahšie dosiahnuteľná aktuálnosť odporúčaní. Problémom pri čisto

pamäťovom prístupe je slabá škálovateľnosť pri riedkych maticiach hodnotení veľkých dátových množín, pri pridaní nového objektu do systému, je nutné prepočítanie odporúčaní.

Odporúčacie systémy pracujúce s pamäťou môžeme ďalej rozdeliť na dve skupiny, a to: založené na *podobnosti používateľov* a založené na *podobnosti objektov*, s ktorými používateľ interagoval. Prvou metódou v kolaboratívnom odporúčaní bolo odporúčanie založené na podobnosti používateľov, využitím algoritmu hľadania k – najbližších susedov – k-NN (Ekstrand 2010). Algoritmus k-NN je možné implementovať jednoducho, dosahuje dostatočne presné odporúčania a tak sa aj v dnešných časoch teší veľkej obľube. Bližší pohľad prinášame v kapitole 2.2.1. Pri rozhodovaní medzi metódami využívajúcimi podobnosť používateľov alebo objektov, je treba zohľadniť viacero aspektov, medzi ktoré patria: prekvapivosť, vysvetliteľnosť, presnosť, efektívnosť a stabilita odporúčaní (Desrosiers and Karypis 2011).

V tradičnom kolaboratívnom odporúčaní na základe podobnosti používateľov sú používatelia reprezentovaní N dimenzionálnym vektorom objektov, kde N je počet objektov v systéme. Jednotlivé zložky vektora potom predstavujú vzťah resp. hodnotenia objektov používateľmi – pozitívne alebo negatívne. Aby systém neodporúčal len globálne najpopulárnejšie objekty, je obvyklé využiť metódu IDF, ktorá zníži váhu pozitívnych hodnotení pri najpopulárnejších objektoch, a menej rozšírené objekty sa tak stanú relevantnejšími. Výsledný vektor je pri tomto spôsobe u väčšiny používateľov príliš riedky. Pri generovaní odporúčaní systém určí najpodobnejších zákazníkov, na základe podobnosti hodnotení jednotlivých objektov, často využitím napr. metriky *kosínusovej podobnosti* (alebo zhlukovaním). Pri využití tohto postupu je výpočet veľmi náročný, aj pri zohľadnení riedkych vektorov používateľov je v priemere ohraničiteľný ako $O(m + n)$, kde n je počet objektov a m je počet používateľov. Vyplývajúci problém slabej škálovateľnosti je možné riešiť technikami redukcie dimenzionality priestoru alebo redukovaním veľkosti dátovej množiny (náhodný výber určitého počtu používateľov). Techniky maticovej faktorizácie, ktoré redukujú počet dimenzií pri odporúčaniach sú veľmi populárne a my ich bližšie analyzujeme v kapitole 2.2.2. Pri využití zhlukovania systém zhlukuje používateľov do segmentov, v ktorých sa nachádzajú používatelia s podobnými preferenciami. Výhodou zhlukovania je jeho škálovateľnosť a výkon, pretože náročnejšiu časť výpočtu je možné realizovať off-line. Nevýhodou je nižšia kvalita odporúčaní.

Ďalším spôsobom ako hľadať podobných používateľov na základe ich doterajších preferencií, je preniesť túto úlohu na hľadanie podobnosti medzi objektami, s ktorými interagovali jednotliví používatelia. Prístup odporúčania založeného na podobnosti objektov sa najprv objavil v akademickej sfére v systéme GroupLens odporúčacieho novinové články (Rucker et al. 1997), pričom v komerčnej sfére bol spopularizovaný americkým gigantom Amazon (Linden et al. 2003). Spoločnosť Amazon personalizuje obsah elektronického obchodu, emailov alebo webových stránok pre každého používateľa. Namiesto toho aby odporúčacie systémy Amazonu určovali podobných používateľov pre konkrétneho používateľa, Amazon využíva porovnávania hodnotených a nakúpených objektov jednotlivých používateľov k iným objektom v systéme. Tento postup veľmi pomáha škálovateľnosti odporúčacieho systému, ktorú analyzujeme v kapitole 4.1.3.

Ak sa vrátime naspäť ku výzvam a problémom kolaboratívneho prístupu, musíme spomenúť aj problém tzv. ovcí, a to čiernej – používateľ, ktorý je unikátny resp. výstredný, nesúhlasí

svojimi preferenciami so žiadnymi inými, a šedej – používateľ, ktorému systém konzistentne nedokáže nájsť podobných používateľov (Su and Khoshgoftaar 2009; Shi et al. 2014a). Nutné je budovať aj mechanizmy ochrany pred manipuláciou hodnotením (angl. shilling attack) – pridávaním enormného množstva pozitívnych hodnotení k vybranému objektu, a zároveň poškodzovanie konkurencie formou negatívnych hodnotení. V určitých metódach sa vyskytuje aj problém dlhého chvosta a prehnanej preferencie populárnych objektov (Cremonesi et al. 2012).

Kolaboratívny prístup bol využitý aj v distribuovaných a paralelných systémoch (Schelter et al. 2013; Zhang et al. 2011; Han et al. 2004), kde je výhodou škálovateľnosť systémov, rýchlejšia odozva systémov, ktoré pracujú s veľkými dátovými množinami. Je tu priestor využiť viacero algoritmov simultánne, vyššia stabilita systémov. Karydi a Margaritas analyzovali prístupy využívajúce paralelné modely OpenMP a MPI, využitie grafických výpočtových jednotiek (GPU) pri kolaboratívnych systémoch (Karydi and Margaritis 2014). Pre odlišnú architektúru GPU tieto systémy prinášajú obrovské zrýchlenia výpočtov, ak sú realizované cez maticovo vektorové operácie. Pri využití GPU je preferovaná technológia CUDA [1] spolu s prístupmi využívajúcimi modely – najčastejšie SVD a SGD. Paralelné a distribuované systémy prinášajú implementačnú záťaž, kedy ju nutné riešiť prístup do pamäte, synchronizáciu a súperenie o zdroje. Bližší pohľad na distribuované odporúčacie systémy a ich škálovateľnosť prinášame v kapitole 3.

Shi et al. ponúkajú sumarizáciu ďalších možností a existujúcich prác, ktoré okrem priamych hodnotení využívajú aj informácie zo sociálnych sietí, tagy, geolokačné informácie, multimediálny obsah, recenzie alebo komentáre (Shi et al. 2014a). Spôsoby využitia sociálnych sietí za účelom zvýšenia presnosti odporúčaní pri kolaboratívnom prístupe skúmali Yang et. al, ktorí upozorňujú na problémy späté so súkromím používateľov a preferenciou off-line prístupu trénovania modelu, kedy systémy neobsahujú najaktuálnejšie informácie o používateľoch (Yang et al. 2013). Okrem populárnych a overených prístupov ku kolaboratívnemu odporúčaniu, ktoré bližšie analyzujeme v najbližších podkapitolách existujú aj odporúčacie systémy využívajúce napr. grafové alebo genetické algoritmy, hĺbkové učenie neurónových sietí alebo faktorizáciu tenzorov (Bobadilla et al. 2013; Shi et al. 2014a; Deshpande and Chitrakant 2014; Demovic et al. 2013).

2.2.1 Algoritmy hľadajúce najbližších susedov

Najpopulárnejšou metódou z kategórie kolaboratívneho odporúčania využívajúceho pamäť boli dlhodobo odporúčania pomocou hľadania najbližších susedov. Medzi výhody patrí jednoduchá implementácia a efektívnosť – bez zdĺhavých fáz trénovania modelov. Práve tento prístup je analogicky najpodobnejší bežnému životu, kde sa spoliehame na rady a odporúčanie našich známych alebo priateľov.

Pri realizácii metódy hľadania najbližších susedov je nutné definovať, či využijeme prístup založený na podobnosti objektov alebo používateľov, a či využijeme regresnú alebo klasifikačnú metódu. Pri diskrétnych hodnoteniach typu páči sa mi a nepáči sa mi, je vhodnejšie použiť klasifikačnú metódu. Ak pracujeme so širšou škálou číselných hodnotení – napr. ohodnotenie od 0 do 10 regresná metóda môže byť vhodnejšia (Tintarev et al. 2011).

[1] http://www.nvidia.com/object/cuda_home_new.html

Pri kolaboratívnom odporúčaní na základe podobnosti používateľov predikujeme hodnotenie objektu i používateľom p r_{ui}. Využívame hodnotenia objektu i používateľmi, ktorí sú najpodobnejší používateľovi u. Týchto používateľov nazývame najbližšími susedmi. Potom rovnica

$$\hat{r}_{ui} = h^{-1}\left(\frac{\sum_{v \in N_i(u)} w_{uv} h(r_{vi})}{\sum_{v \in N_i(u)} |w_{uv}|}\right)$$

vyjadruje predikované hodnotenie $\hat{r}_{ui}$ využitím normalizácie hodnotení $h(r_{vi})$ susedov používateľa N_i s ováhovanými podobnosťami w_{uv} medzi používateľom u a jeho susedom v (Tintarev et al. 2011).

Algoritmus hľadania najbližších susedov pozostáva z 3 častí, ktorými sú normalizácia hodnotení používateľov, výpočet ováhovaných podobností a selekcia susedov.

Normalizácia hodnotení

Tak ako môže každý človek ohodnotiť jednotlivé objekty rôzne, rovnako má aj každý používateľ svoju vlastnú škálu hodnotení. Niektorí používatelia môžu pri päť hviezdičkovej hodnotiacej škále udeľovať hodnotenie 5 (najlepší) všetkým objektom, ktoré ich zaujali, zatiaľ čo iní používatelia udelia rovnaké hodnotenia len pár objektom za ich celú dobu pôsobenia v odporúčacom systéme. Vznikajú tak rozdiely v hodnoteniach, a na to aby sme odstránili subjektívny prvok hodnotiacej škály musia byť používateľské hodnotenia normalizované. V praxi sú najpoužívanejšie 2 postupy: *z-hodnotenie* (angl. Z-score) a *Centrovanie na stred* (angl. mean-centering).

Myšlienka pri metóde *centrovania na stred* je jednoduchá – zisti, či je hodnotenie pozitívne alebo negatívne pri porovnaní s priemerným hodnotením konkrétneho používateľa. V opise metód uvažujme odporúčania založené na podobnosti používateľov. Pri centrovaní na stred vieme transformovať hodnotenie používateľa r_{ui} na normalizované hodnotenie $h(r_{ui})$ odčítaním priemerného hodnotenia všetkých objektov $\overline{r_u}$ od hodnotenia r_{ui}.

$$h(r_{ui}) = r_{ui} - \overline{r_u}$$

kde ako $\overline{r_i}$ figurujú priemerné hodnotenia objektu i všetkých používateľov v systéme. Výhodou centrovania na stred je, že priamo vidíme používateľovu polaritu vzťahu k objektu resp. či je pozitívny alebo negatívny. Pri využití metódy *z-hodnotenia* okrem odstránenia rôzneho vnímania priemernej hodnoty hodnotenia používateľmi pracujeme aj s rozdelením hodnotení používateľa na hodnotiacej škále. Do rovnice tak vstupuje smerodajná odchýlka hodnotení používateľa u označovaná σ_u.

$$h(r_{ui}) = \frac{r_{ui} - \overline{r_u}}{\sigma_u}$$

Desroisiers a Karypis uvádzajú v (Desrosiers and Karypis 2011) aj hraničné prípady, kedy môže mať normalizácia hodnotení negatívne alebo neočakávané účinky. Sú to napr. situácie, kedy máme k dispozícii pre používateľa málo hodnotení, alebo väčšina z nich je príliš pozitívna.

Výpočet váh susedov

Výpočet váh susedov, resp. ich vyplzvu poskytuje možnosť priradiť váhu vybraným susedom pri predikcii hodnotenia objektu. Výpočet ováhovaných podobností je kritický pre algoritmus hľadania najbližších susedov, pretože vplýva na presnosť a výkon odporúčacieho systému. V kapitole kolaboratívneho odporúčania sme si už predstavili kosínusovú podobnosť. Zaužívanou metódou výpočtu podobnosti je aj *Pearsonovou korelačný koeficient*. Pri zohľadnení a eliminácii vplyvu globálneho priemerného hodnotenia a rozložení hodnotení je možné podobnosť medzi používateľom u a susedom v pomocou *Pearsonovho koeficientu* vypočítať nasledovne:

$$sim(u,v) = \frac{\sum_{i \epsilon I_{uv}} (r_{ui} - \overline{r_u})(r_{vi} - \overline{r_v})}{\sqrt{\sum_{i \epsilon I_{uv}} (r_{ui} - \overline{r_u})^2 \sum_{i \epsilon I_{uv}} (r_{vi} - \overline{r_v})^2}}$$

kde I_{uv} predstavuje zjednotene množinu objektov hodotenú používateľmi u a v. Do úvahy je vhodné brať silu ováhovaných podobností, kde je časté znížiť váhu podobností, ak vznikla výpočtom malej vzorky spoločných hodnotení. Druhým aspektom, ktorý umožňuje zvýšiť presnosť odporúčaní pri váhovaní podobností je ich variácia resp. rozloženie hodnotení, kedy upravujeme variáciu o frekvenciu výskytov vybraných hodnotení metódou IDF.

Selekcia susedov

Kľúčovou časťou algoritmu je aj počet k najbližších susedov a podmienky, podľa ktorých ich algoritmus vyberie. Proces selekcie je obvykle realizovaný v dvoch fázach:

1. Všeobecný krok výberu resp. pred filtrovanie najpravdepodobnejších susedov

2. Výber susedov pre konkrétne odporúčanie jednotlivých objektov.

Predfiltrovanie kandidátov na k –najbližších susedov je nutné realizovať najmä kvoli veľkosti množiny používateľov alebo objektov v systéme, pretože nie je možné pracovať so všetkými dátami v pamäti. Spracovanie všetkých používateľov v systéme je pri algoritme hľadania k najbližších susedov zbytočné, pretože do úvahy sa berie len množina najpodobnejších používateľov. Tento krok je realizovaný off-line a v určitých časových intervaloch býva množina kandidátov aktualizovaná. Na predfiltrovanie je možné využiť niektorú z 3 známych metód, prípadne ich kombinácie:

- *Top-N filtrovanie* – pre každého používateľa (alebo objekt) eviduje zoznam iba N najbližších susedov aj s ováhovaním podobnosti.

- *Filtrovanie hranicou* – výber všetkých najbližších susedov, ktorých ováhovaná podobnosť presahuje určitú hranicu. Jedná sa o flexibilnejší prístup, ale aj tu je problém správneho nastavenia hranice výberu.

- *Negatívne filtrovanie* – odstraňovanie vybratých susedov, ktorý spadajú do vopred vymedzených kritérií, napr. príslušnosť k inej vekovej skupine.

Po vybratí kandidátov na najbližších susedov nasleduje finálny krok výberu, a to určenie k – *najbližších* z nich, ktorí majú najvyššie váhy podobností. Otázkou zostáva ako určiť

parameter k tak, aby sme dosiahli čo najvyššiu presnosť odporúčaní. Parametrizovaním a aj hľadaním optimálneho k v algoritme k – najbližších susedov sa zaoberali Batista a Silva, kde pri evaluácií na 31 datasetoch vyhodnotil ako najlepšie parametre pre k 5 a 11 (Batista and Silva 2009).

Hlavným problémom pri využití algoritmov hľadania najbližších susedov v on-line odporúčacích systémoch je ich náročný a slabo škálovateľný proces výpočtu. Využitím hybridného algoritmu ClustKnn na rozhraní prístupov využívajúcich model a pamäť je realizovať výpočtové náročné operácie off-line, pomocou parametrizovateľného zhlukovania (parametrom je možné určiť pomery medzi presnosťou alebo časovou efektívnosťou) (Rashid et al. 2006). Na zhlukovaním vytvorené modely Rashid et. al aplikujú štandardnú metódu hľadania najbližších susedov, využívajúcu Pearsonov korelačný koeficient, čím redukujú on-line zložku výpočtu z obvyklých $O(mn)$ na $O(m)$. m predstavuje veľkosť množiny odporúčaných objektov a n veľkosť množiny používateľov odporúčacieho systému.

Jedným z riešení slabej škálovateľnosti prístupu hľadania najbližších susedov je distribúcia výpočtov na viacero strojov a delenie dát na partície (Formoso et al. 2014). Distribúciou dát na jednotlivé stroje a pridaním invertovaných indexov dokázali zvýšiť škálovateľnosť tohto prístupu, odozvu a efektivitu odporúčacieho systému. Jednotlivé stroje pracovali s podmnožinou profilov používateľov, čo sa pri štandardnom objeme hodnotení objektov ukázalo ako efektívnejší prístup, pri porovnaní s distribúciou na základe objektov v odporúčacom systéme.

Pri tvorbe grafov najbližších susedov Park et. al využili postup tzv. nenásytného odporúčania (angl. greedy filtering), ktorým vytvorili efektívny a škálovateľný systém, dosahujúci až 6 násobné zrýchlenie voči ustáleným metódam hľadania najbližších susedov pri veľkých dátových množinách (Park et al. 2013). Ich metóda sa spolieha prefixovú kontrolu podobnosti vektorov používateľov, vďaka ktorej je možné vytvoriť aproximatívne grafy reprezentácie najbližších susedov.

2.2.2 Metódy využívajúce latentné vlastnosti

Tieto metódy sa snažia odhaliť a využiť latentné vlastnosti resp. identifikovať vzory správania, ktoré vedú k zaznamenaným hodnoteniam objektov používateľmi v odporúčacích systémoch. Spadajú sem metódy využívajúce neurónové siete, Latentnú Dirichletovu Alokáciu (Lin et al. 2013; Chen 2010), PLSA alebo faktorizácie matíc. Masové rozšírenie využitia prístupov na redukovanie dimenzií (Sarwar et al. 2000) v dátových množinách priniesla súťaž z roku 2006 o 1 milión dolárov organizovaná spoločnosťou Netflix (Bell and Koren 2007). Spopularizovala najmä metódy faktorizácie matíc a ukázala ich presnosť a škálovateľnosť.

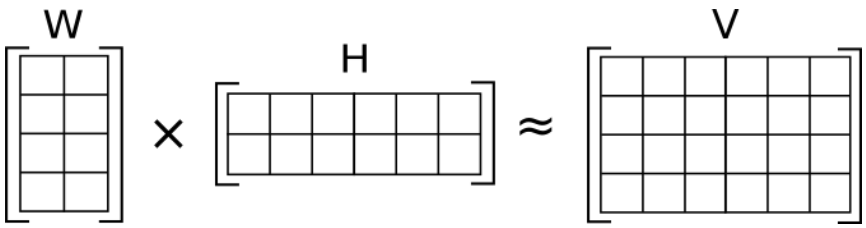
Faktorizácia matíc

Faktorizácia matíc je technika rozkladania a redukcie dimenzie matíc definovaná v lineárnej algebre. V doméne odporúčacích systémov faktorizácie matíc modelujú používateľov a objekty do spoločného priestoru, kde sú interakcie používateľov s objektmi modelované ako ich produkt. Vznikajú tak vektory jednotlivých objektov q_i a používateľov p_u. Pre jednotlivé objekty alebo používateľov zložky vektorov definujú nakoľko napĺňa objekt alebo používateľ zložky vektora – latentné faktory, či už pozitívne alebo negatívne. Výsledný

skalárny súčin týchto dvoch vektorov odráža používateľov záujem o vlastnosti konkrétneho objektu i a predikuje hodnotenie používateľa $\hat{r}_{ui}$.

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

Zaujímavé je pozrieť sa práve na to, ako je možné definovať latentné vlastnosti interakcie používateľov s objektmi. V doméne vyhľadávania informácií je často využívaná metóda SVD (angl. Singular Value Decomposition), ktorá je frekventovaná aj kvôli svojej škálovateľnosti (Sarwar et al. 2002; Spiegel et al. 2009). Na Obr. 2 je znázornená dekompozícia matice V, na dve menšie matice W a H, ktorých súčinom je možné získať naspäť pôvodnú maticu V.



Obr. 2 Faktorizácia Matice V

SVD (angl. Singular Value Decomposition)

Pri využití metódy SVD v súťaži Netflix Koren et. al obohatili všeobecnú definíciu z úvodu kapitoly faktorizácie matíc o pridanie prvkov predikujúcich hodnotenie objektov, závislé čisto na používateľovi alebo objekte, a teda nie na vzájomnej interakcii (Koren et al. 2009). Použitá funkcia predikcie hodnotenia $\hat{r}_{ui}$ je tak obohatená o základnú predikciu hodnotení formou využitia priemerných hodnotení.

$$\hat{r}_{ui} = \mu + b_i + b_u + q_i^T p_u$$

Na to aby sa model naučil predikovať zložky b_u, b_i, q_i a p_u minimalizujú chybu hodnotení nad už známymi trénovacími hodnoteniami. Cieľom systému je ale predikovať nové hodnotenia a tak je minimalizácia chyby doplnená o regularizačný parameter α.

$$min \sum_{(u,i)\epsilon K} (r_{ui} - \mu - b_i - b_u)^2 + \alpha_4 (\, {b_u}^2 + {b_i}^2 + \| \, q_i\|^2 + \|p_u\|^2)$$

Na samotnú minimalizáciu sa často využíva buď algoritmus SGD (angl. Stochastic gradient descent) alebo ALS (angl. alternating least squares). SGD je populárnejšou voľbou z dôvodu relatívnej rýchlosti, efektivity a nenáročnosti na implementáciu. Algoritmus SGD prechádza všetky hodnotenia v trénovacej množine dát, pričom pre každé hodnotenie r_{ui} vypočíta predikované hodnotenie $\hat{r}_{ui}$ a chybu predikcie $e_{ui} = r_{ui} - \hat{r}_{ui}$. Následne algoritmus modifikuje parametre posunom k dotyčnici derivácie:

- $b_u \leftarrow b_\mathrm{u} + \gamma \cdot (e_{ui} - \alpha_4 \cdot b_u)$
- $b_u \leftarrow b_i + \gamma \cdot (e_{ui} - \alpha_4 \cdot b_i)$
- $q_i \leftarrow q_i + \gamma \cdot (e_{ui} \cdot p_u - \alpha_4 \cdot q_i)$

- $p_u \leftarrow p_u + \gamma \cdot (e_{ui} \cdot q_i - \alpha_4 \cdot p_u)$

Výhodou použitia ALS je možnosť jednoduchšej paralelizácie výpočtu, pretože pracuje oddelene s jednotlivými vektormi objektov, prípadne používateľov. ALS je vhodné použiť aj v systémoch pracujúcich s implicitnou spätnou väzbou, kde sú matice interakcií riedke a algoritmus SGD prechádzajúci všetky latentné vlastnosti nie je vhodné použiť (Koren et al. 2009).

SVD++

Koren neskôr vylepšil svoju pôvodnú metódu SVD o zapracovanie implicitnej spätnej väzby, ktorú je ľahšie získať od používateľa ako explicitnú (Koren 2008). SVD ++ využíva analogický postup k metóde SVD, ale funkcia predikujúca hodnotenie je doplnená o zložku interpretujúcu normalizovanú implicitnú spätnú väzbu $|R(u)|^{-\frac{1}{2}} \sum_{j \epsilon R(u)} y_j$.

$$\hat{r}_{ui} = \mu + b_i + b_u + q_i^T (p_u + |R(u)|^{-\frac{1}{2}} \sum_{j \epsilon R(u)} y_j)$$

Metódy využívajúce SVD sú ľahko rozšíriteľné o funkciu modelovaniu času a dynamických zmien modelu používateľa v priebehu času (Koren et al. 2009), čo sa ukazuje ako veľmi efektívne zlepšenie kvality resp. presnosti odporúčaní.

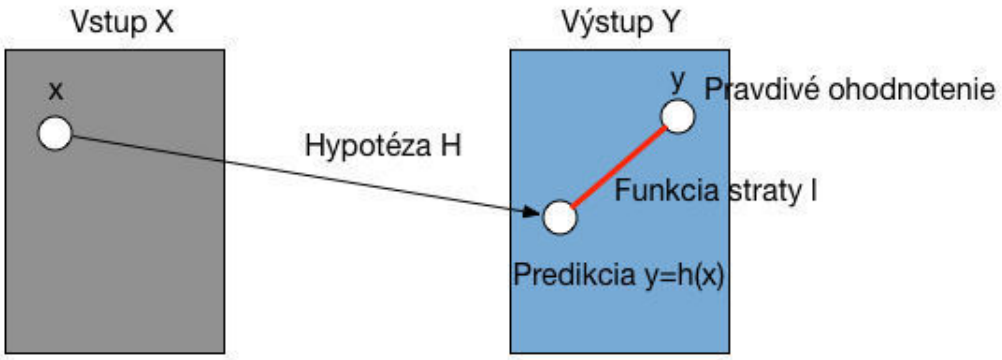
2.2.3 Learning to rank

Predchádzajúce prístupy pri odporúčaniach sa snažili čo najpresnejšie predikovať hodnotenia objektov používateľmi. Alternatívny postoj k odporúčaniu pochádza z oblasti vyhľadávania informácií, kde je cieľom usporiadať objekty v takom poradí, aby ich používateľ čo najrýchlejšie dokázal zúžitkovať. Alternatívny pohľad na funkciu, resp. cieľ odporúčacieho systému je nájdenie najlepšieho možného usporiadania množiny objektov pre jednotlivých používateľov. Učenie usporadúvania je technika strojového učenia s učiteľom (angl. supervised learning), čiže je nutná prítomnosť trénovacej dátovej množiny.

Pri určovaní kvality a efektívnosti sa na vyhodnocovanie používajú iné metriky ako pri predchádzajúcich prístupoch. Populárnymi sa stali MRR (angl. Mean reciprocal rank), MAP (angl. Mean average rank) alebo (N)DCG (angl. (Normalized) Discontinued cumulative gain). Spoločnou črtou metrík je explicitné použitie pozície usporiadania. Normalizovné DCG (NDCG), využíva normalizáciu parametrom Z_k (maximálna dosiahnuteľná hodnota) a dosahuje hodnoty z rozmedzia 0 až 1 Pri evaluácií k-tej pozície zoradených objektov π ju vieme vypočítať pomocou nasledujúceho vzťahu, kde $g()$ predstavuje hodnotenie objektu a $\eta(j)$ je logaritmický parameter znevýhodňujúci klesajúcu pozíciu:

$$NDCG@k(\pi, l) = \frac{1}{Z_k} \sum_{j=1}^{k} G(l_\pi - 1_{(j)}) \eta(j)$$

NDCG je často využívaná priamo aj pri minimalizácií funkcií straty (angl. loss function), resp. systémy sa práve snažia túto metriku optimalizovať (Sun et al. 2012; Balakrishnan and Chopra 2012). Existujúce prístupy metódy učenia sa usporadúvania je možné rozdeliť do troch kategórií. Kritériom delenia sú 4 základne komponenty techník strojového učenia využívaných pri metóde learning to rank, viditeľné na Obr. 3. Kľúčovými komponentmi sú priestor vstupu, priestor výstupu, priestor hypotézy a funkcia straty (angl. loss function).



Obr. 3 Komponenty strojového učenia metódy learning to rank

Bodové (angl. pointwise)

Pri bodovom prístupe môžeme hovoriť o redukovaní problému zoraďovania objektov na regresiu jednotlivých objektov, kde odhadujeme skóre objektu. Do učenia vstupujú jednotlivé vektory vlastností objektov, pričom výstupom je relevancia, určená pre každý dokument zvlášť. Tento prístup je veľmi podobný k využitiu spätnej väzby k relevancii. Bodový prístup nezohľadňuje závislosti medzi dokumentmi, a tak je finálna pozícia neviditeľná pre funkciu straty. Bodový prístup je podľa použitých algoritmov možné deliť na algoritmy založené na regresii, klasifikácii alebo ordinálnej regresii. Zástupcom kategórie využívajúci ordinálny bodový prístup pri kolaboratívnom filtrovaní je rámec OrdRec, v ktorom Koren a Sill predikovali distribúciu hodnotení objektov nie priamou s hodnotou, ale jej pozíciou v zoradenom zozname hodnotení (Koren and Sill 2011).

Párové (angl. pairwise)

Pokročilejší a populárnejší je párový princíp, v ktorom algoritmy nepredikujú stupeň relevancie jednotlivých objektov, ale určujú relatívne poradie medzi pármi objektov s ohľadom na predikované hodnotenie používateľa. V párovom prístupe algoritmy definujú, ktorý z dvojice objektov je pre používateľa relevantnejší, resp. ktorý ohodnotí vyšším hodnotením. Cieľom pri párovom prístupe je minimalizovať množstvo nesprávne predikovaných resp. klasifikovaných dvojíc dokumentov. Populárnymi sa stali algoritmy využívajúce neurónové siete, perceptróny, podporovanie (angl. boosting) alebo SVM, akými sú napr. Frank, AdaBoost, RankSVM (Chapelle and Chang 2011). V kolaboratívnom odporúčaní bol párový prístup prvý raz využitý v (Pessiot et al. 2004). Použili funkciu straty, ktorá brala do úvahový párové preferencie medzi jednotlivými odporúčanými dvojicami objektov resp. minimalizovali klesajúci gradient. Trénovanie modelu realizovali offline so zložitosťou $O(npvk)$, a generovanie samotných odporúčaní on-line s $O(p(k+h))$, pričom $n-$ počet používateľov, $p-$počet párov objektov, $v-$ veľkosť množiny možných hodnotení, $k-$ počet dimenzií objektov, $h-$ top h najlepších odporúčaní, čím dosiahli lineárnu zložitosť a škálovateľnosť metódy. Ich evaluácie priniesli v metrike MRE, v porovnaní s vtedy štandardnými metódami kolaboratívneho odporúčania (SVD), nečakane neuspokojivé výsledky, ktoré navrhovali vylepšiť normalizáciou a regularizáciou. Algoritmus RankSVM využili v (Sun et al. 2012), kde využili metódu learning to rank prvýkrát v hybridnom odporúčaní. Použili metódu $TFI-DF$ na reprezentáciu používateľov a objektov vo vektorovej forme a minimalizovali metriku NDCG. Hybridnou kombináciou dokázali na

štandardnej dátovej množine Movielens[2] prekonať jednotlivé metódy alebo iné learning to rank algoritmy takmer o 8%.

Zoznamové (angl. listwise)

Najpokročilejšou metódou je zoznamový prístup, ktorého vstupom je celá množina objektov a algoritmus predikuje ich tzv. pravdivé ohodnotenie (angl. ground truth). Podľa použitej funkcie straty je možné rozdeliť tento prístup do dvoch kategórií, a to postupy *priamo optimalizujúce evaluačné metriky* a postupy, ktoré *neoptimalizujú funkciu straty na základe priameho vzťahu s evaluačnými metrikami*. Populárnymi metódami sú napr. RankCosine, alebo Listnet (Shukla et al. 2012; Xia et al. 2008; Niek 2014).

Zrýchlením procesu učenia algoritmu Listnet použitím technológie Apache Spark sa zaoberali (Shukla et al. 2012), ktorí pomocou paralelizácie jednotlivých iterácií klesajúceho gradientu zefektívnili proces učenia sa Listnet algoritmu. Distribúciou procesu učenia algoritmu ListNet realizoval (Niek 2014). Niek využil programovací model MapReduce, ktorý priniesol očakávané zlepšenie pri obrovských dátových množinách. Pri malých dátových množinách sa samotná réžia a distribúcia výpočtov resp. úloh ukázala ako spomaľujúci prvok celého systému. Obe práce preukazovali lineárnu závislosť od rastu množiny dát. Môžeme povedať, že paralelizácia alebo distribúcia výpočtu pri prístupe learning to rank, je perspektívnou metódou, ktorá spĺňa priame požiadavky škálovateľnosti odporúčacích systémov.

2.3 Hybridný prístup

V predchádzajúcich kapitolách sme predstavili metódy kolaboratívneho odporúčania a odporúčania založenom na obsahu. Okrem týchto prístupov, s ktorými sú späté ich nevýhody a výhody, existujú aj ďalšie metódy tvorby odporúčaní - napr. demografické, komunitné alebo znalostné. Správnym krokom sa ukazuje kombinácia viacerých prístupov, kde je možné eliminovať nedostatky jednotlivých metód a vytvoriť tak úspešnejšiu metódu odporúčania (Bell and Koren 2007).

Odporúčacie systémy kombinujúce viacero prístupov klasifikujeme ako *hybridné odporúčacie systémy*. Kombinované metódy nemusia byť nutne viacerých druhov. Tieto systémy je možné kategorizovať do 7 nasledujúcich kategórii (Kaššák 2014; Burke 2007). Obaja autori prízvukujú dôslednú analýzu a pozorovanie konkrétnej domény, za účelom výberu najvhodnejšieho druhu hybridného odporúčacieho systému, ako aj funkcie jednotlivých komponentov odporúčania v hybridnom systéme. Nižšie uvádzame zoznam kategórií s opisom metódy, príkladmi existujúcich systémov a výsledkov Burkeho evaluácií systému odporúčajúceho reštaurácie realizovanom na dátovej množine 280 000 hodnotení.

2.3.1 Váhované

Burke považuje váhovanú metódu za najjednoduchšiu a najuniverzálnejšiu. Lineárnou kombináciou viacerých hodnotení (ováhovaním týchto hodnotení) objektov pochádzajúcich z rôznych odporúčacích metód vzniká váhovaná metóda odporúčania. Pri výbere statickej lineárnej váhovanej rovnice sa obvykle uvažuje o zjednotení alebo prieseku odporúčaných objektov. Pri Burkeho evaluácii sa ukázalo empirické nastavenie váh jednotlivých metód ako

[2] https://movielens.org/

nečakane neefektívne a iba v 1/3 prípadov prinieslo zlepšenie oproti použitiu jednotlivých metód samostatne. Ukážkovým príkladom je systém P-Tango, využívajúci priemer ováhovaných metód, ktorý odporúčal novinové články v denníkoch Worcester Telegram a Gazette Online (Claypool et al. 1999). Váhy v systéme P-Tango sú dynamicky určované na báze používateľa, s možnosťou ich definície aj na úrovni objektov. Modely využívané pri kombinácií kolaboratívneho prístupu a odporúčaní na základe obsahu sú udržiavané separátne, čo umožňuje využitie špecifických vylepšení jednotlivých metód. Claypool et. al pre nového používateľa definujú rovnakú váhu jednotlivých modelov, ktorá je neskôr optimalizovaná znižovaním chyby predikcie hodnotení.

2.3.2 Zmiešané

Zmiešaný prístup pozostáva zo zostavovania zoznamu odporúčaných objektovo, ktoré sú vygenerované rôznymi odporúčacími metódami do jedného unifikovaného zoznamu. Otázne je, ako tieto odporúčania správne zoraďovať, obvykle je používaná miera istoty prítomná pri jednotlivých odporúčaniach. Problémom pri zmiešanom prístupe je náročnosť spätného overenia jednotlivých použitých metód. Príkladom odporúčacieho systému využívajúceho zmiešaný prístup je systém na odporúčanie krstných mien, ktorý vyhral súťaž pre webový portál nameling.net (Glauber et al. 2013). Systém vytvára množinu odporúčaní z troch prístupov, a to obsahovo založeného, kolaboratívneho prístupu a zvyšnú vzorku mien usporiada podľa všeobecnej popularity. Úspešné sa ukázalo použitie tohto prístupu v doméne odporúčania multimediálneho obsahu skupinám (Kaššák 2014), kde zvýšil presnosť odporúčaní na popredných priečkach voči využitiu metód samostatne. Kaššák využíva dve metódy generovania odporúčaní, a to kolaboratívne odporúčania jednotlivcom a obsahové odporúčania jednotlivcom v skupinách, ktoré sú neskôr agregované do výsledných hybridných odporúčaní prezentovaných používateľom.

2.3.3 Prepínacie

Prepínací prístup vyberá len jednu z viacerých odporúčacích metód v systéme na základe situácie. Pre rôznych používateľov, môže byť vybratá odlišná metóda. Predpokladom pri prepínacom prístupe je prítomnosť spoľahlivého kritéria na základe ktorej, sa rozhodne o výbere odporúčacej metódy. Obvykle sa využíva miera istoty odporučeného objektu. Príkladom prepínacieho odporúčacieho systému je práca Jajvand et. al, ktorí vytvorili systém na odporúčanie objavovania služieb v prostredí SOA (angl. service oriented architecture) (Jajvand et al. 2013). Zameriavajú sa na využitie histórie používateľa, namiesto opisovania existujúcich služieb. Preferovanou metódou odporúčania je kolaboratívne odporúčanie, ktoré je v prípade neschopnosti generovania odporúčaní nahradené kontextovo obohateným odporúčaním. Najväčšou výhodou takéhoto hybridného systému je eliminácia problému studeného štartu, a to z pohľadu používateľa, ale aj webovej služby. Prepínacie prístupy môžu okrem presnosti a pokrytia odporúčaní zvýšiť aj škálovateľnosť odporúčacieho systému (Ghazanfar and Prügel-Bennett 2010).

2.3.4 Kombinované

Pri kombinovanom prístupe sa využívajú informácie a vlastnosti získané z jedného zdroja, na doplnenie metódy resp. jej algoritmu generovania odporúčaní založenom na inom zdroji. Tento druh hybridného odporúčania obsahuje iba jeden komponent generujúci odporúčania, naopak odporúčací systém si len prepožičia informácie z ďalšieho zdroja, ktoré sám využíva pri generovaní odporúčaní. Vhodné je použiť ho pri už efektívnych odporúčacích systémoch,

ktoré chceme rozširovať. Burke spozoroval pri využití kombinovaného postupu len minimálne zlepšenie. Hydra je príkladom kombinovaného prístupu, v ktorom je kolaboratívne odporúčanie obohatené o obsahové vlastnosti objektov (Spiegel et al. 2009). Konkrétne experimenty boli realizované v doméne filmov, a suplementárnymi informáciami pri kolaboratívnom prístupe sú informácie o filmoch a používateľoch z databázy IMDB. Najlepším sa ukázal prístup využívajúci kombinovanú metódu implementovanú využitím algoritmov SVD a k-NN, ktorá je vďaka faktorizácii cez SVD efektívna, rýchla a zároveň umožňuje extrakciu latentných vlastností medzi používateľmi a filmami.

2.3.5 Rozšírené

Podobne ku kombinovanému spôsobu Burke definuje kategóriu rozšírených hybridných odporúčacích systémov. Pri tomto spôsobe je vstupom pre metódu odporúčania množina obohatená o prídavné informácie. Jedná sa ale už o obohatenie používateľských dát vo forme modelov, alebo kandidátov na odporúčanie formou určitého výpočtu realizovaného inou odporúčacou metódou resp. komponentom. Proces rozširovania je možné realizovať off-line, nezaťažuje a nezvyšuje komplexitu (dimenzionalitu) hlavnej odporúčacej metódy a je flexibilnejší ako kombinovaný prístup. Príkladom takéhoto prístupu pri odporúčaní reštaurácií je napr. zhlukovanie reštaurácii do zhlukov vedľajšou metódou odporúčania. Zhluk, ku ktorému reštaurácia prislúcha môžeme považovať za rozšírenie odporúčaných objektov – reštaurácií. Následne obohatíme metódu odporúčania o preferenciu reštaurácií z rovnakého zhluku. Praktickým príkladom tohto prístupu je kombinovanie obsahovo založeného odporúčania, ktoré neskôr spracúva prídavnú informáciu o príslušnosti k jednotlivým zhlukom metódou kolaboratívneho odporúčania (Kim 2003). Problémom rozšíreného prístupu je hľadanie vhodnej reprezentácie výstupu a informácie ako takej z vedľajších rozširujúcich odporúčacích metód.

2.3.6 Kaskádové

Sekvenčným zoradením a vytvorením striktnej hierarchickej štruktúry vzniká kaskádový model. V tomto modely sa využívajú ďalšie metódy iba v situácii, kedy nadradená odporúčacia metóda priradila viacerým objektom rovnaké hodnotenie, resp. užitočnosť pre používateľa. Už existujúce predchádzajúce odporúčania nezmení, iba ich preusporiada. Situácia s priradenou rovnakou hodnotou funkcie užitočnosti nie je až taká častá, pretože obvykle vieme vyjadriť užitočnosť v presnejšej mierke ako napr. hodnotenia na škále 1 až 5.

Burke pri evaluácii kaskádového prístupu zmenil funkciu užitočnosti, ktorej výstupom boli diskrétne hodnoty, bez straty kvality odporúčaní. Využil kaskádovanie dvoch odporúčacích metód a výsledky boli priaznivé, prekonávajúce iné hybridné kombinácie. Príkladom hybridného kaskádového odporúčacieho systému je systém odporúčajúci hudbu pre mobilné zariadenia (Lampropoulos et al. 2012). Využívajú dvojúrovňovú hierarchiu, kde základom je klasifikovanie žánru pomocou SVM, ktoré je následne upravené o ďalšie kolaboratívne odporúčanie. Týmto spôsobom je kaskádový model využívaný vždy. V porovnaní s čisto kolaboratívnym prístupom s Pearsonovou koreláciou dosahujú až o 30% presnejšie usporiadanie odporúčaných pesničiek a metriku MAE znižujú o 10%.

2.3.7 Meta-úrovňové

Systémy využívajúce model používateľa naučený jednou odporúčacou metódou ako vstup pre ďalšiu odporúčaciu metódu, generujúcu odporúčania, sa nazývajú meta-úrovňové. Od

kombinujúceho prístupu sa odlišujú v tom, že vstup do metódy generujúcej odporúčania úplne nahradzuje model prislúchajúci metóde generujúcej odporúčania, ale používa sa naučený model z prvej metódy. Meta-úrovňový prístup je limitovaný aplikovateľnosťou vzhľadom na doménu a získané dáta o používateľoch. Rovnako je limitovaná aj množina použiteľných metód pri odporúčaní (Burke 2007), pretože nutným výstupom jednej z použitých metód je model, ktorý vstupuje do ďalšej techniky resp. fázy generovania odporúčaní. Burke pri evaluácií 6 meta-úrovňových kombinácií ani pri jednom nepozoroval synergiu. Problémom bola slabá efektivita prístupov, ak boli použité jednotlivo, čo porušuje predpoklad nutný pre efektívnosť meta-úrovňových systémov. Zástupcom meta-úrovňového systému je systém Fab, ktorý odporúča webový obsah na základe analýzy dokumentov (webové stránky) (Balabanović 1997). Systém Fab využíva kolaboratívne odporúčania založené na obsahu. Každý používateľ systému má priradeného vlastného vyhľadávajúceho agenta, ktorý udržiava model používateľa a generuje odporúčania na základe informácií o dokumentoch získaných zberajúcim agentom, prehľadávajúcim webové indexy Alta Vista, Inktomi a Excite. Balabanović navrhol danú architektúru, ktorá okrem škálovateľnosti prináša ďalšie výhody. Napríklad umožňuje vytváranie modelu priemerného používateľa a rieši problém studeného štartu.

2.4 Obohatenie odporúčaní o kontext používateľa

Kontext je interdisciplinárne skúmaný subjekt, na ktorého definíciu existuje viacero pohľadov. Vo svete počítačov sa vyskytuje napr. pri dolovaní dát, personalizácii, marketingu, vyhľadávaní informácií. Pri elektronickom nakupovaní je možné bližšie určiť vďaka kontextu zámer nakupujúceho. Tento prístup zvolili v (Palmisano et al. 2008), kde vytvárali pre používateľov viacero modelov (toho istého používateľa) na základe kontextu nákupu. Pre príklad, rozlišovali nákup knihy za účelom sebavzdelávania sa v určitom smere a nákup knihy ako darček pre tretiu osobu. Adomavicius a Tuzhilin (Tintarev et al. 2011) považujú za najvhodnejšiu definíciu a klasifikáciu kontextu v doméne odporúčacích systémov prácu Dourisha, ktorý klasifikuje kontext z dvoch uhlov pohľadu, a to *reprezentačného* a *interakčného* (Dourish 2004). Pri *reprezentačnom* pohľade je kontext definovaný ako vopred známa, preddefinovaná množina pozorovateľných a identifikovateľných atribútov nad štruktúrou, ktorá sa výrazne s časom nemení. Pri *interakčnom* pohľade predpokladáme, že správanie používateľa závisí od kontextu, ktorý nemusí byť nutne pozorovateľný. Dourish ďalej predpokladá bidirekcionálny vzťah medzi aktivitami a prináležiacim kontextom, pri ktorom boli aktivity realizované.

Majoritná časť výskumu odporúčacích systémov sa zameriava na odporúčanie bez zohľadňovania kontextu prislúchajúcemu k používateľovi. Ak uvažujeme o kontexte v doméne odporúčacích systémov, môžeme upraviť hodnotiacu funkciu R z úvodu kapitoly 2 nasledovne:

$$R: User \times Item \times Context \rightarrow Rating$$

$User$ a $Item$ predstavujú množinu používateľov a odporúčaných objektov, kontextuálna informácia asociovaná s aplikáciou je zahrnutá v kontexte $Context$ a $Rating$ predstavuje výslednú množinu hodnotení objektov používateľmi. Kontext môže byť rôzneho druhu, a každý druh môže byť definovaný špecifickou štruktúrou, ktorá odráža kontextuálnu informáciu. Jedným z populárnych spôsobov reprezentácie kontextu je forma hierarchickej

štruktúry (napr. stromov) (Palmisano et al. 2008; Adomavicius et al. 2005). Pri aplikácií odporúčajúcej hudbu predstavujú množinu objektov pesničky (názov, autor, vydavateľstvo, žáner, dĺžka, rok vydania) a množinu používateľov používatelia(ID, meno, adresa, pohlavie). Kontextom môže byť napr. čas prehrávania alebo miesto prehrávania. Hierarchická reprezentácia času potom vyzerá takto (granularitu je možné určiť): deň, týždeň, mesiac, rok.

2.4.1 Získavanie kontextuálnej informácie

Množinu spôsobov, ako je možné získať kontext používateľa resp. kontextuálnu informáciu je možné definovať nasledovne (Tintarev et al. 2011):

- *Explicitne* – priamym oslovovaním relevantných používateľov a zdrojov kontextuálnej informácie, za účelom získania informácie (vypĺňanie webových formulárov alebo dotazníkov),

- *Implicitne* – priamo zo získaných dát alebo prostredia, bez interakcie s používateľom (zmena polohy, časová pečiatka transakcie),

- *Odvodením* – na základe štatistických metód alebo pomocou metód dolovania dát, ako napr. *Bayesianovských sietí* (Palmisano et al. 2008) (identifikácia používateľa na zdieľanom počítači spôsobom písania na klávesnici). Pri tomto spôsobe je možné využiť aj latentné faktory resp. premenné, na implicitné odvodenie kontextuálnej informácie bez akejkoľvek explicitnej znalosti kontextu.

Pri voľbe vhodných kontextuálnych atribútov odporúčajú Adomavicius et al. na začiatok zvoliť širokú množinu aspektov, definovanú doménovým expertom. Potom na základe získaných dát a realizovaných štatistických testov môžeme naozaj zistiť, či konkrétny kontext používateľa vplýva na aktivitu používateľa.

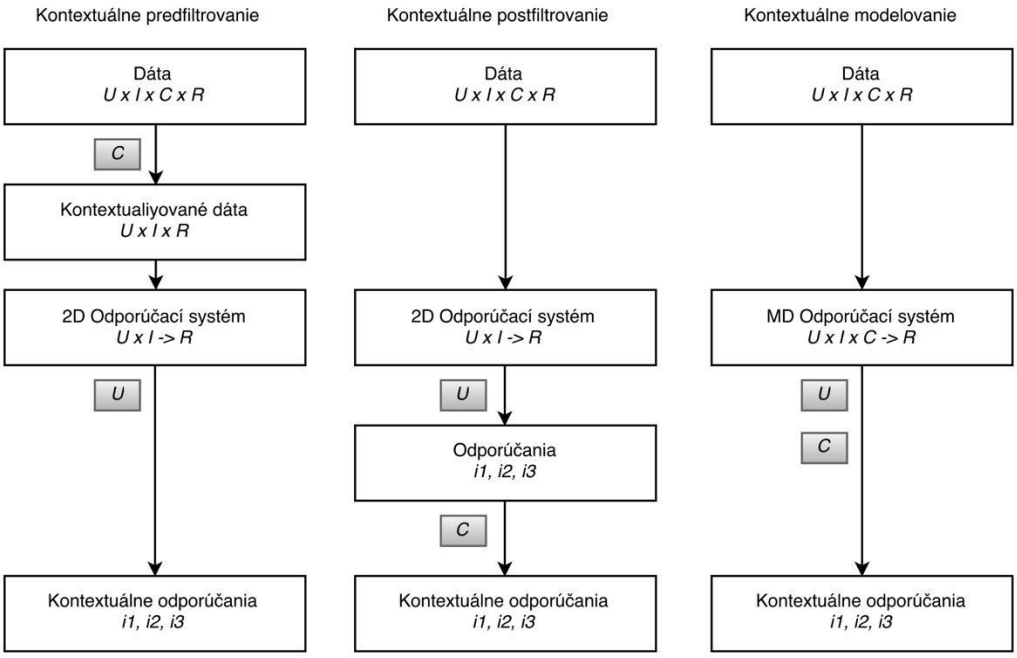
2.4.2 Využitie kontextuálnej informácie v procese odporúčania

Rozličné prístupy využitia kontextuálnych informácií pri generovaní odporúčaní možno podľa (Adomavicius et al. 2005)rozdeliť do dvoch abstraktných skupín:

- *Odporúčania na základe kontextovo riadeného dopytovania* - často využívané v doméne turistických sprievodcov, kde najprv explicitne alebo implicitne získame kontext (nálada používateľa, počasie) a následne prezentujeme používateľovi najlepšie výsledky,

- *Odporúčania na základe výberu a odhadu kontextuálnej preferencie* – novší spôsob (Adomavicius et al. 2005), kedy je cieľom modelovať a naučiť sa používateľove kontextuálne preferencie pozorovaním jeho interakcie so systémom za pomoci strojového učenia alebo dolovania dát.

Adomavicius a Tuzhilin druhú skupinu ďalej rozdeľujú do troch kategórií (Obr. 4), podľa toho v ktorej fáze generovania odporúčaní sa kontext spracúva.



Obr. 4 Spôsoby obohacovania odporúčaní o kontext pri ich generovaní (Tintarev et al. 2011)

Identifikované tri paradigmy využitia kontextu sú nasledovné:

- *Filtrovanie kontextom pred generovaním odporúčaní* alebo kontextualizácia odporúčaných objektov na vstupe je paradigma, kde kontextuálna informácia riadi výber dát pre špecifický kontext. Najprv pomocou kontextu zvolíme vhodnú množinu hodnotených objektov, resp. ich hodnotení, u ktorých následne predikujeme hodnotenia vybraného používateľa (ak chceme odporúčať film v kontexte času na sobotu večer, do úvahy zoberieme iba hodnotenia vyjadrené v sobotu večer). Pri tomto spôsobe, kde môže byť určený kontext príliš špecializovaný, sú využívané techniky *kontextového zovšeobecňovania* alebo *redukcie*.

- *Filtrovanie kontextom po generovaní odporúčaní* je opačným prístupom, kde sú už vygenerované odporúčania kontextualizované pre jednotlivých používateľov pomocou kontextuálnej informácie. Úpravy formou kontextualizácie môžu byť buď *filtrovacie* – odstránenie irelevantných odporúčaní v danom kontexte, alebo *modifikujúce* poradie odporúčaní. Tento prístup je ďalej možné rozdeliť na techniky využívajúce *model* alebo *heuristiky*.

- *Modelovanie kontextom* alebo kontextualizácia odporúčania zasahuje priamo do modelovacích techník a je časťou funkcie predikovaného hodnotenia pri odporúčaných objektoch.

Tieto paradigmy, resp. formy predikcie je možné kombinovať. Kombinovaním viacerých kontextuálnych filtrov pred generovaním odporúčaní sa zaoberali Adomavicius et al. a dosiahli približne o 20% lepšie výsledky pri ich metóde segmentácie filtrovaním (Adomavicius et al. 2005) v porovnaní s klasickou metódou kolaboratívneho odporúčania. Výskum v tejto oblasti poukazuje na zefektívnenie kvality odporúčaní pri zahrnutí kontextu do procesu ich generovania (Adomavicius et al. 2005; Palmisano et al. 2008; Zeng et al. 2012;

Shi et al. 2014b). Adomavicius a Tuzhilin poukazujú na dva hlavné problémy resp. vlastnosti kontextovo obohatených odporúčacích systémov – *komplexitu* a *interaktívnosť.* Kombinovaním týchto aspektov vzniká potreba flexibilných odporúčacích metód, ktoré umožňujú používateľom získať pre nich zaujímavé odporúčania pomocou vhodných používateľských rozhraní. Zaujímavým flexibilným odporúčacím systémom je FlexRecs (Koutrika et al. 2009), ktorý oddeľuje definíciu a samotné generovanie odporúčaní. Umožňuje deklaratívne vyjadrenie odporúčaní vysoko úrovňovým parametrizovateľným prístupom. Ďalším príkladom flexibilného prístupu ku kontextovo obohatenému procesu odporúčania je REQUEST[3] - dopytovací jazyk na odporúčania (Adomavicius et al. 2011). Založený je na modeli inšpirovanom multidimenzionálnym kontextovo obohateným odporúčaním, ktorý využíva techniky OLAP[4]. Pomocou dopytovacieho jazyka umožňuje presnejšie prispôsobenie odporúčaní jednotlivým používateľom vďaka flexibilnému spôsobu špecifikácie dopytu odrážajúceho záujmy používateľov.

2.5 Problém studeného štartu

Studený štart môže nastať v prípade, že odporúčací systém nie je schopný vygenerovať spoľahlivé odporúčania kvôli nedostatku hodnotení. Tento problém je možné identifikovať v troch rovinách, a to pridanie nového objektu do dátového setu (Saveski and Mantrach 2014) pridanie nového používateľa, pre ktorého budeme musieť byť schopní generovať nové odporúčania (Nicta et al. 2014) a vznik novej komunity formou nového odporúčacieho systému (Bobadilla et al. 2013). Problém nového objektu vzniká, ak do odporúčacieho systému prichádza nový objekt, ktorý obvykle nemá priradené hodnotenia používateľov, čiže je malá miera jeho odporučenia používateľom. Ak objekt nie je odporúčaný, používatelia si ho nemusia všimnúť a môžu ho ignorovať, čo znamená že nezískame hodnotenia používateľov. Týmto vzniká problém, keď sa odporúčací systém dostane do nekonečného kruhu. Častým riešením je určenie a motivácia vybraných používateľov, ktorí hodnotia nové objekty v systéme. Väčším problémom je príchod nového používateľa do systému. Používatelia nehodnotili žiadne objekty, takže odporúčací systém im nedokáže vygenerovať odporúčania. Aj po pár iniciálnych hodnoteniach je náročné vygenerovať presné odporúčania a používatelia zostávajú frustrovaní alebo opúšťajú systémy alebo webové portály využívajúce odporúčacie systémy. Riešením je využitie a obohatenie informácií z iných zdrojov alebo využitie hybridných odporúčacích systémov.

Využitím dát zo sociálnej siete Facebook[5] a základných demografických dát Nicta et al. zvýšili úspešnosť odporúčaní pre tzv. "studených" používateľov 3 až 6 násobne v porovnaní so štandardizovanými metrikami (angl. RMSE – Root mean square error) (Nicta et al. 2014). Tento problém je prítomný v oboch rozšírených typoch generovaní odporúčaní (Liu 2011). Liu ho rieši využívaním reprezentatívnej vzorky objektov a používateľov, ktoré prechádzajú maticovou faktorizáciou. Výsledky overení realizované v doméne multimédií (film a hudba) vykazujú vysokú diverzitu a pokrytie odporúčaní jeho metódou a zároveň v oboch typoch problému studeného štartu prekonávajú zaužívané heuristické metódy.

Problém studeného štartu je výraznejšie prítomný v doméne kolaboratívneho odporúčania, kde je model založený iba na predošlých hodnoteniach objektov v systéme (Schein et al.

[3] angl. Recommendation QUEry STatements
[4] angl. Online Analytical Processing
[5] https://www.facebook.com/

2002). Hybridnou kombináciou a pridaním odporúčaní na základe obsahu môžeme eliminovať problém studeného štartu nových objektov dátových setov (Saveski and Mantrach 2014; Gunawardana and Meek 2008). Využitím hybridného prístupu a pomocou faktorizácie matíc Saveski a Mantrach vytvorili rámec, ktorý podľa overení v doméne novinových článkov prekonal existujúce prístupy k riešeniu studeného štartu odporúčaných objektov. Gunawardna a Meek použili na riešenie štatistický prístup využívajúci Boltzmannove automaty pri modelovaní korelácií medzi náhodnými hodnoteniami (párová interakcia) objektov používateľmi, ktoré neskôr prepájajú s asociovanými obsahovými informáciami o objekte. Alternatívnym prístupom, overeným v doméne odporúčania mobilných aplikácií je získavanie informácii zo sociálnych sietí (Twitter[6]) (Lin et al. 2013), kde autori pracovali s fanúšikmi na profiloch jednotlivých aplikácií a vytvárali modelové virtuálne skupiny. Týmto latentným prístupom dokázali Lin et al. vo svojich experimentoch až o 33% vyššie odporúčaní voči zaužívaným metódam.

2.6 Model používateľa

Model používateľa reprezentuje informácie o individuálnych požívateľoch a je nutný pre schopnosť adaptácie, resp. personalizovania generovaných odporúčaní. V adaptívnych webových systémoch modelujeme používateľovu znalosť, ciele, úlohy, predchádzajúcu skúsenosť alebo individuálne rysy používateľa (povahové, osobnostné alebo kognitívne) (Brusilovsky and Millán 2007). V personalizovanom odporúčaní modelujeme používateľove záujmy. Proces vytvorenia a udržiavania aktuálneho modelu používateľa získavaním dát rôznymi spôsobmi, ako napr. implicitné sledovanie správania používateľa alebo explicitné vyžiadanie vstupu od používateľa sa nazýva modelovanie používateľa.

Pri nových používateľoch v systéme existuje problém studeného štartu, diskutovaný v kapitole 2.5. Jedným z možných riešení, je možnosť naúčiť sa model používateľa od ostatných, jemu najpodobnejších používateľov v systéme. Tento proces pomocou EM algoritmov (angl. Expectation-maximization) je ale výpočtovo veľmi náročný, najmä kvôli riedkej distribúcií dát. Zhang a Koren predstavili alternatívnu metódu modifikáciou EM algoritmov na získavanie informácii pre model používateľa pomocou Bayesovskeho hierarchického modelu (Zhang and Koren 2007). Ich metóda priniesla násobné zníženie nutného počtu iterácií algoritmu pri riedkych dátových množinách. Škálovateľnosť ich metódy umožňuje vytvorenie pol milióna používateľských profilov zo sto miliónov hodnotení v priebehu pár hodín na jednojadrovom CPU. Pri hybridnom odporúčaní ako prví využili váhovaný model používateľových vlastností Symeonidis et al., ktorí kombinovali a určovali váhy aj z pohľadu jednotlivých používateľov, ale aj medzi používateľmi na základe súvisiacich podobných vlastností (Symeonidis et al. 2007). Tento model aplikovali na už existujúci model vytvorený na základe predchádzajúcich hodnotení a vlastností objektov. Pri evaluácií nad štandardizovanými dátovými množinami (MovieLens[7] a EachMovie[8]) úspešne prekonali (o 60% vyššia presnosť, o 20% vyššie pokrytie) klasické metódy tvorby používateľských profilov v kolaboratívnom a obsahovo založenom odporúčaní.

Wang et al. použili pri kolaboratívnom odporúčaní hybridný model používateľa, kde je konštrukcia modelu uskutočňovaná off-line na základe obsahu a demografických informácii

[6] https://twitter.com/
[7] http://grouplens.org/datasets/movielens/
[8] http://grouplens.org/datasets/eachmovie/

(Wang et al. 2010). Vďaka tomu redukovali veľkosť dimenzií matíc vlastností objektov a používateľov, ktoré využívali pri výpočte podobnosti medzi používateľmi. Hybridný model používateľa overovali v doméne televíznych programov aj Ardissono et al, kde využili kombináciu troch nezávislých heterogénnych expertných modelov (Ardissono et al. 2004). Explicitný model používateľa vypĺňaný pri registrácii, model využívajúci stereotypy správania sa používateľov (dáta získali z predchádzajúcich sledovaní správania sa populácie talianskych občanov) a tretím expertným modelom bol dynamický pravdepodobnostný model, založený na modelovaní preferencií a kontextu používateľov pomocou Bayesianovských sietí. Tieto modely boli následne kombinované a ich efekt váhovaný mierou istoty, pričom s postupom času bol výsledný model ovplyvňovaný najviac tretím, dynamickým modelom. Kombináciou čiastkových modelov dokázali zvýšiť efektivitu odporúčaní voči stereotypnému modelu až o 80%. V doméne elektronického nakupovania sa ako efektívny ukázal behaviorálny model používateľa od Iwata et al., ktorý modeluje jednoduchú vlastnosť zákazníkov – predchádzajúce nákupy, a je založený na princípe maximálnej entropie (Iwata et al. 2007).

González et al. vytvorili inteligentný model používateľa (angl. smart user model), ktorý zvyšuje úroveň a presnosť personalizácie, zmenšuje objem spracovaných informácií a odhaľuje používateľove preferencie a emocionálny stav v otvorených, distribuovaných a prepojených prostrediach (González et al. 2005). Využitím multi-agentových systémov a techník strojového učenia (SVM angl. Support Vector Machines) navrhujú inteligentný model využívajúci znalosti zo systémov obklopujúcich používateľa.

Zjednocovaním a agregáciou modelov používateľa z viacerých služieb sa zaoberal aj Berkovsky (Berkovsky 2005; Berkovsky et al. 2008). Berkovsky definoval štyri najväčšie problémy pri mediácií resp. spájaní používateľských profilov z rôznych systémov:

- Trhové a biznisové obmedzenia (napr. konkurencia),
- Zachovanie používateľovho súkromia,
- Technické a praktické prekážky pri komunikácii s externými systémami,
- Štrukturálna heterogénnosť a nekompletnosť dát používateľských modelov.

Berkovsky v (Berkovsky et al. 2008) prezentuje rámec na riešenie problému heterogénnosti používateľských modelov.

2.6.1 Distribuovaný model používateľa

V doteraz prezentovaných prístupoch sa jednalo o centralizované modelovanie používateľa (vo výslednej podobe), ktoré je v doméne odporúčacích systémov dominantné. Distribuované modelovanie používateľa je preskúmanou oblasťou, kde boli navrhnuté rôzne rámce alebo systémy na využitie vo všadeprítomnom prostredí pre modelovanie používateľa (Heckmann 2005; Specht et al. 2005; Vassileva 2001). Uniformnú reprezentáciu

distribuovaných modelov je možné prehliadať pomocou GUMO[9], v ktorom je ako štandard zvolený modelovací jazyk UserML[10].

Presunutím modelu používateľa na stranu klienta sa zaoberali Bilenko a Richardson, a to v doméne internetovej reklamy (Bilenko and Richardson 2011). Model je umiestnený v cookies internetového prehliadača, je transparentný a umožňuje vykonať zmeny v modely priamo používateľmi. Ich overenie realizovali vo vyhľadávacom nástroji Bing[11] spoločnosti Microsoft[12] a prišli k záverom, ktoré ukazujú že odstránením serverovej časti z procesu modelovania používateľových preferencií nestráca personalizácia reklamy na efektivite. Ako negatívum vidíme obmedzenosť veľkosti lokálneho úložiska – cookies dovoľujú ukladať obsah len do veľkosti 4KB, čo v prípade zvyšného miesta využitom vyhľadávačom Bing predstavovalo miesto len pre približne 100 kľúčových slov.

BrUMo[13] predstavuje špecializovaný decentralizovaný rámec na modelovanie používateľa a personalizáciu (Šajgalík et al. 2013). Šajgalík et al. zvolili webový prehliadač, konkrétne doplnok webového prehliadača Mozzila Firefox ako vhodné úložisko pre model používateľa. Výhodou je jednoduchý prístup ku kompletnej histórii a správaniu sa používateľa na webe. BrUMo je využiteľný aj pre odporúčanie na základe obsahu, aj pre kolaboratívne odporúčanie, vďaka zdieľaniu modelu medzi službami eliminuje problém studeného štartu. V obmedzených možnostiach úložiska webového prehliadača efektívne ukladajú stromy doménových a používateľových záujmov, kde listy predstavujú identifikované kľúčové slová charakterizujúce záujmy používateľa. Efektívnou implementáciou formou označených radixových stromov je možné získavať množiny najrelevantnejších váhovaných termov v stovkách mikrosekúnd. Modelovanie za účelom personalizácie na strane klienta v mobilnom zariadení realizovali Gerber et al. vytvorením PersonisJ (Gerber et al. 2010). Jedná sa o rámec využívajúci výpočtovú silu mobilných zariadení na vytvorenie kontextovo obohatených personalizovaných aplikácií. Nezanedbateľnou výhodou pre používateľov je plné vlastníctvo ich modelu, s čím sú späté ale aj bezpečnostné riziká súvisiace s mobilnými zariadeniami. Cudzie aplikácie komunikujú s verejne dostupným API, ktoré obsluhuje PersonsisJ jadro aplikácie, ktoré umožňuje import a export ontológií. Rámec PersonisJ je postavený na základoch systému PersonisAD – aktívneho, skúmateľného, distribuovaného modelovacieho rámca pre kontextovo závislé služby (Assad et al. 2007). Využitím systémov P2P (angl. Peer to peer) sietí na distribúciu modelu používateľa vytvorili Paraskevopoulos and Mentzas návrh plne decentralizovanej architektúry na vytváranie a udržiavanie modelov používateľov (Paraskevopoulos and Mentzas 2014).

2.7 Spätná väzba

Na vytvorenie používateľských modelov je obvykle nutné zachytávať používateľské preferencie a správanie pomocou spätnej väzby. V praxi sa na tvorbu modelov používajú často kombinácie oboch typov spätných väzieb: implicitnej – nepriame akcie používateľa aj explicitnej – priame hodnotenia. Kombináciou oboch prístupov pomocou faktorizácie Koren dokázal zlepšenie kvality generovaných odporúčaní (Bell and Koren 2007; Koren 2010). Defacto štandardným prístupom vyjadrenie používateľských preferencií sa v doméne

[9] The General User Model Ontology
[10] User Modeling Markup Language
[11] www.bing.com
[12] www.microsoft.com
[13] http://brumo.fiit.stuba.sk/?home.html

odporúčacích systémov ustálila explicitná spätná väzba vo forme hodnotení (Jawaheer et al. 2014). Jawaheer et al. uvádzajú porovnanie prístupov z pohľadu aspektov dôležitých pre odporúčacie systémy v tabuľke 1. K určitým aspektom ako napr. negatívna polarita pri implicitnej spätnej väzbe alebo škála merania uvádzame nižšie práce, ktoré ukazujú aj širšie pojatie uvádzaných aspektov.

Tab. 1 Vlastnosti spätnej väzby v odporúčacích systémoch (Jawaheer et al. 2014)

Aspekt	Explicitná spätná väzba	Implicitná spätná väzba
Kognitívna námaha	Áno	Nie
Model používateľa	Preferencie	Miera istoty
Škála merania	Poradie	Pomer
Doménová závislosť	Závislá	Nezávislá
Náchylnosť na šum	Áno	Áno
Polarita	Pozitívna aj negatívna	Pozitívna
Rozsah používateľov	Podmnožina používateľov	Všetci používatelia
Transparentnosť pre používateľa	Áno	Nie
Zaujatosť (angl. bias)	Skúsení používatelia	Nie

2.7.1 Explicitná spätná väzba

Preferovaným prístupom k zachyteniu spätnej väzby v doméne odporúčacích systémov je explicitná spätná väzba (Jawaheer et al. 2014), kde používateľ obvykle vyjadruje svoje preferencie nad objektami formou hodnotení. Tieto hodnotenia môžu byť rôznej škály, či už binárne ako „áno, páči sa mi to" a "nie, nepáči sa mi to" alebo viacúrovňové hodnotenia na škále 1-10, percentuálne hodnotenie alebo využitie Likertovej škály. Z pohľadu používateľa je systém využívajúci explicitnú spätnú väzbu transparentnejší a ďalšou výhodou je ľahké zachytávanie pozitívnej aj negatívnej spätnej väzby. Negatívum tohto prístupu pre používateľa je fakt, že nemá z týchto hodnotení žiaden priamy benefit, oberá ho to o čas a preto je nutné používateľov stimulovať a motivovať k explicitnému prejavu ich preferencií. Na prvý pohľad sa môže zdať, že explicitné hodnotenia používateľov budú presnejším vyjadrením ich preferencií, ale v praxi a aj pri tomto spôsobe zberu spätnej väzby sa objavuje nestabilita hodnotení používateľov a určitý šum (Amatriain et al. 2009a; Jawaheer et al. 2010; Amatriain et al. 2009b), ktorý je podľa (Amatriain et al. 2009b) možné odstrániť opätovným prehodnocovaním objektov používateľmi, čo je v praxi aplikovateľné iba na obmedzenej vzorke dát. Druhým prístupom, ktorý realizovali bola čiastočná eliminácia šumu aj na strane dát, prípadne identifikácia "rušivých" používateľov, čo nevyžaduje ďalšiu interakciu zo strany používateľa, ale stačí na to, aby signifikantne vylepšili konzistentnosť získaných preferencií.

Okrem explicitných hodnotení odporúčaných objektov, je možné pri budovaní používateľských modelov využiť aj iné postupy zapracovania explicitnej spätnej väzby. V (Szomszor et al. 2007) sa na malej vzorke dát v doméne filmov ukázalo vhodné obohatiť množinu odporúčaných objektov o tagy z externej služby kde bolo ich hlavnou motiváciou vytvoriť bohatšie profily používateľov. Ďalšou entitou využiteľnou na zber explicitnej spätnej väzby sú recenzie a komentáre používateľov prislúchajúce k odporúčaným objektom. Garcia Esparza et al. experimentoval so zaniknutou mikro blogovacou platformou Blippr, v ktorej používatelia vytvárali okrem priamych hodnotení pomocou štvorstupňovej škály aj krátke recenzie a publikovali ich na svojom profile (Garcia Esparza et al. 2012). Evaluáciou ich metódy na platforme Blippr prekonali tradičné kolaboratívne odporúčanie vo všetkých

dátových množinách, pri porovnávaní metrík presnosti a pokrytia odporúčaní. Aciar et al. realizovali odporúčanie na základe používateľských recenzií v obmedzenej doméne digitálnych fotoaparátov (Aciar et al. 2007). Numerické explicitné hodnotenia sú využívané pri výpočte podobnosti medzi objektami alebo používateľmi. Desroisiers a Karypis vylepšili známy algoritmus výpočtu podobnosti a korelácie *SimRank* (Jeh and Widom 2002) o možnosť využitia nie numerických hodnotení pri výpočte podobnosti objektov. Ich ďalším prínosom bola efektivita generovania odporúčaní v riedkych dátových množinách, kde ich prístup výpočtu podobnosti používateľov uvažuje všetky dostupné hodnotenia vybraných používateľov a nie len spoločné pre oboch používateľov (Desrosiers and Karypis 2010).

Explicitné hodnotenie numerickými hodnotami alebo intervalmi sa ukazuje ako intuitívne, ale podľa (Koren and Sill 2011) nereprezentuje používateľskú spätnú väzbu prirodzeným spôsobom. Každý používateľ má individuálne mentálne nastavenú stupnicu pre hodnotenia, čo vyžaduje normalizáciu hodnotení. Koren a Sill vytvorili rámec *OrdRec* v ktorom nevyužívajú priame hodnotenia objektov (ich číselnú hodnotu), ale pri faktorizácii matíc pomocou SVD++ algoritmu pracujú s poradím ohodnotených objektov používateľmi.

2.7.2 Implicitná spätná väzba

V praxi je implicitná spätná väzba často jedinou, ktorá je prítomná v mnohých odporúčacích systémoch. Od používateľov nevyžaduje žiadne ďalšie úsilie a kognitívnu námahu, navyše eliminuje možný nátlak a ovplyvnenie preferencií používateľa komunitou. Obvykle nevieme priamo zachytiť negatívnu spätnú väzbu (pasivita používateľa k objektom neznamená nutne negatívnu preferenciu), ale domény, v ktorých je dostatočná variabilita a objekty majú podobnú granularitu je možné negatívnu spätnú väzbu správne predpokladať (Parra and Amatriain 2011). Parra a Amatriain uvádzajú televízne seriály ako príklad vhodnej domény, kde ak používateľ zhliadol iba jednu časť, môžeme povedať, že seriál ho nezaujal a vyjadril tým svoje negatívne preferencie. Tento prístup nie je možné opakovať v napr. v podobnej doméne filmov. Najčastejším spôsobom, ako môžeme pracovať s implicitnou spätnou väzbou je monitorovanie času, ktorý používatelia strávia pri prezeraní konkrétneho objektu, percentuálny objem prečítaného obsahu alebo interakcia vo forme klikov alebo pohybov myši. Sledovanie používateľovho správania definovali Oard a Kim formou rámca aplikovateľného bez závislosti od domény (Oard and Kim 2001). Vymenované správanie používateľa analyzovali Claypool et al. so zameraním sa na odporúčacie systémy, pričom čas strávený na stránke určili ako najspoľahlivejší faktor (Claypool et al. 2001). Hu et al. odporúčajú mapovať získanú implicitnú spätnú väzbu používateľa na explicitnú preferenciu používateľa, nižšie uvedeným vzťahom (je vhodné ho upraviť pre konkrétnu doménu),

$$c_{ui} = 1 + \alpha \, log\,(1 + \frac{r_{ui}}{\epsilon}) \qquad \textit{(Hu et al. 2008)}$$

kde c_{ui} je pozorované správanie a r_{ui} vyjadrenie miery istoty zaznamenaného používateľského správania (Hu et al. 2008). Implicitnú spätnú väzbu je podľa spomenutých autorov v kapitole 2.7.2 vhodné modelovať binárne (Oard and Kim 2001; Koren 2010; Hu et al. 2008). Rendle et al. modelovali odporúčanie pomocou implicitnej spätnej väzby ako predikciu personalizovaného usporadúvania objektov, využitím algoritmu *BPR* (angl. Bayesian personalized ranking) (Rendle et al. 2009). Porovnaním s metódami *k*-NN a faktorizácie matíc nad rovnakou dátovou množinou, kde pôvodnú explicitnú spätnú väzbu upravili na pseudo implicitnú, zvýšili úspešnosť predikovaných hodnotení. Problémom pri pracovaní s implicitnou spätnou väzbou je vysoká miera šumu (Gadanho and Lhuillier 2007).

Koreláciu implicitnej spätnej väzby a explicitného hodnotenia používateľom skúmali v doméne multimédií Parra a Amatriain, kde prišli k záveru o vysokom prepojení týchto dvoch faktorov (Parra and Amatriain 2011). S pridaním sledovania času prehrávania hudby (aktuálnosť – angl. recentness) pomocou svojej metódy vylepšili schopnosť predikcie hodnotenia objektov až o 10%.

3 Škálovateľnosť odporúčacích systémov

V kapitole skúmame pojem škálovateľnosť odporúčacích systémov. Ukážeme si, ako zvoliť vhodnú architektúru systému, tak by bol škálovateľný. Predstavíme si špecifiká distribuovaných systémov a modelov, kde nadviažeme na kapitolu 2.6.1 Distribuovaný model používateľa. Pozrieme sa na používané postupy a technológie, ktoré využíva aj komerčná oblasť, ako uvidíme v kapitole 4.1.

3.1 Architektúra distribuovaných systémov

Pod distribuovaným systémom rozumieme systém, ktorého komponenty komunikujú pomocou siete a ich akcie sú koordinované prostredníctvom vymieňania správ. Medzi problémy, ktoré sa vyskytujú pri distribuovaných systémoch sa radí otvorenosť systému, zlyhanie komponentov, transparentnosť, škálovateľnosť, komplexnosť, bezpečnosť, heterogenita komponentov, paralelný prístup ku zdrojom - konkurencia, manažovateľnosť a nepredvídateľnosť. Z pohľadu architektúry môžeme distribuované systémy rozdeliť do dvoch skupín, využívajúcich architektonický štýl *klient-server* alebo *peer to peer* (Coulouris et al. 2011).

Klient-server architektúra

Tento typ architektúry je v súčasnosti najrozšírenejší. Vystupujú v ňom dve entity, *server* – ktorý poskytuje vopred definovanú množinu služieb a *klient* – ktorý tieto služby využíva. Každá entita plní odlišnú funkciu, pričom v pozícií klienta sa môže vyskytnúť aj iný server. Architektúru klient-server je ďalej možné rozlišovať, na základe toho, kde sa vykonáva väčšia a komplexnejšia časť spracovania úloh, čím sa zadefinovali architektonické vzory. Pri tenkom klientovi je snaha presunúť čo najväčšie množstvo operácií na stranu servera, zatiaľ čo klient iba zadáva úlohy a požiadavky na spracovanie. Tento scenár je bežný v doméne distribuovaných odporúčacích systémov. Alternatívou je použitie tučného klienta, kedy je určitá časť výpočtov presunutá priamo na stranu klienta. Vynikajúcou ukážkou z domény odporúčacích systémov využívajúcich architektúru tučného klienta je systém HyRec (Boutet et al. 2014), ktorý si bližšie predstavíme v kapitole 4.2.2.

Peer to peer architektúra

Decentralizované systémy, v ktorých môže byť výpočtová úloha vykonaná ktorýmkoľvek uzlom v sieti využívajú tzv. peer to peer architektúru. Všetky uzly prijímajú a odosielajú dáta, realizujú výpočty a zohrávajú rovnakú rolu v systéme. Architektúru typu peer to peer je možné využiť na distribúciu modelu používateľa (Paraskevopoulos and Mentzas 2014) alebo aj pri návrhu škálovateľného kolaboratívneho odporúčacieho systému (Gong et al. 2009). Gong et al. rozdeľovali dátové množiny hodnotení objektov v systéme priamo na jednotlivých klientov odporúčacieho systému, ktorí boli zodpovední aj za výpočet podobnosti a hľadanie najbližších susedov.

V súvislosti so škálovateľnosťou distribuovaných systémov definoval Brewer CAP teorém (angl. consistency, availability, partition tolerance) (Fox and Brewer 1999), ktorý súvisí so zvýšenou komplexitou distribuovaných a škálovateľných systémov. CAP hovorí o tom, že každý distribuovaný, sieťovo prepojený systém, využívajúci zdieľané dáta môže garantovať najviac 2 z 3 vlastností CAP, ktorými sú:

- *Konzistencia (angl. consistency)* – v danom čase vidia všetky uzly distribuovaného systému rovnaké dáta.

- *Dostupnosť (angl. availability)* – každá požiadavka dostane spätnú väzbu o tom, či bola vykonaná úspešne alebo zlyhala.

- *Tolerancia výpadkov (angl. partition tolerance)* – distribuovaný systém pracuje korektne naďalej aj po strate správy alebo výpadku časti systému.

Brewer poskytuje po 12 rokoch aj aktuálnejší pohľad na problém (Brewer 2012) a objasňuje koncepty ACID (atomicita, konzistencia, izolácia a durabilita), BASE – „byť druhým" (angl. basically available, soft state, eventually consistent) a CAP. Je vhodné uviesť problémy, ktoré sú spojené so spoľahlivosťou sietí v distribuovaných systémoch. P. Deutsch identifikoval 8 mylných predpokladov[14], ktoré systémoví architekti a vývojári očakávajú pri distribuovaných systémoch:

1. Sieť je spoľahlivá.
2. Latencia je nulová.
3. Prenosové pásmo je neobmedzené.
4. Sieť je zabezpečená.
5. Topológia siete sa s časom nemení.
6. Systém spravuje jeden administrátor.
7. Cena prenosu dát je nulová.
8. Sieť je homogénna.

Zlyhaniam siete sa v distribuovaných systémoch využívajúcich komunikáciu prostredníctvom siete nevyhneme. Aj keď sú len relatívne zriedkavé, je nutné ich explicitne očakávať. Brewer objasňuje, že počas výpadkov siete nie je nutné obetovať aspekty konzistencie alebo dostupnosti úplne, ale využitím flexibilných a premyslených techník je možné optimalizovať obe vlastnosti – konzistenciu, aj dostupnosť distribuovaných systémov.

3.2 Škálovateľnosť systémov

Systém môžeme označiť za škálovateľný vtedy, ak pri zvýšení objemu používateľov alebo dát dokáže ďalej fungovať bez badateľného poklesu výkonu a nutných výrazných architektonických alebo administratívnych zásahov. Z iného hľadiska je škálovateľný taký systém, ktorého výkon narastie proporčne vzhľadom k pridanému alebo vylepšenému hardvéru. Algoritmus je škálovateľný ak je efektívny aj pri veľkých dátových množinách a dokáže využiť potenciál systému – napr. distribúciou alebo paralelizáciou výpočtu. Techniky škálovania systémov je možné rozdeliť do dvoch kategórií, a to *vertikálne škálovanie* a *horizontálne škálovanie*.

Vertikálne škálovanie, inak označované ako aj škálovanie do výšky resp. hore, je zvyšovanie výkonu pridávaním alebo vylepšovaním existujúcich hardvérových zdrojov napr. CPU alebo pamäte jedného počítača alebo servera. Pri tomto prístupe je maximálna snaha o paralelizáciu a využitie všetkých jadier CPU, využíva sa virtualizácia. Pri vertikálnom škálovaní existuje pevná hranica, kedy dosiahneme limity jedného stroja a ďalej už nie je

[14] http://www.rgoarchitects.com/Files/fallacies.pdf

možné systém rozšíriť. *Horizontálne škálovanie*, inak označované ako aj škálovanie do šírky resp. von, je proces pridávania ďalších uzlov do systému. Častou praktikou je využitie bežne dostupných počítačov alebo serverov, namiesto obstarávania výkonných serverov alebo superpočítačov, ktoré spolu vytvoria výkonný klaster. Klaster je flexibilný, umožňuje kontinuálne vylepšovanie systému, distribúciu strojov na rôzne lokácie (do viacerých dátových centier) a najmä kontinuálnu dostupnosť systému.

Proces škálovania systému je podľa (Neuman 1994) možné rozdeliť do troch dimenzií – numerická (počet používateľov alebo objektov v systéme), geografická (vzdialenosť medzi jednotlivými strojmi v systéme, prípadne ich distribúcia a administratívna (počet organizačných jednotiek alebo administrátorov spravujúcich systém). Škálovanie prináša zmeny, ktoré ovplyvňujú systémy a to najmä ich zaťaženie, výkon, dostupnosť, spoľahlivosť, administratívu a cenu.

3.3 Databázy a distribuované úložiská veľkého objemu dát

Dáta sú v odporúčacích systémoch veľmi cenným artiklom. Spoločnosti si ich chránia a zdráhajú sa ich verejne poskytovať. Tieto dáta sú ukladané v dátových úložiskách resp. databázach, ktoré si od ich počiatku prešli výraznými zmenami spätými s narastajúcimi nárokmi na ich výkonnosť.

Ak hovoríme o škálovaní databázových systémov, hovoríme väčšinou o rozdeľovaní dát do menších množín. *Delenie* (angl. partitioning) dátovej množiny je proces rozbíjania logických dátových elementov do väčšieho množstva entít za účelom zvýšenia výkonu, dostupnosti alebo jednoduchšej údržby. Už špecifickejšie môžeme hovoriť o *horizontálnom delení* (angl. sharding). Pri horizontálnom delení databázy duplikujeme databázovú schému na jednotlivé stroje a následne dáta rozdeľujeme na stroje v závislosti od kľúča dát, ktorý je vhodné zvoliť tak aby bola distribúcia dát čo možno najrovnomernejšia.

Výhodou relačných databáz je pomerne flexibilná možnosť zadefinovania dátovej schémy, ktorá je neskôr pri dopytovaní sa do databázy obmedzená prílišnou fixáciou na dátovú normalizáciu. Výsledkom takéhoto modelovania dát je veľké množstvo tabuliek, ktoré sú spájané pri dopytovaní a znižujú výkon systému. Klasické relačné databázy sú vhodnejšie a ľahšie škálovateľné vertikálne. Dodržiavaním ACID a dbaním najmä na konzistenciu je výkon a škálovateľnosť relačných databáz obmedzená. Celkovo môžeme povedať, že implementácia delenia dátovej množiny je pri relačných databáz oveľa náročnejšia ako pri systémoch využívajúcich NoSQL, ktoré boli navrhované od začiatku s ohľadom na ich horizontálnu škálovateľnosť.

NoSQL predstavuje alternatívu ku klasickým relačným databázam využívajúcich štandard SQL na dopytovanie. Vznikli väčšinou ako prispôsobené riešenia rozličných firiem a organizácií, ktoré spĺňali ich špecifické požiadavky. Jednotlivé NoSQL databázy sú veľmi špecifické, nie sú univerzálne ako relačné databázy a výrazne sa od seba odlišujú. Spája ich prakticky iba absencia relačnej schémy. NoSQL databázy je možné kategorizovať do 4 nasledujúcich skupín:

- Stĺpcovo orientované databázy,
- Dokumentové databaźy,

- Databázy typu kľúč/hodnota (angl. key/value),
- Grafové databázy.

Ako si ukážeme v kapitole 4.1 v doméne odporúčacích systémov je priestor pre kombinované využitie jednotlivých typov NoSQL databáz.

Stĺpcovo orientované databázy

Stĺpcové databázy sú najpodobnejšie relačným databázam. Dáta organizujú do riadkov a stĺpcov, ale využívajú denormalizovaný prístup. Tabuľky bývajú v stĺpcových databázach veľmi široké a často aj riedke, pretože jednotlivé riadky môžu v rámci databázovej tabuľky obsahovať rozličné stĺpce. Populárnymi riešeniami sú distribuované úložiska Apache HBase [15] a Cassandra [16], ktoré sú inšpirované projektom BigTable [17] spoločnosti Google. V databázach je umožnené vytvárať tzv. stĺpcové rodiny, ktoré agregujú skupiny stĺpcov.

Dokumentové databázy

Do dokumentových databáz sú ukladané priamo celé objekty vo forme dokumentov, často vo formáte JSON alebo XML. Dokumentové databázy nemajú pevne stanovenú schému a preto sú veľmi flexibilným riešením, či už na fulltextové vyhľadávanie alebo dátovú analytiku. Implementujú REST rozhrania a komunikujú pomocou HTTP protokolu. Vzťahy medzi dokumentmi je možné modelovať množinou vnorených sub-dokumentov alebo využitím vzťahu otec – dieťa. Populárnymi zástupcami sú napr. MongoDB[18], Elasticsearch[19] alebo CouchDB[20].

Databázy typu kľúč/hodnota

Najjednoduchším (z pohľadu poskytovanej množiny operácií) zástupcom NoSQL technológií sú úložíská typu kľúč/hodnota. Jedná sa v princípe o veľkú hashovaciu tabuľku, ktorá obsahuje unikátne kľúče pre jednotlivé záznamy. Ukladané dáta sú binárneho charakteru a môžu predstavovať číselné hodnoty, množiny alebo polia. Najznámejšou databázou tohto typu je úložisko Redis[21], ktoré od verzie 3 podporuje aj distribuovanú implementáciu Redis Cluster. O popularite rýchlych úložísk typu kľúč/hodnota v doméne odporúčacích systémov svedčí aj použitie modulárnej databázy Voldemort spoločnosťou LinkedIn (Sumbaly et al. 2013), ktorá bola inšpirovaná distribuovaným úložiskom spoločnosti Amazon DynamoDB[22].

Grafové databázy

Grafové databázy sú založené na myšlienke existencie vzťahu medzi jednotlivými entitami v databáze. Dáta sú uchovávané formou grafu s vrcholmi – objektmi a hranami – vzťahmi medzi objektmi. Hrany môžu byť orientované a rovnako ako aj vrcholy, môžu mať priradené atribúty. Výhodou grafových databáz je efektívne vykonávanie grafových algoritmov alebo traverzovanie grafov. Rovnaké údaje je možné ukladať aj v relačných databázach, ale samotné dopytovanie je implementačne aj výkonovo náročnejšie. Populárnymi zástupcami

[15] http://hbase.apache.org/
[16] http://cassandra.apache.org/
[17] http://static.googleusercontent.com/media/research.google.com/cs//archive/bigtable-osdi06.pdf
[18] https://www.mongodb.org/
[19] https://www.elastic.co/
[20] http://couchdb.apache.org/
[21] http://redis.io/
[22] http://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Introduction.html

kategórie grafových databáz sú Neo4j[23] alebo robustná distribuovaná grafová databáza Titan[24]. Efektívnosť grafových algoritmov v databáze Neo4j na väčších dátových množinách skúmali v (Demovic et al. 2013), kde sa vhodne ukázal algoritmus zjednocovania farieb, s ktorým dosiahli uspokojivú presnosť a škálovateľnosť odporúčacieho systému.

3.4 Distribuované súborové systémy a spracovanie dát

Mnohé zo spomenutých distribuovaných NoSQL databáz využívajú na ukladanie dát distribuované súborové systémy, ktoré ukladajú neštrukturalizované dáta. V praxi sa podarilo presadiť trom technológiám a to GFS od spoločnosti Google, S3 od spoločnosti Amazon a HDFS (Hadoop Distributed File System) od organizácie Apache (Šeleng et al. 2011). My sa bližšie pozrieme na distribuovaný súborový systém HDFS. HDFS bol navrhnutý na ukladanie veľkých dátových množín a sprístupnenie dát používateľom alebo aplikáciám s veľkou priepustnosťou. Dizajnovaný bol pre nasadenie v bežne dostupných počítačoch alebo serveroch a je prispôsobený na vysokú úroveň chýb uzlov v sieti. V HDFS sú metadáta ukladané separátne od samotných dát na dedikovanom serveri - Namenode, ktoré popisujú dáta uskladnené na iných serverov nazývaných Datanodes. Serveri sú plne prepojené a komunikujú na základe TCP protokolu. HDFS využíva hierarchickú štruktúru ukladania dát a vnútornú replikáciu dát na zabezpečenie spoľahlivosti a zvýšenie priepustnosti systému. Spoločnosť Yahoo[25] využíva veľké dátové klastre obsahujúce až 40000 strojov poskytujúcich 40PB on-line priestoru na ukladanie dát.

Mapreduce a Hadoop

Apache Hadoop [26] predstavuje softvérový rámec napísaný v jazyku Java, umožňujúci škálovateľné distribuované spracovanie veľkých dátovych množín využitím viacerých výpočtových jednotiek organizovaných do klastrov. Platforma Hadoop je veľmi populárna a je možné na nej prevádzkovať široké portfólio Apache projektov ako napr. Mahout[27] (nástroj implementujúci techniky strojového učenia), Hive [28] (dátový sklad), Pig [29] (paralelizácia výpočtových úloh) alebo Spark, na ktorý sa pozrieme detailnejšie. Najpopulárnejšou sa ale stala integrácia s jedným z jeho už dnes vnútorných komponentov – rámcom MapReduce spracúvajúcim dáta z HDFS.

MapReduce je programovací model, ktorý bol prvotne vyvinutý v spoločnosti Google (Dean and Ghemawat 2008) na spracovanie paralelizovateľných problémov nad veľkými dátovými množinami. Umožňuje spracovávať štruktúrované aj neštruktúrované dáta. Programovací model je inšpirovaný funkcionálnym jazykom Lisp, do ktorého vstupuje a vystupuje množina kľúčov a hodnôt a pozostáva a z dvoch operácií. Funkcia *map* vykonáva filtrovanie a usporadúvanie dát, pričom funkcia *reduce* má za úlohu spojiť čiastkové výsledky. Tieto procesy sú jednoducho paralelizovateľné a ľahko distribuovateľné medzi viacero strojov.

Bližší ilustrovaný pohľad na priebeh spracovania v modeli MapReduce ponúkame na Obr. 5, ktorý pozostáva z nasledujúcich operácií (Dean and Ghemawat 2008):

[23] http://neo4j.com/
[24] http://thinkaurelius.github.io/titan/
[25] http://www.aosabook.org/en/hdfs.html
[26] https://hadoop.apache.org/
[27] http://mahout.apache.org/
[28] http://hive.apache.org/
[29] http://pig.apache.org/

1. Knižnica MapReduce rozdelí dáta na vstupe do M častí. Toto rozdelovanie je parametrizovateľné používateľom a manažuje ho používateľova aplikácia. Následne systém MapReduce vytvorí vhodný počet inštancií na jednotlivých strojoch.

2. Jedna z inštancií sa stane riadiacou (angl. master) – nadradenou riadiacou jednotkou, ktorá bude pracujúcim inštanciám zadeľovať úlohy spracúvajúce podmnožiny dát, či už vo forme map alebo reduce operácií.

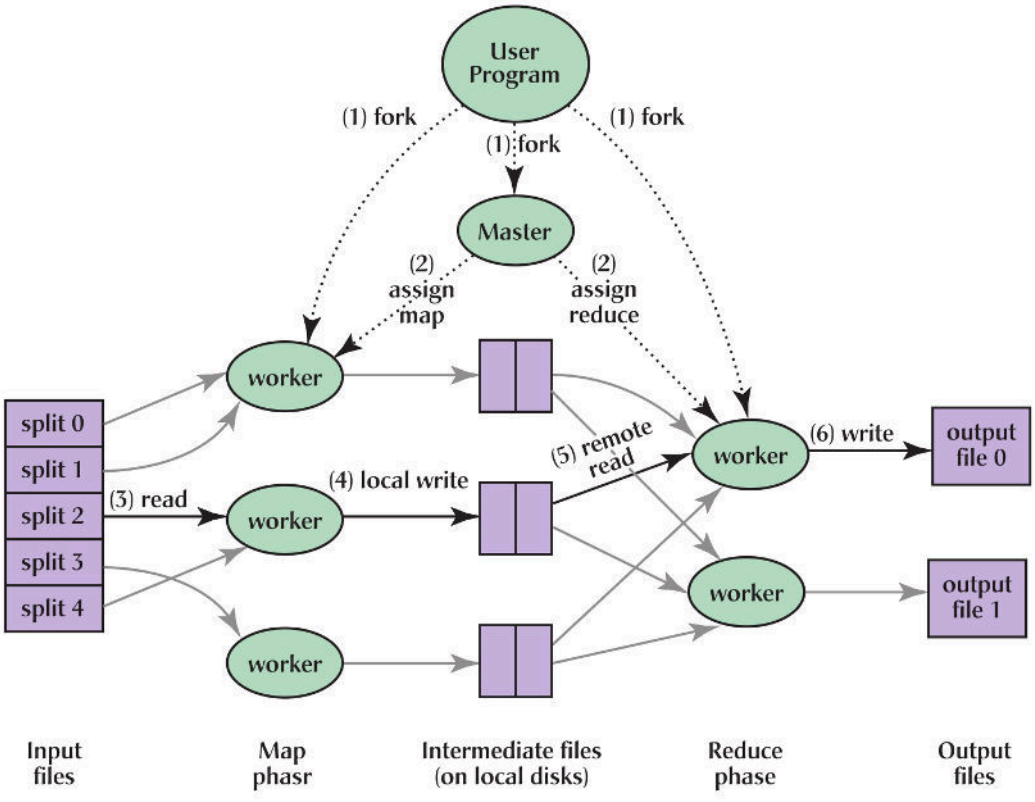
3. Pracujúca inštancia, ktorej je priradená map operácia sparsuje vstupnú podmnožinu dát. Množinu kľúčov a hodnôt spracuje a medzivýsledok načíta do pamäte.

4. Takto načítane medzivýsledky sú periodicky zapisované na disk, delené do R častí na základe deliacej funkcie. Toto rozdelenie je preposielané riadiacej inštancii, ktorý sa stará o preposlanie informácií ďalej pracujúcim inštanciám s úlohou reduce.

5. V momente, keď čakajúca inštancia s úlohou reduce získa informácie o lokácii medzivýsledkov na disku, ich pomocou volania vzdialenej procedúry prečíta a preusporiada.

6. Redukujúca (angl. reduce) inštancia iteruje cez usporiadané medzivýsledky a posúva ďalej množinu kľúčov aj hodnôt. Tento výstup je zapísaný na koniec výstupu danej dátovej partície.

7. Ak všetky map inštancie dokončili svoju prácu, master sa vráti do klientskej aplikácie a pokračuje v behu aplikácie.



Obr. 5 Spracovanie úlohy v modely MapReduce (Dean and Ghemawat 2008)

Hadoop a Mapreduce je v doméne odporúčacích systémov využívaný najmä na dávkové spracovanie off-line trénovania modelu používateľa alebo učenia sa odporúčacieho algoritmu (Das 2007; Sumbaly et al. 2013; Amatriain 2013; Niek 2014; Schelter et al. 2013).

Spark

Apache Spark[30] predstavuje relatívne mladú platformu využiteľnú na rýchle výpočty priamo v pamäti. Umožňuje distribúciu výpočtových úloh medzi uzlami klastra a poskytuje rozhranie na implementáciu programov nad klastrom. Oproti programovaciemu modelu MapReduce Spark poskytuje aj prúdové spracovanie dát a interaktívne dopyty, ďalej je vhodný aj na výpočet iteratívnych algoritmov. Spark umožňuje podobne ako MapReduce nie len dávkové spracovanie dát, ktoré je násobné rýchlejšie v porovnaní s MapReduce, ale aj spracovanie dát priamo v pamäti. Spark je možné prevádzkovať na platforme Hadoop, YARN alebo aj úplne nezávisle. Využíva koncept RDD (angl. resilient distributed dataset), ktorý modeluje kolekciu elementov spracovateľnú paralelne, s toleranciou chýb. Zdrojmi dát môžu byť stĺpcové databázy Cassandra, HBase alebo distribuované súborové systémy ako S3 alebo HDFS. Poskytuje 4 vysoko úrovňové nástroje – SparkSQL (procesovanie štruktúrovaých dát), Mlib (strojové učenie), GraphX (spracovanie grafov) a Spark Streaming (prúdové spracovanie dát). Spark bol v doméne odporúčacích systémov využitý napr. na zrýchlenie trénovania algoritmu ListNet metódy learning to rank (Shukla et al. 2012).

[30] https://spark.apache.org/

4 Architektúry existujúcich odporúčacích systémov

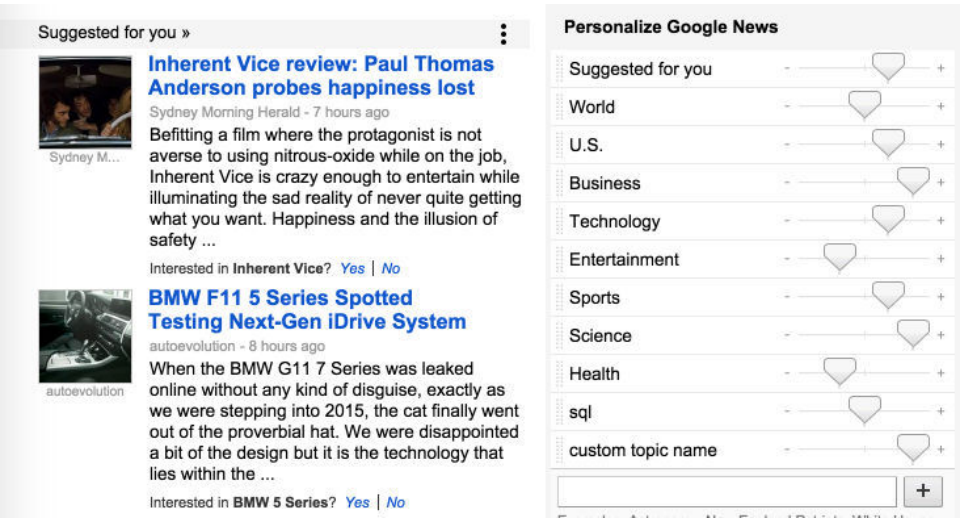
Analýzou personalizovaných odporúčacích systémov sa zaoberali vo svojich prácach mnohí autori (Tintarev et al. 2011; Bobadilla et al. 2013; Adomavicius and Tuzhilin 2005). My sa v tejto kapitole práce zameriame na existujúce odporúčacie systémy komerčných aj akademických oblastí a budeme sa orientovať na ich softvérový návrh, architektúru a škálovateľnosť. Pozrieme sa bližšie na použité prístupy, technológie, spôsoby ukladania dát, algoritmy a ich vhodnosť pre škálovateľné odporúčacie systémy.

4.1 Komerčné odporúčacie systémy

V tejto podkapitole sme analyzovali systémy pionierskych spoločností, ktoré do samotného výskumu v oblasti odporúčania investujú nemalé prostriedky a denne obsluhujú enormne množstvá používateľov, čo vypovedá aj o škálovateľnosti ich riešení.

4.1.1 Google News

Google News [31] je dnes už neaktívny produkt spoločnosti Google, ktorý fungoval ako agregátor novinových článkov z rôznych zdrojov a oblastí. Zaujímavou možnosťou systému je prispôsobenie si obsahu, zobrazené na Obr. 6. Spôsob, ktorým v Google odporúčajú články, označujú (Das et al. 2007) za doménovo nezávislé riešenie, agnostické od obsahu článkov overené na miliónoch požívateľoch a objektoch.



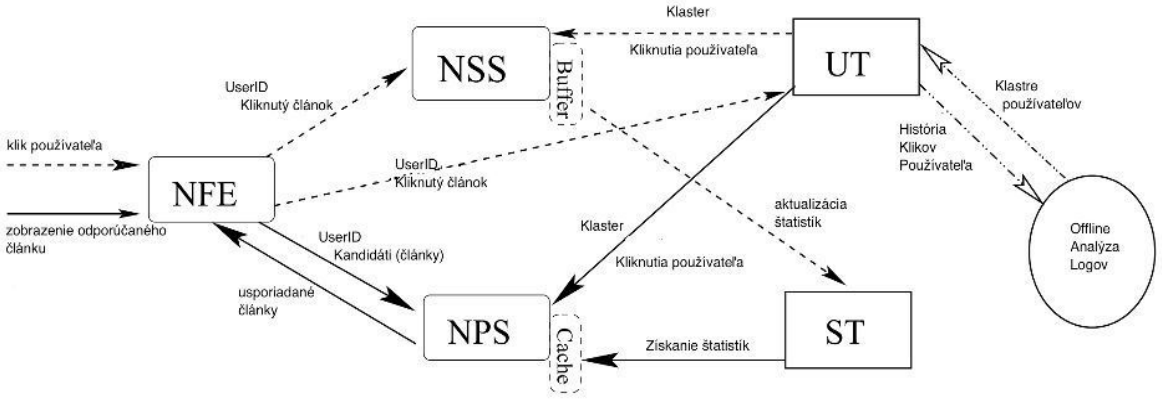
Obr. 6 Možnosti dodatočnej úpravy používateľského profilu v službe Google News

Doména aktuálnych správ sa vyznačuje frekventovanými zmenami už existujúcich príspevkov, a tak je nutné časté prepočítavanie existujúcich modelov, ktoré stoja za odporúčaniami. Spolu s požiadavkami na škálovateľnosť, schopnosť generovať odporúčania v reálnom čase (max. stovky ms) aj to bol dôvod, pre ktorý sa rozhodli vytvoriť vlastnú formu kolaboratívneho odporúčania založenú na lineárnej kombinácii viacerých techník. Využívaná je modifikovaná verzia PLSI, ktorá redukuje výpočtovú zložitosť a tak pri paralelizácii je

[31] https://news.google.com

možné aj frekventované prepočítavanie modelu s ohľadom na dynamickosť využitej metódy zhlukovanie pomocou pravdepodobnostnej metódy hašovania (angl. MinHash). Ďalšou technikou sú spoločné návštevy – kliknutia na dva príspevky v krátkom časovom intervale, kde sa do úvahy berie ich normalizovaný počet. Spoločné návštevy sú ukladané ako graf v úložisku *Bigtable*. Bližší pohľad na distribuované úložisko *Bigtable* je dostupný v (Chang et al. 2006), kde je okrem detailného popisu internej štruktúry a spôsobu ukladania dát opísané, ako s ňou pracujú aj iné systémy spoločnosti Google.

Odporúčací systém je zložený z troch hlavných komponentov. Off-line komponent je zodpovedný za periodické zhlukovanie používateľov na základe ich predchádzajúceho správania. S týmto komponentom pracujú dva druhy online serverov, kde jeden slúži na aktualizáciu štatistík jednotlivých príspevkov (NSS) a používateľov a druhý slúži na generovanie odporúčaní pre používateľov (NPS). Tretím online prvkom je tzv. modul novinového stánku (NFE), ktorý slúži na interakciu systému s používateľom a zaznamenávanie používateľských akcií. V off-line procesovaní systém využíva model *MapReduce*, a to pri výpočte modelov oboma technikami – *PLSI* a *MinHash*. Na Obr. 7 je okrem popisovaných komponentov viditeľný aj tok akcií a dát medzi komponentami.



Obr. 7 Komponenty odporúčacieho systému v Google News (Das et al. 2007).

V porovnaní s najpopulárnejšími riešeniami (napr. Spoločný výskyt dvoch udalostí) dosahovali personalizované odporúčania využitím *PLSI* a *MinHash* približne o 1/3 vyšší CTR (angl. Click through rate). Najlepšie výsledky dosahovali odporúčania generované pomocou PLSI a ako prekvapivé sa ukázalo nekombinovať jednotlivé metódy odporúčaní, pretože vedú k horším výsledkom. Napriek limitom architektúry súvisiacich s obmedzenou pamäťou serverov dokázali škálovateľnosť ich riešenia, pri zachovaní vysokej kvality odporúčaní.

Na vyššie uvedený výskum nadviazali Liu et al v roku 2010, kedy obohatili existujúci systém generovaním odporúčaní na základe obsahu (Liu et al. 2010). Sledovali vplyv lokálnych novinových článkov na preferencie používateľa. Rozhodli sa ho oddeliť od skutočných vlastných preferencií používateľa. Na modelovanie týchto preferencií a generovanie odporúčaní použili *Bayesov klasifikátor*. Týmto hybridným prístupom sa im podarilo zvýšiť návštevnosť a kvalitu odporúčaní.

4.1.2 Netflix

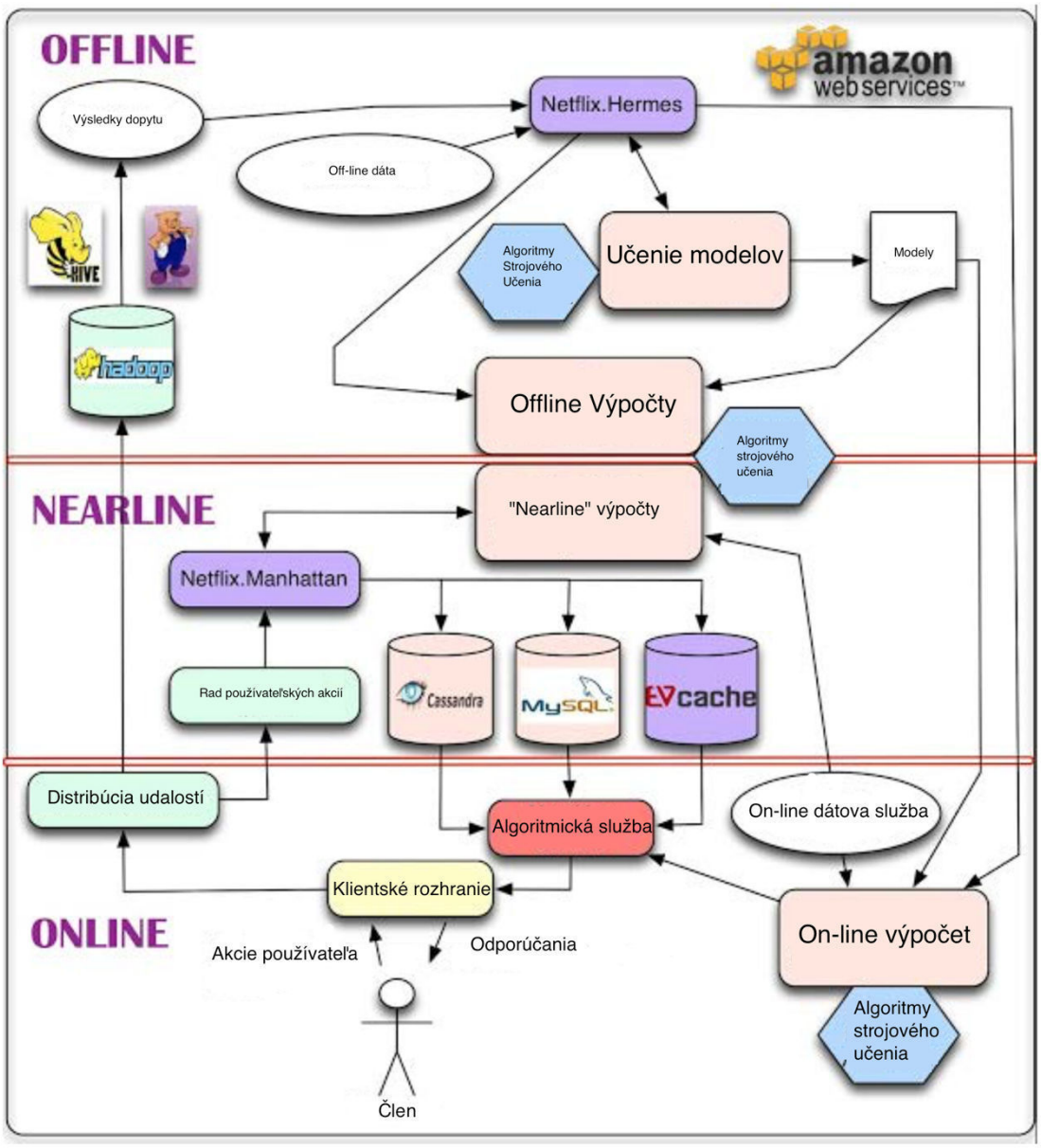
Spoločnosť Netflix vyhlásila v roku 2006 súťaž[32] o 1milión USD, ktorej cieľom bolo vylepšenie existujúceho personalizovaného odporúčania donášky DVD filmov do poštových schránok. Napriek tomu, že víťaza našli a riešenie bolo úspešnejšie o viac ako 8% voči existujúcemu systému Cinematch v metrike RMSE, konkrétne riešenie nebolo nikdy nasadené kompletne, ale niektoré využívajú aj dnes. Bell a Koren diskutujú vo svojej práci použité prístupy, ktorými dosiahli požadované riešenie (Bell and Koren 2007). Z ich skúseností sa ukazuje najvýhodnejšie použiť kombináciu viacerých prístupov pri odporúčaní. Najvyššie zlepšenie pripisujú rozdeleniu tvorby používateľských modelov do troch častí. Ako prvú uvádzajú zvyšovanie kvality modelov pomocou ich stavania na hlbších a detailnejších informáciách. Druhou častou je využívanie dvoch oddelených prístupov pri hľadaní najbližších susedov, kde pri vzdialenejších používajú latentné vlastnosti modelov. Poslednou časťou je binárny pohľad na správanie sa používateľa, kde sa zameriavajú na predikciu, či daný film používateľ hodnotil, namiesto stupňa samotného hodnotenia. Pri generovaní odporúčaní využívajú techniky ako k-NN, faktorizácia s ohľadom na okolie susedov, NSVD alebo Boltzmannove automaty. Ďalším prínosom od Koren et al bolo vylepšenie existujúcich techník faktorizácie matíc pomocou SVD++ (Koren 2008), kde prezentovali vysokú škálovateľnosť procesu faktorizácie, možnosti kombinácie implicitnej aj explicitnej spätnej väzby, pridanie schopnosti generovania vysvetlení pri jednotlivých odporúčaniach a pri ich algoritme nie je potrebná parametrizácia používateľov, ani nutnosť pretrénovania modelov pri pridávaní nových používateľov do systému.

Objem dát narástol nielen o hodnotenia, ale aj o detailné analýzy správania sa používateľov v ich systéme. Amatarian v (Amatriain 2013) diskutuje využívané postupy na spracovanie veľkého množstva dát zahŕňajúce strojové učenie a architektúry, ktoré im umožňujú kombinovať off-line dávkové spracovanie s prúdmi odporúčaní reálneho času. Netlflix generuje odporúčanie nie len jednotlivým používateľom, ale napr. aj domácnosti ako skupine, takže dôležitú rolu pri odporúčaní hrá aj rôznorodosť a transparentné vysvetlenie odporúčaní používateľom.

Z pohľadu dát obsahuje systém Netflix miliardy hodnotení, a milióny pribúdajú denne, pričom okrem priamych hodnotení pracujú aj s impresiami – zobrazeniami jednotlivých odporúčaní objektov, najnovšie aj informáciami zo sociálnych sietí. Na predikciu hodnotenia používajú aj demografické a časové údaje, do úvahy berú globálnu popularitu jednotlivých objektov. Pri modelovaní odporúčaní využívajú strojové učenie, zhlukovanie, lineárnu regresiu a ďalšie metódy uvedené v (Amatriain 2013).

Pri charaktere našej práce je najzaujímavejšie pozrieť sa detailnejšie na použitú architektúru systému, načrtnutú na Obr. 8.

[32] http://www.thoughtgadgets.com/why-netflix-walked-away-from-personalization/



Obr. 8 Architektúra odporúčacích systémov v systéme Netflix (Amatriain 2013)

Z pohľadu architektúry využíva Netflix pri off-line predpočítavaní škálovateľné riešenie Apache Hadoop, pri ktorom nedeklarujú limity architektúry. Nad ním sú dáta dávkovo spracovávané nástrojmi Hive a Pig. Na tejto úrovni systém vytvára a trénuje modely a generuje medzivýsledky použiteľné pri „Nearline“ spracovaní. Pri tzv. “Nearline” spracovaní, ktoré je kompromisom medzi off-line a on-line počítaním, je výpočet realizovaný rovnako ako v on-line počítaní, bez zavedenia časového limitu, čo umožňuje ukladanie výsledkov a aplikovanie asynchrónneho prístupu k spracovaniu dát a poskytnutí odpovede vo forme odporúčaní. Pri on-line časti generovania odporúčaní sú opäť využívané algoritmy strojového učenia, a nastáva tu kombinácia off-line výsledkov s online akciami používateľa. Pri on-line spracovaní využívajú klasickú relačnú databázu (MySQL), stĺpcovú databázu z kategórie NoSQL (Cassandra) a pri volatílnom spracovaní vybranej temporálnej množiny dát využívajú distribuované úložisko (EVCache [33]). Z proprietárnych interných riešení využívajú Netflix Hermes – ktorý plní funkciu komunikačnej služby na základe správ

[33] https://github.com/Netflix/EVCache

(návrhový vzor Publish-Subscribe) a distribuovaný systém Netflix Manhattan, ktorý slúži na zachytávanie akcií používateľov v systéme.

4.1.3 Amazon

G.Linden, B.Smith, a J. York diskutovali vo svojom reporte riešenie využívané americkým obchodným gigantom Amazon, ktoré využíva kolaboratívne filtrovanie hľadaním využitím podobnosti objektov (angl. *item-to-item collaborative filtering)* (Linden et al. 2003). Spomínaný algoritmus označujú za nezávislý od počtu používateľov alebo produktov v obchode, schopný odporúčania v reálnom čase, generujúci vynikajúce používateľské odporúčania a škálovateľný až do úrovne masívnych dátových množín. Pomocou personalizovaného odporúčania dosahujú vyššie (bližšie neuvedené) marketingové výsledky v metrikách, ako pri odporúčaní napr. všeobecne najpredávanejších produktov v danom období. Podstatou ich algoritmu je pri kolaboratívnom odporúčaní hľadanie podobných produktov, ktoré budú neskôr priamo odporučené, namiesto hľadania podobných používateľov a porovnávania hodnotení ich produktov. Náročná časť výpočtu podobnosti medzi produktami zakúpenými zákazníkmi je realizovaná off-line, pričom v reálnom čase je realizované len vyhľadávanie medzi podobnými produktami k tým, ktoré si zákazník už zakúpil alebo hodnotil. Tým pádom je použitý prístup vysoko škálovateľný. Ich algoritmus funguje rovnako dobre aj pri používateľovi s malým objemom hodnotených produktov. Ukážka odporúčaných produktov je zobrazená na Obr. 9.

Pri určovaní najpodobnejších podobných produktov algoritmus vytvára tabuľky podobných produktov tým, že vyhľadáva tovar, ktorý zákazníci obvykle nakupujú spolu. Podobnosť medzi dvoma produktami je následne vypočítaná cez kosínusovú podobnosť ich vektorov. Tento výpočet je časovo náročná operácia, ktorá je ohraničená ako $O = (n^2 m)$, kde n je počet rozličných produktov a m je počet dimenzií vektora reprezentujúcich produkty, pričom priemerne je možné ju ohraničiť ako $O = (nm)$, pretože väčšina zákazníkov realizovala minimálny objem nakúpených produktov v porovnaní s celým dostupným katalógom. Ak uvážime pri výpočte podobnosti zúžiť množinu používateľov, a použiť iba ich reprezentatívnu vzorku, je výpočet oveľa jednoduchší a kvalita odporúčaní týmto zásahom výrazne netrpí. S takto získanou tabuľkou podobných produktov už ľahko algoritmus nájde produkty podobné tým, ktoré používateľ kladne hodnotil alebo zakúpil v minulosti, agreguje ich a odporučí tie najpopulárnejšie, prípadne tie, ktoré dosahujú najvyššiu úroveň korelácie.

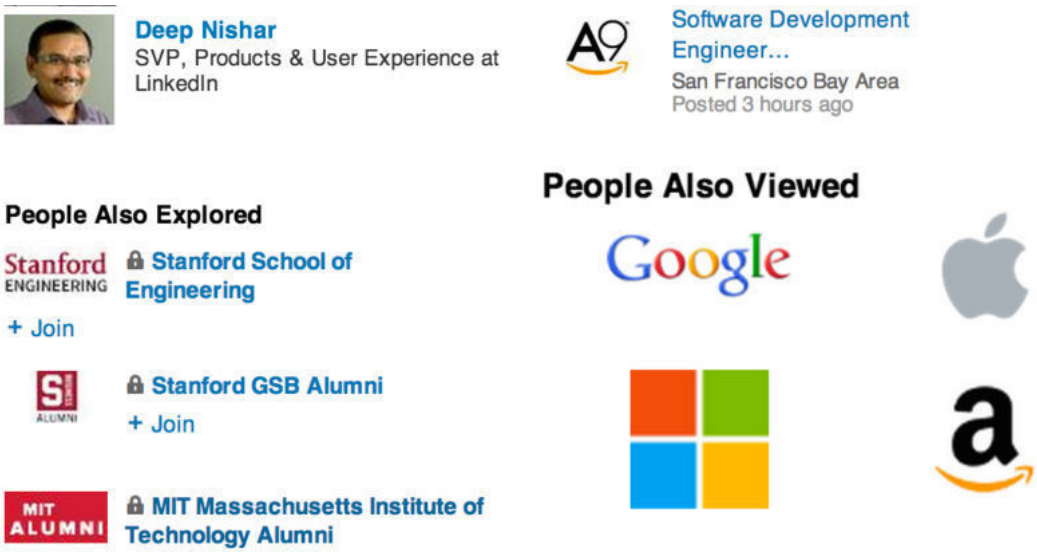
Algoritmus 1 Výpočet podobnosti medzi produktom a množinou podobných produktov.

```
Pre každý produkt i₁ z produktového katalógu
 Pre každého zákazníka C, ktorý kúpil produkt i₁
  Pre každý produkt i₂ zakúpený zákazníkom C
   B << i₂
   Pre každý produkt i₂ z množiny B
    Vypočítaj podobnosť medzi i₁ a i₂
```

Obr. 9 Odporúčania pri nákupe vygenerované v systéme Amazon

4.1.4 LinkedIn

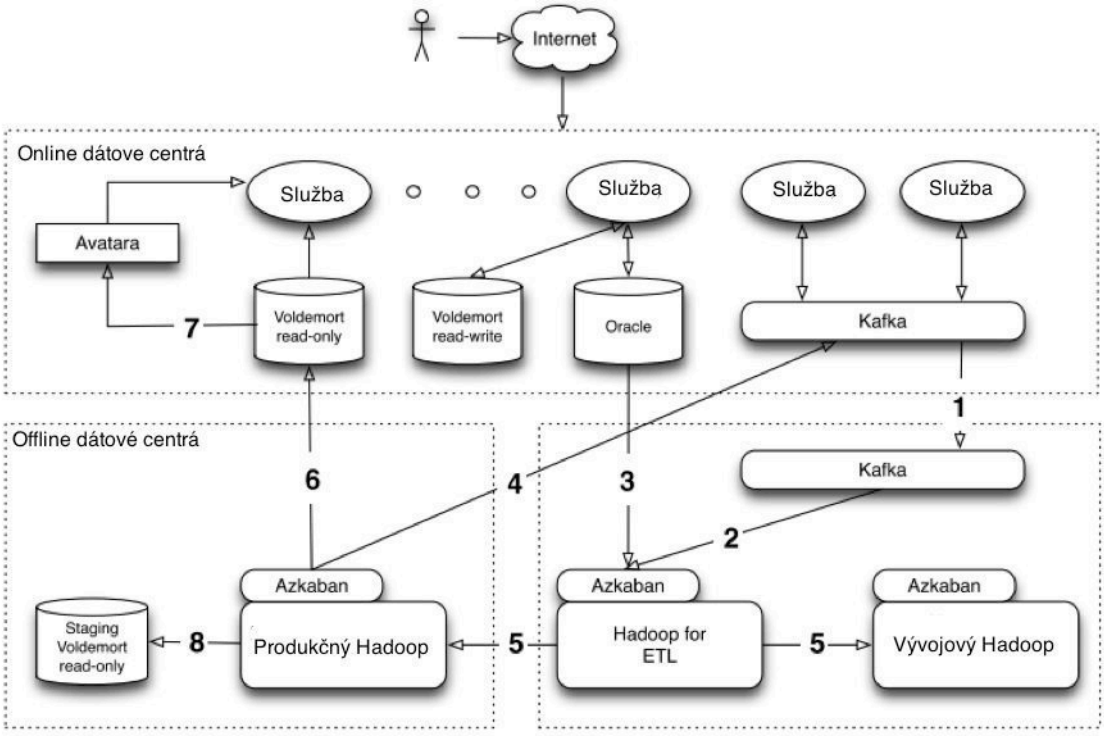
LinkedIn[34] je globálna sociálna sieť so zameraním sa na profesný život, kariéru a prácu. Používateľská základňa obsahuje 350 miliónov používateľov, z ktorých až 40% navštevuje sociálnu sieť denne. LinkedIn využíva odporúčania v rôznych formách, niektoré zobrazené na Obr. 10 – pridanie nových priateľov alebo personalizované upozornenie na zaujímavé pracovné ponuky, skupiny a spoločnosti na sociálnej sieti. V posledných rokoch systém LinkedIn čiastočne prechádza transformáciou z off-line dávkového predpočítavania pomocou nástroja Hadoop na architektúru schopnú generovania odporúčaní v reálnom čase.



Obr. 10 Pohľad na rôzne formy odporúčaní na sociálnej sieti LinkedIn

[34] https://www.linkedin.com/

Spoločnosť LinkedIn publikovala veľmi detailný opis svojej distribuovanej architektúry na zber a analýzu dát, viditeľný na Obr. 11 (Sumbaly et al. 2013). Nad takto získanými dátami pracujú odporúčacie systémy alebo rôzne analytické techniky strojového učenia. Pre výskumníkov vytvorili prostredie, v ktorom dokážu sebestačne pracovať s dátami, bez nutnosti existencie vývojárskeho tímu alebo akejkoľvek pomoci. Využívajú rôzne moderné technológie, napr. Pig, Hive, MapReduce, Mahout, pričom niektoré z nich boli vyvinuté priamo spoločnosťou LinkedIn (Kafka[35]). Všetky časti systému sú horizontálne škálovateľné, multiúrovňové a odolné voči zlyhaniu a chybám pri spracovaní. Dáta získané interakciou používateľov so sociálnou sieťou putujú do off-line systémov za účelom vytvorenia rôznych dátových reprezentácií, s ktorými je možné neskôr pracovať v reálnom čase. LinkedIn používa pre Hadoop súborovú štruktúru HDFS.



Obr. 11 Architektúra a tok dát v sociálnej sieti LinkedIn (Sumbaly et al. 2013)

Primárnym prenášačom dát je práve Kafka, ktorá je navrhnutá využitím vzoru komunikácie pomocou správ – *Publish-Subscribe*. Udalosti používateľov sú zhlukované na základe typu a pomocou správ sú publikované do systému. Druhým zdrojom dát sú kópie databáz, ktoré sú uchovávané formou denných verzií. Prichádzajúce dáta sú validované a je tak dodržaná dátová integrita, celý proces je monitorovaný. Ukladané sú do hierarchických štruktúr súborového systému HDFS. Keď sú už dáta prístupne v ETL nástrojoch, sú replikované do dvoch inštancií systému Hadoop – produkčnej a vývojovej, kde dátami pracujú analytici a výskumníci. Vkladanie dát do systému Hadoop je realizované pomocou načasovaných úloh definovaných v nástroji Azkaban, ktoré poskytuje monitorovanie a zber logov. Dáta sú zo systému Hadoop ukladané do distribuovaného on-line úložiska *Voldemort*[36] typu kľúč-hodnota a multidimenzionálnych OLAP[37] kociek, použitím technológie Avatara[38]. Následne

[35] http://kafka.apache.org/
[36] http://www.project-voldemort.com/voldemort/
[37] Online Analytical Processing

umožňujú zjednodušenú prácu s ďalšími systémami ako Pig. Dáta sú potom finálne nahrávané naspäť do online systémov.

LinkedIn Browsemaps

Browsemaps je platforma odporúčacieho systému vyvinutá spoločnosťou LinekdIn, založená na kolaboratívnom prístupe využívajúceho podobnosť objektov. Jedná sa o hybridný systém, ktorý kombinuje off-line a on-line spracovanie. Na základe predchádzajúceho správania sa a navigácie používateľa vytvára latentne grafy spoločných výskytov entít. Jeho unikátnosť spočíva v jednoduchej implementácií a pridávaní ďalších typov odporúčaní. Browsemaps je použitý pri 12 rozličných typoch odporúčaní na sociálnej sieti (podobní používatelia, prislúchajúce vyhľadávania, návrhy na aktualizáciu používateľských profilov, alebo odporúčania vo forme profilov spoločností, ktoré používateľa zaujímajú. Z pohľadu architektúry sa jedná o horizontálny systém s viacerými zdieľanými komponentami, kde biznis logika (konkr. odporúčania) je stavaná ako vertikálna špecifickosť jednotlivých typov odporúčaní. Podporuje centralizované monitorovanie a jednoduchú škálovateľnosť. Využitím systému Browsemap je možné pridať nové typy odporúčaní do produkčného prostredia v priebehu 1-2 dní.

Na strane klienta vytvára front-endový rámec udalosti aktivít pri každom zobrazení webovej stránky. Tieto aktivity sú vopred definovaným prúdom dát nahrávané do systému za účelom konštrukcie matíc spoločného výskytu entít. Jednotlivé mapy (angl. browsemaps) sú inkrementálne vypočítané off-line pomocou systému *Hadoop* – oddelene pre každú entitu (130 rozličných úloh generujúcich terabajty dát týždenne). Dáta ďalej dávkovo smerujú do distribuovaného on-line úložiska *Voldem*ort typu kľúč-hodnota, odkiaľ sú dotazované pomocou online API. Online Api je agnostické od typu odporúčaných objektov, pričom 99% požiadaviek je obslúžených do 10ms.

Detailnejší pohľad a konkrétne implementácie (Amin et al. 2012; Huang et al. 2013) a vyhodnocovanie odporúčaní (Rodriguez et al. 2012) sú publikované a voľne dostupné. Optimalizáciou kvality odporúčaní pri zohľadnení viacerých kritérií sa zaoberali Rodriguez et al (Rodriguez et al. 2012). Ich model optimalizácie prezentujú ako ľahko pridateľný do existujúcich prúdov spracovania dát, model nepotrebuje žiadnu trénovaciu množinu a je nezávislý od domény, alebo vlastností odporúčaných objektov. V práci uvádzajú konkrétny príklad implementácie v systéme TalentMatch, ktorého úlohou je hľadanie vhodných kandidátov na pracovné pozície.

4.1.5 Tivo

TiVo [39] predstavuje službu, ktorej úlohou je nahrávanie televíznych programov a ich kategorizácia pre ľahšie vyhľadávanie (Ali and van Stam 2004). Odporúčací systém využíva klient-server architektúru, pri ktorej pracuje s Tivo klientskymi boxami v domácnostiach. Odporúčania sú v systéme generované pre domácnosti, Tivo nerozlišuje jednotlivých používateľov a neodporúča televízne vysielanie pre skupiny. Ich systém neperzistuje žiadne používateľské dáta na serveroch, čím zabezpečuje vysokú ochranu súkromia. TiVo hodnotí série televíznych programov ako jeden objekt. Preferencie používateľov resp. domácnosti sa ukazujú ako stabilné až do úrovne mesiacov.

[38] http://engineering.linkedin.com/olap/avatara-olap-web-scale-analytics-products
[39] http://www.tivo.com/

Na hodnotenie televíznych programov používateľmi Tivo používa implicitnú (nahratie televízneho programu, ktorý používateľ ešte nevidel) aj explicitnú spätnú väzbu (3 úrovne pre pozitívne hodnotenie, 3 pre negatívne). Generovanie odporúčaní je realizované kombináciou dvoch algoritmov – *Bayeseianovský algoritmus* na základe obsahu, ktorý generuje odporúčania primárne pri probléme studeného štartu, a kolaboratívne filtrovanie formou modelovania vlastností jednotlivých televíznych nahrávok.

Sekvencia tvorby odporúčaní v systéme TiVo je prezentovaná postupnosťou nasledovných krokov:

1. Získavanie používateľskej spätnej väzby.
2. Prenos profilu používateľa z TiVo boxu na server (prenáša sa celý používateľský profil, výmena dát nie je inkrementálna).
3. Anonymizácia používateľského profilu.
4. Výpočet na strane servera – výpočet korelácii medzi dvojicami televíznych programov, ktoré spadajú do sekcie záujmu používateľa.
5. Výpočet na strane klienta – kolaboratívne odporúčanie nehodnotených televíznych programov, bez zaťažovania servera.
6. Vytvorenie zoznamu odporúčaní – obohatenie kroku č.6 o odporúčanie založené na obsahu
7. Automatické nahratie odporúčaných programov TiVo boxom.

Odporúčania sú generované periodicky, lokálne umiestnenými TiVo boxami na pozadí práce so systémom. Vstupom pre algoritmus uvedený ako pseudo kód v Algoritmus 2 Kolaboratívne odporúčanie v systéme TiVo, je množina televíznych programov, u ktorých sa očakáva predikcia hodnotenia – `Programy`, a množina korelácií dvojíc objektov – `Páry`.

Algoritmus 2 Kolaboratívne odporúčanie v systéme TiVo

```
Kolaboratívne(Programy, Páry):

 Pre každý nehodnotenie program P v Programoch

  nech Páry(P) sú podmnožinou Párov predikovaných pre P

  uvažuj elementy <P1, P, r(P1,P)> in Páry(P)

  zoraď Páry(P) s ohľadom na hodnotenie r

  vypočítaj váhovaný lineárny priemer ů

Predikcie := Predikcie <P, ů>

Výstup: predikcie zoradené na základe ů
```

Použitý algoritmus odporúčania je vhodne škálovateľný, lebo každé zariadenie TiVo generuje odporúčanie na pozadí len pre domácnosť v ktorej je umiestnené. Architektúra systému na strane serverov je postavená na horizontálne škálovateľnom distribuovanom modely, ktorý im umožňuje výpočet veľkého množstva párov televíznych programov a ich korelácií. Servery sú rozdelené do troch úrovní. Úlohou prvej úrovne je akceptovanie logov a správ z TiVo zariadení. Prichádzajúcim logom je priraďovaný fikčný identifikátor používateľa, uchovávaný len počas doby generovania aktuálnych korelácií. Výpočet korelácií je možné rozdeliť medzi viacero strojov druhej úrovne, ktorých výsledky sú spájané na základe fikčného identifikátora používateľa.

Servery druhej úrovne majú priradené rozsahy identifikátorov televíznych programov, a ich úlohou je spočítanie počtu unikátnych hodnotení pre televízne programy spadajúce do rozsahu jednotlivých serverov. Na tretej, a zároveň finálnej úrovni sú servery rozdelené nad priestorom párov televíznych programov. Prvotným rýchlym filtračným mechanizmom určí, či sa bude generovať korelácia medzi dvoma televíznymi programami. Ak áno, nasleduje drahšie počítanie korelácie pomocou matíc hodnotení rozmerov 7x7. Výsledkom je výpočet vo forme lineárnej korelácie, ktorú zvolili tvorcovia systému TiVo, pretože k hodnoteniam programov používateľmi sa stavia ako ku diskrétnym rádovým premenným, namiesto nominálnych premenným.

Škálovateľnosť systému TiVo je dosiahnutá dvomi cestami. Prvou z nich je spôsob, kedy sa negenerujú korelácie denne pre celú množinu televíznych programov, ale sú rozdeľované na podmnožiny veľkosti 1/N, ktoré sú aktualizované každých N dní, kedy obetúvajú aktuálnosť korelácií, za cenu zníženia nákladov. Druhým spôsobom je zavedenie parametra min. počtu párov hodnotení pri filtrovaní množiny, označovanom "min#pár" na základe počtu hodnotení párov televíznych programov, čím redukujú objem možných korelácií z 10^11 na 30 000 párov. Manipuláciou s týmto parametrom "min#pár"-hodnotení zvláda architektúra systému TiVo upravovať kvalitu odporúčaní s ohľadom na dostupné výpočtové zdroje. Druhým parametrom na redukciu množiny korelácií je min#hodnotení, ktorý určuje spodnú hranicu minimálneho počtu hodnotení unikátnymi používateľmi. Pomocou týchto parametrov je možné konfigurovať systém TiVo pre zvolenie vhodnej hladiny kvality odporúčaní pri náraste počtu používateľov.

4.1.6 Yahoo

V spoločnosti Yahoo[40] využívajú techniky strojového učenia a formy odporúčaní na veľkom množstve prvkov ich webových portálov alebo aj ich systéme Yahoo Recommends[41] (viď Obr. 12) využiteľnom aj v externých systémoch. V (Agarwal et al. 2010) vytvorili metódu *FOBFM* (angl. *Fast Online Bilinear Factor Model*), ktorej cieľom je rýchle učenie sa vlastností odporúčaných objektov pomocou on-line regresie. Jej využitie je vhodné pri odporúčaní webových stránok reklám alebo novinových článkov, kedy je nutné pozerať sa aj na modelovanie času pri generovaní odporúčaní. V nej je on-line výpočet pre jednotlivé objekty nezávislý od ostatných a preto je metóda rýchla, ľahko paralelizovateľná a škálovateľná. Hlavný princíp, ktorý metóda využíva je použitie veľkého množstva historických dát na inicializáciu on-line modelov na základe off-line získaných vlastností. Tým získavajú lineárne

[40] https://www.yahoo.com/
[41] http://yahoo.tumblr.com/post/96977561949/a-new-publisher-solution-yahoo-recommends

projekcie, ktoré efektívne redukujú dimenzionalitu priestoru, z ktorého sú odporúčania generované.

Pri off-line fitovaní modelu využívajú algoritmus založený na algoritme očakávania-maximalizovania (angl. EM – expectation-maximization), kombináciami Gaussovského modelu, Gibbsovým vzorkovaním a pravdepodobnostného modelu Monte Carlo. Práve pri vytváraní Gibssových vzoriek je celý proces paralelizovateľný a nezávislý od ostatných používateľských modelov. Hlavným prínosom práce je rýchla online regresia nad získanými dátami z off-line trénovania modelov. Výstupom online spracovania sú regresné váhy $(b - váha\ posunu, A - váha\ preferencií, B - váha\ regresných\ váh)$, používateľské vlastnosti u_i, a predchádzajúca variancia. Online reprezentácia objektu je daná ako $s_{ijt} = x_{ij}b + u_{it}Ax_j + u_{it}B\theta_{jt}$, kde $x_{ij}b$ predstavuje posun, $u_{it}Ax_j$ je dimenzionálne zredukovaný vektor preferencií používateľa a θ_{jt} je vektor regresných váh. Keďže online model pre každý z objektov j je nezávislý od iných objektov a dimenzionalita θ_j je nízka, tak proces aktualizácie modelu preferencií používateľa je veľmi efektívny, škálovateľný a paralelizovateľný.

Pri overeniach na Yahoo a MovieLens datasetoch je najvýraznejšie zlepšenie v oblasti správneho usporadúvania odporúčaní (2-3 násobné). Pri sledovaní ROC (angl. Receiver Operating Characteristic) kriviek vykazuje najlepšie výsledky pri porovnaní s klasickým kolaboratívnym prístupom alebo prístupom využívajúcim on-line PLSI (Das et al. 2007) práve ich model FOBFM.

Obr. 12 Odporúčania generované systémom Yahoo Recommends

4.2 Akademické odporúčacie systémy

Z akademickej sféry sme analyzovali tri existujúce systémy, využívajúce odlišné prístupy k tvorbe odporúčaní, ktoré sú prezentované ako škálovateľné: Yoda, HyRec a Scene.

4.2.1 Yoda

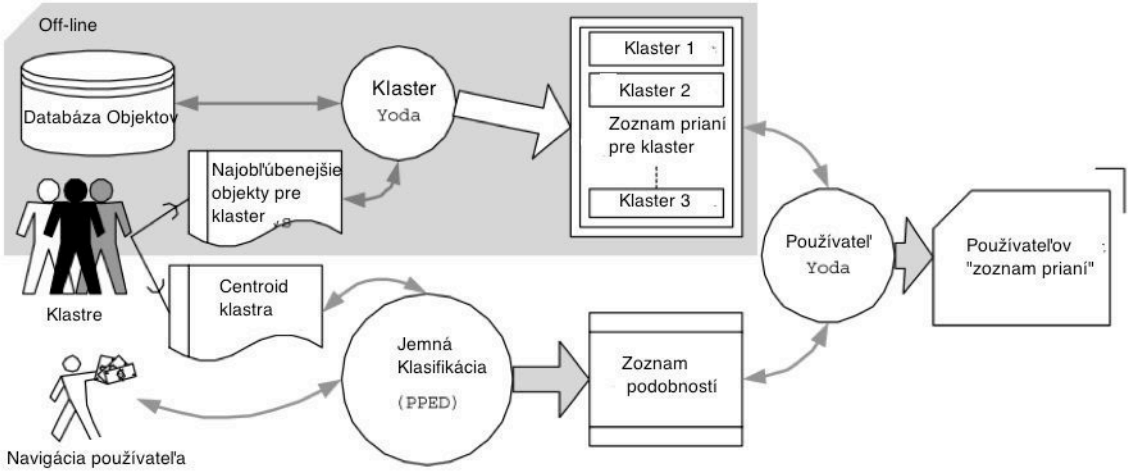
Prvým zástupcom existujúceho systému z akademickej oblasti je systém Yoda, ktorý jeho autori prezentujú ako odporúčací systém využiteľný najmä vo veľkých webových aplikáciách, vyžadujúcich presné odporúčania v reálnom čase (Shahabi et al. 2001). Yoda bol vytvorený za účelom odporúčania hudobných CD nosičov. Ich cieľom pri tvorbe systému bola vysoká škálovateľnosť pre mohutné webové systémy, čo pre nich znamenalo nutnosť využitia off-line

predpočítania používateľských preferencií vo forme modelov. Ich postup odporúčania pozostáva z kolaboratívneho odporúčania, ktoré delia na 2 fázy:

1. Klasifikácia používateľov – získanie preferencií a klasifikácia používateľov formou zhlukovania.

2. Ohodnotenie objektov a generovanie odporúčaní – ich výsledné zoradenie.

Generovanie odporúčaní, viditeľné na Obr. 13, je realizované off-line predpočítaním podobnosti CD nosičov k množine vlastností (napr. hudobný žáner), ktorá je určená fuzzy pojmami. V časti off-line procesovania systém Yoda identifikuje skupiny používateľov pomocou zhlukovania z trénovacej množiny. Každému zhluku priradí preferencie (angl. favorite property values) a použije ich na vytvorenie odporúčaní pre zhluky. V on-line fáze (v reálnom čase) je používateľ priradený do zhlukov na základe jeho správania a spôsobu navigácie. *Yoda* následne použije klasifikačné faktory na vygenerovanie zoradeného zoznamu odporúčaní získaného váhovanou agregáciou odporúčaní pre jednotlivé zhluky používateľov, do ktorých bol používateľ priradený. Popisovaný algoritmus redukuje časovú komplexitu tvorby agregácií z pôvodne definovanej ako $O(||I||\times||P||)$ na $O(||I||)$, kde I predstavuje celkovú množinu objektov (konkr. CD nosičov) a P množinu vlastností priradených každému objektu.



Obr. 13 Pohľad na architektúru odporúčacieho systému Yoda (Shahabi et al. 2001)

Pri tvorbe používateľovho modelu a klasifikácii používateľov využili ich predchádzajúci výskum (Shahabi et al. 1995), v ktorom používajú matice vlastností, ktoré sú reprezentované hyperkockovými dátovými štruktúrami. Tie môžu reprezentovať rôzne agregované vlastnosti s požadovanými presnosťami. Matice vlastností môžu byť použité na modelovanie preferencií jednotlivých používateľov alebo zhluku používateľov. Yoda využíva matice vlastností na získanie individuálnych preferencií aktívnych používateľov a pri trénovacej off-line množine pracuje nad zhlukmi používateľov. Zaujímavejšia je ale ich metóda generovania odporúčaní, a to konkrétne mechanizmy na ich filtrovanie. Rozširujú *lokálne senzitívne hashovanie* (angl. LSH), ktoré slúži na redukciu komplexnosti vo viac dimenziálnom vyhľadávaní najbližších susedov. Nahradením Hammingovej vzdialenosti fuzzy metódou

vytvorili algoritmus *lokálne fuzzy senzitívneho hashovania*, ktorý redukuje potenciálny priestor riešenia generovania odporúčaní využitím agregačnej schopnosti zhlukov.

Výsledky ich experimentov ukazujú škálovateľnosť s rastúcim počtom používateľov (dokonca deklarujú až nezávislosť od množstva používateľov) alebo objektov v systéme. V aspekte presnosti dosahujú až 120% nárast oproti základným metódam využívajúcich podobnosť najbližších susedov pri generovaní odporúčaní.

4.2.2 HyRec

HyRec[42] je voľne dostupným príspevkom do kategórie akademických personalizovaných odporúčacích systémov, využívajúci architektúru typu klient-server (Boutet et al. 2014). Využíva kolaboratívne odporúčanie podobnosťou používateľov založené na hybridnej architektúre systému, ktorej náčrt uvádzame na

Obr. 14, ktorá kombinuje výhody centralizovaných a distribuovaných systémov. Pri implementácii riešenia Boutet využil k-NN (k – najbližších susedov) algoritmus, pričom existuje možnosť jeho nahradenia inými algoritmami, ktoré sú schopné rozdeliť výpočty do webových prehliadačov používateľov. Systém HyRec je vhodný pre domény v ktorých pribúda obsah rýchlo, očakáva sa frekventovaná zmena používateľových preferencií a odporúčania generované v reálnom čase.

Obr. 14 Hybridná architektúra odporúčacieho systému HyRec (Boutet et al. 2014)

Architektonické riešenie systému HyRec používa pri výpočte najbližších susedov vždy len vzorku okolitých používateľov – kandidátov. Tento spôsob vytvárania a selekcie používateľov zo vzoriek je inšpirovaný "klebetným" prístupom (Das et al. 2007; Bertier et al. 2010). Proces výpočtu je iteratívny a pri každom vypočítaní k*-NN* na strane používateľa systém komunikuje zo serverom a získava novú, aktualizovanú vzorku používateľov za účelom kvalitnejších odporúčaní. Systém sa skladá z dvoch modulov, ktorými sú HyRec server a HyRec klient. HyRec je implementovaný v jazykoch Java a Javascript a poskytuje webové API.

HyRec server

Úlohou servera je orchestrácia celého procesu prideľovania úloh klientom, udržiavanie globálnych dátových štruktúr vo forme používateľských a k-NN tabuliek. Server vytvára personalizované úlohy pre klientov, ktoré pozostávajú zo správy obsahujúcej profil používateľa a profily kandidátov, ktoré odošle klientovi. Systém je vďaka iteratívnemu formovaniu najbližších susedov pomocou vzorkovania odosielaných profilov kandidátov dynamickejší a presnejšie zachytáva aktuálne preferencie používateľa. Vedľajším efektom je znížený prenos dát, zníženie výpočtovej náročnosti (Dong et al. 2011) bez efektu na výslednú kvalitu a presnosť odporúčaní. Systém pridáva do množiny kandidátov aj náhodné profily používateľov, aby sa zamedzilo uviaznutiu na lokálnom maxime.

HyRec klient

Používatelia interagujú s odporúčacím systémom prostredníctvom webového rozhrania, ktorý je na strane klienta implementovaný javascriptovým skriptom bežiacim vo webovom

[42] http://gossple2.irisa.fr/~aboutet/hyrec/index.htm#bin

prehliadači. Klient odosiela serveru požiadavku vždy, keď používateľ potrebuje odporúčania. Server odpovedá klientovi s personalizovanou úlohou, obsahujúcej množinu kandidátov na najbližších susedov. Cieľom úlohy je vypočítať odporúčania pre používateľa a vybrať najbližších susedov, ktorých opäť odosiela naspať na server cez webové API, za účelom vytvorenia presnejších odporúčaní pomocou vylepšovania kvality vybraných najbližších susedov. Vďaka hybridnej asynchrónnej architektúre sa na strane klienta nemusia ukladať žiadne lokálne dáta a komunikácia je bezstavová.

Podľa experimentov a výpočtov Boutet et al. dosahujú až 100 násobne vyššiu škálovateľnosť v porovnaní s čisto centralizovanými systémami. Operačné náklady na infraštruktúru dosahujú len 50% nákladov centralizovaných systémov. HyRec využíva len 3% objemu dát prenášaných pri systémoch založených na báze P2P[43], systémové zdroje používateľov vyťažuje len minimálne a na krátku dobu (menej ako 60ms) a je využiteľný aj na mobilných zariadeniach vďaka asynchrónnej architektúre riešenia. Pre bežné použitie prináša aj bezpečnostné opatrenia proti získavaniu informácií o používateľských profiloch – dynamické pregenerovanie identifikátorov profilov, každý používateľ generuje odporúčania iba pre seba, čo v prípade P2P neplatí. Autori uvádzajú nezávislosť efektívnosti systému od veľkosti profilov používateľov alebo počtu používateľov. HyRec je doménovo nezávislé riešenie, ktoré poskytuje priestor na optimalizáciu používateľských profilov v závislosti od domény. Pri overeniach využili dátové sety služieb Digg [44] a Movielens.

4.2.3 Scene

SCENE (angl. SCalable two-stage pErsonalized News rEcommendation) je dvojúrovňový škálovateľný odporúčací systém vytvorený špeciálne za účelom využitia v doméne webových novinových článkov a správ (Li et al. 2011). V prvej úrovni selektuje vybrané témy zaujímavé a špecifické pre používateľa a v druhej úrovni odporúča konkrétne články. Pri bližšom pohľade, zobrazenom na Obr. 15, je možné aktivity systému rozčleniť do troch častí, a to: zhlukovanie novo publikovaných novinových článkov, vytváranie používateľských profilov a personalizované odporúčanie novinových článkov. Zhlukovanie novinových článkov prebieha online, zatiaľ čo dynamická konštrukcia používateľského modelu a tvorba odporúčaní formou hierarchického zhlukovania je vykonávaná online v reálnom čase.

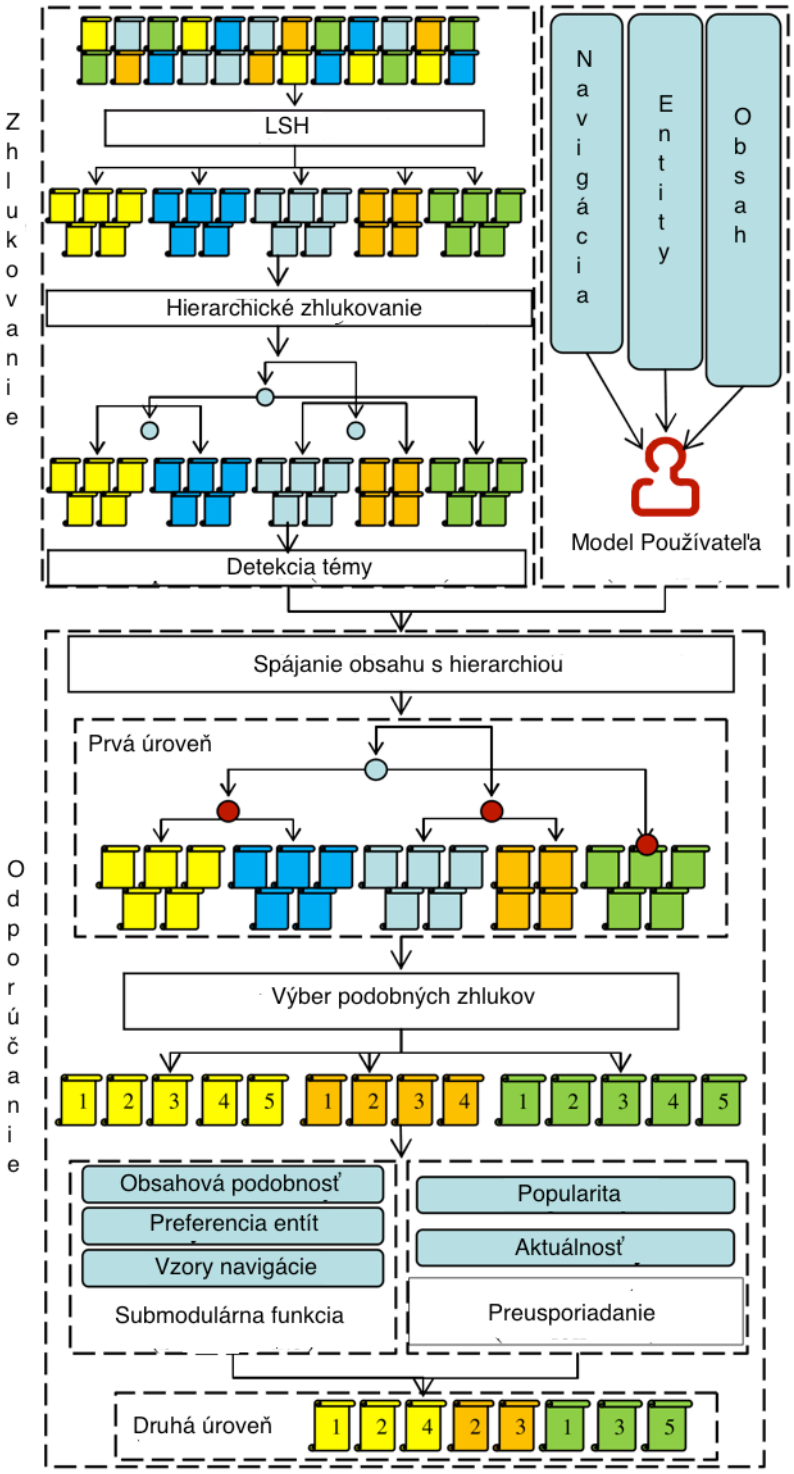
Na zhlukovanie novinových článkov do malých skupín využíva systém SCENE metódu hashovania závislého od umiestnenia (angl. locality sensitive hashing), kde tieto zhluky hierarchicky usporadúva do dočasných medzi-krokových zhlukov. Tieto temporálne zhluky sú zosumarizované technikami latentného sémantického indexovania (angl. Latent Semantic Indexing (LSI)) a latentného Dirichletovho priraďovania (angl. Latent Dirichlet Allocation (LDA)). Pri vytváraní používateľských modelov berie do úvahy tri dimenzie, a to: preferenciu entít článkov, distribúcie tém článkov a podobnosť vzorov navigácie a správania sa používateľov. Na základe vygenerovanej distribúcie tém systém následne sekvenčne vyberá zhluky článkov podobné používateľským profilom, ktoré reprezentujú výsledok prvej úrovne odporúčania. V každom zhluku sa následne pomocou submodulárnej funkcie vyberajú vlastnosti z dimenzií používateľských profilov a vytvára sa druhá úroveň reprezentácie odporúčaní, kde sú jednotlivé odporúčania zoraďované na základe ich vhodnosti pre používateľa.

[43] Peer to peer (distribuovaný model systému)
[44] http://digg.com/

SCENE predstavuje zástupcu z domény hybridných odporúčacích systémov. Rozdielom ktorý prináša Li et al. je ich postoj ku generovaniu personalizovaných odporúčaní, kde tento problém vnímajú ako problém maximálneho pokrytia s obmedzenými zdrojmi (Khuller et al. 1999). V tomto pohľade na generovanie odporúčaní ovplyvní výber aktuálneho článku generovanie odporúčaní pre nasledovné články. Ich prístup je podobný predošlej práci (Li et al. 2010), kde aplikovali prístup "zlodeja so zameraním na kontext" (angl. Contextual bandit). V tomto pravdepodobnostnom modely sa jedná o výber nasledujúcej akcie na základe predchádzajúcich skúsenosti a získaného profitu z výberu jednotlivých možností s ich kontextuálnymi vektormi. Cieľom je naučiť sa koreláciu medzi profitom a kontextom za účelom jej predikcie na základe vektora vlastností objektu. V systéme SCENE sú ale odporúčania generované s ohľadom na aktuálnu konkrétnu používateľskú časť práce so systémom (angl. session), zatiaľ čo v predošlej práci sa zameriavali na generovanie odporúčaní dlhodobého charakteru.

Pri zameraní sa na aspekt škálovateľnosti je vhodné uviesť využitie minhashovania (Indyk 1999) na zmenšenie veľkosti matice postupností slov za účelom jej reprezentácii priamo v pamäti systému, kde je pravdepodobnosť rovnakej reprezentácie dvoch článkov vygenerovanej hashovacej funkcie priamo úmerná Jaccardovej podobnosti medzi dvoma článkami. Ďalšími prvkami sú aplikovanie hashovania závislého od umiestnenia a hierarchického zhlukovania – využitie priemeru spojení medzi zhlukmi, kde systém získa prvotnú reprezentáciu odporúčaní článkov.

Prínosom ich práce je aj identifikovanie intistrických preferencií používateľa za účelom generovania odporúčaní s vyváženou úrovňou aktuálnosti, diverzity a novoty jednotlivých odporúčaní. Na overenie systému použili dáta (112,380 novinových článkov a 4630 používateľských profilov) získané z reálnych webových serverov so širokým obsahom tém. Realizované overenia ukazujú zlepšenie pri zhlukovaní voči štandardnému *K-means* algoritmu. Priame vysvetlenie autori nachádzajú v reprezentácii článkov – postupností slov (angl. shingles), ktoré predstavujú k-postupnosti nasledujúcich slov. Pri evaluácii modelovania používateľských preferencií autori zdôrazňujú výhodu hybridného prístupu a kombinácie viacerých aspektov modelu voči využitiu singulárnych faktorov používateľov. Autori realizovali aj používateľskú štúdiu v ktorej majorita používateľov hodnotila systém ako výnimočné dobrý v oblastiach rýchlosti, diverzity, usporadúvania a tvorbe preferencií pri generovaní odporúčaní.



Obr. 15 Dvojúrovňový rámec odporúčacieho systému SCENE (Li et al. 2011)

5 Ciele práce

V kapitole analýzy sme sa pozreli na rozmanitosť existujúcich prístupov k odporúčaniu. Každý z nich má svoje špecifiká, kladné aj negatívne stránky. Hybridné systémy vedia efektívne skombinovať viacero prístupov a vytvoriť tak veľmi presné a efektívne odporúčacie systémy, ktoré eliminujú napr. problém studeného štartu. Za popularitu výskumu oblasti odporúčacích systémov môžeme do určitej miery vďačiť priamo komerčnej sfére, ktorá investuje veľké objemy financií do vylepšenia odporúčaní. Súvisí s tým aj množstvo proprietárnych technológií, ktoré neskôr verejne publikujú aj s voľne dostupným zdrojovým kódom.

Problém, na ktorý sa v našej práci zameriavame je škálovateľnosť odporúčacích systémov. Na to, aby mohli odporúčacie systémy škálovať je nutné vytvoriť pokročilejšie architektúry, ktoré sú náročnejšie na implementáciu alebo ich správu. Ukázali sme, že riešením môže byť napr. distribúcia výpočtu medzi viacero serverov (Han et al. 2004), distribúcia výpočtu na stranu klienta (Boutet et al. 2014), paralelizácia (Shukla et al. 2012) alebo využitie GPU (Karydi and Margaritis 2014).

V našej práci sa zamierame na vytvorenie hybridného odporúčacieho systému, ktorý bude poskytovať aktuálne odporúčania, nielen vzhľadom na kontext používateľa, ale aj na dynamiku aktuálnych udalostí vo svete. Medzi kľúčové vlastnosti navrhovanej metódy patrí rýchla odozva odporúčacieho systému, flexibilta (možnosť konfigurácie medzi rýchlosťou a presnosťou) výberom odporúčacích algoritmov a v neposlednom rade škálovateľnosť. Využitím distribúcie a paralelizácie výpočtov pri modeloch strojového učenia, kolaboratívnom filtrovaní a určovaní stupňa záujmu odporúčaných položiek na základe ich obsahu, vytvoríme efektívny a škálovateľný odporúčací systém, spĺňajúci definované ciele a aktuálne požiadavky odporúčaní na webe.

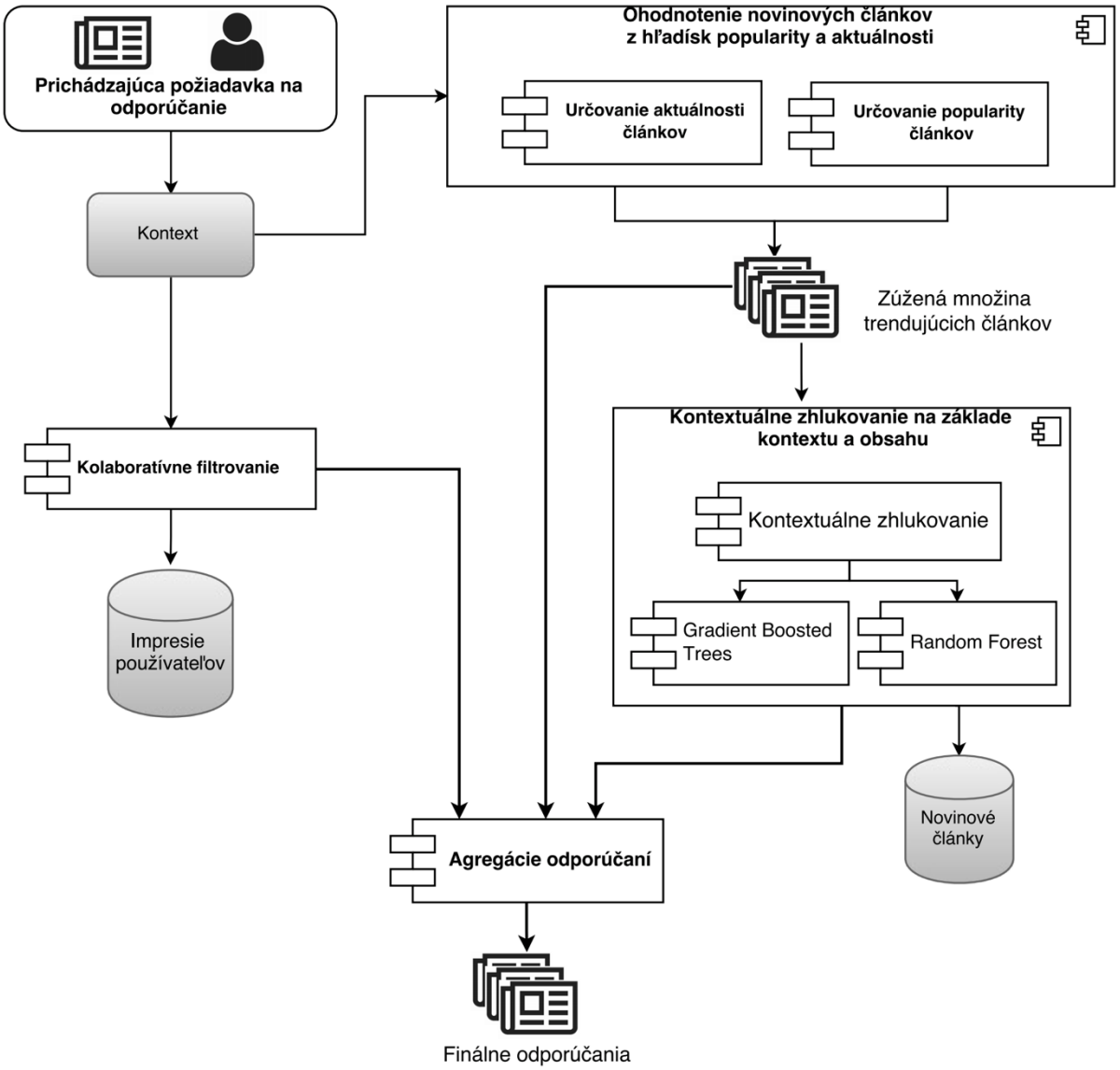
Ako ciele práce identifikujeme:

- Navrhnutie a vybudovanie škálovateľnej architektúry za účelom spracovania prúdu dát v reálnom čase.
- Efektívna reprezentácia (vektorizácia) používateľovych preferencií a samotných odporúčaných prvkov.
- Hybridné odporúčanie so zameraním na rýchlosť odozvy a aktuálnosť odporúčaní vzhľadom k preferenciám používateľa.
- Realizáciu odporúčaní v konkrétnej doméne v praxi (novinové články) – spracovanie požiadaviek a generovanie odporúčaní v reálnom čase.

6 Návrh metódy

Vychádzajúc zo stanovených cieľov sme navrhli metódu škálovateľného personalizovaného odporúčania so zohľadnením kontextu používateľa v doméne novinových článkov na webe. Metódu navrhujeme ako hybridný odporúčací systém pre prácu s prúdom dát, kde časť výpočtu odporúčaní bude prebiehať on-line, v reálnom alebo takmer reálnom čase. Druhá časť hybridného odporúčacieho systému pozostáva z off-line učenia modelov strojového učenia a generovania odporúčaní.

Nami navrhované hybridné odporúčanie je škálovateľné, flexibilné, s cieľom poskytnutia rýchlej odozvy používateľovi, prípadne externému systému. Hybridný systém (Obr. 16) je v závislosti od zvolenej stratégie odporúčania alebo informáciách o používateľovi možné klasifikovať ako zmiešaný, váhovaný, alebo aj ako kaskádový, kedy výstup jednej metódy odporúčania alebo filtrovania obsahu je vstupom pre iný modul odporúčania.



Obr. 16 Návrh a tok dát metódy hybridného odporúčania

Proces odporúčania pozostáva z nasledovných krokov (postupnosť a prítomnosť krokov závisí od použitej stratégie odporúčania):

- *Prijatie požiadavky na odporúčanie* – pri prijatí požiadavky na odporúčenie v prvom kroku extrahujeme kontextuálne informácie, s ktorými pracujeme neskôr pri zhlukovaní.

- *Určenie zhluku* – v závislosti od kontextu systém predikuje zhluk, do ktorého bude požiadavka na odporúčanie zaradená.

- *Získanie populárnych a aktuálnych článkov* – systém získa z periodicky aktualizovných množín najpopulárnejšie a najaktuálnejšie články, v závislosti od vybraného algoritmu resp. stratégie odporúčania.

- *Získanie odporúčaní pomocou kolaboratívneho filtrovania* – na základe predošlého správania (prečítané články) určíme pre používateľa relevantné články, ktoré systém predpočítava na pozadí.

- *Predikcia relevantných článkov na základe obsahu* – systém použije buď predpočítanú množinu relevantných dokumentov pre identifikovaný zhluk alebo ohodnotí vstupnú množinu aktuálnych a populárnych článkov.

- *Agregácia odporúčaní* – poďla zvolenej stratégie agregácie odporúčaní, sformujeme množinu finálnych odporúčaní, ktoré posielame externému systému ako odpoveď na požiadavku na odporúčanie.

6.1 Reprezentácia kontextu a obsahu

V navrhovanej metóde predpokladáme prístup k plnému obsahu samotných odporúčaných prvkov – novinových článkov. Druhou množinou informácií, s ktorou naša metóda pracuje, je kontext prislúchajúci k jednotlivým zobrazeniam (impresiám) novinových článkov. Pod kontextom myslíme situáciu a aktivitu používateľa, v ktorej sa nachádza pri požiadavke na odporučenie ďalších novinových článkov.

6.1.1 Obsah objektov

Novinovým článkom môže byť napr. rozsiahla dlhodobá reportáž, komentár, krátka tlačová správa, recenzia alebo test, aktualita alebo dynamický komentár športového zápasu, či spoločenskej udalosti. Spoločnými črtami sú prítomnosť nadpisu, samotného textu a autora (prípadne kolektívu autorov). Pri odporúčaniach a reklame je často k dispozícií aj sumár alebo perex (úvodná časť článku), prípadne iný kratší text, ktorý sumarizuje obsah a jeho cieľom je čitateľa nalákať na prečítanie článku. Pri väčšine odporúčaní je k dispozícii aj obrázok alebo fotografia, ktorá spĺňa podobný účel. Tieto články sú na webových portáloch často kategorizované na základe obsahu alebo polohy, či príslušnosti k regiónom do rôznych kategórií a domén.

Akonáhle máme k dispozícii textový obsah článkov, otvárajú sa možnosti pre obohatenie článkov ďalšími informáciami. Jedná sa o identifikovanie pomenovaných entít (napr. osoby, krajiny, mestá), konceptov alebo tém, extrakcia kľúčových slov prípadne sentimentálne ohodnotenie vymenovaných subjektov. Tieto informácie vieme pre vybrané a rozšírené

jazyky získať rôznymi nástrojmi pomerne spoľahlivo a jednoducho, či už využitím rôznych nástrojov alebo služieb tretích strán. Naopak, pre jazyky ako je napr. Slovenčina, môže byť získanie pomocných charakteristík s aktuálnymi nástrojmi náročné a nepresné. Príkladom informácií, ktoré budú vstupom pre obsahovo založený komponent našej metódy odporúčania je ukážka článku v Tab. 2.

Tab. 2 Ukážka obsahu novinového článku obohateného o extrakciu informácií

Charakteristika	*Charakteristika článku (príklad)*
Nadpis	Vedci objavili záhadného predka Európanov.
Text	LONDÝN, BRATISLAVA. Ak sa rozhodnete navštíviť niektoré gruzínske kláštory zapísané na zozname svetového kultúrneho dedičstva,asi vás nadchne ich izolovaná krása...
Sumár	Vedci našli v nálezoch kľúč k dosiaľ neznámemu pôvodu časti európskeho genetického materiálu.
Kategórie	človek, biológia, história
Entity	Afrika {relevancia: 0.92, sentiment: zmiešaný, typ: krajina } …
Kľúčové slová	človek {relevancia: 0.83, sentiment: neutrálny}, kopať {relevancia: 0.1, sentiment: pozitívny} …
Sentiment dokumentu	pozitívny: 0.64

6.1.2 Kontext interakcie používateľa s objektom

Kontextom označujeme charakteristiky, ktoré buď priamo popisujú používateľa, alebo nepriamo charakterizujú podmienky a situáciu, v ktorých používateľ videl zobrazené novinové články. Pri používateľovi poznáme alebo odhadujeme jeho pohlavie, vek, demografické údaje, vzdelanie alebo ročný príjem. K dispozícii máme informácie o webovom prehliadači, operačnom systéme, používanom jazyku, nastavení blokovania reklám. Ďalšou skupinou sú informácie o počasí, lokácii, čase, počte zobrazení článkov z konkrétneho portálu. Pri kontexte rovnako vieme určiť aj to, ktorý článok používateľ práve číta, takže sú dostupné aj obsahové informácie jednotlivých článkov.

Experimentovaním hľadáme najvhodnejšie kombinácie charakteristík, ktoré budú vstupom do algoritmov odporúčania tak, aby sme odstránili informácie, ktoré spôsobujú iba šum a naopak, zachovali tie, ktoré pre navrhovanú metódu majú pozitívny vplyv a optimalizujú odporúčania – zvyšujú presnosť odporúčaní, s ohľadom na výpočtovú náročnosť.

Tab. 3 Ukážka kontextu dostupného pri požiadavke na odporúčanie

Charakteristika	Charakteristika článku (príklad)
Operačný systém	Microsoft Windows 10
Internetový prehliadač	Google Chrome 46
Typ zariadenia	PC
Pohlavie	Muž
Vek	20-25
Ročný príjem	25000
Poloha	Berlín, Nemecko, 52° 31' 14.3220" N, 13° 24' 35.2044" E
Poskytovateľ internetu	SWAN Slovakia s.r.o
ID článku	34221

6.1.3 Reprezentácia objektov a ich vektorizácia

Na to, aby sme mohli efektívne pracovať so zaznamenaným kontextom, zobrazeniami článkov alebo samotným obsahom článkov, je nutné tieto objekty efektívne reprezentovať pre neskoršie spracovanie v pamäti počítača. Objekty reprezentujeme formou vektorového modelu (angl. VSM – vector space model). Týmto spôsobom získame univerzálnu reprezentáciu, na ktorú môžeme aplikovať rôzne metriky podobnosti alebo využiť ich pri algoritmoch strojového učenia.

Vektorizácia pomocou kódovania „jeden z"

Ako sme si ukázali, väčšina atribútov pre charakteristiky, či už kontextu alebo obsahu, sú nominálne alebo kategorické atibúty. Na to, aby sme vedeli efektívne precovať aj s takýmito atribútmi, musíme ich obsah konvertovať do inej formy, najmä ak chceme pracovať s prúdmi dát dynamicky. V situáciách, kedy pracujeme s prúdom dát, prípadne to bude vyžadovať použitý algoritmus používame kódovanie "jeden z" (angl. one hot encoding). Princíp kódovania spočíva v konvertovaní jednej premennej do n binárnych premenných, kde n je počet unikátnych hodnôt danej premennej. Nové premenné potom nadobúdajú hodnoty 0 alebo 1 podľa toho, či sa zhodujú s hodnotou pôvodnej premennej. Uvádzame príklad reprezentácie typu zariadenia, ktoré je kategorickou premennou a nadobúda jednu z hodnôt *mobil*, *tablet* alebo *počítač*.

Tab. 4 Príklad kódovania "jeden z n" pre premennú typ zariadenia

Hodnota premennej	*Vektor*
mobil	*[1,0,0]*
tablet	*[0,1,0]*
počítač	*[0,0,1]*

Pri práci s numerickými premennými, ktorými môžu byť napr. doba od otvorenia stránky po kliknutia na odporúčanie alebo počty doteraz odporučených objektov, využívame normalizáciu premenných tak, aby extrémne hodnoty v jednotlivých premenných nemali príliš silný efekt v našej metóde (napr. pri určovaní podobnosti). Normalizácia hodnoty premennej do intervalu $< 0,1 >$, s maximálnou hodnotou premennej max a minimálnou hodnotou premennej min bude prebiehať pre premennú var s hodnotou x nasledovne.

$$norm(var_x) = \frac{x - min}{max - \min}$$

Pre reprezentáciu textového obsahu (text, sumár a nadpis) použijeme ustálenú metriku *tf-idf*, po predspracovaní textu pre príslušný jazyk (odstránenie stop slov, lematizácia).

6.2 Preferencie a model používateľa

Z pohľadu kategorizácie modelu používateľa pracujeme s dynamickým prekryvným modelom používateľa. Najpodstatnejšou informáciou, ktorá bude odrážať používateľove preferencie sú ním v minulosti prečítané články. Druhou množinou prefencií, ktoré využijeme pri odporúčaniach z hľadiska aktuality alebo popularity, bude uchovávanie preferencií kategórií, portálov alebo domén článkov, ktoré autor čítal. Jedná sa o denormalizovanú reprezentáciu informácií, ktorú získame z histórie používateľových zobrazení novinových článkov. Dôvodom je efektívnejšie získanie používateľových preferencií pri výpočtoch v reálnom čase.

Na to, aby bol model používateľa sofistikovanejší a bral do úvahy aj vplyv času a možnosť zmeny preferencií, použijeme váhovanie jednotlivých návštev používateľa, podľa dátumov a časov návštev. V prípade viacnásobného zobrazenia novinových článkov, berieme do úvahy čas poslednej návštevy novinového článku.

6.3 Hybridné odporúčanie

Po vysoko úrovňovom predstavení navrhovanej metódy, ukážkovej domény a dát, modelovaní preferencií používateľa sa dostávame k najpodstatnejšej časti metódy, a to samotnému odporúčaniu. Hybridné odporúčanie je realizované vo viacerých rovinách, a to vzhľadom k aktuálnemu času a spôsobu výpočtu. Odporúčania zohľadňujúce popularitu a aktualitu objektov sú realizované v aktuálnom, alebo takmer aktuálnom čase za behu odporúčacieho systému. Pri obsahovo založenom odporúčaní rozdeľujeme výpočty podobností ako off-line inkrementálny proces, pričom výpočet samotných odporúčaní prebieha v aktuálnom čase. Poslednými dvoma zložkami sú off-line kolaboratívne filtrovanie a odporúčanie založené na zhlukovaní kontextu pre následnú aplikáciu metód strojového učenia algoritmami náhodného lesa (angl. random forest) a učenia sa usporadúvania (angl. learning to rank).

6.3.1 Popularita a aktuálnosť obsahu

Používatelia navštevujú novinové portály viackrát denne. Zaujíma ich aktuálne dianie vo svete alebo v najbližšom okolí. Na väčšine portálov môžeme nájsť zoznamy resp. rebríčky článkov, ktoré boli nedávno publikované alebo aktualizované. Ďalšími rebríčkami, ktoré je možné na portáloch nájsť sú najpopulárnejšie články. Obvykle je dostupná aj informácia o časovej jednotke, a to napr. najpopulárnejšie články za aktuálny týždeň, deň alebo hodinu. Rovnako sú zvýrazňované aj články, ktoré môžu byť aj staršieho dáta, ale z dôvodu určitých udalostí sa ich čitateľnosť rýchlo zvýši a vzrastie ich trend (angl. trending articles). V nami navrhovanej metóde monitorujeme a udržiavame spomínané množiny článkov, ktoré budú vplývať na výsledné odporúčania, prípadne za nepriaznivých okolností a vyťaženia systému ich môžu priamo predstavovať.

Uvedené množiny článkov agregujeme v rôznych množinách, v našom systéme teda máme okrem globálnej množiny najpopulárnejších alebo najaktuálnejších článkov aj množinu článkov prislúchajúcich konkrétnemu portálu, kategórii doméne, ale rovnako aj danému zhluku požiadaviek s kontextom resp. používateľov.

Aktuálnosť

Ohodnotenie aktuálnosti článku navrhujeme pomocou funkcie [45] , ktorá umožní sofistikovanejšie váhovanie ako pri obyčajnom lineárnom penalizovaní z pohľadu doby publikovania článku. Je dôležité uviesť, že článok, ktorý bol prvý-krát publikovaný pred 10 minútami je z nášho pohľadu pre používateľa rovnako aktuálny, ako článok, ktorý bol aktualizovaný napr. pred dvoma hodinami. Rovnako vnímame aj opačnú stranu tejto problematiky, kedy článok publikovaný pred 6 mesiacmi je pre používateľa hľadajúceho najnovšie informácie rovnako neaktuálny, ako taký, ktorý bol publikovaný pred rokom. Preto si stanovujeme bod v časovom vnímaní, ktorý bude zohľadňovať vyššie uvedené

[45] https://www.elastic.co/guide/en/elasticsearch/guide/current/decay-functions.html

predpoklady. Bod $penaltyStart$ predstavuje bod v čase nasledujúci po aktuálnom čase $currentTime$, po ktorý nebudú články penalizované.

$$recency(article) = w_1 * timePenalty(\text{published}_{at}) - w_2 * timePenalty(updated_{at})$$

$$timePenalty(time) = exp(-\frac{max\ (0, |time - published_{at}| - penaltyStart)^2}{2(-\frac{scale^2}{2} * log(decay))}$$

Články s časmi publikácie alebo aktualizácie medzi týmito dvoma hraničnými bodmi budú penalizované na základe tretieho vstupného parametra $decay$, ktorý určuje sklon funkcie a silu penalizácie. O túto hodnotu bude penalizovaný článok, ktorého časový údaj je zhodný s parametrom $scale$. Pre výslednú hodnotu aktuálnosti článku používame kombináciu času prvej publikácie článku a času poslednej aktualizácie obsahu článku.

Tab. 5 Ukážka ohodnotenia aktuálnosti odporúčaných článkov, akt. čas 1447799955 (epoch)

	Čas publikovania	*Čas aktualizácie*	*Skóre*
článok 1	1447799955	1447799955	1
článok 2	1447799900	1447799952	1
článok 3	1443600000	1443600000	0.5

Popularita

Pri súčasnej škále a počte návštevníkov jednotlivých portálov aj jednoduché operácie alebo štatistiky o návštevnosti alebo počte obslúžených požiadaviek vyžadujú netriviálne riešenia. Naskytá sa možnosť využitia existujúcich pravdepodobnostných štruktúr akými sú bloomovské filtre, hyperloglog alebo Count-min-sketch (Cormode 2003). V našom návrhu metódy nepotrebujeme mať informácie o prítomnosti prvkov v množine, ale stačia nám počítadlá návštev jednotlivých objektov resp. článkov. Najpopulárnejšie články agregujeme podľa ich charakteristík do rôznych množín a do rôznych časových období. Navrhujeme dve riešenia, ktoré je možné použiť pre nami definované meranie popularity. Prvý spôsob je založený na udržiavaní si usporiadaných množín s exspiráciou obsahujúcich dvojice $article{:}\,click\_count$, ktoré inkrementujeme pri každej návšteve článku. Inkrementujeme globálne množiny pre rôzne časové obdobia, ale aj jednotlivé množiny monitorujúce konkrétne charakteristiky objektov.

Tab. 6 Spôsob uchovávania najpopulárnejších článkov v kategórii šport

Množina: najpopulárnejšie:šport:01022015		
1. najpopulárnejší článok	článok:54531	2322
2. najpopulárnejší článok	článok:57004	931
3. najpopulárnejší článok	článok:54587	438

Druhou možnosťou je periodické dopytovanie a vytváranie agregácií nad všetkými zaznamenanými udalosťami vo vybranom časovom období. Vďaka dnes rozšíreným ľahko škálovateľným NoSQL databázam nie je problém realizovať navrhovaný spôsob aj s veľkým a rýchlo rastúcim objemom dát.

Výstupom modulov určujúcich popularitu alebo aktuálnosť budú v závislosti od aktuálneho zaťaženia systému priamo aj výsledné odporúčania, čo je aplikovateľné aj v prípade, ak nemáme dosť informácií o používateľovi alebo kontexte prichádzajúcej požiadavky. Ak bude

mať odporúčací systém dostatok času na vygenerovanie aj iných odporúčaní, akými sú obsahové alebo kolaboratívne, tak vieme ovplyvniť váhu týchto odporúčaní výstupom modulu popularity a aktuálnosti, prípadne využiť spomínané odporúčania ako vstup pre ďalšie komponenty hybridného odporučacieho systému opisované v nasledujúcich podkapitolách.

6.3.2 Odporúčanie so zohľadnením kontextu

Pod kontextom rozumieme stav používateľa (kapitola 6.1.2), v ktorom sa nachádzal v momente, kedy odporúčací systém prijal požiadavku na vygenerovanie odporúčaní. Požiadavky reprezentujeme formou vektorov, ktoré prejdú vektorizáciou spomínanou v predchádzajúcej kapitole. Takto získané vektory zhlukujeme využitím algoritmu modifikovanej paralelnej verzie algoritmu k-means, a to kmeans||, ktorý je založený na algoritme kmeans++ využívajúcom efektívnu inicializáciu centroidov (Bahmani et al. 2012). Centroidy bude naša navrhovaná metóda aktualizovať v reálnom čase, čím dosiahneme aktuálnosť modelu zhlukovania. Výsledkom bude agregovanie prichádzajúcich požiadaviek na základe blízkosti resp. podobnosti vlastností kontextu, ktoré je možné obohatiť aj o samotný obsah novinového článku alebo históriu používateľom prečítaných článkov. Navrhovaná metóda vie zúžitkovať informácie získané zhlukovaním a na základe príslušnosti k predikovanému zhluku vybrať jeden z naučených modelov, podľa voľby algoritmu odporúčania. Príslušnosť požiadavky k zhluku je možné využiť aj ako ďalší atribút, podľa ktorého modelujeme podmnožinu populárnych článkov (impresie článkov prislúchajúce danému zhluku požiadaviek).

Pre každý takto vytvorený zhluk využívame metódy strojového učenia, za účelom predikcie pravdepodobnosti, podľa ktorej používateľa daný článok zaujme a eventuálne naň aj klikne. Navrhujeme využitie dvoch metód z rodiny CART (angl. classification and regression trees), a to rozhodovacích stromov vo forme náhodného lesa (angl. Random forest) a stupňovaných stromov (angl. Gradient boosted trees). Spomenuté algoritmy budú v našom systéme figurovať formou modelov pre zhluky, pričom modely budú vytvárané na pozadí, off-line a periodicky aktualizované.

Pri rozhodovacích stromoch využijeme paralelizovateľnú a škálovateľnú metódu náhodného lesa. Tá umožňuje kombináciu nezávislých rozhodovacích stromov, často s menšou hĺbkou, ktoré je možné učiť paralelne. Náhodnosť zohráva úlohu pri vytváraní jednotlivých stromov lesa, ktoré sú pri jednotlivých iteráciách algoritmu menené, či už zmenou podmnožiny vlastností, na základe ktorých stromy určujú výsledok alebo využitím rozličných podmnožín dát. Výstupom náhodného lesa nad množinou dát môže byť buď regresia alebo klasifikácia. V našej metóde pracujeme s regresiou, predikujeme relevanciu článku pre používateľa, resp. pre kontext používateľa, ohraničenú na intervale $< 0{,}1 >$, pričom 1 bude označovať články, u ktorých predpokladáme najväčší používateľov záujem o ich prečítanie. Druhou možnosťou je aj využitie klasifikácie do dvoch tried, a to články, ktoré používateľa zaujmú a články, u ktorých nepredpokladáme záujem medzi používateľmi v danom zhluku. Z dôvodov agregovania odporúčaní z viacerých komponentov je vhodnejšie použiť jemnejšie členenie formou regresie.

Ako sekundárnu voľbu algoritmu rozhodovacích stromov sme vybrali GBTs (angl. Gradient boosted trees). Na rozdiel od algoritmu náhodného lesa, sa pri GBTs trénuje len jeden strom, a to iteratívne, čiže na rozdiel od náhodného lesa nie je implementácia až zahanbujúco

paralelizovateľným problémom. Vstupom do rozhodovacích stromov sú počty impresií a kliknutí článkov v jednotlivých zhlukoch. Experimentálne overíme vhodné parametre (počet stromov, hĺbka, počet iterácií) a aj hranicu, od ktorej považujeme objekt ako pozitívny príklad pre zhluk a jeho model rozhodovacích stromov.

6.3.3 Kolaboratívne filtrovanie

V doplnkovom kolaboratívnom komponente odporúčacieho systému použijeme overenú metódu faktorizácií matíc – ALS (angl. alternating least squares) (Koren et al. 2009). Pomocou nej, bude kolaboratívny komponent schopný odhaliť skryté latentné faktory z implicitných preferencií získaných z akcií používateľov a vygenerovať pre používateľov odporúčania. Tvorba modelu faktorizácie matíc bude prebiehať off-line, za pomoci využitia distribuovaných technológií, čím sa stane komponent škálovateľným.

Ako sme uviedli v kapitole 2, pri faktorizácií matíc ALS iteratívným spôsobom využíva určovanie latentných faktorov jednej z dekomponovaných matíc, pričom druhá zostáva v danej iterácii nezmenená. V našej metóde pracujeme s dvoma maticami faktorov, a to maticou faktorov používateľov U a maticou faktorov novinových článkov I, ktoré vytvárameť z matice R. Každý riadok z týchto matíc charakterizuje väzbu medzi používateľmi alebo článkami a latentnými faktormi. Používateľské akcie vo forme zobrazení novinových článkov budú reprezentované hodnotou 1, články s ktorými používateľ neinteragoval hodnotou 0 (v prípade binárnych preferencií), prípadne môžu zostať prázdne. Výhodou tohto modelu je aj možnosť použitia rozličných váh, pre implicitné preferencie používateľa, ak by sa ukázalo, že je vhodné rozlišovať medzi kliknutím na novinový článok zo zoznamu odporúčaní alebo neovplyvnené prirodzené zobrazenie článkov navigáciou používateľa.

V navrhovanej metóde predpokladáme periodické aktualizovanie modelu na pozadí, v čo možno najfrekventovanejších periódach, aby sme zabezpečili aktuálnosť modelu.

6.3.4 Agregácia odporúčaní

V predchádzajúcich kapitolách sme opísali rozličné metódy odporúčania, ktoré kombinujeme do finálnych odporúčaní zobrazovaných používateľom odporúčacieho systému. Ako sme naznačili, v našej metóde je priestor pre aplikáciu a experimentovanie s viacerými spôsobmi kombinácií odporúčaní v hybridnom odporúčacom systéme. Jedným z cieľov navrhovanej metódy je flexibilita a rýchla odozva, čo nám hybridný systém umožňuje.

V ideálnych podmienkach, ak máme k dispozícii históriu návštev používateľa, a dostatok výpočtového výkonu, tak využijeme všetky opísané odporúčania. Aspekty popularity a aktuálnosti budú figurovať ako meta-úrovňové odporúčanie, ktoré vyprodukuje vstupnú množinu pre ďalšie odporúčania (napr. kontextové). Zvlášť bude v hybridnom systéme vystupovať komponent kolaboratívneho filtrovania, u ktorého očakávame najmenej aktuálny model, z dôvodu náročnosti jeho učenia a aktualizovania. Často sa tak môže stať, že novým návštevníkom alebo anonymným návštevníkom nevieme cez komponent kolaboratívneho odporúčania vygenerovať žiadne odporúčania. Tento problém je možné vyriešiť za určitých okolností vybraním náhodného reprezentanta zo zhluku, do ktorého bol anonymný používateľ priradený na základe kontextu, pre ktorého vieme vygenerovať odporúčania. Odporúčania z kolaboratívneho aj obsahového komponentu váhujeme pomocou ich aspektov popularity a aktuálnosti.

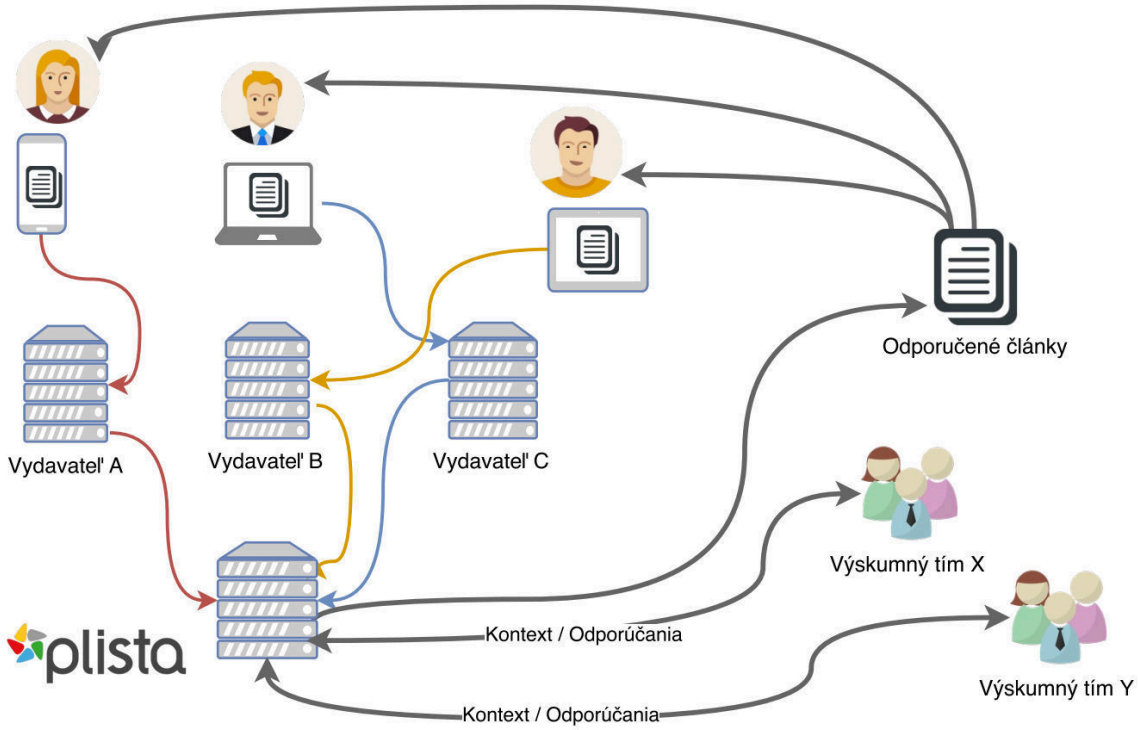
Ak sa vrátime naspäť k prípadu, kde sa jedná o pre náš systém nového, resp. anonymného používateľa, naše odporúčania nebudú ovplyvnené problémom studeného štartu, pretože stále máme dostatok informácií z kontextu a obsahu aktuálneho článku, ktorý používateľ práve číta. Ak sa odporúčací systém ocitne v situácii, kedy je príliš zahltený a nestíha odpovedať na prichádzajúce požiadavky, bude možné vygenerovať odporúčania použitím výpočtovo nenáročných odporúčaní na základe popularity a aktuálnosti novinových článkov.

Finálna kombinácia odporúčaní tak bude zložená z viacerých nezávislých odporúčacích metód, formujúcich skupinu odporúčacích metód. Zmiešaním váhovaných odporúčaní jednotlivými komponentami (kontextový, obsahový, kolaboratívny) a ich zoradením na základe zosumarizovaných indikátorov relevancie pre používateľov dopyt, ktorý je reprezentovaný kontextom a modelom používateľa, získame výsledné odporúčania.

7 Realizácia metódy personalizovaného odporúčania

Hybridnú metódu škálovateľného odporúčania sme realizovali ako samostnaný systém, ktorého vstupom je prúd dát. Týmto spôsobom je možné využívať odporúčací systém ako externý komponent pre on-line systémy, no odporúčací systém vie pracovať aj s off-line množinami dát. V druhom prípade, je do istej miery možné aj zachovať časovú postupnosť a emulovať tak podmienky a situáciu ekvivalentné s podmienkami, za ktorých boli dáta získané. V prípade takých overení ale dochádza k situácii, kedy odporúčací systém nedokáže spätne ovplyvniť minulosť, čo spôsobí, že používateľom nevie odporúčať spätne novo generované odporúčania. Jediným východiskom je sledovať pasívne správanie a aktivitu používateľov. Toto negatívum je bohužiaľ neodstrániteľné a v prípade opakovateľných experimentov nad statickou množinou dát očakávané. Metódu sme realizovali a overli v doméne novinových článkov.

Metódu aj jej overenie sme uskutočnili na platforme Plista ORP [46] (angl. Open Recommendation Platform) a off-line dátových množinách, ktoré uvádzame v nasledujúcej kapitole. Plista je nemecká spoločnosť, ktorá prevádzkuje reklamné a odporúčacie systémy pre väčšinu nemeckých on-line portálov. Okrem ich komečných aktivít poskytujú aj platformu pre výskumnícke tímy alebo jednotlivcov, kde môžu zaregistrovať svoje algoritmy odporúčania a určitá malá časť skutočných on-line požiadaviek bude smerovaná práve na ich odporúčací systém (Obr. 17). Touto cestou môžu individuálni výskumníci, ale aj väčšie tímy vyskúšať presnosť svojich odporúčaní v on-line prostredí.



Obr. 17 Platforma ORP - náčrt rozdeľovania požiadaviek na odporúčania

[46] http://orp.plista.com/

Zaujímavosťou Plisty je ich postoj k odporúčaniam, kde odporúčania na novinové články vnímajú skôr ako priamu reklamu – a tak je možné v reklamných banneroch na stránkach on-line denníkov nájsť aj odporúčania na novinové články konkurenčných denníkov. Aj vďaka tomuto faktu je samotná miera prekliku na novinové články nízka a pohybuje sa okolo hranice 1% (podľa dát z ORP).

7.1 Zdroje dát

Celkovo sme pri realizácií metódy pracovali s tromi zdrojmi dát. Vo všetkých troch prípadoch sa jednalo o implicitné preferencie návštevníkov novinových portálov vo forme impresií a kliknutí. Na úvodné potvrdenie vplyvu popularity a čítanosti novinových článkov v procese odporúčania sme využili off-line data set z Denníka SME, v ktorom figurovalo 11 331 129 impresií a 834 416 unikátnych používateľov, resp. cookies v pehliadači.

Druhým a tretím zdrojom dát sú dáta získané z platformy ORP. Pri opakovateľných experimentoch sme pracovali s dátami zozbieranými z 30 dní počas Júna 2013, resp. s podmnožinami dát. Dáta pre jednotlivé dni pozostávali zo štyroch štruktúrovaných súborov vo formáte JSON. Obsahom boli informácie o vytvorení nových novinových článkov (Ukážka dát 1), ich aktualizácií, impresií a kliknutí používateľov (Ukážka dát 2). V prípade posledných dvoch typov dát boli k dispozícii aj informácie o kontexte. Off-line dáta obsahovali denne v priemere približne 2GB dát, čo predstavuje zhruba 2.5 milióna impresií denne. Kontextuálne informácie pozostávali z 57 vlastností, ktoré boli obvykle vyplnené a jednalo sa napr. o informácie o operačnom systéme, type zariadenia, geolokácii používateľa.


```
{"domainid":"1677","created_at":"2013-06-01 00:00:29", "flag":8,
"title":"\"Berlin Tag &amp;amp; Nacht\" Der Die Das Fan-Tour",
"url":"http:\/\/www.tagesspiegel.de/344334.html","img":"","text":"",
"published_at":null,"version":1,"updated_at":"2013-06-01 00:00:29",
"id":"12796523322323616"}
```


Ukážka dát 1 Vytvorenie nového článku


```
{"type":"click","context":{"simple":{"4":457399,"5":317849,"6":10,"7
":18849,"9":26886,42":0,"47":654013,"49":47,"52":1,"56":1138207,"57"
:2309723554},"lists":{"8":[1841,18842],"10":[3,9,14],"11":[157987]},
"clusters":{"1":{"8":255},"2":[19,15,79,53,65,15,6],"3":[86,40,22,72
,20,12],"33":{"35112":"23":1"69418":0},"46":{"793434":255},"51":{"5"
:255}}},"recs":{"ints":{"3":[127691561]}},"timestamp":1370037599973}
```


Ukážka dát 2 Kliknutie na novinový článok aj s kontextuálnymi informáciami

Interaktívnejšou variantou ako pracovať s dátami ORP bolo priame zapojenie sa do on-line odporúčaní. V čase integrácie s ORP [47] sme využívali verziu 0.4.1-x. Plista neuvádza akým spôsobom distribuuje požiadavky na odporúčanie medzi zapojené tímy. Jednou z požiadaviek na odporúčací systém zapojený do platformy ORP je schopnosť odpovedať v priemere do 100ms na prichádzajúcu požiadavku, čo je dosť limitujúci faktor. Z tohto dôvodu je nutné niektoré časovo náročnejšie odporúčania predpočítavať. Kontextuálne

[47] http://orp.plista.com/documentation/download

informácie zákodovaných vo forme čísel z dôvodu efektívneho prenosu dát po sieti, je možné pomocou prekladača ORP spätne získať originálnu hodnotu, čiže vieme napr. určiť presný názov a verziu operačného prehliadača. Túto funkcionalitu sme pri spracovaní dát implementovali, avšak v procese odporúčania sme ju nevyužili.

7.2 Metóda odporúčania

Navrhnutú a realizovanú metódu klasifikujeme ako hybridný odporúčací systém, ktorý je v závislosti od zvolenej stratégie možné považovať obvykle za váhovaný, ale napr. v metóde zhlukovania na základe kontextu s využitím popularity sa jedná o kaskádový odporúčací systém. Určitá časť odporúčaní je predpočítavaná, iné stratégie odporúčania generujú aj odporúčania v reálnom čase. Od charakteru objemu dát a frekvencii požiadaviek na odporúčanie v externom systéme je vhodné vybrať práve takú metódu, ktorá bude spĺňať rýchlostné požiadavky. V prípade ORP sme pracovali s oboma spôsobmi generovania odporúčaní (v reálnom čase, predpočítané). Výber stratégie je možné zadefinovať zvlášť pre každú z prichádzajúcich požiadaviek, čo robí náš odporúčací systém flexibilným. Posledným krokom v procese odporúčania je spôsob agregácie odporúčaní. Pri váhovaných odporúčaniach generovanými jednotlivými komponentami odprúčacieho systému sme realizovali tri stratégie agregácie, a to *suma*, *súčin* a *výber maximálnej miery istoty*. Na základe overení v úvodných stranách kapitoly 8 sme určili ako najvhodnejší *súčinový* agregátor, pri ktorom manuálne posúvame hranicu neutrálnej hodnoty váhy resp. miery istoty odporúčania z 0 na 1, aby sme matematickými operáciami súčinu nestratili výnimočne dobré položky v jednom z odporúčacích komponentov, ktoré iné odporúčacie komponenty vyhodnotili relatívne slabo.

7.2.1 Popularita a trendovosť novinových článkov

Pri počítaní tzv. trendovosti novinových článkov sme využili periodické prepočítavanie popularity a aktuálnosti, ako sme ho opísali v kapitole 6. Pri popularite sme pracovali s viacerými časovými intervalmi, od 1 hodiny až po týždeň. Pri overení na dátach novinového portálu SME.sk sa ukázalo, že najlepšie výsledky poskytujú práve kratšie dynamickejšie intervaly (čo môže byť spôsobené aj prednastaveným zobrazovaním najčítanejších článkov za posledné 4 hodiny pri otvorení titulnej stránky na portále SME.sk). Okrem globálnej popularity sme modelovali aj popularitu filtrovanú na základe atribútov impresií. Vybrali sme tieto atribúty: *doména novinového článku*, *identifikovaný zhluk impresie* a *vydavateľ*. Výsledky vplyvu trendovosti, aj z pohľadu vymenovaných atribútov, uvádzame v kapitole Popularita a aktuálnosť novinových článkov. Trendovosť, resp. obe jej zložky je možné použiť aj samostatne ako jediný odporúčací komponent. V našej metóde však figurujú v procese odporúčania ako sekundárne odporúčacie metódy, ovplyvňujúce kolaboratívne filtrovanie alebo odporúčanie založené na základe kontextuálneho zhlukovania.

7.2.2 Kolaboratívne filtrovanie

Pri kolaboratívnom filtrovaní sme siahli po škálovateľnom algoritme maticových faktorizácií ALS. Pracujeme s implicitnými preferenciami, takže hodnotenie novinových článkov je realizované binárnou formou. V skutočnosti, ale neurčujeme negatívne príklady, nakoľko z charakteru dát by potenciálnym zdrojom boli novinové články, ktoré používateľ neprečítal alebo tie, na ktoré neklikol pri zobrazení odporúčania (miera prekliku je len 1%), čo vyústi v priveľkú množinu dát, ktorá by diskreditovala aj ostatné pozitívne príklady. Informačný priestor na novinových portáloch je priveľký na to, aby používateľ navštívil skutočne všetky

články, ktoré ho potenciálne môžu zaujímať. Kandidátom bol aj algoritmus kolaboratívneho filtrovania založenom na podobnosti objektov, založený na výpočte spoločného výskytu interakcií [48]medzi množinou používateľov a odporúčaných objektov. Jeho rýchlosť sme overili pri experimentoch nad dátami z portálu SME.sk, kde sme zistili rádovo dlhšiu dobu výpočtu, v porovnaní s ALS algoritmom. Pri spracovaní 8 miliónov impresií vo forme párov *používateľ : identifikátor článku*, z dát z portálu SME.sk sa výpočet na 4 jadrovom osobnom počítači pohyboval okolo hranice 2 hodín, zatiaľ čo algoritmus ALS bol schopný spracovať 9 miliónov impresií z platformy ORP už za 113 sekúnd (pri dosiahnutí vyššej presnosti odporúčaní, síce na odlišných dátových množinách, ale rovnakého charakteru). Takýto radikálny časový rozdiel by zamedzoval frekventovanému prepočítavaniu naučených modelov a komponent kolaboratívneho filtrovania by nebol schopný generovať aktuálne odporúčania.

Pri kolaboratívnom odporúčaní sme implementovali 2 spôsoby generovania odporúčaní, a to:

- *Generovanie top N odporúčaní pre používateľa* – na základe prečítaných článkov v minulosti odporučíme pre používateľa top N najlepších odporúčaní.

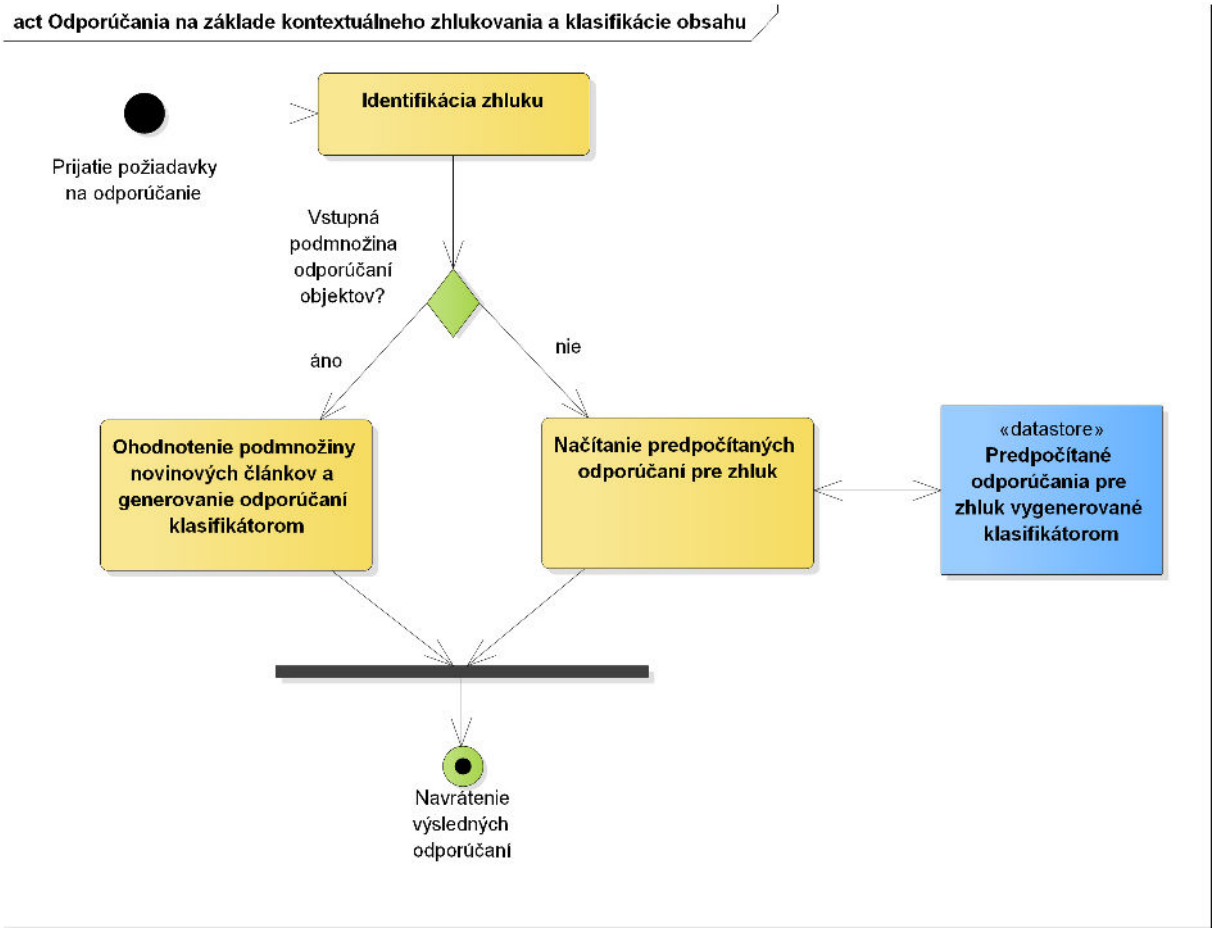
- *Ohodnotenie vstupnej množiny objektov* – na vstupe v kolaboratívnom filtrovaní môže byť podmnožina novinových článkov, získaná z iného odporúčacieho komponentu. Výstupom sú miery istoty odporúčaní pre jednotlivé objekty zo vstupnej množiny, z pohľadu kolaboratívneho filtrovania.

V prípade použitia prvej možnosti je vhodnejšie odporúčania generovať a predpočítavať na pozadí, v prípade druhej možnosti je možné poskytnúť odporúčania v reálnom čase v desiatkach milisekúnd aj na osobnom počítači (Predpokladaná vstupná množina < 20 obejtkov). Algoritmus ALS je parametrizovateľný, preto sme experimentálne overovali presnosť odporúčaní v kapitole 8.3 cez MAP (angl. Mean Average Precision), kde sme hľadali ideálne parametre počtu iterácií a počtu latentných faktorov pre ALS model s dátami z ORP. V procese trénovania modelu ALS prikladáme identickú váhu prirodzeným impresiám ako aj konverziám cez reklamu resp. odporúčania všetkých tímov na platforme ORP.

7.2.3 Odporúčania na základe kontextuálneho zhlukovania a predikcie relevancie obsahu

Jednou z našich kľúčových hypotéz je tvrdenie, že kontext je cennou informáciou pri odporúčaní a dokáže pozitívne ovplyvniť presnosť odporúčaní. V nami realizovanej metóde sme vytvorili samostatný odporúčací komponent, ktorý je založený práve na kontextuálnych informáciách prináležiacim impresiám a požiadavkám na odporúčanie. Vstupom do procesu odporúčania môže byť voliteľne podmnožina článkov resp. odporúčaní, ktoré boli vygenerované iným odporúčacím komponentom. Najpodstatnejším rozdielom, voči zaužívaným metódam (napr. kolaboratívne filtrovanie, k-nn), je odlišný spôsob vnímania používateľa. Používateľa v tejto časti metódy personalizovaného odporúčania nahrádzame zhlukom, ktorý bol identifikovaný na základe kontextuálnych informácií, resp. samotným kontextom, ktorý prislúcha požiadavke na odporúčanie.

[48] https://mahout.apache.org/users/algorithms/intro-cooccurrence-spark.html



Obr. 18 Proces odporúčania v komponente využívajucom kontextuálne zhlukovanie

Samotný odporúčací proces, resp. resp. odpoveď na požiadavku na odporúčanie pozostáva z dvoch krokov (Obr. 18):

- *Identifikácia zhluku* – Podľa kontextuálnych informácií požiadavky na odporúčanie, využijeme vopred naučený model zhlukovania na to, aby sme vedeli požiadavku na odporúčanie zaradiť do zhluku požiadaviek a impresií. Výstupom je identifikovaný zhluk.
- *Odporučenie novinových článkov* – Na základe identifikovaného zhluku vyberieme naučený model rozhodovacích stromov, konkr. náhodný les a pomocou regresie ohodnotíme množinu článkov, ktoré boli identifikované ako potenciálni kandidáti na základe ich popularity a aktuálnosti. Druhou variantou, ktorú sme tiež implementovali je predpočítavanie odporúčaní pre jednotlivé zhluky off-line, a to v momente pridávania alebo aktualizovania novinových článkov do odporúčacieho systému. Druhá alternatíva je časovo vhodnejšia v prostrediach, kde je vyžadovaná odpoveď v pár desiatkach milisekúnd, nakoľko v procese odporúčania vynechávame ohodnotenie novinových článkov a použijeme už uložené predpočítané hodnotenia pre jednotlivé zhluky (získateľné z úložiska do 1ms). Naopak, v prostrediach, kde je očakávaná odpoveď systému rádovo v stovkách milisekúnd je vhodnejšie použiť prvú spomínanú variantu (ak chceme odporúčať na základe trendovosti novinových článkov). S použitím vhodného cachovacieho mechanizmu je možné ale aj takýto výpočet urýchliť pri opakovaných požiadavkách na odporúčanie. Na pozadí sa totižto

vyhneme pravidelnému prepočítavaniu regresie pre všetky novinové články, ktoré v systéme figurujú a sú označené na odporúčanie pri aktualizácií prediktorov.

Pri zhlukovaní na základe kontextu sme využili existujúcu implementáciu škálovateľného algoritmu k-means v nástroji Apache Spark. Zhlukovanie impresií a požiadaviek na základe kontextu priebeha on-line, v reálnom čase, nakoľko tento krok nie je možné predpočítavať. Parametre pri učení modelu sme určili experimentálne, aj s ohľadom na dobu výpočtu. Ďalším dôvodom, pre ktorý sme sa rozhodli uberať cestou kontexuálneho zhlukovania a následného predikovania relevantných objektov, je radikálna redukcia dimenzionality vstupných dát. Namiesto miliónov používateľov figuruje v procese odporúčania (a v procese strojového učenia) konštantné k – počet uzlov. Nie je tak nutné učiť veľké množstvo prediktorov, ale v našej metóde využívame opäť maximálne k rozličných modelov strojového učenia, od čoho očakávame rýchlejšie počítanie, čo vedie k pravidelnejšiemu aktualizovaniu modelov. Práve tieto aspekty metódy overujeme v kapitole 8.

Pri výbere kontextuálnych atribútov, na základe ktorých prebieha indetifikovanie zhluku, sme vybrali množinu s ohľadom najmä na dostatočnú frekvenciu výskytu atribútov pri jednotlivých impresiách a požiadavkách na odporúčanie. Jedná sa o nasledovné atribúty: *odhadované pohlavie, odhadovaný vek, ohadovaný príjem domácnosti, počasie, deň v týždni, hodina dňa, typ zariadenia, jazyk používateľovho systému, preferencie cookies - sledovanie, geografická poloha a internetový poskytovateľ.* Spoločnou charakteristikou atribútov je aj ich pomerná stálosť, v zmysle relatívne konštantného počtu hodnôt pre jednotlivé atribúty. Využívame enkódovanie „jeden z“ (angl. one-hot encoding) vo vlastnej implementácii s rastúcim indexom pozície párov atribút:hodnota (Algoritmus 3).

```
aktuálny_kontextuálny_index = 0

kontext_cache = {}

enkódovaný_kontext = []

pre každý atribút a, hodnotu h z kontextuálnych atribútov

  ak cache[a:h]

    enkódovaný_kontext[cache[a:h]] = 1

  inak

    Mutex.uzamkni

      kontext_cache[a:h] = aktuálny_kontextuálny_index

      enkódovaný_kontext[aktuálny_kontextuálny_index] = 1

      aktuálny_kontextuálny_index++

    Mutex.odomni
```

Algoritmus 3 Proces enkódovania atribútov do vektora

Ako atribúty figurujúce v prediktoroch novinových článkoch sme zvolili tieto atribúty: *doména novinového článku, vydavateľ, obsah článku formou tf vektorov.* Pri klasikátoroch nemusíme riešiť enkódovanie atríbutov, nakoľko existujúce implementácie poskytujú možnosti zadefinovania kategorických atribútov. Pri trénovaní modelov regresie pomocou rozhodovacích stromov sme definovali ako pozitívne príklady pre zhluk tie články, ktoré sa nachádzali medzi 50% najčítanejších v danom zhluku.

8 Overenie metódy personalizovaného odporúčania

V rámci overenia riešenia realizovanej metódy sme si stanovili tieto ciele:

- Experimentálne nájsť vhodnú kombináciu atribútov pre algoritmy strojového učenia, aj v pomere na ich náročnosť na výpočtový čas,
- Evaluáciami zistiť vplyv popularity a jej efekt pri segmentácii impresií,
- Vyhodnotenie presnosti jednotlivých odporúčaných komponentov metódy,
- Vyhodnenie presnosti hybridnej metódy.

Pri overovaní a experimentoch sme pracovali s rôznymi dátovými zdrojmi, on-line aj off-line charakteru, vďaka čomu sme mali možnosť overiť časť metódy aj v skutočnom prostredí.

8.1 Popularita a aktuálnosť novinových článkov

Prvou hypotézou, ktorú sme overovali v doméne novinových článkov je nasledovné tvrdenie:

> *Pridaním aspektu popularity a aktuálnosti článkov ako sekundárneho zdroja dát dokážeme zvýšiť presnosť generovaných odporúčaní.*

Používatelia webových portálov sú často lákaní aj samotnými tvorcami rozhraní na to, aby ich zaujali najnovšie alebo aktuálnejšie najpopulárnejšie články. Ak by sme odporúčali čisto iba najpopulárnejšie články, samotné metriky a predikcie by boli relatívne vysoké, ale už by sa nejednalo personalizované odporúčanie. V našej metóde figuruje popularita ako pomocný aspekt.

8.1.1 Overenie vplyvu popularity na dátach denníka SME

V prvotnom overení sme pracovali s off-line dátami denníka SME, ktoré obsahovali 11 miliónov impresií, od 800-tisíc požívateľov počas Októbra 2009. Pozorovali sme vplyv popularity na presnosť odporúčaní pri využití jednoduchého algoritmu kolaboratívneho filtrovania založeného na podobnosti medzi objektami, určené cez vzájomný spoločný výskyt impresií u požívateľov. Zaujímavým zistením, ktoré sme pozorovali pri využití 7 dennej trénovacej a 3 dennej testovacej množiny je fakt, že 75% impresií v testovacej množine nevieme predpovedať, pretože navštívené články sa v trénovacej množine nevyskytovali, čo podtrháva dynamickosť domény novinových článkov a portálov. Prínos popularity a aktuálnosti článkov (Tab. 7) sa pohyboval pri najlepšom nastavení algoritmu k-nn pri kolaborantívnom filtrovaní na hranici 100% (zlepšenie presnosti odporúčaní o 95%), v metrikách p@10 a NDCG@10. Pri experimentoch sme vyskúšali tri odlíšné spôsoby metódy agregácie normalizovaných výsledkov odporúčania:

- Výber maximálnej istoty odporúčania,
- Sumácia istôt odporúčaní z k-nn a aspektu popularity,
- Súčin istôt odporúčaní.

Pri overeniach sme dosiahli najvyššiu presnosť práve so súčinom istôt, aj keď výsledky s použitím sumácie prinášali 95% presnosti súčinového prístupu.

Tab. 7 Vplyv popularity a aktuálnosti pri kolaboratívnom filtrovaní

Počet najblížších susedov	*Bez popularity a aktuálnosti článkov*		*S popularitou a aktuálnosťou článkov*			
	p@10	*NDCG@10*	*p@10*		*NDCG@10*	
2	0.0086	0.0410	0.0179	**+ 108%**	0.0803	**+ 96%**
3	0.0093	0.0411	0.0190	**+ 104%**	0.0792	**+ 93%**
5	0.0114	0.0495	0.0222	**+ 95%**	0.0906	**+ 83%**
7	0.0104	0.0453	0.0228	**+ 19%**	0.0913	**+ 102%**
10	0.0108	0.0446	0.0243	**+ 125%**	0.0952	**+ 113%**
15	0.0107	0.0414	0.0229	**+ 114%**	0.0860	**+ 108%**
20	0.0109	0.0418	0.0229	**+ 110%**	0.0840	**+ 101%**
30	0.0103	0.0412	0.0225	**+ 118%**	0.0809	**+ 96%**

8.1.2 Vplyv popularity na mieru prekliku na platforme ORP

Druhým experimentom sledujúcim vplyv popularity a aktuálnosti článkov, bol on-line experiment realizovaný na platforme ORP, v ktorom sme sledovali mieru preklikov odporúčaných článkov. Popularitu určujeme nie len globálnu, ale aj segmentovanú podľa atribútov novinových článkov. Najzaujímavejším atribútom, podľa ktorého určujeme najpopulárnejšie články je zhluk určený na základe kontextu jednotlivých impresií. Počas 6 dní sme vygenerovali 42 112 odporúčaní pomocou 8 odlišných stratégií resp. algoritmov odporúčania, ktoré boli vyberané náhodne s priemernou mierou preklikov 0,69% a maximálnou mierou preklikov 0,96%. Pre nás najzaujímavejším výsledkom je CTR pri počítaní popularity poďla navrhovaného kontextuálneho zhlukovania, kedy identifikovaný zhluk na zakláde prítomného kontextu nahrádza rolu atribútu impresie resp. novinového článku. CTR 0,849% v tomto prípade prekonáva globálnu popularitu, čo je signálom zmysluplného zhlukovania požiadaviek na odporúčanie a impresií pri personalizovanom odporúčaní.

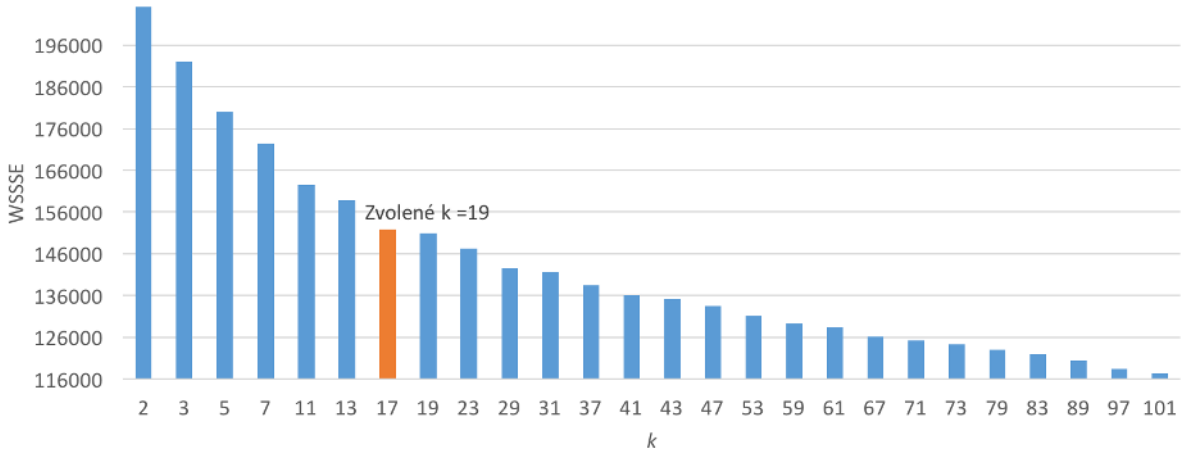
Tab. 8 Miera preklikov pri odporúčaniach na základe popularity na platforme ORP

Algoritmus	*Počet požiadaviek na odporúčanie*	*Počet preklikov*	*CTR*
Globálna aktuálnosť	8432	27	0,32021%
Globálna popularita	4512	38	0,84220%
Globálna popularita a aktuálnosť	5322	47	0,88313%
Popularita poďla kanálu	3367	14	0,41580%
Popoularita poďla kategórie	4839	19	0,39264%
Popularita poďla domény	4433	42	0,94744%
Popularita podľa zhluku	**5184**	**44**	**0,84877%**
Popularita podľa vydavateľa	6023	58	0,96298%

8.2 Zhlukovanie na základe kontextu

Pri zhlukovaní kontextu sme experimentálne určovali parameter *k* v algoritme *k-means,* ktorý určuje počet výsledných centroidov, resp. zhlukov. Ako metriku optimalizácie sme zvolili WSSSE (angl. Within Set Sum of Squared Error), ktorá vyjadruje vzdialenosť vektorov objektov v zhluku od centroidu zhluku. Pri tejto metrike je očakávané, že s počtom centroidov bude klesať. Naším cieľom bolo nájsť hranicu, od ktorej bude tento pokles pomalší a z časového hľadiska bude výpočet ešte relatívne rýchly. Zároveň sme sa snažili nastaviť *k* tak, aby počet príkladov v jednotlivých zlukoch bol dostatočne veľký a obsahoval určitý počet príkladov. Táto požiadavka závisí od vstupnej množiny, pri nami realizovanom overení sme použili dáta preklikov z ORP za prvé tri dni, využitím 5 krížovej validácie. WSSSE počítame ako sumu vzdialeností vektorov p od centroidu c , pre každý objekt v k zhlukoch.

$$WSSSE = \sum_{i=1}^{k} \sum_{p \in C} \|p - c\|^2$$



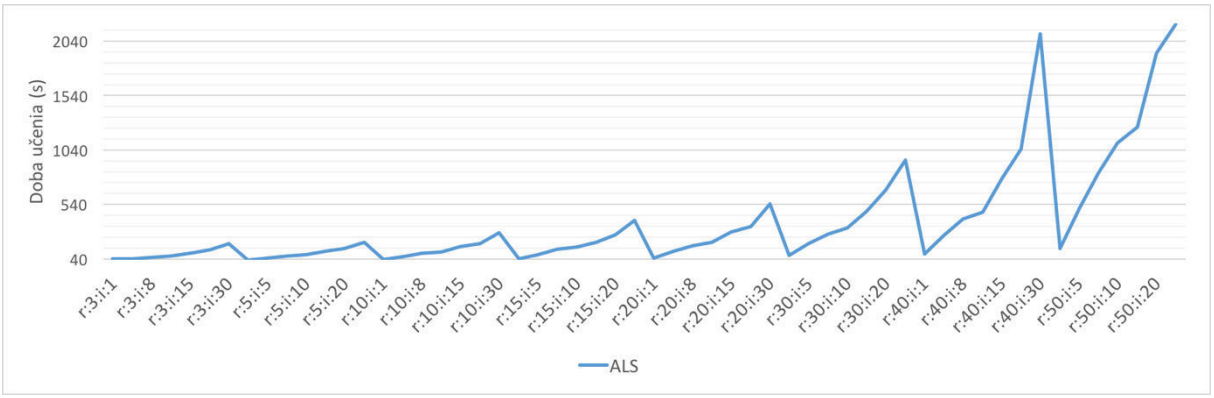
Obr. 19 K-means hľadanie k podľa WSSSE metriky

8.3 Kolaboratívne filtrovanie

Pri kolaboratívnom filtrovaní pomocou algoritmu ALS sú dvomi najvplyvnejšími vstupnými parametrami *rank* – počet latentných faktorov, ktoré bude algoritmus odhaľovať a počet iterácií, prípadne hranica, kedy má algoritmus ALS ukončiť učenie svojho modelu. Pri metrike presnosti (primárne p@10), sme ako najlepšiu vstupnú kombináciu na dátach platformy ORP určili parametre *rank*: 5*, počet iterácií*: 20. Zaujímavým je aj prudký pád prenosti pri náraste počtu latentných faktorov na 10, pri zachovaní počtu iterácii klesne presnosť o 26%. Experimenty sme realizovali pri natrénovaní na 3 dňoch a verifikácií na 4. dni dát z ORP.

Obr. 20 Presnosť odporúčaní pri parametrizácií modelu ALS

Z pohľadu času sa ukázali vyššie počty latentných faktorov ako limitujícim parametrom, pričom nárast počtu iterácií rastie lineárne (Obr. 21).



Obr. 21 Doba výpočtu modelov ALS v závislosti od parametrov rank a počtu iterácií

8.4 Predikcia relevenacie obsahu po zhlukovaní na základe kontextu

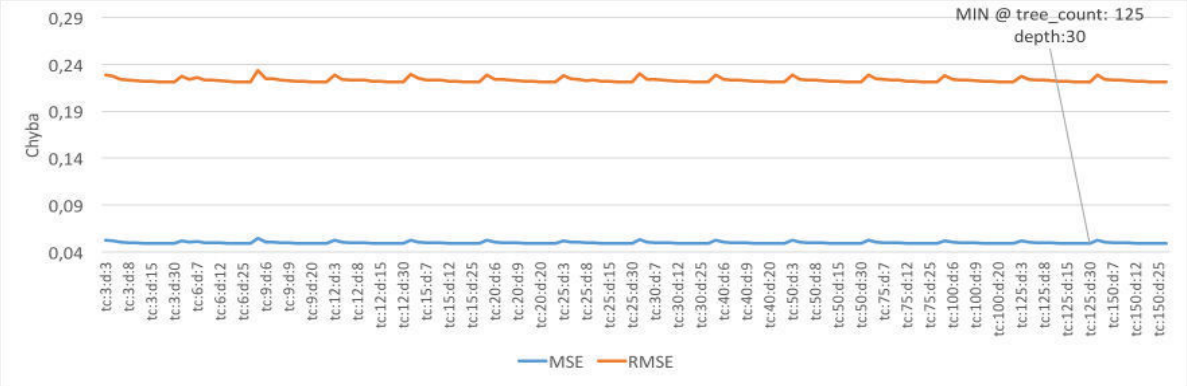
Potom, ako pomocou zhlukovania odhadneme príslušnosť kontextu požiadavky na odporúčanie, prichádza na radu predikcia relevancie novinových článkov pomocou rozhodovacích stromov. V návrhu metódy sme vybrali 2 algoritmy, pomocou ktorých dokážeme predikovať úroveň potenciálneho zaujatia zhluku vybraným článkom. Experimentami sme overili vhodnosť parametrov a sledovali sme čas výpočtu učenia modelov. Ako metriku optimalizácie sme zvolili MSE (angl. Mean squared error), resp. variant RMSE (angl. Root mean squared error), ktorá v porovnaní s MSE zvyšuje penalizácie pri najmenej presných predikciách.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

$$RMSE = \sqrt{MSE}$$

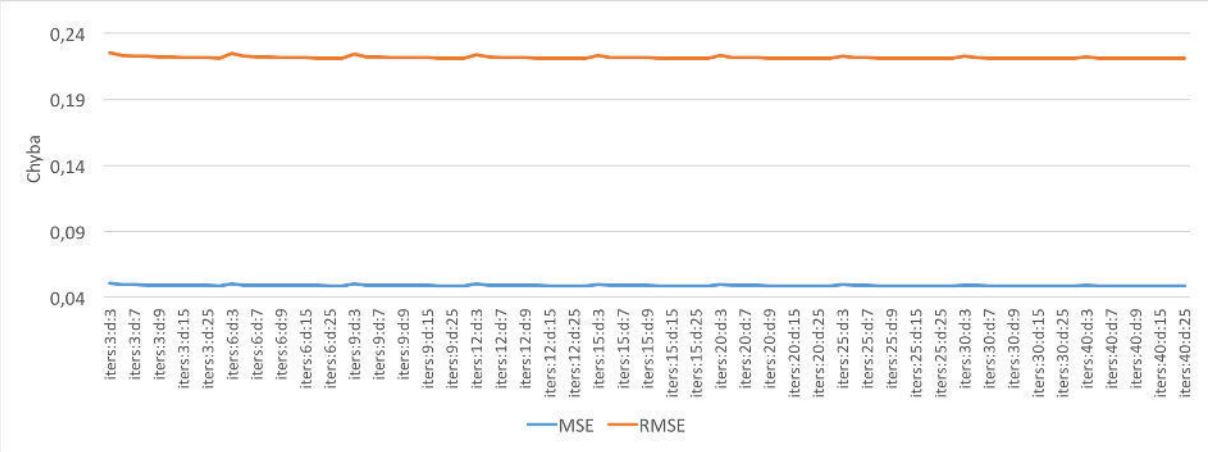
Histogramy návštev pre jednotlivé zhluky mali charakter dlhého chvosta, a tak sme určili hranicu výskytu pozitívnych príkladov >= 3, pre jednotlivé zhluky. Pri experimentoch s parametrami sa vo všetkých troch kritériach – minimálna MSE a RMSE, čas naúčenia

ukázal algoritmus Random Forest ako lepšia voľba, teda pri ďalších overeniach používame v našej metóde na regresiu práve algoritms Random Forest. Učenie GBTs z časového hľadiska, pri vyšších počtoch iterácií, ktoré bolo nutné navoliť aby sa priblížil chybovosti náhodného lesa, trvalo v porovnaní s náhodným lesom aj násobne dlhšie (Obr. 26). Experimenty sme realizovali na podmnožine dát z ORP (na prvých siedmych dňoch). Pri experimentoch sme sledovali varianciu MSE pre jednotlivé z hluky z množiny 19 zhlukov, pre model Random Forest bola hodnota variancie 0,0029, MSE dosiahla hodnotu 0,049.



Obr. 22 MSE a RMSE predikcií modelu Random Forest

Pri pohľade na graf (Obr. 22), kde *tc (angl. tree count)* na osi x označuje počet stromov, *d (angl. depth)* označuje hĺbku stromov, je možné vyčítať silnejší vplyv hĺbky rozhodovacích stromov ako ich počtu v náhodnom lese. Podobný, ale vyváženejší priebeh mali metriky predikcie chyby aj pre GBTs (Obr. 23).

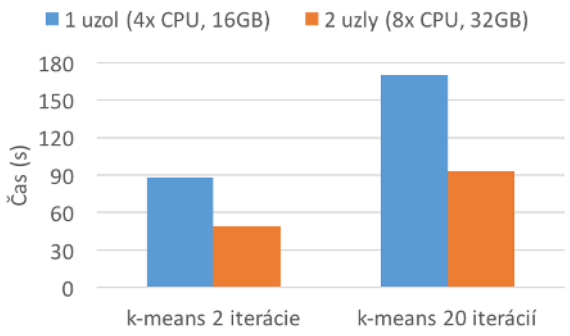


Obr. 23 MSE a RMSE predikcií modelu GBTs

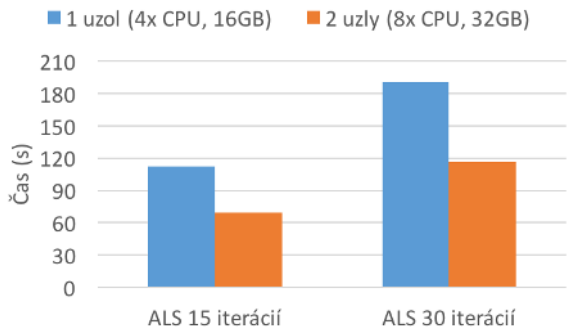
8.5 Škálovateľnosť algoritmov strojového učenia

Jedným z hlavných cieľov našej práce bolo navrhnutie metódy s doubrou škálovateľnosťou. Kritickým miestom v našej metóde môže byť schopnosť učenia modelov strojového učenia s narastajúcim objemom dát. Z tohto dôvodu sme realizovali experimenty, simulujúce zapojenie 2 identicky výkonných strojov (4x 2.2Ghz CPU, 16GB RAM) do klastra pre Apache Spark. V prvých meraniach sme ukázali takmer lineárnu škálovateľnosť algoritmu k-means||, kedy doba výpočtu klesla o 43% (2 iterácií) resp. 46% (20 iterácií) po zapojení dvoch uzlov

klastra, v porovnaní s jedným výpočtovým uzlom. Merania prebiehali pri zhlukovaní kontextuálneho obsahu veľkosti 400MB. Druhým algoritmom, ktorý sme chceli overiť z hľadiska škálovateľnosti bol algoritmus ALS, pri ktorom sme merali dobu výpočtu na 7 miliónoch impresiách, d dvoch variantách počtu iterácii. V prípade 15 iterácií sme zaznamenali zrýchlenie o 40%, v prípade behu s 30 iteráciami sa jednalo o zrýchlenie výpočtu o 39%.

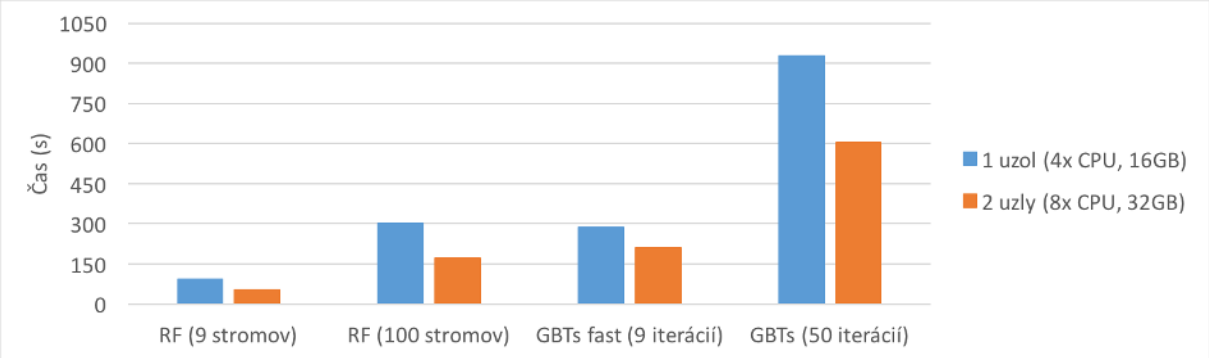


Obr. 2424 Škálovateľnosť algoritmu k-means



Obr. 255 Škálovateľnosť algoritmu ALS

Ďalšími modelmi, ktoré vstupujú do procesu odporúčania sú modely rozhodovacích stromov. Algoritmus Random Forest je ľahšie paralelizovateľný ako GBTs, čím by mohla byť aj jeho škálovateľnosť lepšia. Tento jav sa nám na pozorovaní času výpočtu na vzorke 15 milíonov kontextuálnych dát, extrahovaných z impresií, potvrdil. Pri väčších objemoch dát, kedy je možné distribuovať výpočty v rámci jednotlivých iterácii algoritmu GBTs, je možné očakávať zrýchlenie výpočtov lineárneho charakteru, pretože sieťova réžia sa s narastajúcim objemom dát stane zanedbateľnou. Pri algoritme Random Forest sme dosiahli zrýchlenie o 43% resp. 44%. Pri GBTs prinieslo zvýšenie výpočtových zdrojov skrátenie doby výkonu o 35% resp 38%.



Obr. 26 Vplyv navýšenia výpočtov zdrojov na dobu výpočtu modelov rozhodovacích stromov

8.6 Záverečné overenie hybridnej metódy odporúčania

Posledným overením, ktoré sme realizovali bolo porovnanie presnosti odporúčaní využitím rôznych stratégií odporúčania navrhovanej metódy. Na overenie sme využili off-line dátovú množinu z ORP, ktorá sme rozdelili na 7 trénovacích a 5 testovacích dní. Overenia prebiehali na podmnožine 27 249 používateľov, ktorí počas piatich testovacích dní zaznamenali aspoň

jednu impresiu za každý deň a celkovo aspoň 10 impresií (používatelia identifikovaní pomocou cookie v prehliadači). Presnosť odporúčaní sme sledovali aj zvlášť pre jednotlivé testovacie dni, kde sme očakávali postupnú degradáciu presnosti, pretože naúčené modely strojového učenia neboli počas experimentov aktualizované a tak položky, ktoré pribudli ako nové počas 5 dní nedokázali odporúčať. Sledovali sme 4 metriky, a to presnosť najlepších 3, 5, 10 (p@3, p@5, p@10) odporúčaní a vlastnú metriku počtu používateľov, ktorým sme správne odporúčili aspoň 1 z desiatich odporúčaní. Ako popúlarne články sme vybrali globálne najpopulárnejšie články z testovacej množiny. Pri agregáciách jednotlivých odporúčaní sme využívali súčinovú variantu spomínanú v kapitole 7.

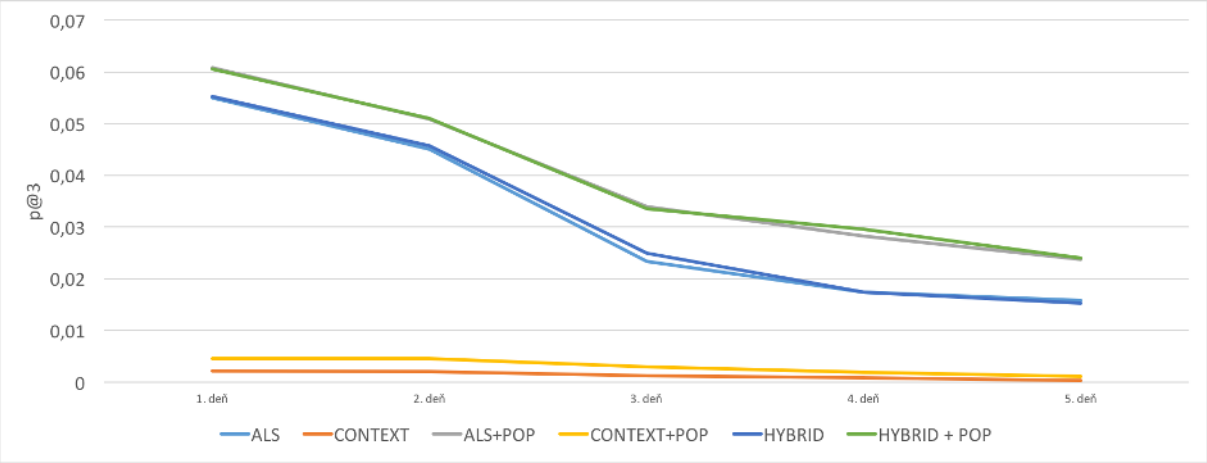
Testovali sme tri stratégie odporúčania: kolaborantívne filtrovanie (ALS), kontextuálne zhlukovanie a klasifikácia (CTX), hybridná metóda, ktorá ich kombinuje (HYBRID). Pri všetkých troch stratégiach sme overovali aj výsledky po pridaní popularity (+POP) do procesu odporúčania. Pri kontexuálnom zhľukovaní sme pomerovo určovali počet odporúčaní pre používateľov na základe impresiám priradeným zhlukom, vďaka čomu sme získali váhu pre odporúčania z daného zhluku pri tvorbe finálnych metrík presnosti. Pre vysvetlenie, ak sme pre požívateľa zaznamenali spolu 5 impresií, pričom 3 pri z nich bol identifikovaný zhluk 9 a pri 3 z nich bol identifikovaný zhluk 11, do výpočtu presnosti vstupovali hodnoty pre zhluk 9 prenásobene 3/5 a pre zhluk 11 prenásobené 2/5 tak, aby sme zachovali maximálnu mieru presnosti (p@k) 1, v prípade odporučenia novinových článkov, ktoré používateľ navštívil.

Tab. 8 Presnosť pri odporúčaniach na základe popularity na platforme ORP

Stratégia odporúčania	*p@3*	*p@5*	*p@10*	*% používateľov s odporučenou položkou*
ALS	0,03797	0,02514	0,01478	13,89%
CTX	0,00122	0,00172	0,00267	0,37%
ALS+POP	0,04120	0,03193	0,03437	16,75%
CTX+POP	0,00318	0,01270	0,02329	1,88%
HYBRID	0,03817	0,02514	0,01484	16,79%
HYBRID+POP	0,04121	0,03193	0,03478	16,92%

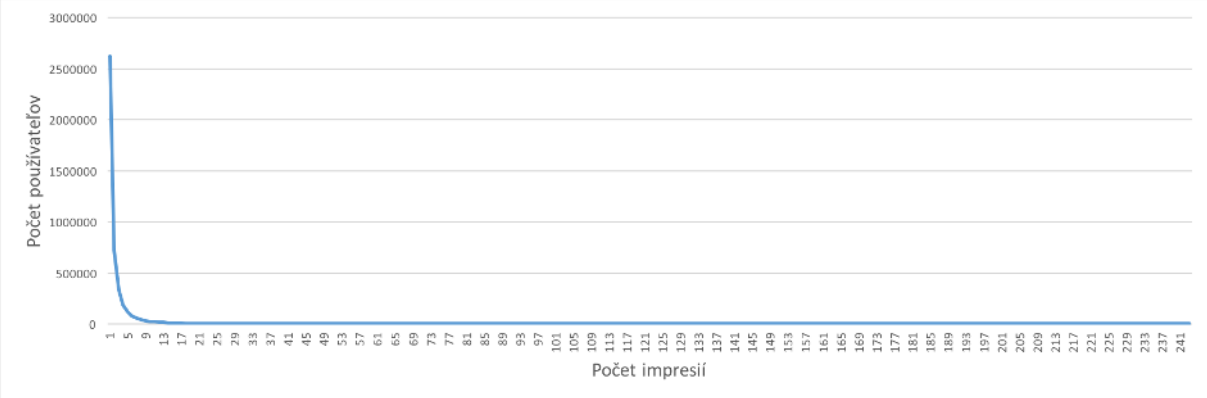
Z pozorovaní na dátovej množine 5 testovacích dní (Tab. 8), vidíme všeobecne nízke výsledky bez ohľadu na algoritmus. V najlepšom prípade sme boli schopní správne z 10 odporúčaní odporučiť novinový článok správne iba pre 16,92% používateľov a dosiahli sme p@3 (najčastejšie požadované množstvo odporúčaní na ORP) 0,03921. Najlepší výsledok dosiahli takmer zhodne metóda kolaboratívneho filtrovania s popularitou novinových článkov s nami navrhovanou hybridnou metódou. To implikuje minimálny prínos kontextuálneho zhlukovania pri overení na off-line dátovej množine ORP. Pozitívny a očakávany je prínos zapracovania popularity pri odporúčaniach, ktorý najviac dopomohol práve kontextuálnemu odporúčaniu. Tým že sa odporúčací systém spolieha na strojové učenie, ktoré v čase experimentov neaktualizujeme prichádzame o veľkú množinu novinových článkov, ktoré používateľom nedokážeme odporučiť, keďže sme zvolili relatívne veľkú testovaciu množinu dát, v dynamickej doméne odporúčaní novinových článkov.

Pri práci s dátami denníka SME, v testovacej množine vytvorenej z 3 dní nasledujúcich po trénovacej množine pozostávalo až 75% impresií z nových novinových článkoch, ktoré sa v trénovacej množine nevyskytli. Vplyv a degradáciu presnosti odporúčaní počas 5 testovacích dní pri metrike p@3 uvádzame nižšie (Obr. 27). Na grafe možeme sledovať minimálnu prenosť odporúčaní pri posledných dňoch z testovacej množiny. Vplyv globálnej popularity pomáha eliminovať neaktúalnosť prediktívných modelov, avšak už pri 4. a 5. dni vplyv popularity klesá. Vyššia degradácia na tretí deň, môže byť spôsobená začiatkom nového týždňa a vysokým objemom nových článkov publikovaných v pondelok.



Obr. 27 Degradácia odporúčacieho systému vzhľadom na čas

Uvádzame aj histogram počtu návštev jednotlivých používateľov v trénovacej množine. Ako môžeme vidieť (Obr. 28), rozloženie má tvar dlhého chvosta, čo znamená, že väčšina používateľov má minimálny počet zaznamenaných impresií, a tým pádom sa tu objavuje aj problém studeného štartu. Až u 60% používateľov patriciach do podmnožiny nie anonymných, sme zaznamenali len jednu impresiu. Charakteristika dát potvrdzuje silný vplyv popularity a zvýšenie p@10 pri kontextovom odporúčaní pri pridaní aspektu popularity, kde pre používateľov s malým počtom návštev algoritmus odporučí globálne najpopulárnejšie články, u ktorých je väčšia pravdepodobnosť ich čítania.



Obr. 28 Histogram počtu impresií nie anonymných používateľov v trénovacích dátach

9 Záver a zhodnotenie

Personalizované odporúčanie, ako jedna z možností na redukciu informačného zahltenia sa v poslednom období teší veľkej popularite. Používateľom prináša vyšší komfort pri práci s informačnými systémami, pre komerčnú sféru predstavuje spôsob na zvýšenie obratu. V súčasnosti objem používateľov informačných systémov prudko narastá, s čím rastú aj nároky na výpočtovú efektívnosť odporúčacích systémov. Požiadavky na spracovanie dát a informácií v reálnom čase alebo aktualitu modelov používateľov nútia tvorcov odporúčacích systémov zamýšľať sa aj nad aspektom škálovateľnosti personalizovaného odporúčania. Populárna je najmä distribúcia výpočtov, ich paralelizácia alebo alternatívne spôsoby využitia hardvéru – výpočet pomocou GPU.

V práci sme uviedli analýzu personalizovaného odporúčania a súvisiacich kvalitatívnych vlastností, modelovania používateľských preferencií, škálovateľnosti a architektúr odporúčacích systémov. V tejto analýze sa potvrdil dôraz na škálovateľnosť odporúčacích systémov, ktorú sa snaží dosiahnuť komerčná aj akademická sféra, nakoľko objem dát aj používateľov prudko rastie. Analýzou najnovších prístupov v odporúčaní založenom na obsahu, kolaboratívnom filtrovaní a hybridnom odporúčaní sme získali dôležité vedomosti a prehľad, ktoré sme pretavili do návrhu metódy odporúčania.

Navrhujeme škálovateľnú a flexibilnú metódu hybridného personalizovaného odporúčania zohľadňujúcu obsah položiek, kontext a správanie používateľov. V navrhovanej metóde je priestor pre rozličné možnosti stratégií odporúčania a agregácie finálnych odporúčaní, čím zabezpečujeme aj flexibilitu systému – schopnosť systému prispôsobiť sa aktuálnemu zaťaženiu. Silnú váhu pri odporúčaniach má popularita a aktuálnosť novinových článkov. Správanie používateľov je vstupnou množinou pre populárne kolaboratívne odporúčanie, ktoré aj v našej metóde dosahuje najlepšiu presnosť odporúčaní, ktorú sme ešte zvýšili pridaním aspektu popularity.

Vysoký dôraz kladieme na aktuálnosť používateľských preferencií, preto pracujeme aj so samotným kontextom, ktorý dnes nie je problém získať vďaka stále rastúcej popularite rôznych technologických zariadení. V súčasnosti používatelia často používajú minimálne dva druhy zariadení (mobil, osobný počítač alebo tablet) pre prehliadanie webu. Používatelia majú rôzne informačné potreby, aj v závislosti od obdobia roka, dňa v týždni alebo hodiny dňa. Ako ďalšie kontextuálne informácie radíme informácie o polohe používateľa, používaný operačný systém, jazyk alebo dobu reakcie pri navigácii vo webovom prostredí. V budúcnosti môžeme očakávať rozširovanie množiny kontextuálnych informácii o údaje akými sú napr. aktuálne informácie o zdravotnom stave používateľa, používateľov tep, teplotu alebo mieru únavy. Práve kontextuálne informácie sú v našej metóde kľúčovým zdrojom informácií pre proces zhlukovania. Na základe kontextu pristupujeme k požiadavkám na odporúčanie pre používateľov ako požiadavkám pre odporúčanie pre zhluk. V komponente kontextuálneho odporúčania tak abstrahujeme od stálych preferencií používateľa. Naopak, zameriavame sa na dynamické vlastnosti formou kontextu, podľa ktorých identifikujeme vhodný zhluk požiadaviek na odporúčanie. Je teda možné povedať, že pri kontextuálnom odporúčaní modelujeme krátkodobé implicitné preferencie používateľov.

Výstupnú informáciu po zhlukovaní využívame ako jednoduchý filter pre agregovanie populárnych článkov v rámci zhlukov. Komplexnejšiu metódu, ktorá je založená na identifikovaní zhlukov požiadaviek na odporúčanie sme realizovali ako predikciu relevancie obsahu položiek pomocou rozhodovacích stromov. Pomocou nej vie náš odporúčací systém ohodnotiť relevanciu položiek z obsahových informácií pre jednotlivé zhluky. Takéto odporúčanie netrpí problémom studeného štartu na strane používateľa, ani položky. Nevýhodou je nutné frekventované aktualizovanie modelov a prepočítavanie preferencií zhluku, aby sme zachovali aktuálnosť a dynamiku preferencií, aj vzhľadom na dianie vo svete. Toto negatívum je ale prítomné aj pri zaužívanom kolaboratívnom filtrovaní a ako sa ukázalo pri experimentoch, implementované prediktory a modely dobre škáľujú, čiže pri využití distribuovaného počítania je zachovanie dynamiky už len otázkou systémových a finančných zdrojov.

Metódu sme realizovali v doméne novinových článkov. Pracujeme s prúdom dát z viacerých novinových portálov v reálnom čase, prevádzkovaných na platforme ORP, ale aj s off-line dátovými množinami. Pri overení navrhnutej metódy sme uskutočnili experimenty a inkrementálne overenia pre jednotlivé komponenty odporúčacieho systému. Experimentálne sme určili parametre vstupujúce do procesu zhlukovania na základe kontextu (počet zhlukov), kolaboratívneho filtrovania (počet latentných faktorov a iterácií) a algoritmov rozhodovacíh stromov (počet stromov, hĺbka). V overeniach sa ukázal silný, pozitívny vplv popularity novinových článkov aj pri zúžitkovaní v procese hybridného odporúčania. Kontextuálne zhlukovanie prinieslo pozitívne výsledky vo forme mierneho zvýšenia miery prekliku, pri segmentácií počítania popularity podľa identifikovaných zhlukov v porovnaní s globálnym počítaním popularity novinových článkov. Prínos kontextuálneho odporúčania sa prejavil aj pri zvýšení množstva používateľov, ktorým vedel náš odporúčací systém správne odporučiť aspoň jednu položku. Ako samostatný komponent pre odporúčanie prinieslo kontextuálne odporúčanie obsahu rádovo slabšiu presnosť ako zaužívané kolaboratívne filtrovanie. Navrhovaná hybridná metóda odporúčania je aplikovateľná aj v iných doménach, kde vieme získať obsahové vlastnosti odporúčaných položiek (napr. multimédia – filmy, hudba), modelovať popularitu a aktuálnosť (využitím odlišných intervalov a parametrov), nakoľko doména novinových článkov je pomerne dynamická.

Zoznam použitej literatúry

ACIAR, Silvana, Debbie ZHANG, Simeon SIMOFF & John DEBENHAM, 2007. Recommender System Based on Consumer Product Reviews. In: *Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence (WI 2006 Main Conference Proceedings)(WI'06)*.

ADOMAVICIUS, Gediminas, Ramesh SANKARANARAYANAN, Shahana SEN & Alexander TUZHILIN, 2005. Incorporating contextual information in recommender systems using a multidimensional approach. *ACM Trans. Inf. Syst.* vol. 23, no. 1, pp. 103–145.

ADOMAVICIUS, Gediminas & Alexander TUZHILIN, 2005. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*. vol. 17, no. 6, pp. 734–749.

ADOMAVICIUS, Gediminas, Alexander TUZHILIN & Rong ZHENG, 2011. REQUEST: A query language for customizing recommendations. *Information Systems Research*. vol. 22, no. 1, pp. 99–117.

AGARWAL, Deepak, Bee-Chung CHEN & Pradheep ELANGO, 2010. Fast online learning through offline initialization for time-sensitive recommendation. *Online*. vol. 29, pp. 703–712.

ALI, Kamal & Wijnand VAN STAM, 2004. TiVo: Making Show Recommendations Using a Distributed Collaborative Filtering Architecture. *Proceedings of the 2004 ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '04*. p. 394.

AMATRIAIN, Xavier, 2013. Big & Personal: data and models behind Netflix recommendations. *Proceedings of the 2nd International Workshop on Big Data*. pp. 1–6.

AMATRIAIN, Xavier, J.M. PUJOL, X. AMATRIAIN, J.M. PUJOL, Nava TINTAREV, N. TINTAREV, Nuria OLIVER & N. OLIVER, 2009a. Rate it Again: Increasing Recommendation Accuracy by User re-Rating. *Proceedings of the 2009 ACM conference on Recommender systems*. pp. 173–180.

AMATRIAIN, Xavier, Josep M. PUJOL & Nuria OLIVER, 2009b. I like it... i like it not: Evaluating user ratings noise in recommender systems. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. vol. 5535 LNCS, pp. 247–258.

AMIN, Mohammad Shafkat, Baoshi YAN, Sripad SRIRAM, Anmol BHASIN & Christian POSSE, 2012. Social referral: leveraging network connections to deliver recommendations. *Proceedings of the sixth ACM conference on Recommender systems*. pp. 273–276.

ARDISSONO, Liliana, Cristina GENA, Pietro TORASSO, Fabio BELLIFEMINE, Angelo DIFINO & Barbara NEGRO, 2004. User Modeling and Recommendation Techniques for Personalized Electronic Program Guides. *Personalized Digital Television*. no. 2002, pp. 3–26.

ASSAD, Mark, David J CARMICHAEL, Judy KAY & Bob KUMMERFELD, 2007. PersonisAD: Distributed, Active, Scrutable Model Framework for Context-Aware Services. *Pervasive Computing*. vol. 4480, pp. 55–72.

BAHMANI, Bahman, Benjamin MOSELEY, Andrea VATTANI, Ravi KUMAR & Sergei VASSILVITSKII, 2012. Scalable K-Means ++. *Proceedings of the VLDB Endowment*. vol. 5, no. 7, pp. 622–633.

BALABANOVIĆ, Marko, 1997. *An adaptive Web page recommendation service*. 1997. ISBN 0897918770.

BALAKRISHNAN, Suhrid & Sumit CHOPRA, 2012. Collaborative ranking. *Proceedings of the fifth ACM international conference on Web search and data mining - WSDM '12*. p. 143.

BATISTA, GEAPA & Diego Furtado SILVA, 2009. How k-nearest neighbor parameters affect its performance. *Argentine Symposium on Artificial Intelligence*. no. Asai, pp. 1–12.

BELL, Robert M. & Yehuda KOREN, 2007. Lessons from the Netflix prize challenge. *ACM SIGKDD Explorations Newsletter*. vol. 9, no. 2, p. 75.

BERKOVSKY, Shlomo, 2005. Ubiquitous User Modeling in Recommender Systems. In: *User Modeling 2005*. B.m.: Springer Berlin Heidelberg, p. 496–498. ISBN 9783540278856.

BERKOVSKY, Shlomo, Tsvi KUFLIK & Francesco RICCI, 2008. Mediation of user models for enhanced personalization in recommender systems. *User Modelling and User-Adapted Interaction*. vol. 18, no. 3, pp. 245–286.

BERTIER, Marin, Davide FREY, Rachid GUERRAOUI, Anne Marie KERMARREC & Vincent LEROY, 2010. The Gossple anonymous social network. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. vol. 6452 LNCS, pp. 191–211.

BILENKO, Mikhail & Matthew RICHARDSON, 2011. Predictive client-side profiles for personalized advertising. *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '11*. p. 413.

BOBADILLA, J., F. ORTEGA, a. HERNANDO & a. GUTIÉRREZ, 2013. Recommender systems survey. *Knowledge-Based Systems*. B.m.: Elsevier B.V., vol. 46, pp. 109–132.

BOUTET, Antoine, Davide FREY & Anne-marie KERMARREC, 2014. HyRec : Leveraging Browsers for Scalable Recommenders Categories and Subject Descriptors. pp. 85–96.

BREWER, E., 2012. *CAP twelve years later: How the 'rules' have changed*. 2012. ISBN 9781605588889.

BRUSILOVSKY, Peter & Eva MILLÁN, 2007. User models for adaptive hypermedia and adaptive educational systems. *The adaptive web*. pp. 3–53.

BURKE, Robin, 2007. Hybrid web recommender systems. In: *The adaptive web*. p. 377–408. ISBN 978-3-540-72078-2. Retrieved z: doi:10.1007/978-3-540-72079-9_12

CHANG, Fay, Jeffrey DEAN, Sanjay GHEMAWAT, Wilson C. HSIEH, Deborah a. WALLACH, Mike BURROWS, Tushar CHANDRA, Andrew FIKES & Robert E. GRUBER, 2006. Bigtable: A distributed storage system for structured data. *7th Symposium on Operating Systems Design and Implementation (OSDI '06), November 6-8, Seattle, WA, USA*. pp. 205–218.

CHAPELLE, Olivier & Yi CHANG, 2011. Yahoo ! Learning to Rank Challenge Overview. *JMLR: Workshop and Conference Proceedings*. vol. 14, pp. 1–24.

CHEN, Bee-chung, 2010. fLDA : Matrix Factorization through Latent Dirichlet Allocation. *New York*. pp. 91–100.

CLAYPOOL, Mark, Phong LE, Makoto WASED & David BROWN, 2001. Implicit interest indicators. *Proceedings of the 6th international conference on Intelligent user interfaces - IUI '01*. pp. 33–40.

CLAYPOOL, Mark, Tim MIRANDA, Anuja GOKHALE, Pavel MURNIKOV, Dmitry NETES & Matthew SARTIN, 1999. Combining content-based and collaborative filters in an online newspaper. In: *Proceedings of Recommender Systems Workshop at ACM SIGIR*. p. 40–48.

CORMODE, Graham, 2003. Count-Min Sketch. *AT&T Labs–Research*. pp. 1–5.

COULOURIS, F, J DOLLIMORE & Tim KINDBERG, 2011. *Distributed Systems: Concepts and Design*. Boston: Addison-Wesley. ISBN 9780132143011.

CREMONESI, Paolo, Franca GARZOTTO & Roberto TURRIN, 2012. Investigating the Persuasion Potential of Recommender Systems from a Quality Perspective. *ACM Transactions on Interactive Intelligent Systems*. vol. 2, no. 2, pp. 1–41.

DAS, Abhinandan, 2007. Google News Personalization : Scalable Online. *WWW 2007*. pp. 271–280.

DAS, Abhinandan S., Mayur DATAR, Ashutosh GARG & Shyam RAJARAM, 2007. Google news personalization. *Proceedings of the 16th international conference on World Wide Web - WWW '07*. p. 271.

DEAN, Jeffrey & Sanjay GHEMAWAT, 2008. MapReduce: Simplified Data Processing on Large Clusters. *Communications of the ACM*. vol. 51, no. 1, pp. 107–113.

DEMOVIC, Lubos, Eduard FRITSCHER, Jakub KRIZ, Ondrej KUZMIK, Ondrej PROKSA, Diana VANDLIKOVA, Dusan ZELENIK & Maria BIELIKOVA, 2013. Movie recommendation based on graph traversal algorithms. *Proceedings - International Workshop on Database and Expert Systems Applications, DEXA*. pp. 152–156.

DESHPANDE, Pallavi k & Banchhor CHITRAKANT, 2014. Survey on Recommender Systems. *International Journal of Engineering Research and Development*. vol. 10, no. 6, pp. 49–54.

DESROSIERS, Christian & George KARYPIS, 2010. A novel approach to compute similarities and its application to item recommendation. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. vol. 6230 LNAI, pp. 39–51.

DESROSIERS, Christian & George KARYPIS, 2011. A comprehensive survey of neighborhood-based recommendation methods. *Recommender Systems Handbook*. vol. 69, no. 11, pp. 107–144.

DONG, Wei, Charikar MOSES & Kai LI, 2011. Efficient k-nearest neighbor graph construction for generic similarity measures. *Proceedings of the 20th international conference on World wide web - WWW '11*. pp. 577–586.

DOURISH, Paul, 2004. What We Talk About When We Talk About Context. *Personal and Ubiquitous Computing*. vol. 8, no. 1, pp. 19–30.

EKSTRAND, Michael D., 2010. Collaborative Filtering Recommender Systems. *Foundations and Trends® in Human–Computer Interaction*. vol. 4, no. 2, pp. 81–173.

FORMOSO, Vreixo, Diego FERNÁNDEZ, Fidel CACHEDA & Victor CARNEIRO, 2014. Distributed architecture for k-nearest neighbors recommender systems. *World Wide Web*. pp. 1–21.

FOX, a. & E.a. BREWER, 1999. Harvest, yield, and scalable tolerant systems. *Proceedings of the Seventh Workshop on Hot Topics in Operating Systems*.

GADANHO, Sandra Clara & Nicolas LHUILLIER, 2007. Addressing uncertainty in implicit preferences. *Proceedings of the 2007 ACM conference on*. pp. 97–104.

GARCIA ESPARZA, Sandra, Michael P. O&APOS;MAHONY & Barry SMYTH, 2012. Mining the real-time web: A novel approach to product recommendation. *Knowledge-Based Systems*. B.m.: Elsevier B.V., vol. 29, pp. 3–11.

GERBER, Simon, Michael FRY, Judy KAY, Bob KUMMERFELD, Glen PINK & Rainer WASINGER, 2010. PersonisJ: Mobile, client-side user modelling. In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. B.m.: Springer Berlin Heidelberg, p. 111–122. ISBN 3642134696.

GHAZANFAR, Mustansar Ali & Adam PRÜGEL-BENNETT, 2010. Building switching hybrid recommender system using machine learning classifiers and collaborative filtering. *IAENG International Journal of Computer Science*. vol. 37, no. 3.

GLAUBER, Rafael, Angelo LOULA & Joao B. ROCHA-JUNIOR, 2013. A mixed hybrid recommender system for given names. *Proceedings of the ECML PKDD Discovery Challenge - Recommending Given Names*. pp. 25–36.

GONG, SongJie Gong SongJie, HongWu Ye HongWu YE & Ping Su Ping SU, 2009. A Peer-to-Peer Based Distributed Collaborative Filtering Architecture. *2009 International Joint Conference on Artificial Intelligence*. pp. 2–4.

GONZÁLEZ, Gustavo, Beatriz LÓPEZ & Josep Lluis DE LA ROSA, 2005. A multi-agent smart user model for cross-domain recommender systems. *Beyond Personalization: A Workshop on the Next Stage of Recommender Systems Research*. pp. 93–94.

GUNAWARDANA, Asela & Christopher MEEK, 2008. Tied boltzmann machines for cold start recommendations. *Proceedings of the 2008 ACM conference on Recommender systems RecSys 08*. p. 19.

HAN, Peng, Bo XIE, Fan YANG & Ruimin SHEN, 2004. A scalable P2P recommender system based on distributed collaborative filtering. *Expert Systems with Applications*. vol. 27, pp. 203–210.

HECKMANN, Dominik, 2005. Distributed User Modeling for Situated Interaction. *35. GI Jahrestagung, Informatik 2005 - Situierung, Individualisierung und Personalisierung, Lecture Notes in Informatics (LNI)*. pp. 266–270.

HU, Yifan, Florham PARK, Yehuda KOREN & Chris VOLINSKY, 2008. Collaborative Filtering for Implicit Feedback Datasets.

HUANG, Xl, M TIWARI & S SHAH, 2013. *Structural Diversity in Social Recommender Systems*.

INDYK, Piotr, 1999. A small approximately min-wise independent family of hash functions. *Journal of Algorithms*. vol. 38, pp. 84–90.

IWATA, Tomoharu, Tomoharu IWATA, Kazumi SAITO, Kazumi SAITO, Takeshi YAMADA & Takeshi YAMADA, 2007. Modeling user behavior in recommender systems based on maximum entropy. *Proceedings of the 16th international conference on World Wide Web - WWW '07*. no. 1, p. 1281.

JAJVAND, Atefeh, Mir Ali SEYYEDI & Afshin SALAJEGHEH, 2013. A Hybrid Recommender System for Service Discovery. *International Journal of Innovative Research in Computer and Communication Engineering*. vol. 1, no. 6, pp. 1342–1347.

JAWAHEER, Gawesh, Martin SZOMSZOR & Patty KOSTKOVA, 2010. Characterisation of Explicit Feedback in an Online Music Recommendation Service Categories and Subject Descriptors. *Analysis*. pp. 317–320.

JAWAHEER, Gawesh, Peter WELLER & Patty KOSTKOVA, 2014. Modeling User Preferences in Recommender Systems. *ACM Transactions on Interactive Intelligent Systems*. vol. 4, no. 2, pp. 1–26.

JEH, Glen & Jennifer WIDOM, 2002. SimRank: a measure of structural-context similarity. *Proceedings of the eighth ACM SIGKDD international conference*. pp. 1–11.

KARYDI, Efthalia & Konstantinos G MARGARITIS, 2014. Parallel and Distributed Collaborative Filtering: A Survey. *CoRR*. vol. 1409.2762, pp. 1–46.

KAŠŠÁK, O, 2014. *Skupinové Odporúčanie Pre Inteligentnú Televíziu*. B.m. b.n.

KHULLER, Samir, Anna MOSS & Joseph (Seffi) NAOR, 1999. The budgeted maximum coverage problem. *Information Processing Letters*. vol. 70, pp. 39–45.

KIM, Byeong Man, 2003. Clustering approach for hybrid recommender system. *Proceedings IEEE/WIC International Conference on Web Intelligence (WI 2003)*. pp. 33–38.

KOREN, Y., R. BELL & C. VOLINSKY, 2009. Matrix Factorization Techniques for Recommender Systems. *Computer*. vol. 42, no. 8, pp. 42–49.

KOREN, Yehuda, 2008. Factorization meets the neighborhood: a multifaceted collaborative filtering model. *Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. pp. 426–434.

KOREN, Yehuda, 2010. Factor in the neighbors: Factor in the Neighbors: Scalable and Accurate Collaborative Filtering. *ACM Transactions on Knowledge Discovery from Data*. vol. 4, no. 1, pp. 1–24.

KOREN, Yehuda & Joe SILL, 2011. OrdRec : An Ordinal Model for Predicting Personalized Item Rating Distributions. *Proceedings of the 5th ACM conference on Recommender systems - RecSys '11*. pp. 117–124.

KOUTRIKA, Georgia, Benjamin BERCOVITZ & Hector GARCIA-MOLINA, 2009. FlexRecs: expressing and combining flexible recommendations. *Proceedings of the 35th SIGMOD international conference on Management of data*. pp. 745–758.

LAMPROPOULOS, Aristomenis S., Paraskevi S. LAMPROPOULOU & George a. TSIHRINTZIS, 2012. A Cascade-Hybrid Music Recommender System for mobile services based on musical genre classification and personality diagnosis. *Multimedia Tools and Applications*. vol. 59, no. 1, pp. 241–258.

LI, L, Dingding WANG, Tao LI, Daniel KNOX & Balaji PADMANABHAN, 2011. SCENE: a scalable two-stage personalized news recommendation system. *Sigir*. pp. 125–134.

LI, Lihong, Wei CHU & John LANGFORD, 2010. A Contextual-Bandit Approach to Personalized News Article Recommendation. pp. 661–670.

LIN, Jovian, Kazunari SUGIYAMA, Min-Yen KAN & Tat-Seng CHUA, 2013. Addressing Cold-Start in App Recommendation : Latent User Models Constructed from Twitter Followers Categories and Subject Descriptors. *Sigir*. pp. 283–292.

LINDEN, Greg, Brent SMITH & Jeremy YORK, 2003. Amazon.com Recommendations Item-to-Item Collaborative Filtering. *IEEE Internet Computing*. vol. 7, no. 1, pp. 76–80.

LIU, Jiahui, Peter DOLAN & Er PEDERSEN, 2010. Personalized news recommendation based on click behavior. *2010 International Conference on Intelligent User Interfaces*. pp. 31–40.

LIU, Nathan, 2011. Wisdom of the Better Few : Cold Start Recommendation via Representative based Rating Elicitation. *Proceedings of the 5th ACM conference on Recommender systems - RecSys '11*. pp. 37–44.

NEUMAN, Clifford, 1994. Scale in Distributed Systems. *Readings in Distributed Computing Systems.* pp. 1–28.

NICTA, A N U, Jordan CHRISTENSEN & Scott SANNER, 2014. Social Collaborative Filtering for Cold-start Recommendations. *RecSys '14 Proceedings of the 8th ACM Conference on Recommender systems*. pp. 345–348.

NIEK, Tax, 2014. *Scaling Learning to Rank to Big Data Using MapReduce to Parallelise Learning to Rank*. Twente, Netherlands. University of Twente.

OARD, D. & J. KIM, 2001. *Modeling information content using observable behavior*. 2001.

PALMISANO, Cosimo, Alexander TUZHILIN & Michele GORGOGLIONE, 2008. Using context to improve predictive modeling of customers in personalization applications. *IEEE Transactions on Knowledge and Data Engineering*. vol. 20, no. 11, pp. 1535–1549.

PARASKEVOPOULOS, Fotis & Gregoris MENTZAS, 2014. A Peer to Peer Architecture for a Distributed User Model. In: *Proceedings of UMAP 2014 posters, demonstrations and late-breaking results*. Athens: National Technical University of Athens, p. 25–28.

PARK, Youngki, Sungchan PARK & Sang-goo LEE, 2013. Scalable K-Nearest Neighbor Graph Construction Based on Greedy Filtering. *Www*. pp. 227–228.

PARRA, Denis & Xavier AMATRIAIN, 2011. Walk the talk: Analyzing the relation between implicit and explicit feedback for preference elicitation. In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. p. 255–268. ISBN 9783642223617.

PAZZANI, Michael J MJ & Daniel BILLSUS, 2007. Content-based recommendation systems. *The adaptive web*. pp. 409–432.

PESSIOT, Jean-francois, Tuong-vinh TRUONG, Nicolas USUNIER, Massih-reza AMINI & Patrick

GALLINARI, 2004. Learning to rank for collaborative filtering.

RASHID, Al Mamunur, Shyong K LAM, George KARYPIS & John RIEDL, 2006. ClustKNN : A Highly Scalable Hybrid Model- & Memory-Based CF Algorithm Categories and Subject Descriptors. *WEBKDD '06*.

RENDLE, Steffen, Christoph FREUDENTHALER, Zeno GANTNER & Lars SCHMIDT-THIEME, 2009. BPR : Bayesian Personalized Ranking from Implicit Feedback. *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*. vol. cs.LG, pp. 452–461.

RODRIGUEZ, Mario, Christian POSSE & Ethan ZHANG, 2012. Multiple objective optimization in recommender systems. *Proceedings of the 6th ACM conference on Recommender systems - RecSys '12*. p. 11.

RUCKER, James, Marcos J POLANCO, W E B PAGE, Recommendation SYSTEM & That USES, 1997. Applying Collaborative Filtering to Usenet News. *Communications of the ACM*. vol. 40, no. 3, pp. 73–75.

ŠAJGALÍK, Márius, Michal BARLA & Mária BIELIKOVÁ, 2013. Efficient Representation of the Lifelong Web Browsing User Characteristics. *Second Workshop on LifeLong User Modelling, in conjunction with UMAP 2013*. pp. 21–30.

SARWAR, Badrul, George KARYPIS, Joseph KONSTAN & John RIEDL, 2002. Incremental Singular Value Decomposition Algorithms for Highly Scalable Recommender Systems. pp. 27–28.

SARWAR, Badrul M, George KARYPIS, Joseph a KONSTAN & John T RIEDL, 2000. *Application of Dimensionality Reduction in Recommender System -- A Case Study*.

SAVESKI, Martin & A MANTRACH, 2014. Item cold-start recommendations: learning local collective embeddings. *RecSys '14 Proceedings of the 8th ACM Conference on Recommender systems*. pp. 89–96.

SCHEIN, Andrew I, Alexandrin POPESCUL, Lyle H UNGAR & David M PENNOCK, 2002. Methods and metrics for cold-start recommendations. *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval SIGIR 02*. vol. 46, pp. 253–260.

SCHELTER, Sebastian, Christoph BODEN, Martin SCHENCK, Alexander ALEXANDROV & Volker MARKL, 2013. Distributed matrix factorization with mapreduce using a series of broadcast-joins. *Proceedings of the 7th ACM conference on Recommender systems - RecSys '13*. no. 1, pp. 281–284.

ŠELENG, Martin, Michal LACLAVÍK, Štefan DLUGOLINSKÝ & Ladislav HLUCHÝ, 2011. Dostupné škálovateľné riešenia pre spracovanie veľkého objemu dát a dátové sklady. In: Jan ZENDULKA and M RYCHLY, eds. *DATAKON 2011*. p. 1–22.

SHAHABI, Cyrus, Farnoush BANAEI-KASHANI, Yi-shin CHEN & Dennis MCLEOD, 2001. Yoda: An accurate and scalable web-based recommendation system. *Cooperative Information Systems*. no. September, pp. 418–432.

SHAHABI, Cyrus, Farnoush BANAEI-KASHANI, Jabed FARUQUE & Adil FAISAL, 1995. *Feature Matrices : A Model for E cient and Anonymous Mining of Web Navigations*. 1995.

SHI, Y Ue, Martha LARSON & Alan HANJALIC, 2014a. Collaborative Filtering beyond the User-Item Matrix : A Survey of the State of the Art and Future Challenges. *ACM Computing Surveys (CSUR)*. vol. 47, no. 1, pp. 1–45.

SHI, Yue, Alexandros KARATZOGLOU, Linas BALTRUNAS & Martha LARSON, 2014b. CARS 2 : Learning Context-aware Representations for Context-aware Recommendations. pp. 291–300.

SHUKLA, Shilpa, Matthew LEASE & Ambuj TEWARI, 2012. Parallelizing ListNet training using spark. *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval - SIGIR '12*. p. 1127.

SPECHT, Marcus, Andreas LORENZ & Andreas ZIMMERMANN, 2005. Towards a framework for distributed user modelling for ubiquitous computing. *Workshop on Decentralized, Agent Based and Social Approaches to User Modelling (DASUM), 9th International Conference on User Modeling*. pp. 31–40.

SPIEGEL, Stephan, Jérôme KUNEGIS & Fang LI, 2009. Hydra: A Hybrid Recommender System. *CNIKM '09*. pp. 75–80.

SU, Xiaoyuan & Taghi M. KHOSHGOFTAAR, 2009. A Survey of Collaborative Filtering Techniques. *Advances in Artificial Intelligence*. vol. 2009, pp. 1–19.

SUMBALY, Roshan, Jay KREPS & Sam SHAH, 2013. The 'Big Data' Ecosystem at LinkedIn. *International Conference on Management of Data (SIGMOD 2013)*. pp. 1–10.

SUN, Jiankai, Shuaiqiang WANG, Byron J. GAO & Jun MA, 2012. Learning to rank for hybrid recommendation. *Proceedings of the 21st ACM international conference on Information and knowledge management - CIKM '12*. p. 2239.

SYMEONIDIS, Panagiotis, Alexandros NANOPOULOS & Yannis MANOLOPOULOS, 2007. Feature-weighted user model for recommender systems. *User Modeling 2007*. pp. 97–106.

SZOMSZOR, Martin, Ciro CATTUTO, Harith ALANI, Kieron O'HARA, Andrea BALDASSARRI, Vittorio LORETO & Vito D.P. SERVEDIO, 2007. Folksonomies, the semantic web, and movie recommendation. pp. 1–14.

TINTAREV, Nava, Judith MASTHOFF, Francesco RICCI, Lior ROKACH, Bracha SHAPIRA & Paul B KANTOR, 2011. *Recommender Systems Handbook*. Boston, MA: Springer US. ISBN 978-0-387-85819-7.

VASSILEVA, Julita, 2001. Distributed User Modelling for Universal Information Access. *Universal Access in HCI: Towards An information Society for All*. vol. 3, p. 122.

WANG, Qian Wang Qian, Xianhu Yuan Xianhu YUAN & Min Sun Min SUN, 2010. Collaborative filtering recommendation algorithm based on hybrid user model. *Fuzzy Systems and Knowledge Discovery (FSKD), 2010 Seventh International Conference on*. vol. 4, no. Fskd, pp. 1985–1990.

XIA, Fen, Tie-Yan LIU, Jue WANG, Wensheng ZHANG & Hang LI, 2008. Listwise approach to learning to rank. In: *Proceedings of the 25th international conference on Machine learning - ICML '08*. p. 1192–1199. ISBN 9781605582054.

YANG, Xiwang, Yang GUO, Yong LIU & Harald STECK, 2013. A Survey of Collaborative Filtering Based Social Recommender Systems. *Computer Communications*. pp. 1–27.

ZENG, Cheng, Dawen JIA, Jian WANG, Liang HONG, Wenhui NIE, Zhihao LI & Jilei TIAN, 2012. Context-aware social media recommendation based on potential group. *Proceedings of the 1st International Workshop on Context Discovery and Data Mining - ContextDD '12*. New York, New York, USA: ACM Press, p. 1.

ZHANG, Yaming, Haiou LIU & Shiyong LI, 2011. A Distributed Collaborative Filtering Recommendation Mechanism for Mobile Commerce Based on Cloud Computing. vol. 16, no. December, pp. 3883–3891.

ZHANG, Yi & Jonathan KOREN, 2007. Efficient Bayesian Hierarchical User Modeling for Recommendation System. *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. pp. 47–54.

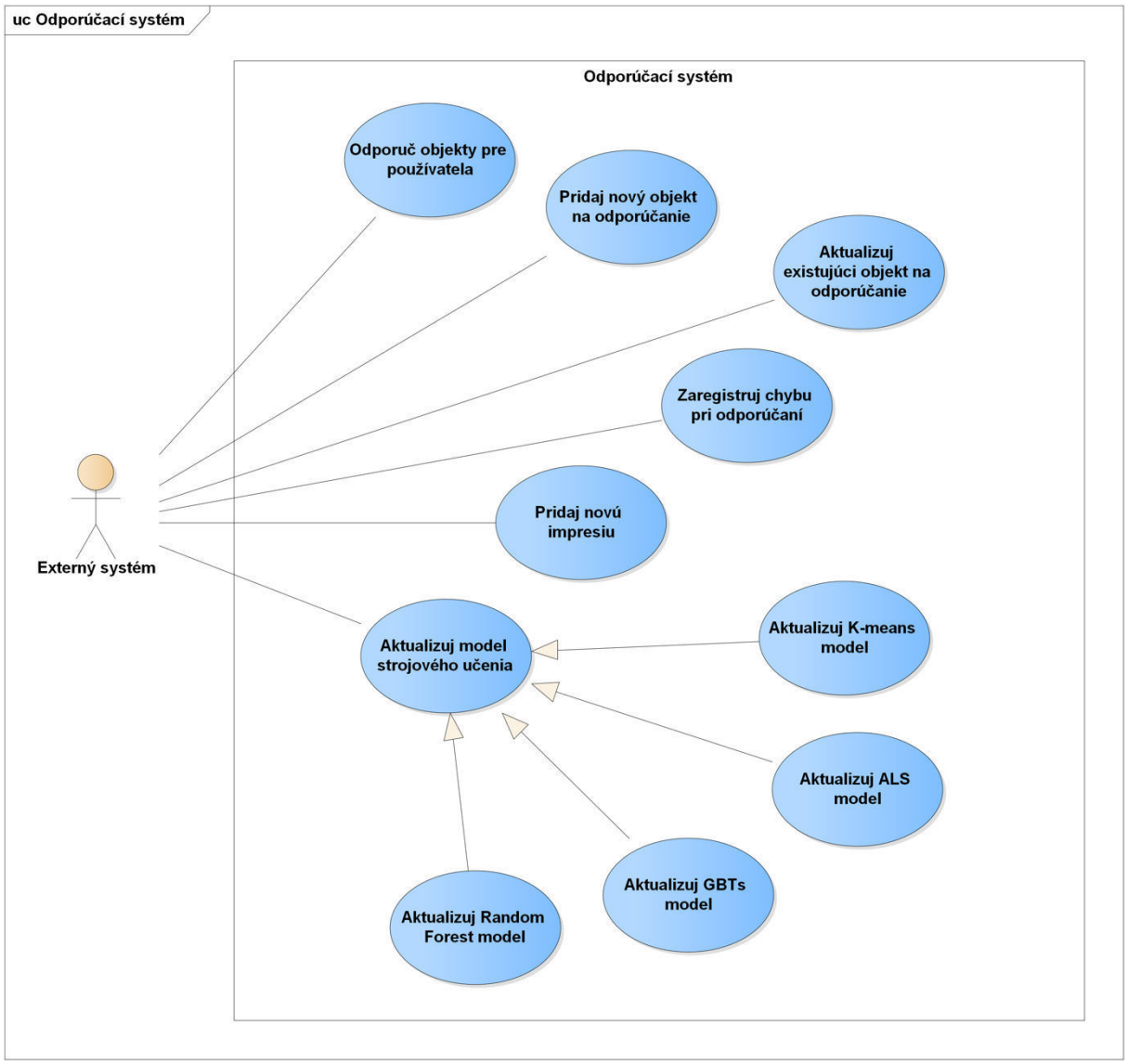
PRÍLOHY

Príloha A. Technická dokumentácia

V kapitole technickej dokumentácie opisujeme nami implementovaný odporúčací systém *ScaRec*. Poskytujeme rozličné pohľady na systém, a to statickej aj dynamickej úrovne formou diagramov UML. V podkapitola Implementácia objasňujeme použité technologie a rozhodnutia, pričom uvádzame aj zaújimavejšie časti implementácie alebo algoritmov formou ukážky zdrojových kódov.

A.1 Diagram prípadov použitia

Odporúčací systém (*Obr. A.1-1*) neposkytuje priame používateľské rozhranie pre interakciu s používateľmi.

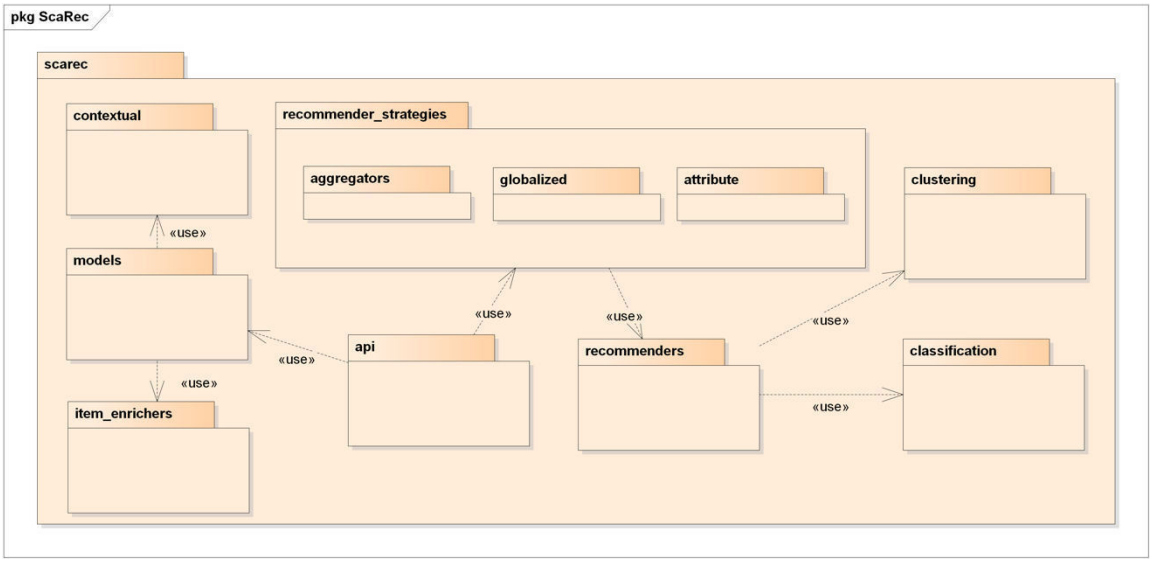


Obr. A.1-1 Odporúčací systém – diagram prípadov použitia.

Figuruje v ňom len serverové API, ktoré poskytuje externému systému nasledujúce služby:

- *Odporuč objekty pre používateľa* – externý systém na základe vstupných požiadaviek odporučí množinu objektov. Rozhodujúce vstupné informácie sú napr. identifikátor používateľa alebo objektu, s ktorým používateľ práve interaguje. Pri požiadavke na odporúčanie je možné zadefinovať akú stratégiu resp. algoritmus použiť na výber odporúčaných položiek. Pri požiadavke na odporúčanie systém spracúva kontextuálne dáta, na základe ktorých sú pri vybraných stratégiách odporúčania vygenerované odporúčania.
- *Pridaj nový objekt na odporúčanie* – externý systém môže pomocou HTTP rozhrania pridať nový objekt do systému na odporúčanie. Takto pridávaný objekt môže obsahovať obsahové informácie, jeho obsah môže byť rozšírený službami tretích strán. Novo pridaný objekt je zaindexoaný do databázy, sú z neho extrahované požadované informácie a od závislosti od zvoleného algoritmu, môže byť okamžite odporúčaný.
- *Aktualizuj existujúci objekt na odporúčanie* – podobne ako pri vytváraní nového objektu, systém cez HTTP rozhranie príjme nové dáta pre už aktualizujúci objekt v databáze – napr. aktualizácia obsahu, domény, času poslednej úpravy.
- Zaregistruj chybu pri odporúčaní – externý systém môže notofikovať odporúčací systém o nevhodnom odporúčaní (napr. neexistujúcej položke alebo príliš pomalej odozve).
- *Pridaj novú impresiu* – odporúčací systém zaregitruje novú impresiu (vo všeobecnom ponímaní sa môže jednať o hodnotenie objektu, zakúpenie) objektu. Pri spracovaní spracuje systém kontextuálne informácie, ktoré preloží do vhodnej, počítačovo spracovateľnej reprezentácie. Nové impresie neskôr ovplyvňujú štatistiky najpopulárnejších objektov.
- *Aktualizuj model strojového učenia* (a jeho špecializácie) – externý systém môže vyvolať aktualizovanie modelov strojového učenia.

A.2 Diagram balíčkov



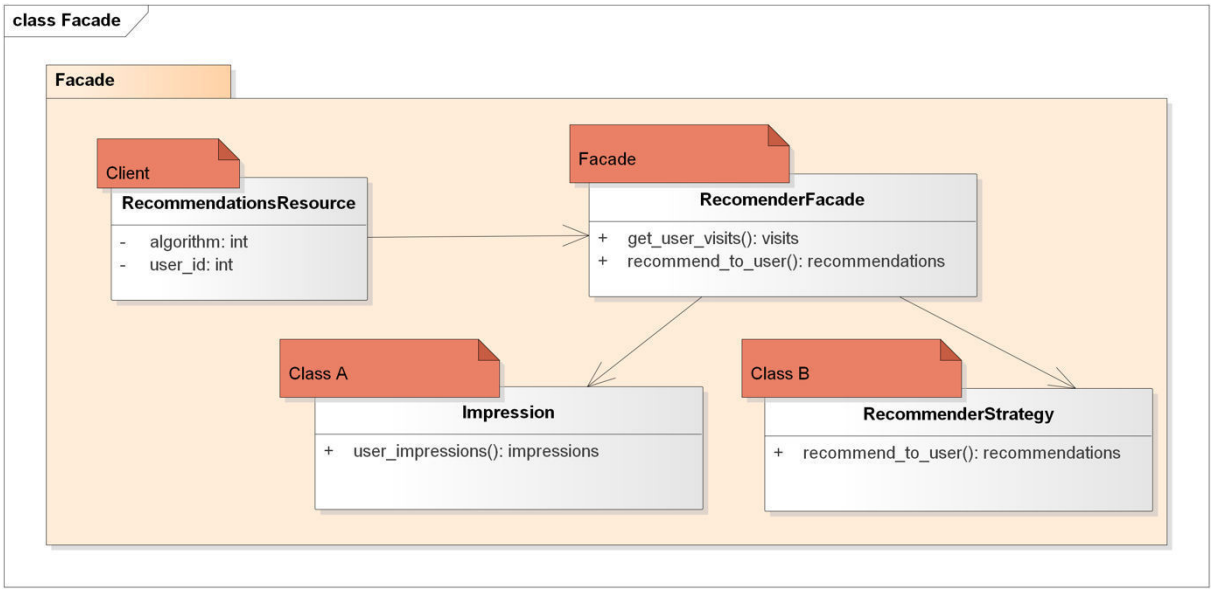
Obr. A.2-1 Diagram balíčkov odporúčacieho systému ScaRec

A.3 Diagram tried

diagram tiried a4

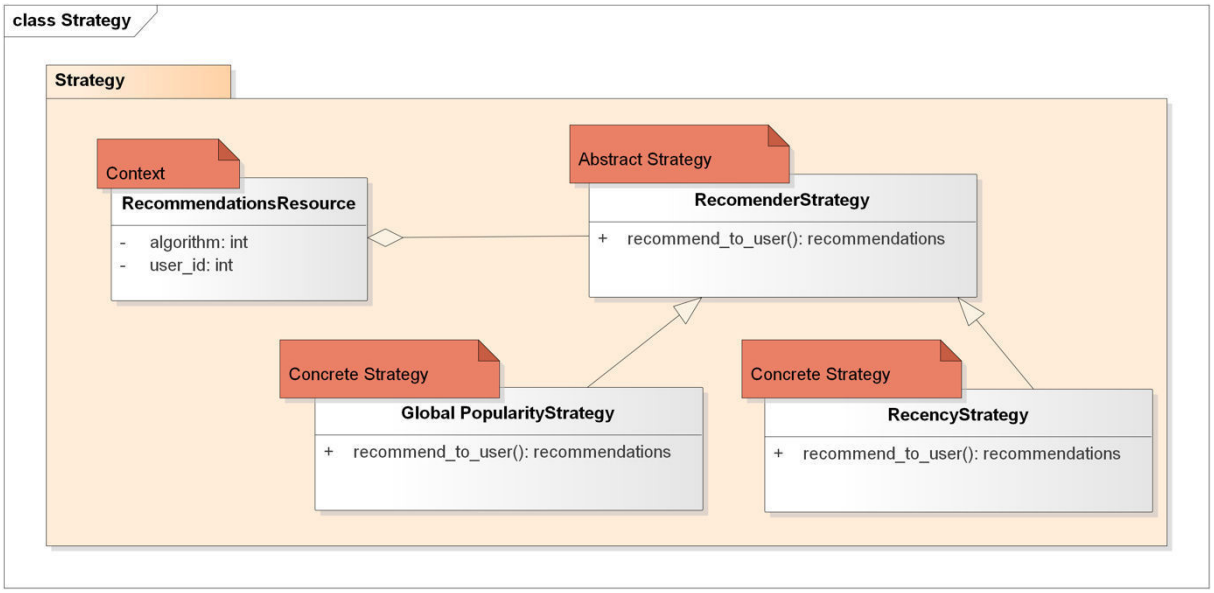
A.4 Aplikácia objektovo orientovaných vzorov

V aplikácii sme identifikovali 2 návrhové objektovo orientované vzory. Prvým z nich je fasáda (obr ref), kde úlohu fasády plní trieda *RecommenderFacade*. Obsluhuje požiadavky HTTP API klientov - *ReccomenderResource* a skrýva komunikáciu s úložiskom návštev používateľa, získaných z databázy z modelu *Impression* a získavanie samotných odporúčaní od konkrétnej odporúčacej stratégie *RecommenderStrategy*.



Obr. A.4-1 Implementácia odporúčacej fasády podľa vzoru Fasáda

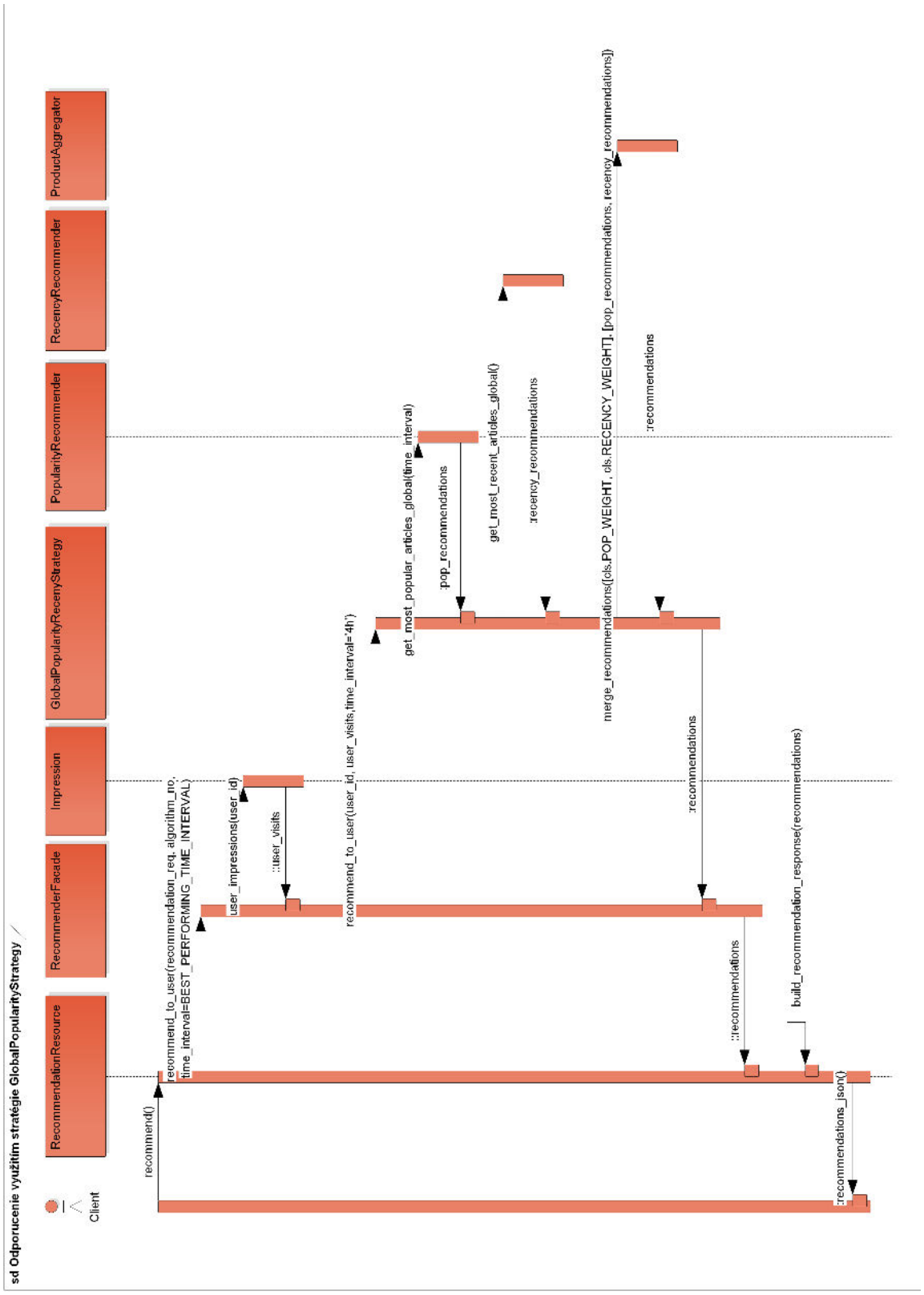
Druhým vzorom je Stratégia (**Error! Reference source not found.**). Rolu abstraktnej stratégie plní trieda *RecommenderStrategy,* kontextom je aktuálne vybraná odporúčacia stratégia definovaná triedou *RecommendationsResource,* ktorá prichádzá od HTTP klienta ako parameter. Implementáciou stratégie sú konkrétne odporúčacie stratégie ako napr. *GlobalPopularityStrategy* alebo *RecenyStrategy.*



Obr. A.4-2 Implementácia algoritmov odporúčania podľa vzoru Stratégia

A.5 Dynamické diagramy správania

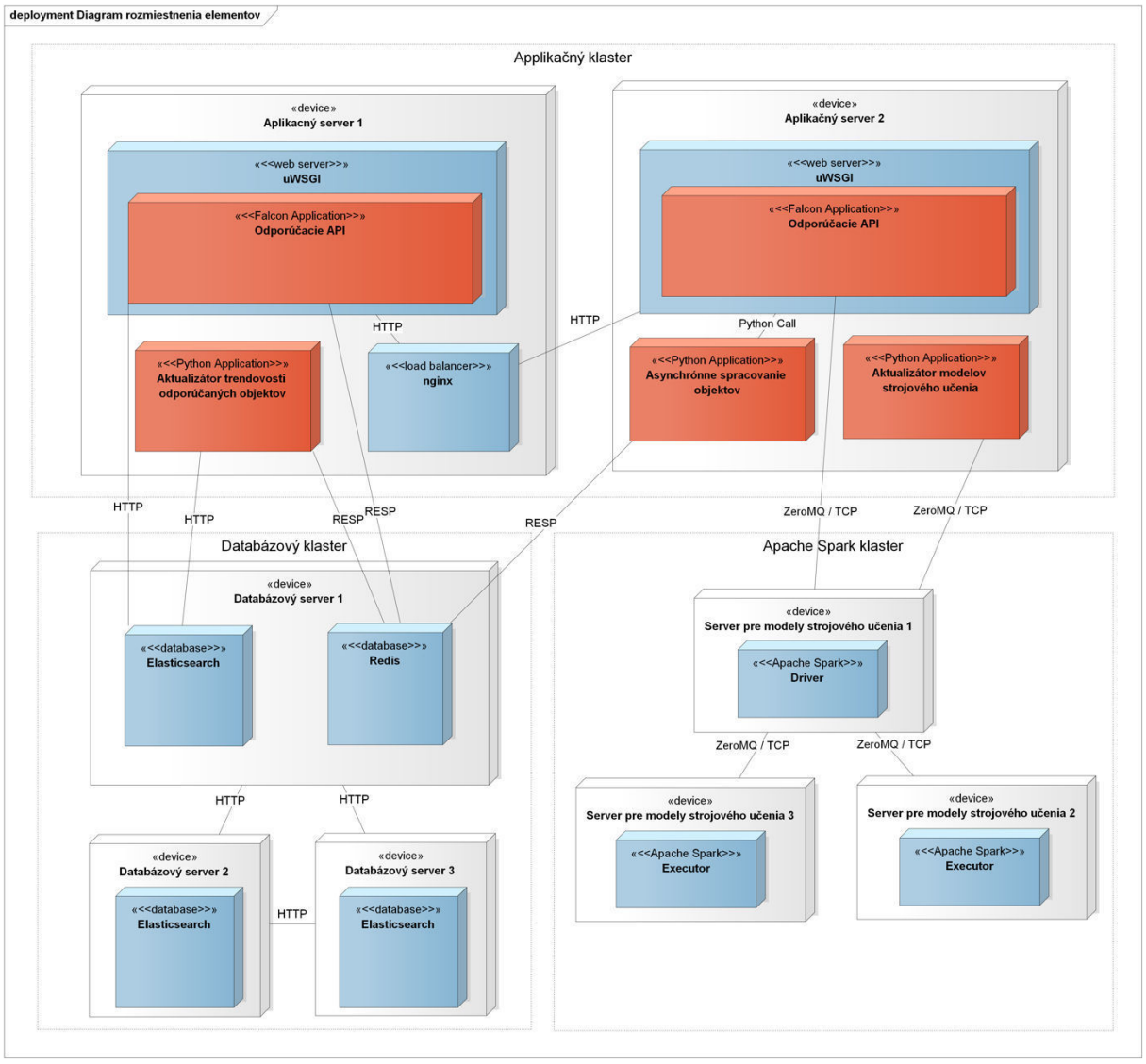
Uvádzame sekvenčný diagram modelujúci správanie systému pri odporúčaní využitím *GlobalPopularityStrategy*.



Obr. A.5-1 Odporúčania využítim GlobalPopularityStrategy

A.6 Diagram rozmiestnenia

Diagramom rozmiestnenia na *Obr. A. 1-8* navrhujeme možné rozmiestnenie elementov v systéme v produkčnom prostredí, s ohľadom na škálovateľnosť a toleranciu chýb siete a systémov. Predpokladáme nasadenie vo virtualizovanom prostredí, s viacerými dedikovanými virtuálnymi strojmi (celkovo 8) umiestnenými v jednej lokálnej sieti. Jednotlivé stroje môžu byť rozlične dimenzované a rozmiestnenie ešte optimálnejšie (výkon, efektívnost, záťaž, náklady), pretože prevádzkované služby majú rôzne nároky na systémové zdroje. Celkový systém možno z najvyššej úrovne rozdeliť do troch klastrov, podľa charakteru prevádzkovaných služieb na serveroch, ktoré bližšie opíšeme.



Obr. A.6-1 Odporúčací systém – diagram rozmiestnenia

Aplikačný klaster

V prvom klastri predkladáme konfiguráciu s dvoma strojmi, na ktorých bežia viaceré služby zastrešujúce API pre externé systémy. Ako prvý komponent obsluhujúci požiadavky odporúčacieho systému využívame Nginx, ktorý jednak ako reverzné proxy správne adresujú

požiadavky jednotlivým aplikáciam. Druhou, nie menej dôležitou funkciou Nginx-u je balancovanie prichádzajúcich požiadaviek (angl. load balancer) a ich rovnomerné smerovanie na jednotlivé aplikačné inštancie nasadené na strojoch.

Prichádzajúce požiadavky sú ďalej smerované cez HTTP protokol webovému serveru, bežiacom na platforme WSGI – uWSGI. Kľúčovým komponentom systéme je Odporúčacie API realizované ako webová aplikácia v jazyku Python použitím rámca Falcon. Na pozadí aplikačných serverov navrhujeme nasadenia procesov, ktoré majú asynchrónny, ale aj periodický charakter – raz za určitú dobu sa spustí aktualizovanie modelov alebo štatistík vyžívaných v procese odporúčania. Všetky komponenty na aplikačných serveroch komunikujú a majú prístup k dvom zostávajúcim klastrom, a to: databázový klaster a klaster pre nástroj Apache Spark.

Databázový klaster

V nami implementovanom systéme využívame ako databázové úložiská dve NoSQL databázy: Redis a Elasticsearch. V aktuálnych verziách poskytujú natívnu podporu prevádzky v multi inštančnom (klastrovom) režime. V navrhovanom modele rozmiestnenia využívame 3 nezávislé stroje, pričom na každom z nich beží inštancia databázy Elasticsearch, vďaka čomu sú dáta replikované, čo okrem zvýšenej spoľahlivosti prináša v správnej konfigurácií a využití aj zvýšenú rýchlosť.

Na jednom z databázových serverov je umiestnené aj rýchle úložisku typu kľúč/hodnota Redis. Redis je v našom systéme využívany rôznymi službami a komponentami, na ukladanie denormalizovaných dát vo forme množín, máp alebo polí a hodnôt. Prichádza do úvahy aj nasadenie priamo na aplikačné stroje, formou klastra, čím redukujeme sieťovú latenciu.

Apache Spark klaster

Dalším izolovaným prvkom systému je klaster určený primárne na výpočet modelov strojového učenia a generovanie samotných odporúčaní pomocou naučených modelov. Navrhujeme využitie 3 serverov, na ktoré sa nasadí nástroj Apache Spark. Jeden zo serverov bude plniť úlohu “Riaditeľa” (angl. Driver / Master), zvyšné dva stroje slúžia ako “Vykonávatelia” (angl. Executor). Použité role nepriamo reflektujú kroky Map a Reduce v algoritme MapReduce. Tento klaster je možné riadiť pomocou viacerých nástrojov, populárnou voľbou je Apache Mesos[49], ktorý rozširuje existujúce monitorovacie rozhranie v Apache Spark a poskytuje vysokoúrovňové rozhrania napr. pre automatické manažovanie inštancií.

Pri výbere vhodných serverových inštancií a konfigurácii[50] je rozumné zamerať sa na konfigurácie poskytujúce rýchle sieťove rozhrania (10 Gbit/s) za účelom zníženie latencie pri komunikácii v sieti. Pre korektné fungovanie tolerancií chýb a perzistenciu medzivýsledkov je nutné poskytnúť aj perzistné úložiská na HDD/SSD diskoch. Nároky na CPU nie sú definované, odporúča sa využitie aspoň 8 jadier na jednotlivých inštanciách, ktoré budú po načítaní dát do pamäte limitujúcim faktorom. Ideálnou veľkosťou pamäte je objem, ktorý

[49] http://mesos.apache.org/
[50] http://spark.apache.org/docs/latest/hardware-provisioning.html

obsiahne všetky dáta využívané pri výpočtoch do pamäte, aj s potenciálnou rezervou pre obsahovo náročnejšie transformácie dát. V prípade nedostaktu pamäte použije Apache Spark perzistentné diskové úložiská, ktoré ale rapídne redukujú výkonosť celého klastra.

A.7 Implementácia

Prototyp metódy sme implementovali v skriptovacom Jazyku Python 3. Python je populárny v doméne dátovej analytiky, strojového učenia pre širokú ponuku existujúcich nástrojov a knižníc. Prototyp pozostáva z viacerých komponentov, pričom hlavným z nich je webové API komunikujúce cez protokol JSON, postavené na rámci Falcon. Voči štandardným zástupcom webových rámcov pre Python akými sú napr. Django alebo Flask poskytuje v metrikách priemernej latencie alebo množstva požiadaviek obslúžených za čas 5-8x vyšší výkon[51]. Ako aplikačný server sme zvolili uWSGI, ktorý svojím minimalizmom a rýchlosťou vyhovoval naším požiadavkám. Pri výbere dátového úložiska bola pre náš kľúčovým faktorom škálovateľnosť, a tak sme siahli po riešeniach ktoré majú natívnu podporu prevádzky v klastroch. Ako úložisko pre dáta statického charakteru (impresie, požiadavky pre odporúčanie, novinové články) sme zvolil NoSQL databázu Elasticsearch, ktorá poskytuje pokročilé rozhranie pre spracovanie textu a vyhľadávanie. Efektívnu tvobru agregácií a štatistík vuyžívame pri prepočítavaní popularity a aktuálnosti novinových článkov. Tieto výsledky, spolu aj s predpočítanými odporúčaniami ukladáme do NoSQL databázy Redis, ktorá predstavuje zástupcu z rodiny databáz typu kľúč hodnota. Redis umožňuje aj ukladanie roližných dátových štruktúr ako sú množiny, mapy alebo zoradené množiny, ktoré nám uľahčujú manipuláciu s dátami.

Druhým komponentom v našom prototype škálovateľného odporúčania je nástroj Apache Spark. Ako sme už uviedli v kapitole analýzy, poskytuje široké možnosti spracovania dát aj v reálnom alebo takmer reálnom čase, implementované algoritmy strojového učenia a vysokoúrovňové API pre prácu s dátovými abstrakciami aj pre Python. V prototype pracujeme s načítavaním dát z vymenonvaných dátových úložísk. Experimentovali sme aj so spracovaním dát (zhlukovanie na základe kontextu) dávkovo, v takmer reálnom čase, ale nutná sieťová réžia neumožňovala využitie spracovania dát týmto spôsobom priamo na platforme ORP. Na asynchrónne spracovanie dát využívame knižnicu rq[52], ktorá ako úložisko pre frontu správ a úloh používa databázu Redis. Nižšie uvádzame niektoré vybrané časti implementácie vo forme zdrojových kódov.

```
class GlobalPopularityRecencyStrategy(RecommenderStrategy):
    POP_WEIGHT = 3
    RECENCY_WEIGHT = 1

    @classmethod
    def recommend_to_user(cls, user_id, user_visits, time_interval='4h'):
        pop_recommendations = PopularityRecommender.get_most_popular_articles_global(time_interval)
        recency_recommendations = RecencyRecommender.get_most_recent_articles_global()

        recommendations = ProductAggregator.merge_recommendations(pop_recommendations, recency_recommendations,
                                                                  weights=[cls.POP_WEIGHT, cls.RECENCY_WEIGHT])
        final_recommendations = sorted([r for r, score in recommendations if r not in user_visits])
```

Ukážka zdroj. kódu 1 Odporúčacia stratégia globálne populárnych a aktuálnych článkov

[51] http://falconframework.org/#Metrics
[52] http://python-rq.org/

```
class ContextEncoder:
    CURRENT_IDX_KEY = 'context_encoder_current_idx'
    DIM_LIMIT = 300

    @classmethod
    def encode_context_to_sparse_vec(cls, context):
        vector_dict = cls.encode_context(context, {})
        return SparseVector(len(vector_dict.keys()), vector_dict)

    @classmethod
    def encode_context_to_dense_vec(cls, context):
        vec_data = numpy.zeros(cls.DIM_LIMIT)  # vector of 0s
        vector = cls.encode_context(context, vec_data)
        return DenseVector(vector)

    @classmethod
    def encode_context(cls, context, vec_data):
        for key, value in context.items():

            # figure out our property:value pairs
            if ':' in key:
                splits = key.split(':')
                prop = splits[0]
                value = splits[1]
            else:
                prop = key
            if isinstance(value, list):
                if len(value) > 0:
                    value = value[0]
                else:
                    continue

            # skip if not included in vector attributes for clustering
            if prop in Context.MAPPINGS.keys():
                prop_name = Context.MAPPINGS[prop]
            else:
                continue

            # get our vector idx of property:value pairs
            if prop_name in Context.CLUSTERING_PROPERTIES:
                property_idx = redis.get(cls.build_context_encoder_key(prop_name, value))
                if property_idx:
                    # cache hit
                    pass
                else:
                    # cache miss
                    property_idx = cls.store_prop_value_pair_to_redis(prop_name, value)
                vec_data[int(property_idx)] = 1
        return vec_data
```

Ukážka zdroj. kódu 2 Implementácia enkódovania "jeden z" (perzistencia v databáze Redis)

```
    @staticmethod  # visit, user attributes encoding for collaborative filtering (long to int conversion)
    def encode_attribute(attr, val):
        key = 'encoded:' + str(attr) + ':' + str(val) + ''
        existing_attr_id = redis.get(key)
        if existing_attr_id:
            return existing_attr_id.decode('utf-8')
        else:
            new_val = redis.incr('encoded_attr_id_idx:' + str(attr))
            r = redis.pipeline()
            r.set(key, new_val)
            r.set('decoded:' + str(attr) + ':' + str(new_val), val)
            r.execute()
            return new_val
```

Ukážka zdroj. kódu 3 Implementácia enkódovania identifikátorov z typu Long na Integer

```
@classmethod
def most_popular_per_attribute_n(cls, attribute, origin_timestamp, count=50):
    body = {
        "size": 0,
        "query": {
            "range": {
                "timestamp": {
                    "gte": origin_timestamp,
                    "format": "epoch_second||dateOptionalTime"
                }
            }
        },
        "aggs": {
            attribute + "_agg_count": {
                "terms": {

                    "field": attribute
                },
                "aggs": {
                    "visits_count": {
                        "terms": {
                            "field": "item_id",
                            "size": count
                        }

                    }

                }
            }
        }
    }
    res = es.search(index='impressions', body=body)
    return cls.build_aggs_dict(res, attribute)

@classmethod
def build_aggs_dict(cls, res, attribute=None):
    res_dict = {}
    for publisher in res['aggregations'][attribute + "_agg_count"]['buckets']:
        key = publisher['key']
        res_dict[key] = {}
        for doc in publisher['visits_count']['buckets']:
            res_dict[key][doc['key']] = doc['doc_count']

    return res_dict
```

Ukážka zdroj. kódu 4 Databázový dopyt pre získanie najpopoulárnejších článkov poďla atribútu

Príloha B. Inštalačná príručka

Opisujeme postup inštalácie nami implementovaného odporúčacieho systému ScaRec. Pri inštalácii na zariadení predpokladáme nainštalovaný operačný systém Ubuntu Linux vo verzii 16.04. Odporúčací systém je možné nainštalovať a prevádzkovať aj na iných platformách (Windows, OS X) alebo distribúciach (Centos, Debian), pričom postupnosť krokov bude podobná, avšak použité príkazy alebo špecifické nastavenia uvádzame iba pre linuxovú distribúciu Ubuntu. Opísaná inštalácia predstavuje postup inštalácie vývojárskeho prostredia. V prípade produkčného nasadenia je vhodná a nutná detailnejšia konfigurácia všetkých komponentov, a to najmä z bezpečnostných ale aj výkonostných dôvodov.

B.1 Požiadavky pre inštaláciu odporúčacieho systému

Pred samotnou inštaláciou požadujeme od konfigurácie systému nasledujúce požiadavky:

- Ako operačný systém je nainštalovaná distribúcia Ubuntu vo verzii 16.x v prednastavenej konfigurácii,
- Porty 5100, 6379, 9200 sú voľné a nepoužívané,
- Používateľ má administrátorské oprávnenia (tkzv. root prístup),
- V premennej `PROJECT_PATH` je nastavená cesta ku koreňovému adresáru pracovného prostredia,
- V priečinku `$PROJECT_PATH` sa nachádza obsah priečinka ml_models z elektronického média
- V priečinku `$PROJECT_PATH/scarec` sa nachádzajú rozbalené zdrojové súbory z priloženého média.

B.2 Aktualizácia apt-get balíčkov a inštalácia prerekvizít

```
$  sudo apt-get update

$  sudo apt-get install build-essential

$  sudo apt-get install tcl8.5

$  sudo apt-get install git
```

B.3 Inštalácia Java 8 (OpenJDK alebo Oracle JDK)

```
$  sudo apt-get install default-jdk
```

B.4 Inštalácia Scala 2

Najprv stiahneme inštalačné súbory a rozbalíme ich do predvoleného adresára.

```
$  cd $PROJECT_PATH

$  wget http://www.scala-lang.org/files/archive/scala-2.10.4.tgz

$  sudo mkdir /usr/local/src/scala
```

```
$ sudo tar xvf scala-2.10.4.tgz -C /usr/local/src/scala/
```

Následne nakonfigurujeme Scalu v prostredí Ubuntu. Otvoríme textový súbor .bashrc a pridáme nastavenia dvoch premenných prostredia.

```
$ vi .bashrc

$ export SCALA_HOME=/usr/local/src/scala/scala-2.10.4

$ export PATH=$SCALA_HOME/bin:$PATH
```

Aktivujeme nové nastavenia prostredia.

```
$ . .bashrc
```

B.5 Inštalácia Redis

```
$ sudo apt-get install redis-server
```

Redis 3.x je nainštalovaný a spustený ako služba a automaticky sa spúšťa po zapnutí-

B.6 Inštalácia a spustenie Elasticsearch

```
$ wget -qO - https://packages.elastic.co/GPG-KEY-elasticsearch | sudo
  apt-key add -

$ echo "deb http://packages.elastic.co/elasticsearch/2.x/debian stable
  main" | sudo tee -a /etc/apt/sources.list.d/elasticsearch-2.x.list

$ sudo apt-get update && sudo apt-get install Elasticsearch

$ sudo update-rc.d elasticsearch defaults 95 10

$ sudo service elasticsearch start
```

Elasticsearch 2.x je nainštalovaný a spustený ako služba a automaticky sa spúšťa po zapnutí.

B.7 Inštalácia a konfigurácia Python 3

```
$ sudo apt-get install python3-pip

$ sudo pip3 install virtualenv

$ mkdir $HOME/.virtualenvs

$ virtualenv $HOME/.virtualenvs/pywork3
```

B.8 Inštalácia Apache Spark

```
$ cd $PROJECT_PATH

$ wget http://www.apache.org/dyn/closer.lua/spark/spark-1.6.1/spark-
  1.6.1-bin-hadoop2.6.tgz
```

```
$  sudo mkdir $PROJECT_PATH/spark

$  sudo tar xvf spark-1.6.1-bin-hadoop2.6.tgz -C $PROJECT_PATH/spark
```

B.9 Konfigurácia prostredia

Nasledujúce riadky pridáme do súboru `.bashrc` a súbor uložíme.

Nastavíme umiestnenie modelov strojového učenia.

```
$  export KMEANS_MODEL_PATH="$PROJECT_PATH/kmeans_clustering_model"

$  export RF_MODEL_PATH_ROOT="$PROJECT_PATH/rf_models/"

$  export ALS_MODEL_PATH="$PROJECT_PATH/als_model"
```

Konfigurujeme premenné pre PySpark.

```
$  export LC_ALL="en_US.UTF-8"

$  export LC_CTYPE="en_US.UTF-8"

$  export PYTHONPATH="$PROJECT_PATH/spark/python"

$  export          PYTHONPATH="$PROJECT_PATH/python/lib/py4j-0.9-src.zip:
   $PYTHONPATH"

$  export PYSPARK_PATH="$ROJECT_PATH/spark/python"

$  export SPARK_HOME="$ROJECT_PATH/spark"

$  export PYSPARK_PYTHON="$HOME/.virtualenvs/pywork3/bin/python"
```

Globálne aktivovanie virtuálneho prostredia Python 3.

```
$  source $HOME/virtualenvs/pywork3/bin/activate
```

Novú konfiguráciu prostredia aktivujeme.

```
$  . .bashrc
```

B.10 Inštalácia Python3 závislostí

```
$  cd $PROJECT_PATH/scarec

$  pip install -r requirements.txt
```

B.11 Spustenie Odprúčacieho systému ScaRec

```
$  cd $PROJECT_PATH/scarec

$  python3 scripts/create_es_indices.py

$  honcho start
```

Príloha C. Vlastnosti a evaluácia odporúčacích systémov

Pri každodennom používaní odporúčacích systémov v komerčnom prostredí sa ukazuje, že exaktné metriky na meranie presností odporúčaní nekorešpondujú priamo s uspokojením potrieb jednotlivých používateľov systémov. V kapitole sa preto zameriame aj na ďalšie vlastnosti a atribúty odporúčacích systémov, ktoré zohrávajú v subjektívnom vnímaní väčšiu úlohu ako matematické metriky.

C.1 Ustálené metriky pri evaluácií odporúčacích systémov

Väčšia časť doterajšieho bádania v oblasti odporúčacích systémov sa zameriavala primárne na zlepšovanie presnosti odporúčaní (angl. accuracy alebo precision) (Sarwar et al. 2001, 2000) a rozširovaní pokrytia (angl. coverage) – množstva objektov, ktoré systém dokáže odporučiť (Adomavicius and Tuzhilin 2005). Metriky definujú ako úspešne dokážu systémy predikovať záujem používateľa k odporučenému objektu. Táto úzka špecializácia podľa (McNee et al. 2006) mohla mať negatívny vplyv na vývoj odporúčacích systémov.

Syntetické off-line overenia resp. experimenty odporúčacích systémov pomocou metrík sú založené na skúmaní hodnotení vzorky objektov vykonanými používateľmi, ktorí reprezentujú len časť celkového objemu dát. Používatelia najčastejšie hodnotia tie objekty, ku ktorým majú pozitívny vzťah. Výsledky týchto evaluácií tým pádom ukazujú iba presnosť hodnotení objektov systémom, ktoré sa používatelia rozhodli hodnotiť, pričom predikované hodnotenia objektov, ktoré používatelia nehodnotili, prípadne náhodné objekty z celej dátovej množiny nie sú pri takomto vyhodnocovaní brané do úvahy. Tento spôsob hodnotenia nekorešponduje s očakávaným využitím odporúčacích systémov pri praktickom bežnom alebo komerčnom nasadení (Adomavicius and Tuzhilin 2005; Yang et al. 2005). Pri ňom sa očakáva schopnosť hodnotenia objektov, s ktorými používateľ ešte nereagoval a do úvahy je nutné brať široké spektrum atribútov, ktoré reprezentujú prínos odporúčaní pre používateľa.

C.2 Evaluácia odporúčacieho systému z pohľadu používateľa

Charakterizácia a evaluácia kvalít odporúčacích systémov je dôležitým krokom pri ich tvorbe, najmä pre ich vysokú popularitu a rozšíriteľnosť. Dnes sa prakticky na každom väčšom webovom portály alebo e-shope objavujú sekcie s obsahom, ktorý je používateľovi odporúčaný na základe jeho predchádzajúceho správania. Dôvodom je ich široká uplatniteľnosť a vysoká úspešnosť. Štúdia spoločnosti ChoiceStream z roku 2008[53] uvádza, že až 58% frekventovane nakupujúcich ľudí pravdepodobnejšie zaujme reklama, ktorá je personalizovaná. Takmer polovica respondentov ale uviedla, že sa stretla s odporúčaniami, ktoré boli neuspokojivé a nevhodné. Aj preto považujeme podstatné pozerať a hodnotiť odporúčacie systémy nie len zo štatistického a matematického hľadiska, ale do úvahy je nutné brať aj názory používateľov.

Medzi hľadiská, ktorým sa venuje akademický výskum patrí identifikácia faktorov, ktoré ovplyvňujú vnímanie odporúčacích systémov (Swearingen and Sinha 2002), motivácia a

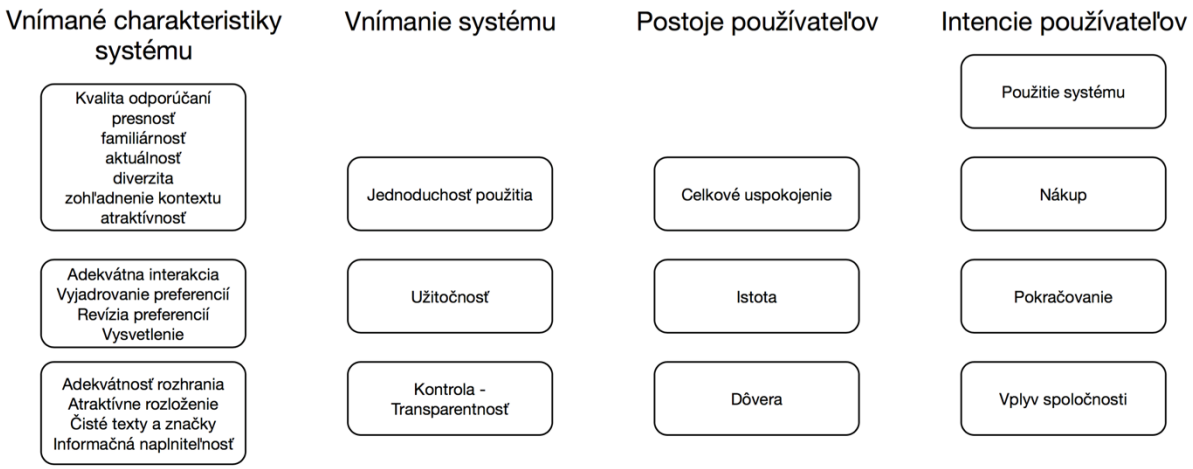
[53] 2008 ChoiceStream Personalization Survey,http://choicestream.com/2013_Staging/wp-content/uploads/2013/10/ChoiceStream_2008_Personalization_survey.pdf

podpora používateľov v hodnotení objektov v odporúčacích systémoch (Beenen et al. 2004; McNee et al. 2003), generovanie originálnych a diverzifikovaných odporúčaní, ktoré vedú k vyššej satisfakcii používateľov (Vargas 2014; McCarthy et al. 2005), poskytovanie vysvetlení a dôvodov pre konkrétne odporúčanie (Tintarev and Masthoff 2007), budovanie dôvery ku odporúčacím systémom (Chen and Pu 2009) a prvkom, alebo dizajnové a interakčné odporúčania pre rozhrania odporúčacích systémov (Norcio 2010).

Spomínané aspekty sa snažili zachytiť v modeloch na evaluáciu odporúčacích systémov ako napr. TAM (Davis 1989) alebo SUMI (Kirakowski and Corbett 1993). Jedným z týchto modelov, ktorý na ne nadväzuje, je modelový rámec ResQue (Pu and Chen 2010) (angl. Recommender systems' Quality of User experience), ktorý kombinuje kritéria hodnotení odporúčacích systémov z pohľadu používateľa. Pozostáva zo 60 otázok a analyzuje 13 aspektov, ktoré je možné rozčleniť do štyroch kategórií, a to:

- Vnímanie charakteristík systému používateľmi
- Subjektívny postoj používateľov k odporúčaciemu systému
- Vnímanie systému používateľmi
- Intencie používateľov.

Jednotlivé aspekty a prislúchajúce kategórie sú znázornené na *Obr. C.1-1.*



Obr. C.2-1 Skladba modelového rámca ResQue (Pu and Chen 2010)

Jeho cieľom je ohodnotiť vnímanú kvalitu odporúčacích systémov, medzi ktoré patrí napr. použiteľnosť, satisfakcia používateľa, rozhranie a interakcia so systémom a predikovať tak predpokladané správanie a akcie používateľa. Medzi akcie patrí výber alebo kúpa odporúčaného objektu, návrat používateľa do daného systému alebo šírenie pozitívnych referencií na systém. V ďalších častiach kapitoly sa pozrieme detailnejšie na vybrané identifikované aspekty prítomné v ResQue modelovom rámci (Pu and Chen 2010; Tintarev et al. 2011), ktorý je primárne zameraný na evaluáciu odporúčacích systémov v sfére elektronického obchodu, ale podstatná väčšina vlastností je aplikovateľná aj v celej sfére odporúčacích systémov.

C.2.1 Kvalita odporúčaní z pohľadu používateľa

Hlavnou funkciou odporúčacích systémov je generovanie odporúčaní, preto je vhodné zamerať sa na ich kvalitu a vlastnosti, ako ich vníma používateľ. Vlastnosti odporúčaní vytvárajú celkovú užitočnosť odporúčaní a ich informačnú kvalitu.

Vnímaná presnosť predstavuje stupeň, do ktorého používatelia vnímajú odporúčania pozitívne, teda odporúčania spĺňajú ich preferencie, záujmy a očakávania. Túto vlastnosť môžeme považovať pri väčšine odporúčacích systémov za najdôležitejšiu, pretože priamo určuje, ako úspešne odporúčací systém predikoval odporúčania. Ak je tento stupeň vysoký, tak rovnako môžeme očakávať aj vysokú mieru pri štatistickej evaluácií algoritmov použitých v systéme, čo Pu a Chen úspešne identifikovali aj v ich predchádzajúcej práci (Pu et al. 2008), kde porovnávali začlenenie kritiky príkladov pri odporúčaní a vytvorili modelový rámec použiteľný primárne vo sfére elektronického obchodu.

Novota a objavovanie nových prekvapivých objektov sú podľa Herlocker et al. dva odlišujúce sa aspekty pozorovateľné pri vnímaní odporúčaní používateľmi (Herlocker et al. 2004). Ako príklad uvádzajú odporúčací systém pre filmy, ktorý generuje odporúčania na základe používateľom obľúbeného režiséra. Ak systém odporučí používateľovi nový film od jeho obľúbeného režiséra, ktorý používateľ nepoznal, bude toto odporúčanie nové, ale nie prekvapivé. Pu et al. tieto dva aspekty vo svojom modeli odporúčajú spojiť, pretože sú úzko prepojené a môžu viesť k pomýleniu používateľov resp. respondentov. Oba aspekty popisujú úroveň, do ktorej používateľ získava odporúčania o pre neho nových, nepoznaných a zaujímavých objektoch. Základným konceptom novoty resp. originálnosti je určitá edukácia používateľov (Pu et al. 2009) vo forme oboznámení sa s doménou, prípadne priame poskytnutie pomocnej navigácie v danej doméne, kedy používateľom odporúčania pomáhajú objavovať nové objekty (Pu et al. 2009; Vargas 2014). Prekvapivosť resp. neočakávanosť je možné predikovať až potom, ako poznáme preferencie používateľa (McNee et al. 2006). Skúmal ich aj Adamopoulos vytvorením algoritmu na základe ekonomickej teórie užitočnosti (Adamopoulos and Tuzhilin 2013). Podarilo sa mu zadefinovať metriky, ktoré merajú neočakávanosť generovaných odporúčaní. Na základe týchto metrík a overení na datasetoch, sa ukázala jeho metóda (Adamopoulos and Tuzhilin 2013) vysoko úspešnejšia v generovaní neočakávaných odporúčaní v porovnaní s inými štandardnými metódami ako napr. *K-NN* (angl. K – nearest neighbors), zatiaľ čo v štandardizovaných metrikách ako *RMSE* (angl. Root mean square error) alebo F-metrike dosahovala minimálne tak dobré výsledky, ako ostatné zaužívané metódy.

Familiárnosť hovorí o tom, či používateľ má predchádzajúcu skúsenosť (prečítanie novinového článku, pozretie filmu) s odporučeným objektom. Pri výskyte familiárnych objektov medzi odporúčaniami zvyšujeme dôveru používateľov v systém, a ukázalo sa, že tieto prvky majú používatelia radi (Swearingen and Sinha 2002). Nebezpečný je ale aj prílišný výskyt takýchto odporúčaní, ktoré používateľa frustrujú a vedú k opusteniu odporúčacieho systému. Veľmi dôležité je preto správne nastaviť hranicu familiárnosti a už spomínanej originálnosti odporúčaní.

Atraktívnosť je z popisovaných vlastností pravdepodobne najabstraktnejším pojmom, ktorý je najnáročnejšie aj odmerať. Predstavuje emocionálnu stimuláciu používateľa, podporuje stimuláciu používateľovej predstavivosti a evokuje pozitívny záujem alebo emóciu k odporúčaniu.

Satisfakcia určuje či mal používateľ z odporúčania, ktoré si vybral aj osoh a priniesol pre neho určité benefity. Satisfakcia je prvok, ktorý najviac súvisí s opakovaným používaním systému resp. navráteniu používateľov. Nie vždy je možné získať priamu okamžitú spätnú väzbu od používateľa, v niektorých systémoch používatelia hodnotia alebo vypĺňajú dotazníky aj s viactýždňovým oneskorením.

Diverzita predstavuje dôležitý aspekt odporúčaní, kedy sa pozeráme na odporúčanie generované používateľovi ako na celkovú množinu odporúčaní. Príliš nízka diverzita odporúčaní môže viesť k opusteniu systému používateľmi. Získavame tak ďalší aspekt hodnotenia, pre ktorý je vhodné hodnotiť celkové vygenerované zoznamy odporúčaní, viac ako jednotlivé odporúčania (Ziegler et al. 2005). Ziegler et al. uvádzajú vo svojej práci metódu, ktorá priamo pri určovaní odporúčaní a výpočte podobnosti jednotlivých objektov berie do úvahy aj diverzitu odporúčaní formou diverzifikácie odporúčaných tém. Pridaním aspektu diverzity pri generovaní odporúčaní sa zaoberali McGinty a Smyth (McGinty and Smyth 2003), kde odporúčali buď podobné objekty alebo naopak diverzifikované objekty na základe používateľov interakcie v minulosti a momentálnych potrieb. Vargas vniesol do sféry personalizovania prvky z vyhľadávania informácií, kde je diverzita jedným z hlavných kritérií systémov (Vargas 2014; Vargas and Castells 2013). Ich prístup spočíva v identifikácii diverzity medzi používateľmi a generovaní čiastočných odporúčaní, založených na homogénnych podmnožinách používateľských preferencií, ktoré neskôr kombinujú do finálnych odporúčaní. Ich metóda sa ukázala ako výnimočné efektívna a prekonáva zaužívané prístupy diverzifikácie založenej na explicitných aspektoch. Jej problémom je ale nízka škálovateľnosť.

Zohľadnenie kontextu je obohacujúcim prvkom, pomocou ktorého vieme zvýšiť kvalitu odporúčaní, čomu zodpovedá vyššia náročnosť na systémové zdroje. Môže sa jednať napr. o zohľadnenie počasia a času počas dňa, na základe čoho vieme predpokladať používateľovu náladu a odporučiť mu filmy alebo hudbu, ktorá bude viac vhodná.

C.2.2 Vnímanie systému používateľmi

Vhodnosť rozhraní a vhodnosť interakcie systémov je frekventovane skúmaná oblasť (Norcio 2010; Tintarev and Masthoff 2007; McNee et al. 2006), ktorá sa v súčasnosti zameriava na optimalizáciu prezentácie odporúčaní (akú formu reprezentácie objektu zvoliť, ako odporúčania rozmiestniť na stránke), výber vhodných techník na získavanie spätnej väzby od používateľa alebo prezentáciu a objasňovanie dôvodov, ktoré stoja za jednotlivými odporúčaniami.

Použiteľnosť systému môžeme rozdeliť hlbšie na naučiteľnosť, ktorá definuje náročnosť interakcie so systémom pre nových používateľov, iniciálne úsilie pred generovaním prvých odporúčaní, aby sme vyriešili problém studeného štartu (McNee et al. 2003; Saveski and Mantrach 2014), ktoré sa odporúča minimalizovať. Je možné ho realizovať napr. vypĺňaním dotazníkov alebo hodnotením vzorky dát. Ak sa jedná už o pravidelného používateľa hovoríme o jednoduchosti používania systému, ktoré vyjadruje objem kognitívnej námahy nutnej na získanie odporúčaní od systému (Chen and Pu 2009).

Užitočnosť hovorí o tom, ako je systém s odporúčaniami výhodnejší pre používateľa, v porovnaní s predchádzajúcou interakciou so systémom bez odporúčaní. Do úvahy sa odporúča brať aj schopnosť pomoci pri rozhodovaní, pretože odporúčacie systémy sú

jedným z prostriedkov na redukciu informačného priestoru, ktorý je inak príliš objemný pre používateľa na to, aby dokázal urobiť časovo efektívne a zároveň vhodné rozhodnutie.

C.2.3 Postoj používateľov

Pu a Chen vo svojej práci definovali postoj ako celkové používateľove cítenie k odporúčaciemu systému, ktoré sa odvíja od interakcie so systémom (Pu and Chen 2010). Pri postojoch používateľa sa predpokladá dlhodobejšie trvanie ako pri domnienkach. Identifikovali tri hlavné prvky, ktoré formujú postoj používateľa, z ktorých najviac skúmaný a dominantný je dôvera v odporúčací systém. Úroveň dôvery je daná aj dôverou v informačné systémy alebo online portály (Broutsou 2012), čo odporúčacie systémy, samé o sebe nemajú príležitosť priamo ovplyvňovať. Naopak, priamo využiteľným prostriedkom pri odporúčacích systémoch je kvalita jednotlivých generovaných odporúčaní a poskytovanie vysvetlení a objasnení dôvodov odporučenia daných produktov (Pu and Chen 2006; Herlocker et al. 2000). Mieru dôvery odporúčajú pozorovať s ohľadom na čas, pretože je postavená na základe dlhšie trvajúceho vzťahu medzi odporúčacím systémom a používateľom, ktorý ho používa opakovane.

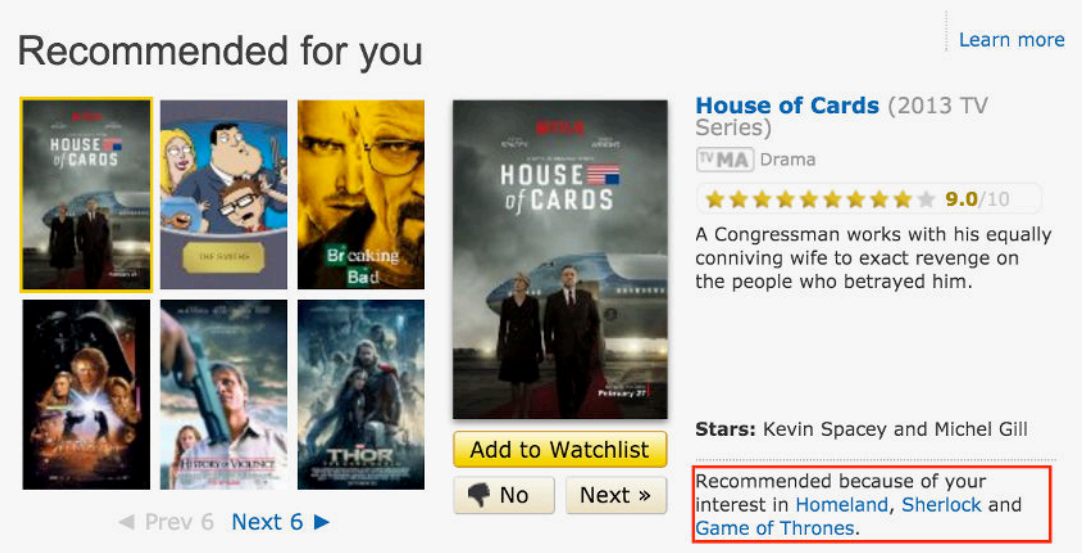
C.2.4 Zámer používateľov

V časti zámerov používateľov pozorujeme, či je systém schopný ovplyvniť používateľovo rozhodovanie používať ho a v konečnom dôsledku vybrať si resp. zakúpiť jeden z odporúčaných objektov. Jedným z hlavných cieľov odporúčacieho systému je návrat používateľov a jeho kontinuálne používanie. Takzvaná lojalita používateľov je najviac ovplyvňovaná predchádzajúcou uspokojivou skúsenosťou so systémom (vo forme odporučenia vhodného objektu).

C.3 Prezentácia a vysvetľovanie odporúčaní

Objasnenie odporúčaní je možné realizovať na základe troch hľadísk, medzi ktoré patria objasnenia založené na podobnosti objektov, (*Obr. C.1-2)* v službe IMDB (páčil sa vám film A, preto by ste si mali pozrieť film B), založené na podobnosti používateľov (vášmu priateľovi X sa páčil film B, pozrite si ho) a založené na podobnosti vlastností odporúčaného objektu (vo filme A, ktorý sa vám páčil hral herec Y, pozrite si film B, v ktorom hral tiež) (Chen et al. 2013; Xu et al. 2012). Chen prezentoval alternatívny prístup prezentácie objasnení vo forme oblaku slov, ktorý obsahuje aj negatívne a neutrálne slová, ktoré popisujú odporúčaný film. Pri využívaní latentných modelov je problémom určenie a definovanie dôvodu odporúčania (Zhang et al. 2014) , ktoré Zhang et al. vo svojej práci eliminovali využívaním sentimentálnej analýzy používateľských recenzií na úrovni fráz. Spájaním týchto používateľských názorov s extrahovanými vlastnosťami objektov boli schopní vytvárať hybridné faktorizačné matice, pričom boli schopní generovať objasnenia pre pozitívne aj negatívne odporúčania v systéme.

Odporúčaniami pre zobrazovanie a prezentáciu objasnení jednotlivých odporúčaní (Pu and Chen 2006, 2007) vieme používateľovi poskytnúť dve podstatné informácie, a to na základe čoho mu systém vygeneroval dané odporúčanie a aké sú jeho ďalšie možnosti, ak nedokázal vygenerovať vhodné odporúčania (Reilly et al. 2007). Pri objasňovaní odporúčaní je možné zobrazovať aj mieru istoty odporúčania vygenerovaného systémom (Reilly et al. 2007), čo umožňuje nastaviť reálnejšie očakávania pre používateľa a zvýšiť tým dôveru v odporúčací systém.



Obr. C.3-1 Zobrazenie dôvodov pre odporúčanie v službe IMDB

Zoznam použitej literatúry

ADAMOPOULOS, Panagiotis & Alexander TUZHILIN, 2013. On Unexpectedness in Recommender Systems : Or How to Better Expect the Unexpected. vol. 5, no. 4, pp. 1–50.

ADOMAVICIUS, Gediminas & Alexander TUZHILIN, 2005. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*. vol. 17, no. 6, pp. 734–749.

BEENEN, Gerard, Kimberly LING, Xiaoqing WANG, Klarissa CHANG, Dan FRANKOWSKI, Paul RESNICK & Robert E. KRAUT, 2004. Using social psychology to motivate contributions to online communities. *Computer Supported Cooperative Work*. pp. 212–221.

BROUTSOU, Andromachi, 2012. Online Trust: The Influence of Perceived Company's Reputation on Consumers' Trust and the Effects of Trust on Intention for Online Transactions. *Journal of Service Science and Management*. vol. 05, no. December, pp. 365–372.

CHEN, Li & Pearl PU, 2009. Interaction design guidelines on critiquing-based recommender systems. *User Modeling and User-Adapted Interaction*. vol. 19, no. 3, pp. 167–206.

CHEN, Wei, Mong Li LEE & Forrest GUMP, 2013. Tagcloud-based Explanation with Feedback for Recommender Systems. *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval - SIGIR '13*. pp. 945–948.

DAVIS, F., 1989. Perceived Usefulness , Perceived Ease Of Use , And User Acceptance. *MIS Quarterly*. vol. 13, pp. 319–339.

HERLOCKER, Jonathan L, Joseph a KONSTAN & John RIEDL, 2000. Explaining collaborative filtering recommendations. *Proceedings of the 2000 ACM conference on Computer supported cooperative work*. pp. 241–250.

HERLOCKER, Jonathan L., Joseph a. KONSTAN, Loren G. TERVEEN & John T. RIEDL, 2004. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems*. 1.1., vol. 22, no. 1, pp. 5–53.

KIRAKOWSKI, J & M CORBETT, 1993. SUMI: The software usability measurement inventory. *British Journal of Educational Technology*. pp. 10–12.

MCCARTHY, Kevin, James REILLY, Lorraine MCGINTY & Barry SMYTH, 2005. Experiments in dynamic critiquing. *Proceedings of the 10th international conference on Intelligent user interfaces - IUI '05*. p. 175.

MCGINTY, Lorraine & Barry SMYTH, 2003. On the Role of Diversity in Conversational Recommender Systems. *Case-Based Reasoning Research and Development*. vol. 2689, p. 1065.

MCNEE, S M, J RIEDL & J a KONSTAN, 2006. Being accurate is not enough: how accuracy metrics have hurt recommender systems. *CHI'06 extended abstracts on Human factors in computing systems*. p. 1101.

MCNEE, Sean, Shyong LAM, Joseph KONSTAN & John RIEDL, 2003. Interfaces for Eliciting New User Preferences in Recommender Systems. *User Modeling 2003*. vol. 2702, p. 148.

NORCIO, Anthony F, 2010. Design Guidelines for Effective E-Commerce Recommender System Interfaces: A Usability and User Preferences Perspective. *Behaviour & Information Technology*. vol. 29, pp. 57–83.

PU, Pearl & Li CHEN, 2006. Trust Building with Explanation Interfaces. *Proceedings of the 11th international conference on Intelligent user interfaces - IUI '06*. pp. 93–100.

PU, Pearl & Li CHEN, 2007. Trust-inspiring explanation interfaces for recommender systems. *Knowledge-Based Systems*. vol. 20, pp. 542–556.

PU, Pearl & Li CHEN, 2010. A user-centric evaluation framework of recommender systems. *CEUR Workshop Proceedings*. vol. 612, pp. 14–21.

PU, Pearl, Li CHEN & Pratyush KUMAR, 2008. Evaluating product search and recommender systems for E-commerce environments. *Electronic Commerce Research*. vol. 8, pp. 1–27.

PU, Pearl, M ZHOU & S CASTAGNOS, 2009. Critiquing recommenders for public taste products. *RecSys '09*. pp. 0–3.

REILLY, James, James REILLY, Jiyong ZHANG, Jiyong ZHANG, Lorraine MCGINTY, Lorraine MCGINTY, Pearl PU, Pearl PU, Barry SMYTH & Barry SMYTH, 2007. A comparison of two compound critiquing systems. *Intelligent User Interfaces*. pp. 317–320.

SARWAR, Badrul, George KARYPIS, Joseph KONSTAN & John RIEDL, 2000. Analysis of recommendation algorithms for e-commerce. *EC '00 Proceedings of the 2nd ACM conference on Electronic commerce*. vol. 5, pp. 158–167.

SARWAR, Badrul, George KARYPIS, Joseph KONSTAN & John RIEDL, 2001. Item-based collaborative filtering recommendation algorithms. *WWW '01 Proceedings of the 10th international conference on World Wide Web*. pp. 285–295.

SAVESKI, Martin & A MANTRACH, 2014. Item cold-start recommendations: learning local collective embeddings. *RecSys '14 Proceedings of the 8th ACM Conference on Recommender systems*. pp. 89–96.

SWEARINGEN, Kirsten & Rashmi SINHA, 2002. Interaction design for recommender systems. *Designing Interactive Systems*. pp. 1–10.

TINTAREV, Nava & Judith MASTHOFF, 2007. A survey of explanations in recommender systems. *Proceedings - International Conference on Data Engineering*. pp. 801–810.

TINTAREV, Nava, Judith MASTHOFF, Francesco RICCI, Lior ROKACH, Bracha SHAPIRA & Paul B KANTOR, 2011. *Recommender Systems Handbook*. Boston, MA: Springer US. ISBN 978-0-387-85819-7.

VARGAS, Saúl, 2014. Novelty and diversity enhancement and evaluation in recommender systems and information retrieval. *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval - SIGIR '14*. pp. 1281–1281.

VARGAS, Saúl & Pablo CASTELLS, 2013. Exploiting the Diversity of User Preferences for

Recommendation. *Proceedings of the 10th Conference on Open Research Areas in Information Retrieval*. pp. 129–136.

XU, Xiaoying, Anindya DATTA & Kaushik DUTTA, 2012. Using Adjective Features from User Reviews to Generate Higher Quality and Explainable Recommendations. *IFIP Advances in Information and Communication Technology*. vol. 389, pp. 18–34.

YANG, Yinghui, Yinghui YANG, Balaji PADMANABHAN & Balaji PADMANABHAN, 2005. Evaluation of online personalization systems: a survey of evaluation schemes and a knowledge-based approach. *Journal of Electronic Commerce Research*. vol. 6, no. 1, pp. 112–122.

ZHANG, Yongfeng, Guokun LAI, Min ZHANG, Yi ZHANG, Yiqun LIU & Shaoping MA, 2014. Explicit Factor Models for Explainable Recommendation based on Phrase-level Sentiment Analysis Categories and Subject Descriptors. *Sigir*. pp. 83–92.

ZIEGLER, Cai-Nicolas C.N., Sean M. S.M. MCNEE, Joseph a. J.a. KONSTAN & Georg LAUSEN, 2005. Improving recommendation lists through topic diversification. *Proceedings of the 14th international conference on World Wide Web WWW 05*. vol. 3, p. 22.

Príloha D. Príspevok prijatý na konferenciu IIT.SRC 2016

Článok prijatý na konferenciu IIT.SRC 2016 do sekcie Software Engineering, organizovanú FIIT STU.

Scalable Personalized Recommender System

Adam Lieskovský*

Slovak University of Technology in Bratislava
Faculty of Informatics and Information Technologies
Ilkovičova 2, 842 16 Bratislava, Slovakia
adamliesko@gmail.com

Abstract. Requirements and priorities for modern recommender systems grow and change hand in hand with the continuous growth of Internet users. In this paper we focus on aspects such as scalability, flexibility and speed of recommender systems, which nowadays play an important role. We propose novel hybrid recommender system, which utilizes user behaviour and context, items' content and items' popularity and recency. We evaluate our approach in the domain of news articles, processing streaming data and computing recommendations online, in real-time. Conducted evaluations showed the benefits of using article trendiness as a strong component in the process of computing recommendations.

10 Introduction and related work

Information overload is nowadays a serious issue on Web. Users don't have the time, neither the resources or skills to manually search and filter for items which they seek. Personalized recommendations are great way to reduce and eliminate this problem. Furthermore, it helps users to discover new items, which they wouldn't be able to find by the means of standard navigation. Other than that, personalized recommendations help with increasing the commercial revenue as well. Therefore, it is a fairly popular topic not only in the academic world, but also between the tech giants like LinkedIn [1], Netflix [2] or Amazon [3], where recommendations play a key role in the user experience.

Another prevailing problem, which appears with the enormous traffic and number of users the popular sites have to serve, is the aspect of scalability. Recently, methods like distributed systems, GPU matrix computations or client-server computation distribution [4] have enabled for real-time processing of streaming data on a large scale.

Majority of established approaches for personalized recommendations fall into the categories of collaborative filtering and content based recommendations. Each of these approaches comes with it's downsides (e.g., cold start problem of a new item or new user, long tail problem), which hybrid recommender systems usually help to eliminate [5]. For long, data processing and computations at large scale have been limited by existing technology or software. With the introduction of map-reduce computation model, now also available through in-memory computation, or the massive growth of NoSQL databases, recommender systems are nowadays able to achieve the required scalability and efficiency needed for the Web scale.

Nowadays, collaborative filtering approaches, for example using a scalable k-NN algorithm [6] or matrix factorization [7, 8], present the state of the art method for recommender systems. Most successful and widely used are matrix factorization processing user-item association matrix [9].

In [9] Shi et al. present a comprehensive survey of the state-of-the-art collaborative filtering and latest challenges in the domain of recommender systems. They identified the incorporation of social recommendations and additional user interaction data (e.g., context), cross domain and group collaborative filtering as the arising key challenges to the future of collaborative filtering.

11 Scalable and flexible recommendation approach

In this paper, we propose a scalable hybrid recommender system, with focus on a fast response time, scalability and flexibility in regard to the system workload and available resources. Abstractly, it can be separated into two subparts, first consisting of a real-time online computation of recommendations, the second of computing offline user models. Our hybrid approach can be classified as a meta-level, mixed or weighted, depending on the form

* Master degree study programme in field: Software Engineering
Supervisor: Dr. Michal Kompan, Institute of Informatics and Software Engineering, Faculty of Informatics and Information Technologies STU in Bratislava

of final recommendation aggregation. In the following sections, we describe respective recommender modules for hybrid recommender system in the domain of news articles, as outlined on the Figure 1.

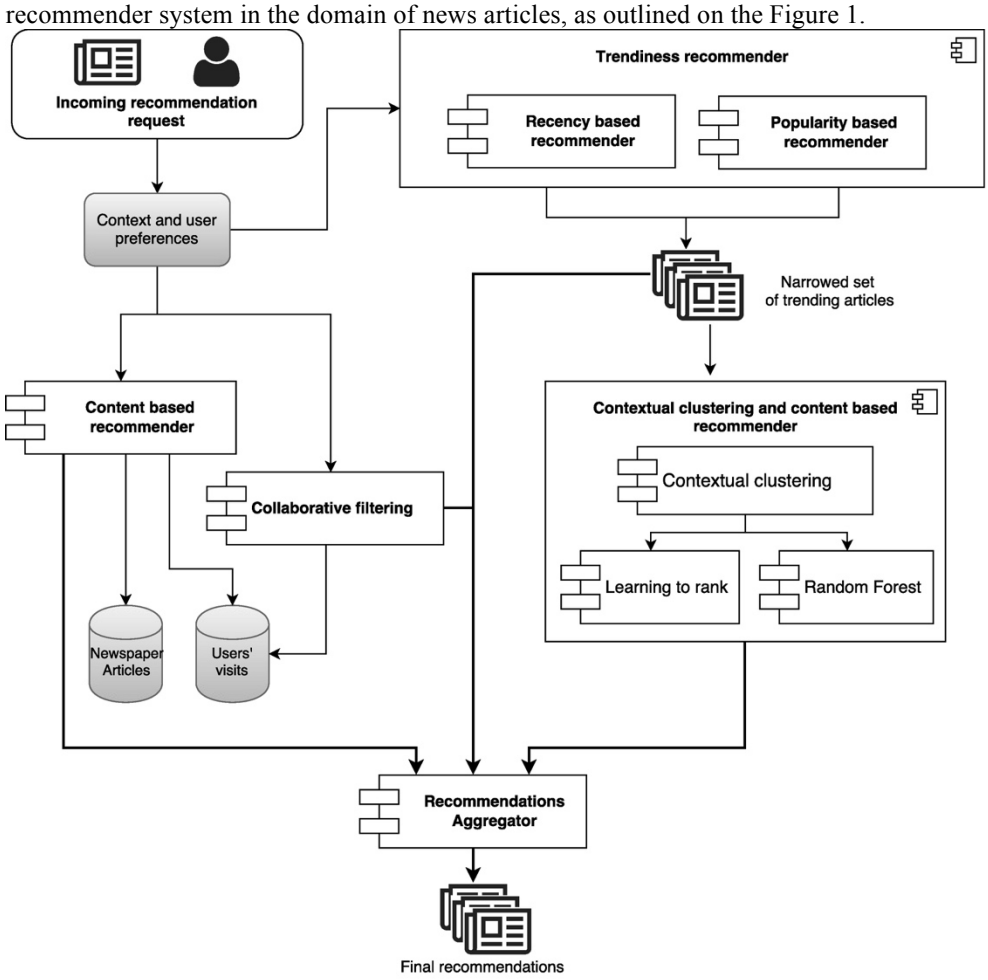


Figure 1. Proposed scalable hybrid recommender approach.

11.1 Content based recommendation

Research in natural language processing (NLP) and its approaches to content similarity estimation using *LDA* (*Latent Dirichlet allocation*) [10] or *LSA* (*Latent semantic analysis*) [11] have recently become fairly popular and successful. If combined with SVD (Singular value decomposition), these algorithms provide a scalable way to cluster items and infer topics based on their content. In our work we opted for a simpler and faster approach, utilizing direct content similarity estimation between item pairs.

On top of static item content comparison, we incorporate sentiment and relevance of extracted keywords or entities into the recommendation process. This information can be extracted from the content of items, e.g., in the domain of news articles we use article title and text as source of keyword extraction. This way, we can identify user's probable attitude towards certain topics (election parties, war etc.), entities (e.g., Steve Jobs, England) or extracted keywords. Given this information, we can narrow the recommendations to the users with items containing similar sentiment or attitude in general towards certain entities or topics, in which the user has previously shown interest.

11.2 Collaborative filtering

Since Netflix prize [8], matrix factorization has become a state-of-the-art method for computing recommendations with collaborative filtering. We chose alternating least squares (*ALS*) algorithm, which can compute latent factors describing users and items from user-item association matrix. ALS algorithm provides us with an option to specify a weight to a user-item interaction, if it turns out that it is desirable to differentiate between user clicking on a certain recommendation or a user naturally visiting an item. More specifically we are using *ALS-WR* approach, as described in [12]. *ALS-WR* uses a normalization parameter lambda, which tackles scalability and sparsity issues in matrix factorization. One of the nice properties of *ALS-WR* is that the accuracy of the method monotonically improves with the number of algorithm iterations.

ALS model is computed periodically in the background with latest subset of user-item interactions narrowed by their occurrence time. Computation of a model is distributable and we can easily set number of iterations based on the current system workload. The periods between model re-computation can be also adjusted according to the workload, which also adds to the flexibility property of our proposed approach.

11.3 Popularity and recency of items

Popularity and recency are two aspects of items, that we combined together to form a so-called trendiness recommender. We model popular and recently created or updated items in certain time intervals. In addition to global counters, we monitor the popularity and recency of articles in the subsets of items, as per their categorization by content. For popularity, we use integer counters of item visits occurrences in the respective time interval. For recency aspect, we propose a more sophisticated approach. Our goal is to be able to assign the same value of recency to an article published 10 minutes ago, as to an article published 1 hour ago (example thresholds). We are able to achieve it, by using Equation 1[54], which creates a gauss like figure of recency in respect to the times of item creation and their last update. Decay and scale parameters enables the fine grained time decay property of recency recommender module.

$$\text{recency}(\text{article}) = w_1 * \text{timePenalty}(\text{published}_{at}) - w_2 * \text{timePenalty}(\text{updated}_{at}) \quad (1)$$

$$\text{timePenalty}(\text{time}) = \exp\left(-\frac{\max(0, |\text{time} - \text{published}_{at}| - \text{penaltyStart})^2}{2\left(-\frac{\text{scale}^2}{2}\right) * \log(\text{decay})}\right)$$

We either use the subsets of our so-called trending articles directly as top-n recommendations, or push them further down our method, where they serve as an input to some of the more computational expensive methods for recommendation generation.

Our reasoning behind this approach is strongly tied with the current trends of navigation on the Web. On majority of sites you can find lists of most popular items, new and current items, that narrows the information and navigation space for users. This especially applies to the domain of newspaper articles, where users are visiting news portals in order to find new and current information. These items could stand a higher chance of actually being visited by user, because on average, majority of users tend to be interested in some of the most popular and recent topics. We proved this assumption correct in our evaluations.

11.4 Contextual recommendations and clustering

Nowadays, context is getting ubiquitous and as seen in other works [13], it is a great source of information, which we can use for clustering user recommendation requests. In our method, we use contextual data like geo-location, gender, estimated age or salary, ISP and similar properties. By using streaming *k-means* algorithm, specifically the *kmeans++* implementation [14], we are able to perform clustering in real-time, while maintaining up-to-date representation of the clusters. Each recommendation request is assigned to a cluster, based on the properties of the request context. Inside of each of these cluster, we hold machine learning models used for classification and ranking items of items. Classification outcome and rank of items depend on the relevance and confidence of respective items being clicked or visited by user.

For classifying and ranking items we use two scalable and parallelizable approaches: decision trees (*Random forest*) and learning to rank (*Listnet*). Learning to rank and *Listnet* algorithm specifically, have already been proved [15, 16] to be a suitable choice for large scale machine learning model construction. In our approach we are using them as estimators of user's probability of actually clicking on a recommendation. Input to machine learning models is a narrowed subset of trending articles and outcome is a set of top-n recommendations with respective confidence weights, based on the learned model for a specific cluster, to which a recommendation request was assigned depending on contextual data. These models are updated and reconstructed periodically in the background with the latest data.

11.5 Aggregation of recommendations

Our proposed approach provides us with variety of possible combinations and aggregation of recommender modules within our hybrid system. The aggregation process is exactly the place, where the flexibility and adaptive properties of our approach can show off. Under ideal conditions and state of a recommender system (low workload), our approach uses all of the described recommender modules. Trendiness module acts as a meta-level recommender, which generates input subsets for other recommender modules (e.g., contextual).

[54] https://www.elastic.co/guide/en/elasticsearch/guide/current/decay-functions.html

Collaborative filtering and content based recommendations, which work with constructed user models provides recommendations based upon the user's behaviour and user models. Recommendations from the respective recommender are then grouped and their confidence is summed up, along with the weight of each of recommender modules.

Our approach eliminates cold start issue of a new user in the system. Even for an unknown or new user, we still have on-line contextual data and information about the currently visited item. Altogether with popularity and recency information, we can identify and recommend relevant articles to the user. Incoming request is assigned to a cluster by the on-line information and a machine learning model (learning to rank, random forest) ranks subset of trending articles according to the predicted cluster and user. As a response to the recommendation request, we present top-n recommendations to the user.

12 Evaluation

As explained in the previous section, one of the key points of our proposed approach is the focus on popularity and recentness of articles. In order to prove this hypothesis, we evaluated the effect of adding popularity and recency aspects to a simple k-nn recommender. In the k-nn algorithm, nearest neighbors were chosen by the co-occurrence factor based on item to item similarity [3]. Similarities were computed using log likelihood ratio of impression occurrences. Evaluations were performed on an offline dataset, consisting of ~12 millions impressions gathered from slovak online newspaper publisher SME.SK during a week in October 2009.

As shown in the Table 1, we were able to improve precision@10 and NDCG@10 metrics by over 100%, when we added popularity and recency aspects into the recommendations process. With items impressions, we have only binary scale (seen/not seen) which is unable to fully utilize NDCG properties. This effect can be seen also by the similar improvement rate of both of the used metrics.

Table 1. Evaluation of article trendiness increase in precision and NDCG metrics

No. of nn	Without popularity and recency		With popularity and recency (exp. boosting)			
	precision @10	NDCG@10	precision @10		NDCG@10	
2	0.0086	0.0410	0.0179	+ **108%**	0.0803	+ **96%**
3	0.0093	0.0411	0.0190	+ **104%**	0.0792	+ **93%**
5	0.0114	0.0495	0.0222	+ **95%**	0.0906	+ **83%**
7	0.0104	0.0453	0.0228	+ **19%**	0.0913	+ **102%**
10	0.0108	0.0446	0.0243	+ **125%**	0.0952	+ **113%**
15	0.0107	0.0414	0.0229	+ **114%**	0.0860	+ **108%**
20	0.0109	0.0418	0.0229	+ **110%**	0.0840	+ **101%**
30	0.0103	0.0412	0.0225	+ **118%**	0.0809	+ **96%**

13 Conclusions and future work

In this paper, we have proposed hybrid recommender system with focus on the scalable and extensible architecture, flexibility of the system and ability to adapt to the system workload. Generated recommendations are computed with the use of up-to-date information, whether it is users behaviour and context or item content. As a unique feature of our approach, we highlight recommendation requests clustering based on the user's context, which reduces information space and enables for frequent and continuous updating of underlying machine learning models used for generating recommendations.

So far, we have performed only a basic and partial evaluation of our proposed hybrid recommender approach. We yet have to evaluate usage of clustering of incoming requests, scalability and flexibility of our approach. We plan to perform online evaluations with streaming data, where we will focus on CTR metric. Furthermore, we will perform rigorous and repeatable evaluations concerning scalability and accuracy of our proposed recommender system with prepared datasets created from acquired streaming data.

In terms of extending our proposed method, there are several opportunities for improvement and experimenting. Inclusion of information gathered from social data could serve as a great addition to the user model. For example, if we had performed latent topic analysis (LDA/LSA), we could map these identified topics to not only topics identified inside the items content, but also to the user's interests, as expressed on their social network profiles.

14 Acknowledgments

This work was partially supported by the Scientific Grant Agency of Slovak Republic, grant No. VG 1/0774/16.

References

[1] R. Sumbaly, J. Kreps, and S. Shah, "The 'Big Data' Ecosystem at LinkedIn," *Int. Conf. Manag. Data (SIGMOD 2013)*, pp. 1–10, 2013.

[2] X. Amatriain, "Big & Personal: data and models behind Netflix recommendations," *Proc. 2nd Int. Work. Big Data*, pp. 1–6, 2013.

[3] G. Linden, B. Smith, and J. York, "Amazon.com Recommendations Item-to-Item Collaborative Filtering," *IEEE Internet Computing.*, vol. 7, no. 1, pp. 76–80, 2003.

[4] A. Boutet, D. Frey, and A. Kermarrec, "HyRec: Leveraging Browsers for Scalable Recommenders Categories and Subject Descriptors," pp. 85–96, 2014.

[5] V. Vekariya and G. R. Kulkarni, "Hybrid recommender systems: Survey and experiments," *2012 2nd Int. Conf. Digit. Inf. Commun. Technol. its Appl. DICTAP 2012*, pp. 469–473, 2012.

[6] X. Yang, Z. Zhang, and K. Wang, "Scalable collaborative filtering using incremental update and local link prediction," *Proc. 21st ACM Int. Conf. Inf. Knowl. Manag. - CIKM '12*, p. 2371, 2012.

[7] G. Takács, I. Pilászy, B. Németh, and D. Tikk, "Scalable Collaborative Filtering Approaches for Large Recommender Systems," *J. Mach. Learn. Res.*, vol. 10, no. 6 , pp. 623–656, 2009.

[8] R. M. Bell and Y. Koren, "Lessons from the Netflix prize challenge," *ACM SIGKDD Explor. Newsl.*, vol. 9, no. 2, p. 75, 2007.

[9] Y. U. Shi, M. Larson, and A. Hanjalic, "Collaborative Filtering beyond the User-Item Matrix: A Survey of the State of the Art and Future Challenges," *ACM Comput. Surv.*, vol. 47, no. 1, pp. 1–45, 2014.

[10] B. Chen, "fLDA : Matrix Factorization through Latent Dirichlet Allocation," *New York*, pp. 91–100, 2010.

[11] F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, *Recommender Systems Handbook*. Boston, MA: Springer US, 2011.

[12] Y. Zhou, D. Wilkinson, R. Schreiber, and R. Pan, "Algorithmic Aspects in Information and Management," Proc. of the 4th Int. Conf., AAIM 2008, Shanghai, China,R. Fleischer and J. Xu, Eds. Berlin, Springer Berlin Heidelberg, 2008, pp. 337–348.

[13] C. Palmisano, A. Tuzhilin, and M. Gorgoglione, "Using context to improve predictive modeling of customers in personalization applications," *IEEE Trans. Knowl. Data Eng.*, vol. 20, no. 11, pp. 1535–1549, 2008.

[14] B. Bahmani, B. Moseley, A. Vattani, R. Kumar, and S. Vassilvitskii, "Scalable K-Means ++," *Proc. VLDB Endow.*, vol. 5, no. 7, pp. 622–633, 2012.

[15] S. Shukla, M. Lease, and A. Tewari, "Parallelizing ListNet training using spark," *Proc. 35th Int. ACM SIGIR Conf. Res. Dev. Inf. Retr. - SIGIR '12*, p. 1127, 2012.

[16] T. Niek, "Scaling Learning to Rank to Big Data Using MapReduce to Parallelise Learning to Rank," University of Twente, Twente, Netherlands, 2014.

Príloha E. Obsah elektronického média

- `doc/` - Diplomová práca (elektornická verzia dokumentu vo formáte pdf a docx)
- `knn-rec/` - Implementácia odporúčacieho systému (zdrojové kódy) využitého pri overeniach dát z denníka SME
- `scarec/` - Implementácia navrhovanej hybridnej metódy v aplikácii ScaRec (zdrojové kódy) využitej na ORP
- `ml-models/` - Predučené modely strojového učenia
- `data/` - Vzorky dát použité na evaluácie a experimenty (Plista ORP, SME.sk)