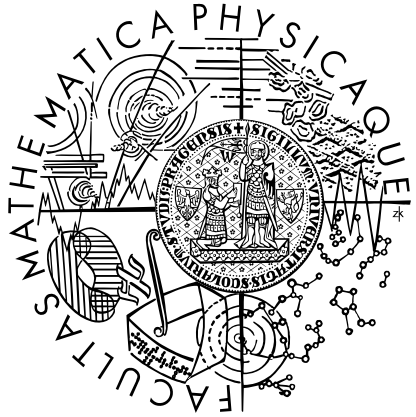
Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

DIPLOMOVÁ PRÁCE



Ladislav Peška

Uživatelské preference v prostředí prodejních webů

Katedra softwarového inženýrství

Vedoucí diplomové práce: prof. RNDr. Peter Vojtáš, DrSC.

Studijní program: Informatika, Softwarové systémy

2010

Děkuji vedoucímu diplomové práce prof. RNDr. Peter Vojtáš, DrSC. Za odporné vedení práce, za čas, nápady, rady a morální podporu, kterou mi věnoval. Cestovní kanceláři SLAN tour a Antikvariátu Ichtys za umožnění testování na svých prodejních webech a Lucii Malečkové za kontrolu práce.

Prohlašuji, že jsem svou diplomovou práci napsal samostatně a výhradně s použitím citovaných pramenů. Souhlasím se zapůjčováním práce.

V Praze dne 9. prosince 2010 Ladislav Peška

Obsah

Název práce: Uživatelské preference v prostředí prodejních webů
Autor: Ladislav Peška
Katedra (ústav): Katedra softwarového inženýrství
Vedoucí diplomové práce: prof. RNDr. Peter Vojtáš, DrSC.
e-mail vedoucího: vojtas@ksi.mff.cuni.cz

Abstrakt: *Cílem práce je nejprve vyhledání dostupných informací o uživatelských preferencích, uživatelské zpětné vazbě a jejich získávání, zpracování, skladování atd. Ze získaných informací pak sestavit návrhy či doporučení pro tvorbu doporučovacích systému v prodejnách webech (především se zaměřením na implicitní zpětné vazby). Další části práce se pak zabývají pak návrhem a implementací UPComp - samostatné doporučovací komponenty pro prodejní weby, která umožňuje dotazování na základě uživatelských preferencí. Komponenta je napsána v programovacím jazyce PHP a využívá databázi MySQL. Součástí práce je také testování komponenty na existujících prodejních webech slantour.cz a antikvariat-ichtys.cz.*

Klíčová slova: *uživatelské preference, prodejní web, doporučovací systémy*

Title: User preferences in the domain of web shops
Author: Ladislav Peska
Department: The Department of Software Engineering
Supervisor: prof. RNDr. Peter Vojtáš, DrSC.
Supervisor's e-mail: vojtas@ksi.mff.cuni.cz

Abstract: *The goal of the thesis is first to find available information about user preferences, user feedback and their acquisition, processing, storing etc. The collected information is then used for making suggestions / advices for the creating an recommender system for the web shops (with special emphasis on implicit feedback). The following chapters introduces UPComp - our solution of the recommender system for the web shops. The UPComp is written in the programming language PHP and uses MySQL database. The thesis also includes testing of the UPComp on real-user web shop sites slantour.cz and antikvariat-ichtys.cz.*

Keywords: *user preference, web shop, recommender systems*

1 Úvod

1.1 Motivace

Žijeme v době, kterou je možné bez nadsázky nazvat dobou internetu. Počet uživatelů internetu celosvětově překročil jednu miliardu, v blízké budoucnosti se očekává překročení hranice dvou miliard, v České republice pak internet pravidelně používá více než polovina populace.

S rostoucím počtem uživatelů také roste síla a důležitost metod, pro zjišťování, analyzování a doporučování na základě uživatelského chování či uživatelských akcí. Danou problematikou se zabývá vědní disciplína "Uživatelské preference", která tyto metody zastřešuje a poskytuje pro ně odpovídající teoretický základ.

Ze správné aplikace takových metod profituje jak uživatel (zrychlení a zjednodušení práce, kratší čas nutný na vyhledání požadované informace), provozovatelé webových serverů (vyšší výnosy, snížení počtu uživatelů, kteří relevantní informaci nenašli).

I přesto, že většina „významných" poskytovatelů webového obsahu (či klíčových hráčů v určitém segmentu trhu) již některé zmíněné metody využívá (například *alza.cz*, *youtube.com*, *yahoo.com* atd.), většina, převážně menších poskytovatelů, žádné podobné metody neaplikuje.

Důvodů pro takový stav může být více, jako pravděpodobné problémy se jeví

- neinformovanost (provozovatel webu neví o existenci popsaných metod)
- neochota (provozovatel metody zná, ale není přesvědčen o jejich významu či důležitosti)
- neschopnost (provozovatel metody zná, ale na svůj web je nedokáže implementovat)
- přílišná náročnost (implementace je možná, ale příliš časově či finančně náročná – nejedná se o rentabilní investici)

„Konečným" cílem, na jehož začátku stojí tato diplomová práce, je proto změna popsané situace, tedy umožnit i menším a středním poskytovatelům webového obsahu informovat o metodách pro analyzování a pro podporu rozhodování na základě uživatelského chování, přesvědčit je o jejich významu a umožnit jim jednoduše metody implementovat.

Diplomová práce se věnuje jednak uživatelskému chování (a jeho důsledcích pro metody pro podporu rozhodování), dále pak možnostem implementace některých metod a prokázání jejich účinnosti.

1.2 Cíl práce

Vzhledem k rozsahu a časové i implementační náročnosti splnění "konečného cíle" popsaného v předchozí části, se diplomová práce zaměří pouze na jeden ze segmentů webových poskytovatelů a sice e-shopy a další prodejní weby (přesná definice prodejního webu ***viz 2.1.2***).

Cílem práce je nejprve popsat chování uživatelů prodejního webu a zjistit důsledky, které z něj plynou pro metody na podporu uživatelova rozhodování. Dále pak navrhnout samostatnou komponentu - UPComp, která provozovateli prodejního webu umožní jednoduše zjišťovat uživatelovy preference a podporovat uživatelovo rozhodování.

Předpokládá se, že UPComp bude implementovat několik spíše méně pokročilých metod pro získávání a analýzu uživatelských preferencí, ale bude snadno rozšiřitelná o další - pokročilejší. Požadavky na komponentu jsou následující:

- UPComp bude působit jako „middle man" mezi databází a prodejním webem

- UPComp bude klást co možná nejméně restrikcí na architekturu prodejního webu či jeho zaměření (umožnit nasazení komponenty na širokém spektru již existujících webů)
- UPComp bude odpovídat v reálném čase – stejně jako klasická databáze
- UPComp by také neměla mít příliš složité rozhraní či implementaci a měla by být dále snadno rozšiřitelná

Další nedílnou součástí práce je pak testování komponenty na reálných uživatelích a prokázání účinnosti navrženého postupu.

1.2.1 Hlavní přínosy práce

Hlavní očekávané přínosy diplomové jsou:

- Sumarizace znalostí o uživatelských preferencích a z nich plynoucí tvorba závěrů, postupů či doporučení pro doporučovací systémy na prodejních webech.
- Vytvoření reálně použitelné doporučovací komponenty implementovatelné do prodejního webu.
- Testování a porovnání doporučovacích metod na reálných uživatelích.
- Získání souboru implicitních uživatelských zpětných vazeb vhodného pro další experimenty.

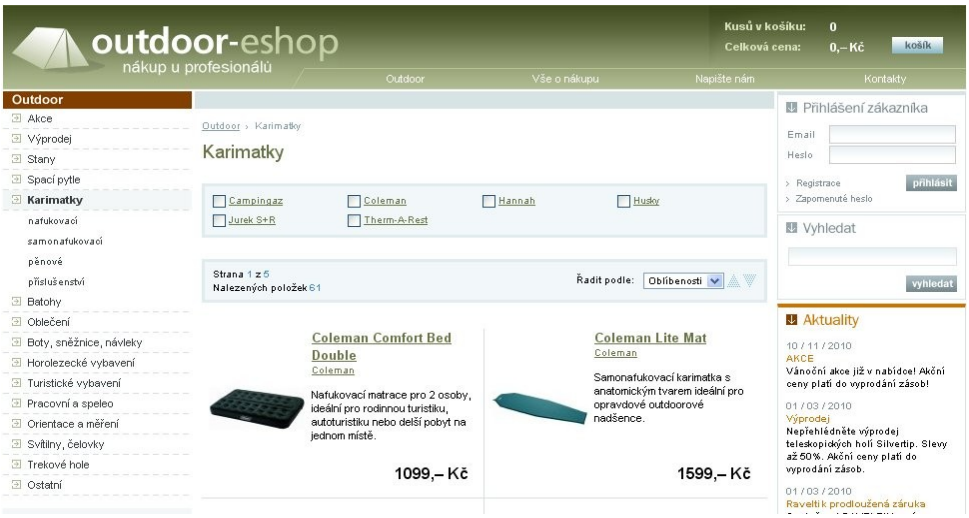
1.3 Příklady použití komponenty

1.3.1 Typické použití: e-shop

Typicky se předpokládá nasazení UPComp na e-shop zaměřený na prodej zboží z určité domény. Předpokládá se, že obchod již nějakou dobu funguje a má už existující zákazníky (v opačném případě se výsledky využití komponenty projeví až po delším čase v závislosti na rychlosti získávání uživatelských dat).

Jako modelový e-shop může sloužit ***http://www.outdoor-eshop.cz***, zaměřený na outdoor a turistiku. E-shop na hlavní straně zobrazuje akční časově omezené nabídky a nové zboží. Dále je možné procházet katalog zboží dle kategorií a v rámci kategorie případně podkategorie zboží filtrovat dle výrobců a řadit dle některých atributů objektu.

Další možnosti vyhledávání výrobku už jsou pouze „TOP 10“ - 10 nejprodávanějších objektů v celém e-shopu a fulltextové vyhledávání, obojí zobrazené v pravém menu.



Obrázek 1: Screenshot z webu www.outdoor-eshop.cz

Nasazení UPComp by v e-shopu umožnilo snadné využití mimo jiné následujících funkcí:

- nejoblíbenější / nejprodávanější objekty v kategorii – při zobrazení kategorie
- objekty související s vybraným objektem – při zobrazení detailu objektu
- vyhledávání dle parametrů v rámci kategorie – při zobrazení kategorie
- pole potencionálně zajímavých objektů vybraných na základě uživatelských akcí

Předpokladem použití výše zmíněných funkcí je to, že usnadní orientaci uživateli e-shopu, zobrazí mu potencionálně zajímavé objekty, které by jinak mohl přehlédnout a případně jinak napomohou uživateli k nalezení a nákupu vhodného objektu.

Využití zmíněných funkcí by e-shop zbavilo některých nevýhod oproti konkurenci (např. *sambarsport.cz* – nejprodávanější produkty v kategorii) a znamenalo by také získání konkurenční výhody oproti ostatním e-shopům zaměřeným na stejnou doménu (samozřejmě, pokud se prokáže účinnost navržených funkcí – více ***viz kapitola 7***).

1.3.2 Příklady netypického použití

UPComp by ovšem mělo jít nasadit i na webové systémy, které neodpovídají zcela definici e-shopu (komponenta nevyžaduje použití některých prvků typických pro e-shopy – např. nákupní košík, sklad, výběr způsobu platby či dopravy u objednávky atd.). Pro přesnou specifikaci očekávaných vlastností prodejního webu ***viz 4.3*** .

Příkladem ne zcela typického použití UPComp může být mimo jiné

- **on-line antikvariát**: objekty jsou obvykle dostupné v jednom či velmi málo exemplářích, po objednávce jsou většinou pro ostatní uživatele nezajímavé. Příkladem on-line antikvariátu je ***www.antikvariat-ichtys.cz***
- **aukční server**: objekty jsou obvykle dostupné v jednom či velmi málo exemplářích, po uplynutí aukce je objekt sám nezajímavý, ale obvykle je možné vyhledat objekty původnímu velmi podobné (stejný typ auta atp.). Příkladem aukčního serveru je ***www.ikup.cz***.
- **rezervační systém zájezdů**: neobsahuje obvyklé prvky e-shopu, typický je dlouhý interval mezi objednávkami od jednoho uživatele. Příkladem rezervačního systému zájezdů je ***www.slantour.cz*****.**

1.4 Alternativní přístupy

V následující kapitole nastíníme další alternativy, které má provozovatel prodejního webu, pokud se rozhodne využívat některých metod pro podporu rozhodování uživatele.

1.4.1 Recommender framework

Alternativní možností jak aplikovat metody pro podporu uživatelských rozhodnutí, je využít některého z frameworků pro doporučování (recommender framework), například **Duine** ***http://duineframework.org/***.

Výhodou frameworků je především větší flexibilita (nejsou omezeny pouze na prodejní weby), obvykle umožňují použití různých funkcí pro tvorbu doporučení a umožňují s doporučením pracovat různými způsoby.

Nevýhody jsou těsně spojeny s výhodami frameworků: vyžadují větší úsilí od programátora, který je implementuje a výsledky, které poskytují je třeba dále zpracovávat, než je možné je poskytnout uživateli. V neposlední řadě pak obecnost frameworku může zapříčinit obtížnost vyjádření některých typických zpětných vazeb uživatelů prodejních webů (například vložení zboží do košíku).

Použití některého recommender frameworku je vhodné především pro netypické projekty, nebo pokud je třeba netypicky zpracovávat vypočtené uživatelské preference.

1.4.2 PrefShop

Zajímavou alternativou pro nové projekty může být použití e-shopů se zabudovanými metodami pro podporu rozhodování uživatele. Základ takového e-shopu položil ve své diplomové práci Bronislav Václav **[Vac10]**. Oproti UPComp se Václav zaměřoval především na přímé preference a tvorbu uživatelského rozhraní pro jejich zadávání (vyhledávání objektů a jejich prezentace) – UPComp se soustředí především na implicitní preference a doporučování objektů.

Potenciální nevýhodou PrefShopu je omezení pouze na „klasické“ e-shopy a především pak nemožnost PrefShop použít pro již běžící projekty.

I přes zmíněné nevýhody je PrefShop zajímavá alternativa především pro nově vznikající projekty.

1.5 Obsah kapitol

V **kapitole 2** se budeme zabývat uživatelskými preferencemi – nejprve zadefinujeme pojmy, které s uživatelskými preferencemi souvisí, následně se budeme zabývat především jejich výpočtem, dělením či časovým vývojem. Zde vycházíme především z použité literatury, kterou vzájemně porovnáváme a snažíme se z ní vyvodit závěry platné pro prodejní weby. Ve **3. kapitole** se pak zaměříme na implicitní uživatelské preference, jejich možné využití a problémy, které přináší. V **kapitole 4** formulujeme návrh nezávislé komponenty UPComp pro doporučování na základě uživatelských preferencí, **kapitola 5** pak obsahuje uživatelskou a **kapitola 6** programátorskou příručku této komponenty. V **7. kapitole** popisujeme průběh a výsledky testování UPComp na reálných uživatelích 2 prodejních webů. **8. kapitola** obsahuje závěr diplomové práce – zhodnocení dosažených výsledků a možnosti pro další rozšíření práce, **9. kapitola** seznam použité literatury a nakonec **10. kapitola** obsahuje přílohy diplomové práce.

1.6 Užívané značení

V dalším textu bude použito následující značení:

- Odkazy na jiná místa diplomové práce budou uvedeny tučným písmem, kurzívou a číslem kapitoly: ***viz 3.7.2*** , ***viz kapitola 4,*** odkazy na použitou literaturu pak tučně v hranatých závorkách: **[KT03]**
- Hypertextové odkazy jsou uváděny kurzívou, podtrženě, modrým písmem: *www.slantour.cz*
- Poznámky či komentáře autora k výsledkům či studiím jiných autorů, případně rozdíly v závěrech budou uváděny kurzívou, šedivě: *Oproti článku je však volba, které dílčí metody budou použity, ponechána na uživateli.*
- Upřesňující či vysvětlující poznámky jsou uvedeny obvykle v závorkách: (ať již vědomě, nebo na základě svého chování).
- Ukázky zdrojových kódů budou uváděny bezpatkovým písmem, zelenou barvou, pokud jsou součástí textu: ComplexQueryHandler ->aggregateExprResults(), v případě delších souvislých bloků navíc v samostatném rámečku. :

```
interface <jméno metody> {
   /** returns $noOfUsers of the best rated users
    */
   public function getBestUsers($noOfUsers, $userList="");
```

```
}
```

2 Uživatelské preference a prodejní weby

V této kapitole se zabýváme především pojmy uživatel, uživatelská preference, uživatelská zpětná vazba a cíl uživatele. Zpětnou vazbu a cíle uživatele rozdělíme podle obvyklých kritérií uváděných v dostupné literatuře a provedeme jejich srovnání s pojmem preference. Dále se zabýváme možnostmi získávání zpětných vazeb v prostředí prodejních webů, časovým vývojem uživatelské preference a v neposlední řadě také možnostmi identifikace uživatele, která je nezbytná pro uvažování o uživatelské preferenci.

Kapitola 2 obsahuje z větší části souhrn informací z použité literatury, případně obecně známých faktů, závěry a doporučení pro prodejní weby jsou pak dílem autora diplomové práce.

2.1 Definice pojmů

2.1.1 World Wide web

World wide web je „distribuovaný hypertextový internetový informační systém, ve kterém dokumenty obsahují odkazy na jiné místní nebo vzdálené dokumenty.“ **[Wiki10a]**

Kolekci místních dokumentů nazveme webovým serverem, nebo zkráceně webem, jednotlivé dokumenty pak webovými stránkami.

2.1.2 Prodejní web

Prodejní web je takový webový server, který především obsahuje a prezentuje kolekci objektů (produktů či služeb) určených k prodeji. Předpokládáme, že prodejní web obsahuje API pro nákup/objednávku jednotlivých objektů a že hlavním cílem provozovatele prodejního webu prodej prezentovaných objektů a tedy zisk.

Mnohem obvyklejším termínem pro webové servery s podobným zaměřením je „e-shop“ **[Wiki10b**]. Termín e-shop je však v obecném chápání více restriktivní – předpokládá existenci specifických funkcí (například sklad, nákupní košík, specifický průběh objednávky atd.). Pro naše potřeby proto zavádíme pojem méně restriktivní (například aukční server nebo rezervační systém zájezdů odpovídají definici prodejního webu, avšak nikoli e-shopu).

2.1.3 Objekty prodejního webu

Jako objekt prodejního webu budeme uvažovat entity (obvykle produkty či služby) určené k prodeji. Předpokládáme, že každý objekt se skládá z jednotlivých atributů, které ho popisují. Množina atributů může být pro různé objekty různá, ale předpokládáme, že všechny objekty prodejního webu je možné jednoznačně identifikovat podle jednoho atributu (id).

Předpokládáme, že v rámci prodejního webu existují funkce:

- **Katalog objektů** – tj. výpis seznamu objektů, případně filtrovaný podle některých atributů.
- **Detail objektu** – webová stránka/stránky prezentující atributy objektu
- **Objednávka objektu** – API umožňující nákup objektu (dle povahy webu – například vyplnění kupní smlouvy)

2.1.4 Uživatel prodejního webu

Uživatelem prodejního webu rozumíme osobu, která nějakým způsobem interagovala s prodejním webem, aniž by to bylo za účelem jeho správy. Mezi uživatele webu tedy obvykle nepočítáme jeho majitele či provozovatele atp. Problém jednoznačného určení uživatele ***viz 2.2.5*** **.**

2.1.5 Cíle uživatele

Cíl uživatele je soubor očekávaných interakcí s webem, případně s konkrétním webovým serverem. Uživatel obvykle dovede svůj cíl formulovat v přirozeném jazyce – například „chci najít všechny dostupné informace o Janu Šedivém“. Podle **[Bro02]** můžeme uživatelské cíle při dotazování na webu přibližně rozdělit do 3 kategorií:

- **Navigační cíle**: nalézt web nebo objekt, který je uživateli znám nebo předpokládá jeho existenci – například „Olympus“, „AAA auto“ atd.
- **Informační cíle**: nalézt informaci, o které uživatel předpokládá, že na webu existuje – například „obchody v New Yorku“, „bitva na Bílé hoře“ atd.
- **Transakční cíle**: cílem uživatele je nalézt web či objekt, se kterým chce provádět další operace (koupit produkt, poslechnout si hudební skladbu...)

Pokud se v pohledu omezíme na konkrétní web, tak můžeme předpokládat, že navigační cíle obvykle slouží k dosažení webu jako takového (případně některé jeho součásti). Po jeho dosažení uživatel svůj cíl upraví buď na informační, nebo transakční a dále se orientuje pomocí dedikovaných navigačních prvků webu (nebo web opouští, pokud neodpovídá jeho požadavkům).

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

2.1.6 Uživatelova preference

Uživatelská preference je obvykle definována jako funkce $P_U(o): O \times U \rightarrow [0,1]$, která pro konkrétního uživatele U a objekt o z množiny objektů O vrací míru „oblíbenosti“ objektu u uživatele. „Oblíbenost“ objektu je doménově závislá vlastnost (*oblíbenost hudební skladby vs. oblíbenost notebooku*). V případě prodejních webů budeme dále předpokládat, že uživatel má transakční cíl „nákup/objednávka objektu s určitými atributy“ (*ostatní cíle zanedbáváme, neboť nejsou příliš důležité pro provozovatele prodejního webu*). Oblíbenost objektu pak můžeme definovat jako míru ochoty uživatele objekt koupit.

2.1.7 Zpětná vazba

Zpětná vazba je informace, kterou vědomě či nevědomě poskytl uživatel během interakce s webovým serverem a na základě které je možné činit závěry či předpoklady o uživatelově preferenci vůči některému objektu.

2.1.8 Krátkodobá preference

Krátkodobá preference uživatele představuje preferenci k objektům na základě aktuálního cíle. Například, je-li aktuální cíl uživatele „nákup levného kompaktního digitálního fotoaparátu pro manželku“, pak preference jakékoli digitální zrcadlovky bude pravděpodobně blízká 0 i přesto, že uživatel je v jiné situaci (v jiném čase) vyhledává.

2.1.9 Dlouhodobá preference

Dlouhodobá preference vyjadřuje obecnější pravidla, kterými se uživatel většinou řídí. Například preference nižší ceny, preference zrcadlovek před kompaktními fotoaparáty, preference značky Škoda před ostatními atp.

2.1.10 Doporučovací systém

Doporučovací systém je taková aplikace, která na základě uživatelské zpětné vazby, případně i dalších informací, formuluje závěry o uživatelově preferenci k některým objektům. Systém obvykle obsahuje jednu či více metod pro výpočet uživatelské preference **[Wiki10c].**

2.2 Vyjádření zpětné vazby

Zpětná vazba je jedním z klíčových pojmů doporučovacích systémů. Uživatelská zpětná vazba je informace, kterou doporučovacímu systému poskytl uživatel (ať již vědomě, nebo na základě svého chování). Obvykle odlišujeme 2 či 3 různé druhy zpětných vazeb: Dle **[KT03]** lze zpětné vazby rozdělit podle zapojení uživatele na **implicitní zpětnou vazbu** – získanou bez vědomého úsilí uživatele a **explicitní zpětnou vazbu**, kterou uživatel poskytl záměrně přes rozhraní webu. Oproti tomuto rozdělení budeme uvažovat ještě **přímou zpětnou vazbu [EV09]**, kde uživatel přímo specifikuje atributy požadovaného objektu. Na přímou zpětnou vazbu je možné nahlížet také jako na druh explicitní zpětné vazby. My jí vyčleňujeme především proto, že je v mnoha prodejních webech zpracovávána výrazně jinak, než ostatní explicitní zpětné vazby.

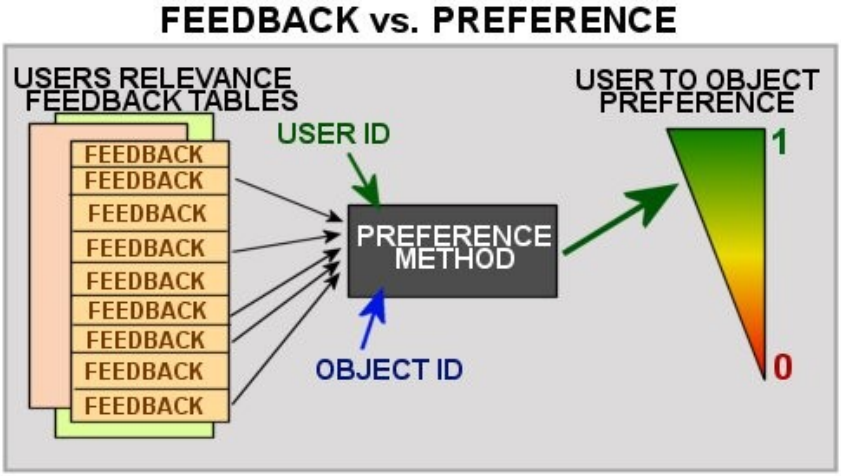
V dalším textu budeme předpokládat, že následující termíny jsou identické:

- implicitní zpětná vazba = implicitní uživatelská data
- explicitní zpětná vazba = explicitní uživatelská data
- přímá zpětná vazba = direct query

2.2.1 Zpětná vazba vs. uživatelova preference

Předpokládáme, že uživatelova zpětná vazba (feedback) představuje spíše surová, či předzpracovaná data a předpokládaná preference uživatele *U* k objektu *O* je počítána nějakou funkcí *F*, kterou obvykle nazýváme metodou pro výpočet uživatelské preference:

$F(feedback,U,O) \rightarrow [0,1]$



Obrázek 2: Vztah zpětné vazby (feedback) a uživatelské preference

2.2.2 Implicitní uživatelské data

Implicitní uživatelská data jsou takové informace, které jsou získávány na základě chování uživatele na webu bez aktivní účasti uživatele na jejich podání. ***Dále viz kapitola 3 .***

2.2.3 Explicitní uživatelské data

Explicitní uživatelská data (nebo také explicitní zpětná vazba) jsou informace, na jejichž poskytnutí musel uživatel vědomě vyvinout určité úsilí - použití dedikovaného rozhraní webu (například hodnocení objektu).

Mezi výhody explicitní zpětné vazby nesporně patří její doménová nezávislost – stejné hodnocení, tedy i stejné doporučovací systémy budované nad explicitní zpětnou vazbou lze použít pro velkou škálu různých projektů (i když v případě prodejních webů dále uvedeme některé limitace tohoto tvrzení).

Hlavní nevýhodou explicitní zpětné vazby je samotná nutnost, aby jí uživatel aktivně poskytnul. Velké množství uživatelů zpětnou vazbu buď poskytnout nechce, nebo k tomu nejsou dostatečně motivováni, což snižuje hodnotu výsledků doporučovacích systémů postavených nad explicitní zpětnou vazbou.

Explicitní vyjádření preference

Typickým příkladem vyjádření explicitních preferencí je hodnocení objektu. Uživateli je umožněno pro každý objekt zvolit na diskrétní škále buď celkové hodnocení objektu (obvyklejší příklad), nebo i hodnocení jednotlivých atributů objektu (například *booking.com*).

Při hodnocení objektu je zásadní zvolit dostatečně jednoduchou stupnici tak, aby nedocházelo k přílišným nekonzistencím v hodnocení a zároveň uživateli srozumitelně stupnici zobrazit.

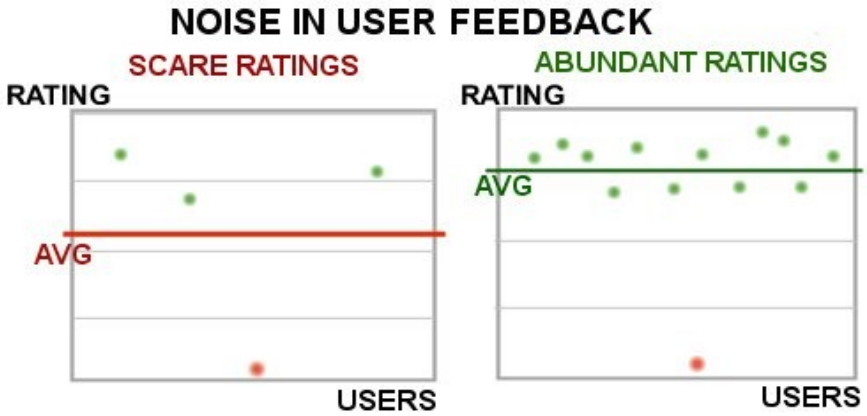
Explicitní zpětná vazba je obvykle považována za přesnější vyjádření uživatelovy preference než implicitní zpětná vazba, nicméně například studie **[JSK10b]** provedená nad uživateli hudebního serveru last.fm tento fakt nepotvrdila. Autoři se spíše přiklánějí k závěru, že – alespoň u zvolených dat – jsou explicitní a implicitní preference stejně přesné. Obdobně studie **[WJR01],** kde měli uživatelé s použitím upraveného vyhledávače Google (změna řazení výsledků na základě explicitní nebo implicitní zpětné vazby) vyřešit několik typických vyhledávacích úkolů, nezjistila žádné statisticky významné rozdíly.

Problémy spojené s explicitními preferencemi

Zásadním problémem explicitní zpětné vazby je neochota některých uživatelů jí poskytovat **[JSK10a]**. Poměrně velké procento uživatelů není ochotno žádnou zpětnou vazbu poskytnout (***viz 7.1.2***), což může zkreslit výsledky kolaborativních algoritmů (***viz 4.5***) celkově a pro dané uživatele znemožní aplikaci kolaborativních algoritmů na základě explicitních preferencí. Především u prodejních webů s velkým poměrem #objektů/#uživatelů nebo s rychlou obměnou nabízených objektů pak může mnoho objektů zůstat neohodnocených. Aplikace kolaborativních algoritmů na základě explicitní zpětné vazby je pak pro tyto objekty rovněž znemožněna. Při menší kvantitě poskytnutých odpovědí se pak mohou více projevit jevy jako šum v hodnocení.

Jako důležitá se jeví osobní zainteresovanost uživatele na poskytnutí zpětné vazby. Ať už ve formě bonusů/slev či jiných výhod nebo pomoci ostatním uživatelům (obvykle v kombinaci s možností napsat komentář k hodnocení v prostém jazyce). Například server ***booking.com*** **–** on-line rezervace hotelů – klienty po skončení jejich objednaného pobytu aktivně vyzývá, aby svůj pobyt ohodnotili a pomohli tím dalším uživatelům s výběrem ubytování.

Rozdíl mezi hodnocením a preferencí



Obrázek 3: Srovnání vlivu osamoceného hodnocení - šumu - na průměrné hodnocení (AVG) u málo a hodně hodnocených objektů

V neposlední řadě je vhodné zmínit určitou nekonzistenci mezi sémantikou hodnocení objektu a námi definovanou preferencí uživatele.

Uživatelovo hodnocení v prodejních webech obvykle vyjadřuje, jak je uživatel s objektem spokojen, případně ještě obsahuje „recenzi" či komentář k objektu (*booking.com*, *mall.cz*, *kasa.cz*...). Předpokládá se tedy, že uživatel je již s objektem seznámen a používá ho. Uživatelova ochota objekt koupit byla proto někdy v minulosti rovna 1 (mimo případy, kdy uživatel nekupoval objekt sám) a ve většině případů je nyní rovna 0 (málokteré objekty uživatel kupuje opakovaně). Uživatelovo hodnocení tak může vyjadřovat spokojenost s objektem, případně ochotu uživatele koupit objekt znovu, nikoli však aktuální ochotu uživatele objekt zakoupit.

Jako ilustraci uvedeme typický průběh nákupu digitálního fotoaparátu:

- Uživatel specifikuje svůj cíl (jaký fotoaparát chce koupit).
- Na prodejním webu (prodejních webech) nalezne vhodné kandidáty na nákup.
- Ze zvolených kandidátů vybere jeden fotoaparát, který zakoupí.
- Po seznámení s fotoaparátem (případně na výzvu prodejního webu) uživatel může vyplnit hodnocení objektu.

Kladné hodnocení objektu znamená, že uživatel splnil svůj cíl a žádný další fotoaparát už po určitou dobu hledat nebude (nicméně, stále ho může zajímat příslušenství k již zakoupenému).

Z toho plyne, že u některých prodejních webů může být důležité – na rozdíl například od hudebních doporučovacích systémů – oddělit doporučování podobných objektů (zjištěných na základě podobnosti atributů) od doporučování souvisejících objektů (zjištěných například na základě kolaborativního filtrování – ***viz 4.5***). Přesněji řečeno ve zmíněných situacích podobné objekty nedoporučovat, i když jsou to zároveň objekty související.

Pokročilé techniky zpracování explicitní zpětné vazby

V případě porovnávání hodnocení objektů od 2 různých uživatelů může být vhodné neporovnávat přímo hodnocení daných objektů, ale odchylky od průměrného hodnocení uživatele **[SKKR01]**.

Autoři studie **[EHV07]** předpokládají, že hodnocení objektu – nachází-li se v seznamu dalších objektů je závislé na „kvalitě" ostatních objektů (uživatel dá průměrnému objektu lepší hodnocení, pokud je obklopen podprůměrnými objekty).

Vzhledem k tomu, že těžiště diplomové práce představují implicitní data, nebudou v dalším textu a v praktické části práce pokročilé techniky zpracování explicitních dat uvažovány. Nicméně doporučovací komponenta by měla být navrhnuta tak, aby do budoucna umožnila rozšíření zpracování explicitních zpětných vazeb.

Další možnosti explicitního vyjádření preference

Někteří autoři **[Cia10]** uvažují i jiné modely vyjádření explicitní zpětné vazby. Například tzv. komparativní model, kde preference uživatele U vůči objektům o_1 a o_2 je definována jako funkce :

$F_U(o_1, o_2)$ = 1, pokud objekt 1 je preferovanější, než objekt 2

-1, pokud objekt 2 je preferovanější, než objekt 1, případně i

0, pokud nelze rozhodnout o preferenci mezi objekty

Tento model uživatelských preferencí není obvykle příliš rozšířený. výjimkou je například porovnávání „oblíbenosti“ osobních fotografií na některých seznamkách (*http://xchat.centrum.cz/duel/*), kde však slouží spíše k hodnocení fotografií, než k získávání uživatelských preferencí.

Obrázek 4: Screenshot z "duelu" - porovnávání oblíbenosti fotografií na xchat.centrum.cz

Komparativní model preferencí navíc může trpět problémy jako cyklické hodnocení objektů, nebo velké množství neporovnatelných objektů.

Další autoři se snaží komparativní model upravit tak, aby výskyt popsaných problémů minimalizovali (například Kiessling**[Kie02]** zavádí místo klasického komparativního modelu relace nad hodnotami atributů: POS(red) = preference červené barvy před ostatními, POS/NEG(red, {blue}) = preference nejlépe červené, není-li červená, tak hlavně ne modrou). Nicméně ani tyto modely nejsou v současné době prakticky příliš používány.

V dalším textu a v praktické části diplomové práce komparativní model uživatelských preferencí neuvažujeme.

2.2.4 Přímé preference

Přímá preference uživatele vůči množině objektů jsou vlastnost či vlastnosti objektu, nebo jeho atributů, u kterých uživatel pomocí dedikovaného rozhraní specifikuje, že je u hledaného objektu vyžaduje (případně upřednostňuje, neočekává, zakazuje atd.).

Uživatel obvykle vyjadřuje přímé preference buď procházením katalogu produktů – zobrazení kategorie = preference objektů této kategorie, vyhledáváním na základě klíčového slova, nebo vyhledáváním podle atributů objektu.

Přímé preference se obvykle přeloží jako dotaz do databáze objektů (uživateli je pak prezentován výsledek dotazu). Do budoucna je možné uvažovat o využití přímých preferencí k dedukci dlouhodobých preferencí uživatele, nebo preferencí na jednotlivých atributech (obdobně jako **[EV09b]** s explicitními daty).

Přímé vyjádření preferencí je obvykle považováno za nejpřesnější, má ovšem několik úskalí:

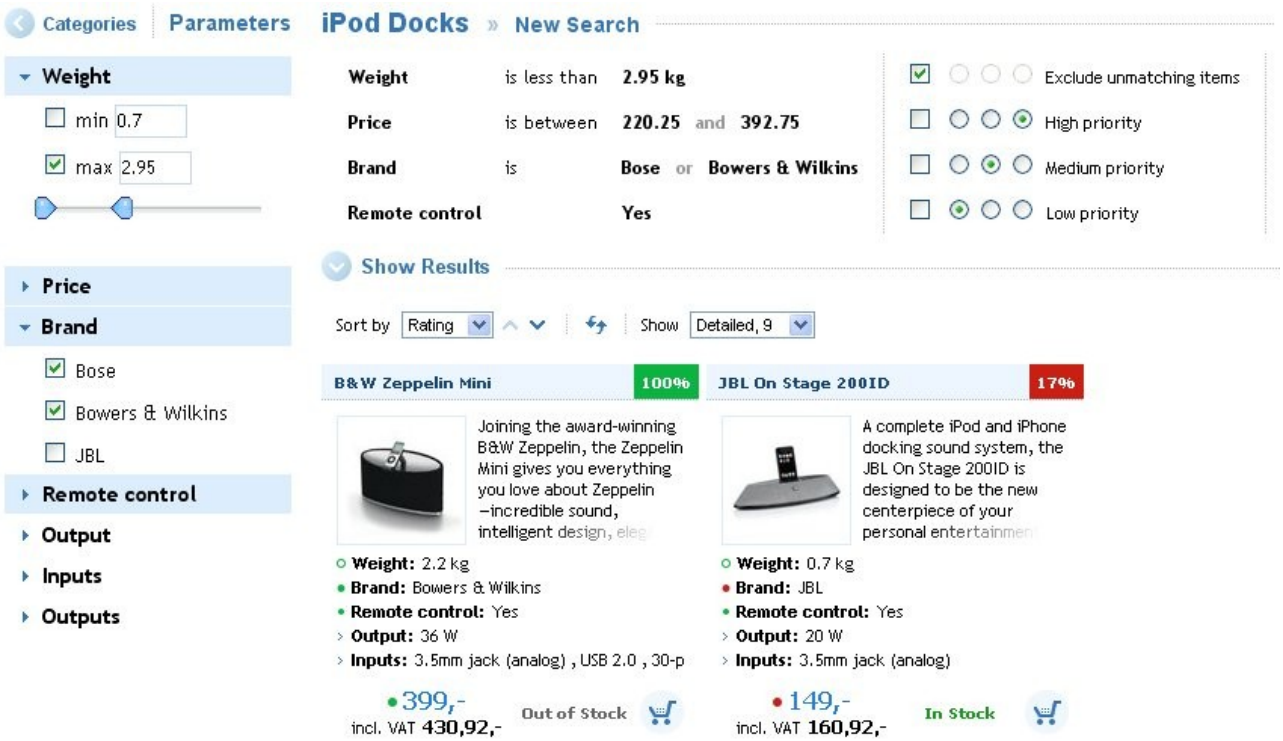
- Uživatel musí své preference na jednotlivých atributech znát nebo je alespoň odhadnout a musí je umět vhodně vyjádřit.
- Web musím mít API navrženo tak, aby umožnilo uživatelům jejich preference vhodně vyjádřit. V tomto bodě je třeba optimalizovat na jednu stranu jednoduchost, pochopitelnost a snadnou použitelnost rozhraní a na druhou stranu komplexnost – umožnit uživateli vyjádřit své preference co nejpřesněji.

  Většina prodejních webů nabízí spíše jednoduché rozhraní – vyhledávání podle klíčového slova, případně konjunktivní vyhledávání dle hodnoty nebo rozsahu hodnot vybraných atributů.

  Uživatelovy preference však mohou mít i další aspekty, které API prodejních webů obvykle neumožňuje vyjádřit: tolerance uvedených hodnot, jiné než rozsahové zadání přijatelných hodnot, důležitost atributu nebo možnost zadat podmínky, které objekt nesmí splňovat. Otázkou však zůstává, zda a kolik uživatelů uvažuje s popsanými aspekty preferencí a jak navrhnout vhodné rozhraní pro vyjádření alespoň některých aspektů těchto preferencí a zároveň nesnížit komfort vyhledávání ostatním uživatelům.
- Vyhodnocování přímých preferencí se obvykle děje konjunktivním rozsahovým dotazem (objekty, které splňují všechny zadané podmínky, jsou ohodnoceny jako vyhovující a zobrazeny uživateli, ostatní objekty jsou ohodnoceny jako nevyhovující a zobrazeny nejsou).

  U některých atributů má však dobrý smysl zařadit do výsledků i ty objekty, které podmínce sice nevyhovují, ale jen „těsně“. Případně je možné zařadit i takové objekty, které sice některou podmínku zcela nesplňují, ale podmínka sama o sobě není příliš důležitá.

  Vyjádřením různých aspektů přímé preference, jejich zpracováním a prezentací výsledků se zabýval například Bronislav Václav **[Vac10]**. Václav navrhl ve své diplomové práci systém PrefShop, což je e-shop rozšířený o možnost zadávání různých aspektů přímé uživatelské preference, jejich vyhodnocování a prezentaci. Nicméně i přes pokrok v této oblasti je třeba ještě dostatečně otestovat v jakých situacích a pro jaké uživatele je obdobné rozhraní přínosné či potřebné, a pro které naopak matoucí a odrazující.



Obrázek 5: Screenshot z PrefShop - vyhledávání objektů

2.2.5 Identifikace uživatele

Pro jakékoli uvažování o preferencích uživatele je klíčová schopnost jej jednoznačně identifikovat. V současné době existují tři postupy, jak identifikaci uživatele provádět:

- **Registrace uživatele:** registrace je nejspolehlivější způsob identifikace uživatele. Systém identifikuje unikátní přihlašovací jméno a heslo – umožní tedy rozeznat jednoho uživatele používajícího web z různých počítačů i více různých uživatelů používajících stejný počítač.

  Hlavní nevýhodou registrace je sama nutnost jí vyplnit: některé uživatele může nutnost registrovat se odradit od používání daného webu, případně pokud registrace není povinná, netriviální část uživatelů jí nevyplní.

- **Identifikace pomocí IP adresy:** systém identifikuje unikátní IP adresu, což je zároveň jeho hlavní slabinou – stejnou IP adresu velmi často sdílí více počítačů v lokální síti a tedy pravděpodobně i více uživatelů. Systém zároveň nerozpozná stejného uživatele, který k webu přistupuje z různých lokací.

  Identifikaci uživatele na základě IP adresy bychom se proto měli raději vyhnout.

- **Identifikace pomocí COOKIES:** systém identifikuje unikátní kombinaci PC + prohlížeč. Nerozpozná tedy stejného uživatele, který k webu přistupuje z různých počítačů, ale za předpokladu, že s počítačem pracuje pouze jeden uživatel, rozpozná dobře jednotlivé uživatele (identifikace uživatelů je „jemnější“ oproti identifikaci pomocí IP).

  Nevýhodou systému je, že cookies, pomocí kterých je identifikace prováděna, jsou ukládány na pevném disku uživatele. Uživatel je proto může kdykoli odstranit nebo jejich ukládání zakázat a znemožnit tak svou identifikaci. I přes tyto problémy se identifikace pomocí COOKIES jeví jako lepší varianta oproti identifikaci pomocí IP.

Jako ideální řešení identifikace uživatele v prodejních webech se jeví kombinace identifikace pomocí cookies pro nepřihlášené uživatele s možností registrovat uživatele (a následná identifikace pomocí přihlášení).

2.3 Získávání uživatelských dat

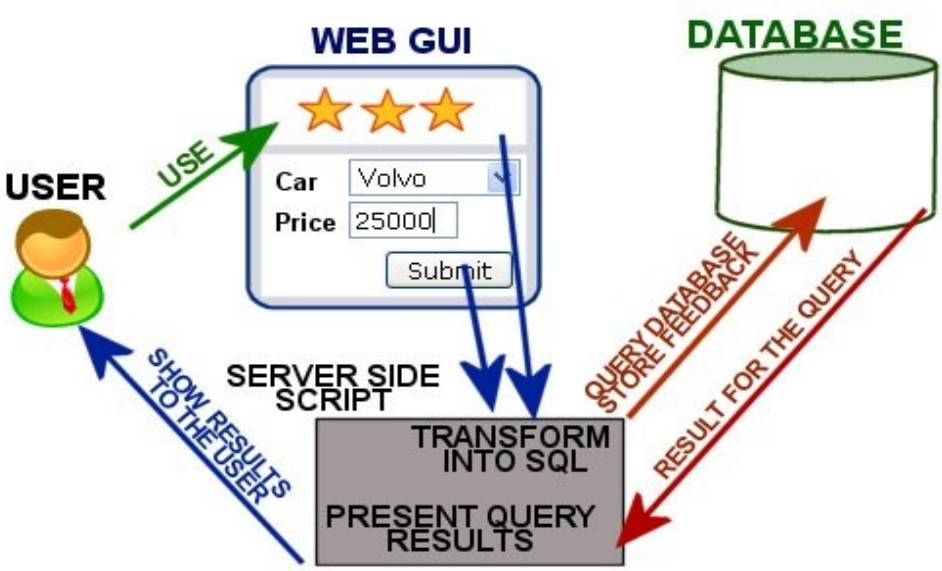
V následující kapitole rozebereme základní metody a postupy sloužící k získání a zpracování uživatelovy zpětné vazby. V minulosti byly zásadní rozdíly ve způsobu zpracování především mezi explicitními a přímými preferencemi na jedné straně (používání formulářů) a implicitními preferencemi (sledování událostí pomocí JavaScriptu aj.) na straně druhé. S postupným rozšiřováním asynchronního zpracování – AJAXu **[Wiki10d]** se však tyto rozdíly začínají stírat.

2.3.1 Získávání explicitních a přímých zpětných vazeb

Získávání přímých a explicitních preferencí je jedna z typických a dobře známých programátorských úloh, proto jí popíšeme jen zevrubně.

Přímé a explicitní preference zadává uživatel pomocí dedikovaného rozhraní webu (obvykle vyplnění a odeslání vyhledávacího formuláře, kliknutím na příslušný odkaz atp.).

Použití rozhraní vyvolá obslužný skript – buď přímo (odkaz směřující na adresu skriptu, formulář s nastaveným skriptem pro zpracování...), nebo asynchronně pomocí AJAXu. Hlavním úkolem obslužného skriptu je obvykle zpracování odpovědi pro uživatele, nicméně je možné pomocí něj i uložit vhodnou reprezentaci přímé či explicitní preference do databáze.



Obrázek 6: Diagram zpracování explicitních a přímých zpětných vazeb

2.3.2 Získávání implicitních zpětných vazeb

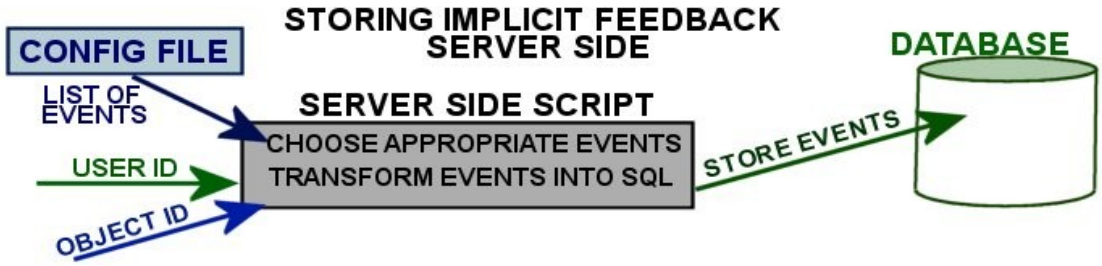
Způsob získávání implicitních preferencí je přímo ovlivněn výběrem uvažovaných implicitních akcí. Z pohledu použitelných nástrojů můžeme implicitní akce rozdělit na:

- **akce zpracovatelné server-side skriptovacím jazykem** (například PHP) **[Wiki10e]** jsou akce, které jsou zaznamenatelné již při vytváření konkrétní stránky na straně serveru – například zobrazení objektu, objednávka objektu atd.

  Hlavní výhodou zpracování akce server-side skriptem je nezávislost na nastavení a funkcích uživatelova prohlížeče. Nevýhodou je omezená paleta akcí, které lze na straně serveru zaznamenat.

  Další problém mohou znamenat crawlery – programy které automaticky procházejí webové stránky **[Wiki10f]**. Vzhledem k tomu, že ukládání preferencí od těchto programů je nežádoucí (přinejmenším proto, že neodpovídají definici uživatele ***viz 2.1.4***), je třeba na

straně serveru rozlišovat mezi „normálním“ uživatelem a crawlerem, což může být netriviální problém. Naproti tomu při použití client-side skriptování tento problém odpadá, neboť crawlery nepodporují žádné client-side skriptovací jazyky.



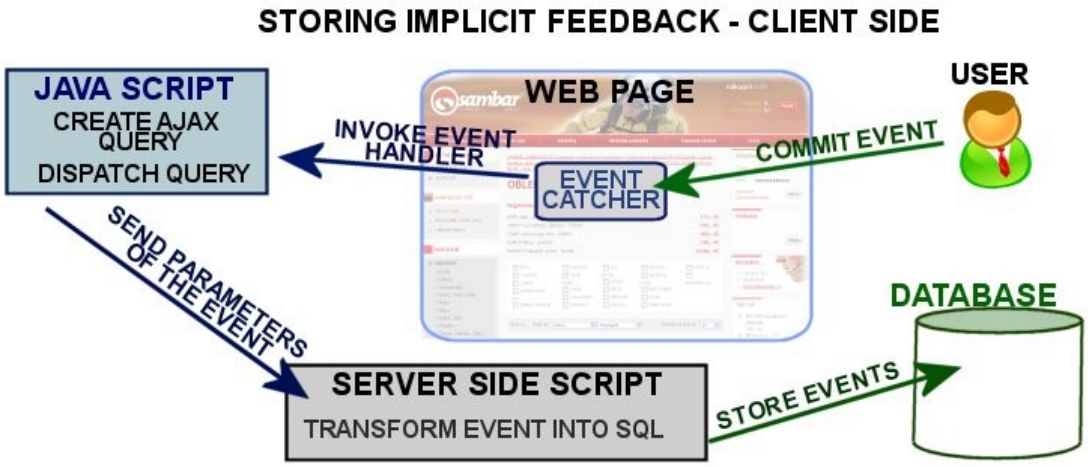
Obrázek 7: Diagram zpracování implicitní zpětné vazby na straně serveru

- **akce ke kterým je nutné použití client-side skriptovacího jazyku** (například JavaScript) **[Wiki10g]** jsou akce, které mohou být vyvolány pouze určitou činností uživatele na dané stránce (kliknutí na odkaz, kopírování části textu, uložení do záložek atd.)

  Sledované uživatelovy akce jsou zachyceny pomocí nastavených DOM událostí (viz *www.w3schools.com/jsref/dom_obj_event.asp*) a zpracovány příslušným client-side skriptem – ten obvykle na základě události vygeneruje vhodnou reprezentaci preference uživatele a asynchronně pomocí AJAXu vyvolá server-side skript, který provede uložení preference do databáze.

  Mezi největší problémy client-side skriptování obecně patří vysoká závislost na uživateli (potažmo jeho prohlížeči). Uživatel může client-side skriptování zcela zakázat, jednotlivé prohlížeče pak kvůli odchylkám od standardu DOM nabízejí různé sady DOM událostí, které je možné zachytit (viz *http://help.dottoro.com/larrqqck.php*). Při návrhu relevantních implicitních akcí je proto třeba brát v úvahu jen ty DOM události, které jsou mezi prohlížeči dostatečně široce podporovány.

  I přes popsané problémy se použití client-side zpracování implicitních zpětných vazeb jeví jako nezbytné – především kvůli omezenému množství zpětných vazeb, které lze zpracovávat na straně serveru. Oba postupy je nicméně možné kombinovat.



Obrázek 8: Diagram zpracování implicitní zpětné vazby na straně klienta

2.4 Časový vývoj preferencí uživatele

Uživatelova preference k objektu je v průběhu času nestabilní. Důvody změny preferencí je možné

rozdělit do dvou kategorií:

- **Změna situace u uživatele:** například ztráta práce uživatele může ovlivnit ochotu utrácet, změna nálady může ovlivnit hodnocení objektů, skrytá vada může zhoršit hodnocení dříve kladně vnímaného objektu atp.
- **Změna situace na trhu:** zlevnění výrobku může zvýšit jeho preferenci, starší výrobky budou pravděpodobně ztrácet preferenci, s vývojem nových výrobků se mění i vývoj preferovaných hodnot atributů (zvětšení kapacity HDD u PC, snižování spotřeby paliva u automobilů...) i důležitost jednotlivých atributů.

Dá se tedy předpokládat, že význam či spíše konfidence uživatelovy preference (ať už implicitní, explicitní nebo přímé) zachycené v čase t bude s přibývajícím časem klesat.

Dále předpokládáme, že vývoj krátkodobé preference uživatele (za předpokladu, že v průběhu času se nezměnil cíl uživatele) budou ovlivňovat jak změny na trhu, tak změny uživatele, kdežto vývoj dlouhodobé preference by měly ovlivňovat pouze změny u uživatele.

Závěry pro doporučovací systémy

Chceme-li v doporučovacím systému brát v úvahu změnu uživatelovy zpětné vazby v čase, můžeme buď (pokud doporučovací systém uvažuje pojmy konfidence, nebo význam) postupně snižovat hodnoty konfidence či významu zpětné vazby, nebo – pokud doporučovací systém tyto vlastnosti preferencí neuvažuje – můžeme zaznamenané zpětné vazby po určitém časovém úseku smazat.

Rychlost snižování hodnot či mazání musí stanovit na základě znalosti domény doménový expert, případně se můžeme snažit vyvodit ze současného chování uživatele závěry týkající se jeho dřívějších preferencí (pokud se uživatel nyní zajímá o jiný segment domény, než dříve, může to znamenat snížení jeho preference k předchozímu segmentu atp.).

První možnost je vcelku jasně definovaná a implementovatelná, naproti tomu druhá varianta vyžaduje další zkoumání. Implementace v UPComp ***viz 5.7.8*** .

3 Implicitní uživatelské data a prodejní weby

V následující kapitole se budeme zabývat především různými aspekty sběru, zpracování a ukládání implicitních zpětných vazeb v prodejních webech, dále pak jejich interpretací a využitím mimo doporučovací systémy.

Kapitoly **3.1**, **3.2** a **3.6** obsahují především souhrn informací z použité literatury či obecně známá fakta, kapitoly **3.3 – 3.5** a **3.7** pak především vlastní poznatky autora.

3.1 Co jsou implicitní uživatelská data a implicitní preference?

Implicitní uživatelská data

Implicitní zpětné vazby jsou takové informace, které jsou získávány na základě chování uživatele na webu bez aktivní účasti uživatele na jejich podání. Mezi implicitní zpětnou vazbu můžeme zařadit například otevření stránky, přehrání hudební skladby, kliknutí na odkaz atd.

Důležitým aspektem implicitní zpětné vazby je její doménová závislost – pro různé domény má smysl uvažovat různé implicitní uživatelské akce (například přehrání hudební skladby u hudebního doporučovače). Keely a Teevan **[KT03]** se pokusili rozdělit pozorovatelné implicitní akce jednak podle rozsahu na akce spojené s částí objektu, s objektem a se třídou objektů, a jednak podle druhu činnosti na průzkum, uložení, odkazování, anotace a vytváření.

	Část objektu	Celý objekt	Třída objektů
Průzkum	**Zobrazit**, **Poslechnout**, **Vyhledávat**, **Scrollovat přes**, **Dotazovat se na**	**Vybrat**	**Prohlížet**
Ukládání	**Tisk**,	**Přidat do záložek**, **Uložit**, Smazat, **Koupit**, **Poslat e-mailem**	**Odebírat novinky**
Odkazování	**Copy/paste**, Citovat	**Přeposlat** Odpovědět Odkazovat na Citovat	
Anotace	**Označit**	**Ohodnotit**, Publikovat	**Setřídit**
Tvorba	Napsat, Upravit	Být autorem	

*Tabulka 1: Možné implicitní uživatelské akce rozdělené podle typu akce a minimálního rozsahu. Tučně jsou vyznačeny nejzajímavější akce podle autora diplomové práce, zdroj **[KT03]***

Tabulka není úplným výčtem možných implicitních akcí (neobsahuje například zmíněné přehrání skladby nebo vložení objektu do košíku) a některé popsané akce jsou v současné době jen obtížně zjistitelné. Nicméně tabulka obsahuje ucelený přehled doménově relativně nezávislých implicitních faktorů.

3.2 Výhody a nevýhody použití implicitních uživatelských dat

Výhodami a nevýhodami použití implicitních uživatelských dat – především ve srovnání s explicitními daty – se zabývalo mnoho studií, například **[JSK10b, CLWB01, WJR01]**, částečně i **[HKV08]**.

Nejčastěji zmiňovanou výhodou implicitních dat oproti explicitním je poměr získatelného množství implicitních a explicitních dat, který je výrazně ve prospěch implicitních dat. Například Jawaheer, Szomszor a Kostkova ve své práci (charakterizace explicitních dat v hudebním doporučovacím systému) **[JSK10a]** zmiňují, že u 16394 náhodně vybraných uživatelů hudebního doporučovacího systému vyřadili z dalšího zkoumání 6382 (tedy více než třetinu) takových uživatelů, kteří neposkytli žádnou explicitní zpětnou vazbu.

Mezi zmiňované problémy, které užívání implicitních dat přináší řadí autoři velmi často:

- neexistenci či obtížné definování negativní zpětné vazby a tedy i negativní preference uživatele
- nejednoznačnost interpretace implicitních dat **(rozebrána v *kapitole 3.6*)** a s ní související
- nutnost použít jiné vyhodnocovací algoritmy, než u explicitních dat

I přes zmíněné problémy je pro mnoho prodejních webů použití implicitních preferencí klíčové. Toto platí především pro weby které:

- mají vysoký poměr #objektů / #uživatelů

- mají vysokou fluktuaci objektů (stávající objekty jsou rychle nahrazovány novými)
- nemohou či nechtějí uživatele motivovat k poskytnutí explicitní zpětné vazby

3.3 Implicitní data v prodejních webech

Pro správné fungování jakéhokoli doporučovacího systému nad implicitními daty je klíčový návrh „implicitních faktorů" tj. událostí, na základě kterých jsou počítány preference uživatele k dalším objektům. Tyto faktory či události jsou doménově závislé, při jejich návrhu by měl doménový expert zvážit především:

- **četnost událostí** (příliš nízká četnost není příliš zajímavá pro výpočet preferencí, příliš vysoká četnost může vést k problémům se zpracováním a ukládáním)
- **jasný vliv na preferenci uživatele** – buď očekávaný nebo přímo otestovaný (*například počet přehrání hudební skladby nebo vyplnění objednávky zájezdu má očekávaný pozitivní vliv na preferenci uživatele k danému objektu*)
- **rozumná složitost zjištění události** (*zde rozumíme především podporu pro zjištění události napříč internetovými prohlížeči*)

I přes značnou doménovou závislost implicitních dat lze identifikovat několik implicitních faktorů, které jsou společné pro většinu webů. Identifikací takových faktorů (na základě porovnání s explicitním hodnocením) se zabýval ve své diplomové práci Vladimír Žák **[Zak10],** který využívá mimo jiné zdroje z **[CLWB01].** V rámci této diplomové práce byla také provedena vlastní studie implicitních faktorů „čas strávený na stránce" a „počet akcí na stránce" ***viz 3.7*** . V následující tabulce se pokusíme provést jejich sumarizaci:

Faktor	Vypovídací hodnota	Poznámky
Čas strávený na stránce	dobrá	Čas na stránce byl vyhodnocen jako dobrý implicitní faktor jak Žákem, tak i vlastním výzkumem
Počet akcí na stránce	dobrá	Počet akcí dosáhl ve ***3.7*** nejvyšší korelace s preferencí uživatele
Čas pohybu myši	špatná	Vliv času pohybu myši na explicitní hodnocení objektu se nepotvrdil (dokáže ovšem odlišit nejhůře hodnocené objekty od ostatních – *je tedy možné, že by se uplatnila jako složka komplexnějšího implicitního faktoru*).
Počet kliknutí myší	špatná	Vliv počtu kliknutí na stránce na explicitní hodnocení objektu se nepotvrdil – *je však možné, že by se uplatnil jako složka komplexnějšího implicitního faktoru – **viz 3.7*** **.**
Scrollování	dobrá	Scrollování bylo identifikováno Žákem jako dobrý implicitní faktor

*Tabulka 2: Implicitní faktory a jejich důležitost - kombinace zdrojů **[Zak10]**, **[CLWB01]** a **3.7.***

Dalšími možnými implicitními faktory pro prodejní weby jsou:

- objednávka objektu
- vložení objektu do košíku
- otevření objektu ze seznamu objektů
- odeslání příspěvku do diskuze k objektu aj.

U těchto faktorů bude třeba provést další výzkum, aby se potvrdil jejich vliv na uživatelskou preferenci, nebo vztah k explicitnímu hodnocení objektu.

3.4 Využití implicitních dat mimo doporučovací systémy

Kromě přímého doporučování uživatelům mohou být implicitní data využita také k dlouhodobým analýzám chování uživatelů v rámci prodejního webu. Z pohledu provozovatele můžou být zajímavé odpovědi například na tyto otázky:

- **Které ovládací prvky jsou uživateli nejvíce používány?**
- **Které ovládací prvky přinášejí nevhodné či neočekávané výsledky?**
- **Jsou preferované objekty pro uživatele zajímavé?**

V rámci přípravy této diplomové práce byl proveden sběr implicitních dat uživatelů webu www.slantour.cz ***(viz příloha A - Obsah CD)***, za účelem zodpovězení otázek/hypotéz nastíněných výše. Ke sběru dat byl použit upravený software od Tomáše Dvořáka. Oproti původnímu řešení, které jako úložiště používalo textový log soubor (což sice nečinilo žádné nároky na přítomnost databáze na zkoumaném webu, nicméně bylo méně pružné co do použití) byla použita MySQL databáze. Navíc bylo ke zjišťovaným atributům implicitních akcí přidáno číslo session uživatele.

Název sloupce	Typ hodnoty	Popis
id_action	Integer, autoincrement	Jednoznačný identifikátor uživatelské akce
user	Integer	Identifikátor uživatele
session_no	Integer	Pořadí session uživatele, ve které došlo k akci. Jedna session = jedna návštěva uživatel. Návštěvy jsou spravovány přímo PHP – pole $_SESSION. Návštěva je obvykle ukončena, pokud uživatel 30 minut nezobrazí žádnou stránku.
datetime	Datetime	Datum a čas zaznamenání uživatelské akce
action	String	Název akce
action_parameter	String	Parametr akce
object	String	Identifikace objektu

Tabulka 3: Přehled sloupců tabulky h0 - tabulka uživatelských zpětných vazeb od uživatelů CK SLAN tour

Sběr uživatelských dat byl prováděn od 23.4. 2010 do 16.5. 2010. Od cca 29000 unikátních uživatelů bylo zaznamenáno celkem cca 384000 implicitních událostí následujících typů.

Název akce	Parametr akce	Popis
PAGE_OPEN	Url odkazující stránky	Otevření stránky
PAGE_CLOSE	---	Zavření stránky
ON_PAGE	Prvek na stránce	Uživatel přejel myší přes určitý prvek na stránce
HREF_CLICK	Url cíle	Uživatel kliknul na odkaz
OBJECT_INTERACTION	Typ interakce	Uživatel vyplnil objednávku, předběžnou poptávku nebo dotaz k zájezdu

Tabulka 4: Přehled hodnot parametrů action a action_parameter tabulky h0

Přesné znění zkoumaných hypotéz a výsledky testování obsahuje ***kapitola 3.7*** **.** Zmíněný soubor dat byl také použit v diplomové práci Vladimíra Žáka **[Zak10]** v části „4: Implementace vybraných metod“.

3.5 Skladování implicitních dat

Jedním z možných problémů při používání implicitních preferencí, který není příliš zmiňovaný, je problematika skladování implicitních preferencí a jejich formátu obecně. Základní požadovanou vlastností pro skladování implicitních dat by měla být především dlouhodobá udržitelnost rychlosti výpočtu a udržitelnost „rozumné“ velikosti objemu dat („rozumná“ velikost dat je závislá především na podmínkách webhostingu jednotlivých webů).

Pro využití implicitních preferencí pro „on-line“ doporučování (doporučování na základě aktuálního požadavku uživatele) je třeba použít nějaký systém pro řízení báze dat (dále jen databáze). Například sběr uživatelských dat popsaný v minulé kapitole problém ukládání dat velmi dobře nastínil: během pouhých 3 týdnů běhu vygeneroval systém téměř 400000 položek – celkový objem surových dat (bez indexů nad daty) byl 52MB. Při předpokladu konstantního toku uživatelských akcí by objem implicitních dat již cca za 9 měsíců přesáhl velikost 2GB, což je obvyklý prostor, který webhostingoví poskytovatelé vyčleňují pro 1 web (*navíc se obvykle jedná o sdílený prostor pro databázi i další obsah webu, zanedbáváme velikost indexů nad daty a nehledíme ani na rychlost práce s podobně objemnou databází*).

Pro dlouhodobé používání systému pro sběr implicitních dat na prodejních webech je vhodné zvážit následující možnosti optimalizace:

- **Optimalizace ukládané informace**: ukládat pouze nutné informace o implicitní události, v co nejvíce zhuštěné podobě (zvážit, zda je třeba ukládat například čas získání informace...)
- **Agregace informací**: zvážit, jaké informace je třeba ukládat jednotlivě, a které je možno agregovat (například neukládat jednotlivé pohyby myši na stránce, ale jejich celkový čas, případně uložit informaci teprve po dosažení určité kritické hodnoty).

  Některé informace je možné také agregovat nejen podle unikátní dvojce uživatel-objekt, ale pouze pro unikátní objekty (nepočítat počet zobrazení objektu uživatelem, ale pouze počet zobrazení objektu jako takového). Tato agregace již však vede k významným ztrátám informace, a proto by měla být použita jen pro data, která není možné či vhodné skladovat separovaně pro jednotlivé uživatele.
- **Mazání starých uživatelských dat:** v ***kapitole 2.4*** jsme nastínili, že konfidence či váha jednotlivých uživatelských dat v čase obvykle klesá. Zdá se proto vhodné, aby dostatečně stará data byla z databáze zcela odstraněna. Tím bychom dosáhli i možnosti omezení horní hranice objemu dat a tedy složitosti výpočtů nad nimi. Postupů pro výběr dat je více, nastíníme několik základních variant:
  - Informace starší než určité datum
  - Informace od uživatelů, kteří byli dlouho neaktivní
  - Informace s nejnižším ID (pokud neukládáme datum získání dat)
  - Informace od uživatelů, kteří odeslali nejméně dat (a pravděpodobně se na web již nevrátí)
  - Informace s nejmenší vypovídací hodnotou (mazat spíše zobrazení objektu, než jeho objednávky)

3.6 Interpretace implicitních dat (modely uživatelských preferencí)

Na rozdíl od explicitních uživatelských dat, kde je většinou sémantika dat dobře definovatelná, může být obtížné interpretovat, jaký je vztah mezi získanými implicitními daty a preferencí uživatele (tedy jaký je model uživatelských preferencí). Následující tři studie představují různé pohledy na interpretaci implicitních uživatelských dat.

3.6.1 Studie o implicitních datech

Hu, Koren a Volinsky se zabývají využitím implicitních uživatelských dat ze sledování digitálních TV pro doporučování dalších zajímavých pořadů **[HKV08].** Autoři navrhli model preferencí uživatele k neznámému objektu s binární proměnnou preference p_{uo} a proměnnou konfidence c_{uo} z intervalu [0,1]. $p_{uo} = 0$ pokud uživatel u objekt o nikdy nezobrazil, jinak =1

c_{uo} roste, se získáváním dalších „důkazů" o pozitivní preferenci uživatele k objektu (více zobrazení atp.). *Idea tohoto modelu byla použita při návrhu komponenty u výpočtu podobností uživatelů a objektů metodou Standard* ***viz 4.5.2*** .

Jawaheer, Szomszor a Kostkova srovnávají ve své studii explicitní a implicitní zpětné vazby v hudebním doporučovacím systému last.fm **[JSK10b]** (jako implicitní akce uvažují počet přehrání skladby, jako explicitní tak ohodnocení skladby jako „like" a „dislike"). Autoři zmiňují, že explicitní uživatelská data jsou obvykle absolutní (mají definovanou horní i dolní hranici možných hodnot), kdežto implicitní data jsou obvykle relativní. Oproti předchozí studii však vyšší hodnoty četnosti implicitních akcí považují za vyšší preferenci, nikoli konfidenci (pokud uživatel U přehraje skladbu A vícekrát oproti skladbě B znamená to, že skladbu A preferuje více). *Upravená idea tohoto modelu byla použita při návrhu komponenty u výpočtu ObjectRating ve většině implementovaných metod* ***viz 4.5.1*** .

Holub a Bieliková [HB10] se zabývali návrhem systému pro doporučování relevantních odkazů na konkrétním webu. V jejich modelu uvažují následující faktory: čas strávený na webové stránce a, počet událostí scrollování. Pro každou webovou stránku spočítají průměrnou hodnotu obou vlastností a definují, že uživatelova preference k objektu o je:

Pozitivní, pokud hodnoty vlastností u současného uživatele jsou o více než X% vyšší než průměrné,

Negativní, pokud jsou o více než X% nižší než průměrné,

Neutrální, pokud neplatí výše zmíněné.

3.6.2 Závěr pro doporučovací systémy

Porovnání zmíněných (a dalších) modelů uživatelských preferencí je velmi obtížné, neboť každý byl testován (a také navržen) na jiné doméně. Doporučovací systém by měl ideálně nabízet výpočet na základě různých modelů uživatelských preferencí – má-li za cíl pokrýt široké spektrum webů (*zde předpokládáme, že pro různé domény objektů budou vhodnější různé modely uživatelských preferencí a doporučovací systém umožní některý z nich zvolit*), případně vycházet z dostatečně ověřených závěrů o vhodnosti konkrétního modelu pro uvažovanou doménu.

3.7 Hypotézy

V této kapitole nastíníme několik hypotéz o chování uživatelů, ze kterých intuitivně vychází většina provozovatelů prodejních webů. Uvažované hypotézy byly otestovány na základě implicitních dat získaných ze sledování uživatelů CK SLAN tour popsaných v ***kapitole 3.4*** **.**

Jako **vyjádření preference** uživatele k objektu považujeme vyplnění objednávky objektu (akce ORDER).

Jako **počet akcí na stránce** uvažujeme počet přejetí myší přes objekt (akce ON_PAGE) + c* počet kliknutí na odkaz na stránce (akce HREF_CLICK), kde c = (celkový počet ON_PAGE akcí)/ (celkový počet HREF_CLICK akcí) = 3.2.

3.7.1 Stanovení hypotéz a jejich motivace

1. **S rostoucím časem na stránce objektu roste preference uživatele k objektu.**
2. **S rostoucím počtem akcí na stránce objektu roste preference uživatele k objektu.**

První dvě hypotézy mají za cíl stanovit, zda čas strávený na stránce produktu, případně to, že byl čas strávený „aktivně“, je dobrým implicitním faktorem. Tento fakt již potvrdily další studie nad jinými daty (například **[CLWB01]**), my jej přidáváme spíše pro potvrzení jejich experimentů.

3. **S rostoucím počtem návštěv detailu objektu od konkrétního uživatele roste jeho preference k objektu.**

Cílem této hypotézy je zkoumat podloženost použití ovládacího prvku „last visited“ - tedy doporučování dříve zobrazených objektů.

4. **Průměrná preference uživatele na objektu roste s počtem zobrazení objektu.**
5. **Průměrná preference uživatele na objektu roste s počtem jeho objednávek.**

Tyto dvě hypotézy zkoumají podloženost ovládacího prvku „best selling“ či „top-viewed“ - tedy zda je vhodné doporučovat všem uživatelům objekty, které jsou podle určitého kritéria nejlepší.

3.7.2 Testování hypotéz

Pro testování zmíněných hypotéz byly vybrána pouze data, která se týkají konkrétního objektu (zájezdu). Data získaná ze stránek katalogu zájezdů aj. mohou být použita pro ověřování dalších hypotéz.

Vztah mezi časem na stránce s detailem objektu / počtem akcí na stránce s detailem a preferencí uživatele k danému objektu:

Pro testování první a druhé hypotézy byly data upraveny do následující podoby:

Název sloupce	Hodnota
user	Id uživatele
object	Identifikace objektu
time/events	Celkový čas, který uživatel strávil prohlížením detailu objektu / počet akcí, které udělal na stránce s detailem objektu
order	Informace, zda a jakou objednávku uživatel vyplnil: 0 = žádná, 1 = dotaz k zájezdu, 2 = předběžná poptávka, 3 = objednávka

Tabulka 5: Přehled sloupců tabulek h1 a h2 - dat pro ověření hypotéz 1 a 2

Časy na stránce, počty událostí a objednávky byly následně ještě normalizovány do intervalu [0,1]. Na výsledných datech byl pak proveden výpočet Pearsonovy korelace.

U porovnávání času a preference uživatele byla zjištěna mírná pozitivní korelace (**cor = 0.12047**), která je však již v 95% intervalu spolehlivosti hypotézy, že korelace mezi časem stráveným na stránce a preferencí uživatele je > 0 (p-value: < 2.2e-16).

U porovnání počtu akcí na stránce a preferencí uživatele byla zjištěna výraznější pozitivní korelace,

než u času (**cor = 0.2355461**), která je také v 95% intervalu spolehlivosti hypotézy, že korelace mezi počtem akcí na stránce a preferencí uživatele je > 0 (p-value: < 2.2e-16).

Vztah mezi počtem návštěv detailu objektu a preferencí uživatele k danému objektu:

Pro testování třetí hypotézy byly data upraveny do následující podoby:

Název sloupce	Hodnota
user	Id uživatele
object	Identifikace objektu
no_of_visits	Celkový počet zobrazení objektu uživatelem
order	Informace, zda a jakou objednávku uživatel vyplnil: 0 = žádná, 1 = dotaz k zájezdu, 2 = předběžná poptávka, 3 = objednávka

Tabulka 6: Přehled sloupců tabulky h3 - dat pro ověření hypotézy 3

Stejně jako v minulém případě byla data ještě normalizována do intervalu [0,1] a na výsledku proveden výpočet Pearsonovy korelace.

I v případě vztahu mezi počtem zobrazení detailu objektu a preferencí uživatele byla zjištěna pozitivní korelace (**cor = 0.1686902**), která je v 95% intervalu spolehlivosti hypotézy, že korelace mezi počtem zobrazení detailu objektu a preferencí uživatele je > 0 (p-value: < 2.2e-16).

Vztah mezi počtem zobrazení, počtem objednávek a průměrnou preferencí uživatele k objektu

Čtvrtá a pátá hypotéza se v podstatě snaží najít odpověď na otázku, zda při rostoucím počtu zobrazení (nebo počtu objednávek) roste také poměr (#objednávek / #uživatelů, kteří s objektem nějak interagovali). Zde předpokládáme že absence uživatelské zpětné vazby není projevem negativní preference.

Získané data jsme agregovali podle objektů následovně:

Název sloupce	Hodnota
object	Identifikace objektu
visits	Počet zobrazení detailu objektu
users	Počet uživatelů, kteří s objektem nějak interagovali (existuje nějaká zpětná vazba uživatele k tomuto objektu)
order	Počet objednávek, které uživatelé k objektu vyplnili (každá objednávka je vždy násobena svou váhou: 1 = dotaz k zájezdu, 2 = předběžná poptávka, 3 = objednávka)

Tabulka 7: Přehled sloupců tabulky h4 - dat pro ověření hypotéz 4,5 a 6

Po dalších úvahách jsme doplnili ještě 6. hypotézu obdobnou předchozím dvěma: **Průměrná preference uživatele na objektu roste s počtem uživatelů, kteří s objektem interagují.**

Pro jednotlivé hypotézy pak byla zkoumána Pearsonova korelace dvojic hodnot (hodnoty byly vždy normalizovány do intervalu [0,1]:

- 4. hypotéza: visits, (order/users)
- 5. hypotéza: order, (order/users)

- 6. hypotéza: users, (order/users)

Vztah počtu objednávek a poměru #objednávek/#uživatelů je mírně pozitivně korelovaný (**cor = 0.1378302**), navíc hodnota korelace se nachází v 95% intervalu spolehlivosti hypotézy, že korelace je > 0 (p-value:8.402e-07).

Naproti tomu vztahy mezi počtem uživatelů, počtem návštěv a poměrem #objednávek/#uživatelů statisticky významné zjištění nepřinesly. Oba jevy se zdají být mírně negativně korelované (**cor = -0.04507224** u návštěv a **-0.05355621** u uživatelů), nemůžeme však zamítnout hypotézu, že hodnota korelace = 0 (p-value: 0.1189 u návštěv a 0.06387 u uživatelů).

Poznatky pro prodejní weby

Testování hypotéz prokázalo (alespoň pro testovaná data) předpoklady, na kterých jsou postaveny některé často používané prvky uživatelského rozhraní v prodejních webech. Jako vhodné prvky UI se jeví „last visited" - zobrazení nedávno navštívených prvků a „best selling objects" - zobrazení nejlépe prodávaných objektů. Naopak použití „top viewed objects" - zobrazování nejvíce navštěvovaných objektů – spíše nedoporučujeme.

Testováním hypotéz bylo dále potvrzeno, že jak čas strávený na stránce objektu, tak i počet provedených akcí na stránce jsou dobré implicitní faktory pro výpočet preference uživatele. Navíc se zdá, že počet provedených akcí je (alespoň pro testovaná data) lepším implicitním faktorem, než čas strávený na stránce.

4 UPComp: Komponenta pro podporu dotazování v prodejních webech na základě preferencí uživatele

Cílem praktické části diplomové práce je navrhnout komponentu pro podporu dotazování v prodejních webech na základě uživatelské preference (dále jen UPComp). V následující kapitole se budeme zabývat především různými aspekty návrhu takové komponenty.

4.1 Shrnutí předchozích kapitol

Předpokládáme, že cílem uživatele na prodejním webu je nákup objektu určitých vlastností - ***viz 2.1.5*** . Proto jsme definovali preferenci uživatele na objektu jako „ochotu uživatele daný objekt zakoupit" **-** ***2.1.6*** . Uživatel poskytuje prodejnímu webu (resp. doporučovacímu systému na prodejním webu) zpětnou vazbu. Termín zpětná vazba budeme dále zaměňovat s termínem „uživatelská akce" který je v některých případech výstižnější.

Doporučovací systém – ***2.1.10*** pak obvykle obsahuje metodu(y) pro výpočet uživatelské preference, které na základě zpětné vazby (případně i dalších informací) vrací předpokládanou preferencí uživatele k objektu (objektům).

Cílem provozovatele prodejního webu by mělo být zobrazit či navrhnout uživateli především ty objekty, které mají nejvyšší preferenci. Zároveň je důležité, aby struktura a ovládací prvky prodejního webu odpovídaly standardu v dané oblasti – tedy aby uživatele nemátly.

V ***kapitole 4.2*** se budeme zabývat požadavky na zamýšlenou komponentu a jejich zohlednění při návrhu, v ***kapitole 4.3*** naopak požadavky UPComp na prodejní web, ***kapitola 4.4*** pak obsahuje návrh a architekturu komponenty jako takové.

4.2 Požadavky na UPComp a způsob jejich realizace

V této kapitole se budeme zabývat různými možnými požadavky na návrh komponenty jak ze strany uživatelů prodejních webů a jejich provozovatelů, tak i z pohledu použití a dalšího vývoje

komponenty.

4.2.1 Specifikace UPComp

Většina následujících požadavků byla zmíněna již v ***kapitole 1.2*** – dají se chápat jako axiomy návrhu komponenty:

1. UPComp bude působit jako **„middle man“** mezi databází a prodejním webem
2. UPComp bude klást **co možná nejméně restrikcí** na architekturu prodejního webu či jeho zaměření

Vzhledem k tomu, že většina webových aplikací obecně i většina prodejních webů je psána v programovacím jazyku PHP a používá MySQL databázi, měla by být UPComp kvůli kompatibilitě psána také v PHP+MySQL. Měla by ale také umožnit snadnou změnu MySQL za jinou SQL databázi. Na zaměření webu komponenta neklade žádné nároky, krom toho, že musí jít o prodejní web dle ***2.1.2*** .

3. UPComp **bude odpovídat v reálném čase**

Použití komponenty by mělo **co nejvíce odpovídat použití běžné databáze** ve stylu Query → Response. Je třeba, aby metody výpočtu preference byly dostatečně rychlé. Předzpracování uživatelských preferencí sice není v rozporu s tímto požadavkem, muselo by však být odolné vůči průběžnému přísunu nových uživatelských dat. V návrhu UPComp se s předzpracováváním nepočítá, nicméně může být pro vybrané metody doplněno později.

4. UPComp by neměla mít **příliš složité rozhraní** a měla by být **dále snadno rozšiřitelná**

Komponenta by měla být rozšiřitelná především přidáním: **dalších druhů uživatelských dat** a **dalších metod výpočtu uživatelské preference**, případně s využitím částí již napsaných metod. To vede k použití objektovému návrhu komponenty.

5. UPComp by měla **využít současné znalosti** provozovatele prodejního webu

Umožnit provozovateli alespoň část dotazu napsat v SQL.

6. UPComp by **neměla dělat to, co už provozovatel umí sám**

Provozovatel nepochybně umí objekty správně uživateli prezentovat a ví jaké ovládací prvky kde zobrazit – komponenta proto nebude obsahovat uživatelská rozhraní ani prezentaci objektů.

4.2.2 Požadavky od provozovatele prodejních webů

Základními požadavky od provozovatele prodejních webů by se daly charakterizovat jako dobrý poměr „cena/výkon“.

- **Nepříliš složitá instalace**
- **Snadné používání komponenty**
  - Minimální zásahy do stávajícího kódu prodejního webu
  - Komunikace s komponentou pomocí dobře definovaného jednoduchého rozhraní
  - Minimální nároky na znalosti provozovatele

Komponentu je v každém případě třeba do stávávajícího webu připojit. Kromě připojení komponenty (ideálně jedním příkazem) by další změny v stávajícím kódu aplikace měly být ideálně pouze tvorba dotazu do komponenty a zpracování odpovědi na dotaz.

Při instalaci a používání komponenty na prodejním webu by neměla být vyžadována znalost komponenty samotné, ale pouze jejího rozhraní. Požadavek jednoduchosti rozhraní jde obvykle

přímo proti vyjadřovací síle – návrh komponenty by se tedy měla snažit o nalezení rozumného kompromisu. Pro kompletní práci s komponentou by měla postačovat znalost uživatelské dokumentace.

- **Prokazatelně dobré výsledky, nejlépe ihned**

Okamžité podávání dobrých výsledků je u mnoha doporučovacích systémů problém **(viz cold start problem [Wiki10h])**, který je ovšem možné odstranit tím, že systém bude nejprve po určitou dobu pouze sbírat uživatelská data, a pak teprve počítat uživatelskou preferenci. Provozovatel by proto měl mít možnost nainstalovat nejprve sběr uživatelských dat (bez jakékoli změny webu viditelné pro uživatele), a pak teprve nainstalovat výpočet preferencí.

Po nasazení komponenty je vhodné mít nástroj ke zjištění úspěšnosti komponenty respektive použitého výpočetního modelu. Je tedy nutné stanovit kritéria úspěšnosti a získávat data pro její výpočet.

4.2.3 Požadavky od uživatele prodejních webů

- **„Neměnit to, co znám."**

Komponenta by neměla zavádět nové ovládací prvky nebo způsoby prezentace objektů konkrétního prodejního webu. Takové rozhodnutí by mělo zůstat výhradně na provozovateli prodejního webu (komponenta pouze poskytuje objekty, které je třeba prezentovat).

- **„Dostat něco navíc."**

Komponenta bude pro uživatele přínosná především v případě, že mu doporučí zajímavý objekt, na který by jinak pravděpodobně nenarazil.

- **„Nebýt chytřejší než jsem já."**

Poslední požadavek vychází z předpokladu, že uživatel obvykle ví, co chce. Není proto vhodné mu vnucovat cokoli proti jeho vůli, ať už je vyjádřená jakkoli (například měnit uspořádání prvků, měnit podmínky vyhledávání atd.).

4.3 Požadavky UPComp na prodejní web

Pro potřeby UPComp předpokládáme, že prodejní web odpovídá definici z ***2.1.2*** (navíc přidáváme několik požadavků na architekturu webu):

- Prodejní web je naprogramován v jazyku PHP, využívá databázi MySQL.
- Všechny **objekty** prodejního webu **lze jednoznačně identifikovat** pomocí jednoho atributu ID.
- Všechny **uživatele lze jednoznačně identifikovat** pomocí jednoho atributu ID.
- Existuje alespoň 1 tabulka databáze, která obsahuje (mimo jiné) jako řádky všechny ID uživatele a 1 tabulka obsahující všechny ID objektů.
- Na webu existuje „**Katalog objektů**" – tj. výpis seznamu objektů, případně filtrovaný podle některých atributů.
- Ke každému objektu existuje „**Detail objektu**" – webová stránka/stránky prezentující atributy objektu.
- Ke každému objektu existuje „**Objednávka objektu**" – API umožňující nákup objektu (dle povahy webu – například vyplnění kupní smlouvy).

První čtyři požadavky se dotýkají samotné možnosti nasadit UPComp na prodejní web (UPComp je schopna ve vlastní režii udržovat seznam uživatelů, pokud prodejní web žádný neobsahuje). Další

tři požadavky popisují očekávané vlastnosti každého prodejního webu. Jejich nesplnění sice neznemožní nasazení UPComp, nicméně drasticky omezí její použitelnost (například bez detailu objektu může UPComp podávat jen velmi omezené předpovědi).

Následující požadavky již nejsou nezbytné pro fungování UPComp, spíše vymezují, kde je dobré jí použít.

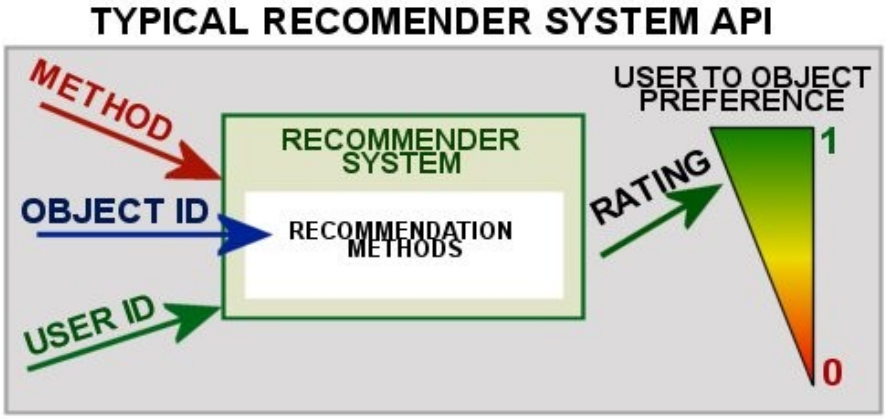
- Předpokládáme, že **prodejní web již byl spuštěn před určitou dobou**, obsahuje ucelenou množinu objektů a je dostatečně navštěvován (nedostatečně navštěvované weby budou trpět tzv. cold start problémem, pro nové weby je vhodné navíc zvážit využití systémů se zabudovanou podporou preferenčního dotazování - například e-shopu od Bronislava Václava ***viz 1.4***).
- Nepředpokládáme, že na prodejním webu již je provozován jiný doporučovací systém (nepředpokládáme konverze či sdílení uživatelských dat atp.).

4.4 Návrh komponenty

V následující kapitole se budeme zabývat návrhem a architekturou UPComp. Nejprve představíme řešení dílčích aspektů a problémů – především těch, které se odlišují od obvyklých postupů při návrhu obdobných systémů, následně představíme celkovou architekturu systému.

4.4.1 Základní rozhraní UPComp

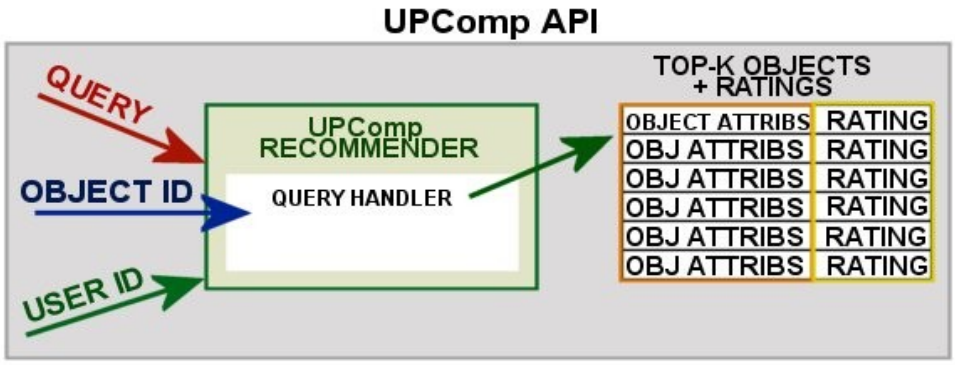
Základní rozhraní (potažmo základní účel) UPComp je poněkud jiný od většiny ostatních doporučovacích systémů. Doporučovací systém obvykle na vstupu očekává objekt a případně metodu výpočtu a jeho výstupem je hodnocení daného objektu specifikovanou metodou výpočtu (viz např. Duine: *http://duineframework.org/)*. Takové rozhraní je dostatečně obecné pro různé úlohy, nechává však mnoho další nutné práce na uživateli.



Obrázek 9: Typické API doporučovacího systému

Při návrhu UPComp vycházíme z předpokladu, že hlavní účel použití doporučovacího systému v prodejních webech je **poskytnout top-k objektů** (tedy k objektů nejlepších podle nějaké metody výpočtu). U běžného doporučovacího systému by za tímto účelem musel uživatel doprogramovat ještě ukládání hodnocení objektů, setřídění a výběr top-k. Navíc musí v každém případě dojít k hodnocení všech uvažovaných objektů.

Proto je základní rozhraní UPComp odlišné: na vstupu očekává počet požadovaných objektů a metodu výpočtu, na výstupu pak vrací seznam top-k objektů a jejich hodnocení. Metody výpočtu preferencí v UPComp tedy vrací pole top-k objektů a jejich hodnocení. *Styl Rozhraní navíc odpovídá dotazu do databáze, nevyžaduje tedy učení příliš mnoha nových věcí od uživatele.*



Obrázek 10: API UPComp

4.4.2 Informace uchovávané systémem

Pro účely výpočtu preferencí je třeba, aby doporučovací systém uchovával zpětné vazby od uživatelů. Zpětné vazby jsme rozdělili na explicitní, implicitní a přímé, kde přímé zpětné vazby jsou v UPComp použity pouze pro tvorbu odpovědi na aktuální dotaz a uchovávány nejsou.

Pro sběr implicitních a explicitních zpětných vazeb jsme zvolili postup popsaný ve ***2.3*** – s použitím AJAXu. Hlavním důvodem byla možnost zaznamenání téměř libovolné uživatelské akce bez nutnosti velkých zásahů do původních zdrojových kódů prodejního webu. Asynchronní zpracování navíc neovlivní (například při chybě) zobrazovanou stránku jako takovou.

Při ukládání implicitní a explicitní zpětné vazby jsme se pokusili stanovit minimum nutných informací o zpětné vazbě a ukládat pak jenom je (vycházíme z doporučení ***2.4***).

- omezili jsme se pouze na zjišťování zpětné vazby uživatele vůči celému objektu (nikoli tedy informace o části objektu, nebo o třídě objektů)
- uchováváme pouze identifikaci uživatele, objektu, druh uživatelské akce a její hodnotu
- jednotlivé druhy uvažovaných uživatelských akcí jsou konfigurovatelné, lze tedy průběžně přidávat nové či je odebírat.

Další možné atributy jako datum a čas zaznamenání akce, či session uživatele byly vyhodnoceny jako málo významné pro výpočet preference.

ID	UserID	ObjectID	EventName	eventValue
113	2425	521	pagewiev	1
114	2425	504	deep_pagewiev	2
115	17	504	order	1

Tabulka 8: příklad obsahu tabulky implicit_events (informace o implicitní zpětná vazbě)

ID	UserID	ObjectID	EventName	eventValue
64	3815	5587	user_rating	0.75
65	3815	5585	user_rating	0.75
67	4457	8413	user_rating	1

Tabulka 9: příklad obsahu tabulky explicit_events (informace o explicitní zpětná vazbě)

Dále dle doporučení ve **3.5** agregujeme některé události (především ty, u kterých se předpokládá vysoká četnost) pouze podle objektu a ukládáme je v separované tabulce.

ObjectID	EventName	eventValue
7539	object_shown_in_list	93
8294	object_opened_from_list	4
8294	object_shown_in_list	232
722	object_ordered	1

Tabulka 10: příklad obsahu tabulky aggregated_events (informace o zpětná vazbě agregované podle objektů)

4.4.3 Postup výpočtu top-k

Postup pro výpočet ideově vychází z článku Alana Eckhardta a Petra Vojtáše **[EV09a].** Autoři navrhli systém kombinující různé metody výpočtu uživatelské preference (využívají různé druhy uživatelských dat). V navrhovaném systému autoři předpokládají použití 4 metod pro výpočet uživatelské preference: Content-based, Collaborative filtering, Clickstream analyse a Direct query. Každá z metod vrací pro zadaný objekt O jeho hodnocení element [0,1].

Výsledné hodnocení objektu je pak dáno váženým průměrem hodnocení od jednotlivých metod, kde váha w je stanovena na základě množstvím dostupných informací v systému.

$$Combined(o) = \frac{w_{CA} * CA(o) + w_{CB} * CB(o) + w_{CF} * CF(o) + w_{DQ} * DQ(o)}{w_{CA} + w_{CB} + w_{CF} + w_{DQ}}$$

V UPComp proto, obdobně jako v popsaném článku, implementujeme jednotlivé metody pro výpočet uživatelských preferencí. Dále je budeme nazývat **dílčí metody výpočtu** uživatelské preference. Výsledná preference objektu je pak dána jejich váženým průměrem. I o částečných metodách předpokládáme, ze vrací na výstupu top-k_i nejlépe hodnocených objektů podle této dílčí metody. Tento závěr byl učiněn především z optimalizačních důvodů, přičemž je však možné nastavit různé (větší) k_i .

Oproti článku je však volba, které dílčí metody budou použity a jaká je jejich váha, ponechána na uživateli (kterému je ale třeba dát dostatek informací o stavu systému pomocí definovaného rozhraní – viz 5.7.7). Navíc uživateli umožníme (v případě, že sémantika metod je 1 „hlavní" metoda s nejvyšší váhou a několik „pomocných" metod), aby pomocné metody uvažovaly při výpočtu top-k_i pouze objekty obsažené v top-k_1 – tedy počítaly preference k objektům vráceným hlavní metodou.

4.4.4 Výpočetní metody v UPComp

V dalším textu se budeme zabývat otázkou, zda je vhodné nějak použité dílčí metody dělit, případně zda je třeba v UPComp i jiných výpočetních metod, než ty, které vrací top-k objektů.

Metody vracející top-k uživatelů

Některé dílčí výpočetní metody vyžadují ke své práci vážený seznam k uživatelů (typický příklad je kolaborativní filtrování). Výpočet top-k uživatelů je možné provést buď v rámci metody, která je používá, nebo zavést samostatné metody pro výpočet top-k uživatelů nezávislé na metodách, které top-k uživatelů používají. Vzhledem k tomu, že první varianta přináší následující problémy:

- přímé spojení metody vracející top-k_j uživatelů s metodou, která je používá.
- nemožnost použít pro získání top-k_j uživatelů více různých metod (jejichž výsledky chceme agregovat).

obsahuje UPComp také metody, které vracejí top-k uživatelů.

Dělení podle vstupních dat

Při otázce možného dělení metod podle vstupních dat jsme jako zajímavé vyhodnotili především závislost metod na znalosti **aktuálního objektu** a **aktuálního uživatele**. Tato závislost má zásadní vliv na možné nasazení metod v rámci prodejního webu. Například metodu závislou na aktuálním objektu můžeme použít pouze na stránce s detailem objektu – jinde není aktuální objekt definován. Naproti tomu metody nezávislé na aktuálním uživateli můžeme použít i pokud se nám nepodařilo uživatele identifikovat.

Návratová hodnota	Závislé pouze na zpětných vazbách	Závislé na ID objektu	Závislé na ID uživatele	Závislé na ID objektu i uživatele
TOP-k objektů	ObjectRating	ObjectSimilarity	Collaborative	
TOP-k uživatelů			UserSimilarity	

Tabulka 11: Typy výpočtu dle návratové hodnoty a vstupních datech

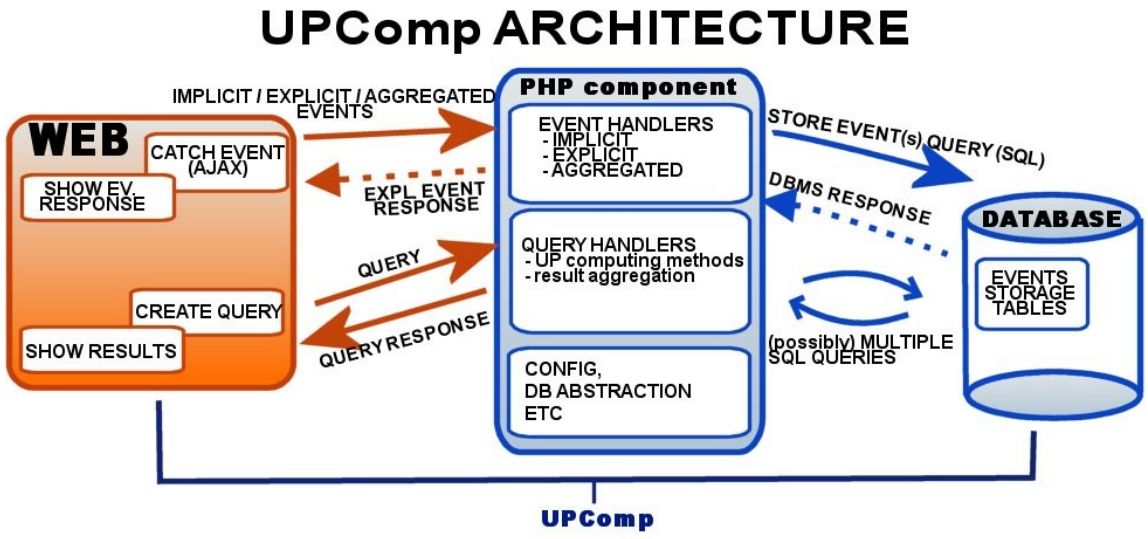
Kombinace vstupních dat a výstupních dat pojmenované v tabulce nazveme **typem metody**. Pole, které zůstaly v tabulce prázdné nejsou dále v diplomové práci uvažovány. Důvodem je především to, že autor nevymyslel (ani žádná dostupná literatura nepopisuje) smysluplné využití metody s takovou kombinací vstupních a výstupních dat. Jednotlivé uvažované typy metod jsou však v UPComp konfigurovatelné a umožňují snadné rozšíření v případě objevení vhodné metody.

Dělení metod dle způsobu výpočtu

Při návrhu UPComp dále vycházíme z předpokladu, že stejný princip (algoritmus) výpočtu je možné použít pro metody závislé i nezávislé na uživatelích či objektech. Zároveň však způsob výpočtu nemusí být vhodný pro všechny tyto kombinace. Především z důvodu přehlednosti pak metody užívající stejný princip výpočtu sdružujeme do tříd. Třídy pak implementují interface jednotlivých typů metod, které obsahují. Konkrétní dílčí metodu (typ výpočtu+algoritmus) pak nazveme jako <algoritmus><typ výpočtu>, např.: **StandardUserSimilarity**, **AggregatedObjectRating** atp.

4.4.5 Celková architektura systému

UPComp se skládá ze 2 funkčně nezávislých celků: **systém pro sběr zpětných vazeb** a **systém pro doporučování** na základě uživatelských preferencí. Vzhledem k řešením, diskutovaným v předchozích kapitolách pak celková architektura systému vypadá následovně:



Obrázek 11: Celková architektura UPComp

4.5 Návrh výpočetních metod

V následující kapitole představíme návrh konkrétních algoritmů použitých v metodách pro výpočet top-k objektů nebo uživatelů. Většina použitých algoritmů (snad s výjimkou StandardUserSimilarity, StandardObjectSimilarity a AttributesObjectSimilarity) jsou obecně známé algoritmy často používané především při zpracování explicitních uživatelských dat. Hlavním problémem návrhu tak bylo vhodně upravit algoritmy tak, aby byly použitelné i pro implicitní či agregované uživatelské data. Předpokládáme, že každá zaznamenaná zpětná vazba (s výjimkou user_rating, *viz 4.5.1*) vyjadřuje pozitivní preferenci uživatele k objektu. Preference má určitou míru, váhu či jistotu (*přesná definice není zásadní pro řešení problému*), která je vyjádřena váhou příslušného typu zpětné vazby (typem zde rozumíme např. otevření stránky, objednávka, zobrazení objektu, hodnocení aj.). Rating objektu – vážený součet příslušných zpětných vazeb - pak používáme na místech, kde se v původních algoritmech vyskytovalo explicitní hodnocení objektu.

Konkrétní metody uvádíme rozřazené podle jednotlivých typů výpočtu.

4.5.1 ObjectRating

- **DummyObjectRating**: výběr k objektů náhodně ze základní množiny.

```
"select distinct `<id_objektu>` from `<tabulka objektů>`
        where `<id_objektu>` in <základní množina objektů> order by RAND() limit <k>"
```

- **StandardObjectRating**: Standard metoda operuje nad implicitními a explicitními zpětnými vazbami. Metoda počítá rating objektů jako vážený součet hodnot zpětných vazeb F uživatele U o objektu o. Výsledkem je pak k-objektů s nejvyšším ratingem. Váha je přiřazována druhům zpětných vazeb buď v rámci volání metody, nebo defaultně. Uvažované druhy zpětných vazeb a základní množina objektů jsou přiřazovány v rámci volání metody. Výpočet ratingu pro jeden objekt na základě implicitních zpětných vazeb vypadá následovně:

$$StandardObjectRating(o) = \sum_{\forall F,U} w_{F(U,o)} * F(U,o)$$

  U explicitní zpětné vazby „user_rating“ je navíc uvažováno, že nízký rating vyjadřuje malou preferenci uživatele k objektu a na ratingu by se tedy měl podílet záporně. Pro user_rating vypadá vzorec pro výpočet následovně:

$$StandardObjectRating(o) = \sum_{\forall F,U} w_{F(U,o)} * 2 * (F(U,o) - \frac{1}{2})$$

- **RandomizedObjectRating**: Randomized metoda používá stejný algoritmus, jako Standard. Jediným rozdílem je, že Randomized omezuje počet uvažovaných zpětných vazeb shora číslem L (nastavitelné v konfiguraci). Překročí-li počet zpětných vazeb L, jsou vybírány náhodně.
- **AggregatedObjectRating**: Aggregated metoda funguje velmi podobně, jako Standard, pouze uvažuje místo implicitních / explicitních pouze agregované zpětné vazby. Výpočet probíhá následovně:

$$AggregatedObjectRating(o) = \sum_{\forall F} w_{F(o)} * F(o)$$

4.5.2 ObjectSimilarity

- **DummyObjectSimilarity:** náhodný výběr k objektů různých od aktuálního ze základní množiny.

```
"select distinct `<id_objektu>` from `<tabulka objektů>`  where `<id_objektu>`  != <aktuální id>
        and `<id_objektu>` in <základní množina objektů> order by RAND() limit <k>"
```

- **StandardObjectSimilarity**: Standard metoda je založena na podobnosti uživatelských zpětných vazeb jednotlivých objektů: Nejprve je vypočítán rating aktuálního objektu pro všechny uživatele U, u kterých existuje nějaká zpětná vazba F k aktuálnímu objektu o – váhy a uvažované druhy zpětných vazeb shodně s StandardObjectRating (jen je hodnocení agregované nejen podle objektu, ale i uživatele):

$$UserObjectRating(U,o)=\sum_{\forall F} w_{F(U,o)} * F(U,o)$$

  Dále jsou stejným způsobem vypočítány ratingy uživatelů U k objektům o_i, ke kterým existuje zpětná vazba.

  Těžiště algoritmu je v předpokladu, že jakýkoli pozitivní rating objektu vyjadřuje pozitivní preferenci uživatele, hodnota ratingu pak značí jistotu či váhu tohoto předpokladu (algoritmus ideově vychází ze článku **[HKV08]**). Výsledný rating podobnosti objektu o_i, a o je pak dán sumou součinu ratingu objektů přes všechny uživatele (podobnost je tedy třeba ještě normalizovat):

$$StandardObjectSimilarity(o_i,o)=\sum_{\forall U} UserObjectRating(U,o) * UserObjectRating(U,o_i)$$

  Stejný algoritmus jako Standard využívá i **RandomizedObjectSimilarity** – obdobně jako v případě ObjectRating však omezuje maximální přípustný počet zpětných vazeb.

- **PearsonCorrelationObjectSimilarity**: PearsonCorrelation metoda postupuje stejně jako Standard při získávání *UserObjectRating*. Následně však počítá Pearsonovu korelaci mezi ratingy jednotlivých uživatelů:

$$PCObjectSimilarity(o_i,o)=cor_{\forall U}(UserObjectRating(U,o), UserObjectRating(U,o_i))$$

- **NrmseObjectSimilarity:** NRMSE metoda postupuje stejně jako Standard při získávání *UserObjectRating*. Následně však počítá normalizovanou střední kvadratickou chybu **[Wiki10i]** objektů:

$$NrmseObjectSimilarity(o_i,o)=NRMSE_{\forall U}(UserObjectRating(U,o), UserObjectRating(U,o_i))$$

- **AttributesObjectSimilarity**: Attributes je od ostatních metod výrazně odlišná. Nevyužívá uživatelskou zpětnou vazbu, ale hledá objekty s podobnými atributy. Pro zadané atributy A_i, jejich váhu w_i, a funkci $Similarity_i$ pro výpočet vzdálenosti mezi hodnotami atributů je pak podobnost objektů definována následovně:

$$AttributesObjectSimilarity(o_j,o)=\frac{\sum w_i * Similarity_i(A_i(o_j), A_i(o))}{\sum_{\forall i} w_i}$$

  V implementaci je výpočet podobnosti převeden do SQL dotazu, funkce Similarity není předávána přímo – pouze její parametry (tolerance aj.).

4.5.3 UserSimilarity

- **DummyUserSimilarity:** náhodný výběr k uživatelů různých od aktuálního ze základní množiny.

```
"select distinct `<id_uživatele>` from `<tabulka uživatelů>`
        where `<id_uživatele>`  != <aktuální uživatel> and
```

`<id_uživatele>` in <základní množina uživatelů> order by RAND() limit <k>"

- **StandardUserSimilarity:** Výpočet UserSimilarity pomocí Standard, NRMSE a PearsonCorrelation metodami probíhá velmi podobně, jako výpočet ObjectSimilarity. Rozdíl je pouze v „prohození“ pojmů uživatel a objekt. Zjištění UserObjectRating probíhá stejně jako v ObjectSimilarity. Výsledný rating podobnosti uživatele U_i, a U je pak dán sumou součinu ratingu objektu o_j od uživatele U_i a U přes všechny objekty, ke kterým existuje zpětná vazba uživatele U.

$$StandardUserSimilarity(U_i, U) = \sum_{\forall j} UserObjectRating(U_i, o_j) * UserObjectRating(U, o_j)$$

- **PearsonCorrelationUserSimilarity:** Výsledný rating podobnosti uživatele U_i, a U je dán Pearsonovu korelací mezi ratingy uživatelů k jednotlivým objektům.

$$PCUserSimilarity(U_i, U) = cor_{\forall o j}(UserObjectRating(U_i, o_j), UserObjectRating(U, o_j))$$

- **NrmseUserSimilarity**:Výsledný rating podobnosti uživatele U_i, a U je dán normalizovanou střední kvadratickou chybou mezi ratingy uživatelů k jednotlivým objektům.

$$NrmseUserSimilarity(U_i, U) = NRMSE_{\forall o j}(UserObjectRating(U_i, o_j), UserObjectRating(U, o_j))$$

4.5.4 Collaborative

- **DummyCollaborative** je shodná s metodou DummyObjectRating
- **StandardCollaborative** je obdoba metody StandardObjectRating. Na rozdíl od ObjectRating však bere v úvahu pouze zpětné vazby F od vybraných uživatelů U_i (výstup některé z UserSimilarity metody), zpětná vazba od uživatele U_i je navíc násobena váhou uživatele w_i (*váha = normalizovaná podobnost uživatele U_i k aktuálnímu uživateli*).

$$StandardCollaborative(o) = \sum_{\forall i, j} w_{Fi} * w_{Uj} * F_i(U_j, o)$$

5 Uživatelská dokumentace

Uživatelská dokumentace je určena především provozovatelům a programátorům prodejních webů – tedy osobám, které budou fakticky UPComp nasazovat na prodejní web. Tomu je přizpůsobena i forma prezentovaného textu.

Pro provozovatele prodejního webu bez programátorských zkušeností jsou určeny především kapitoly **5.1** – **5.4**, ostatní kapitoly jsou určeny především programátorům prodejních webů či osobám, které budou UPComp na daný prodejní web instalovat.

Předpokládá se, že uživatelská dokumentace může být distribuována i samostatně bez ostatních součástí diplomové práce (odkazy na jiná místa v práci by pak byly nahrazeny například za URL).

5.1 Co je to UPComp?

UPComp je nezávislá vyhledávací komponenta (můžeme jí také nazvat doporučovací systém) určená pro prodejní weby (e-shopy, rezervační systémy, aukční servery atp). UPComp Vám umožní zobrazit uživatelům právě ty produkty, které se jim líbí, které hledali, nebo dokonce i ty, o kterých ani nevěděli, že je chtějí najít.

5.2 Jak UPComp funguje?

UPComp identifikuje jednotlivé uživatele Vašeho systému a ukládá si data (uživatelskou zpětnou vazbu) o jejich chování: otevřené objekty, objednávky, chování na stránce atd. UPComp obsahuje metody, které na základě získaných zpětných vazeb zjišťují preference uživatele k ostatním objektům (preference = pravděpodobnost, že objekt uživatele zajímá). Pomocí dotazů do UPComp (obdobných jako SQL dotazy do databáze) můžete vybrané metody aktivovat a získat jako jejich výstup top-k nejlepších objektů (tedy objekty, které budou nejpravděpodobněji uživatele zajímat).

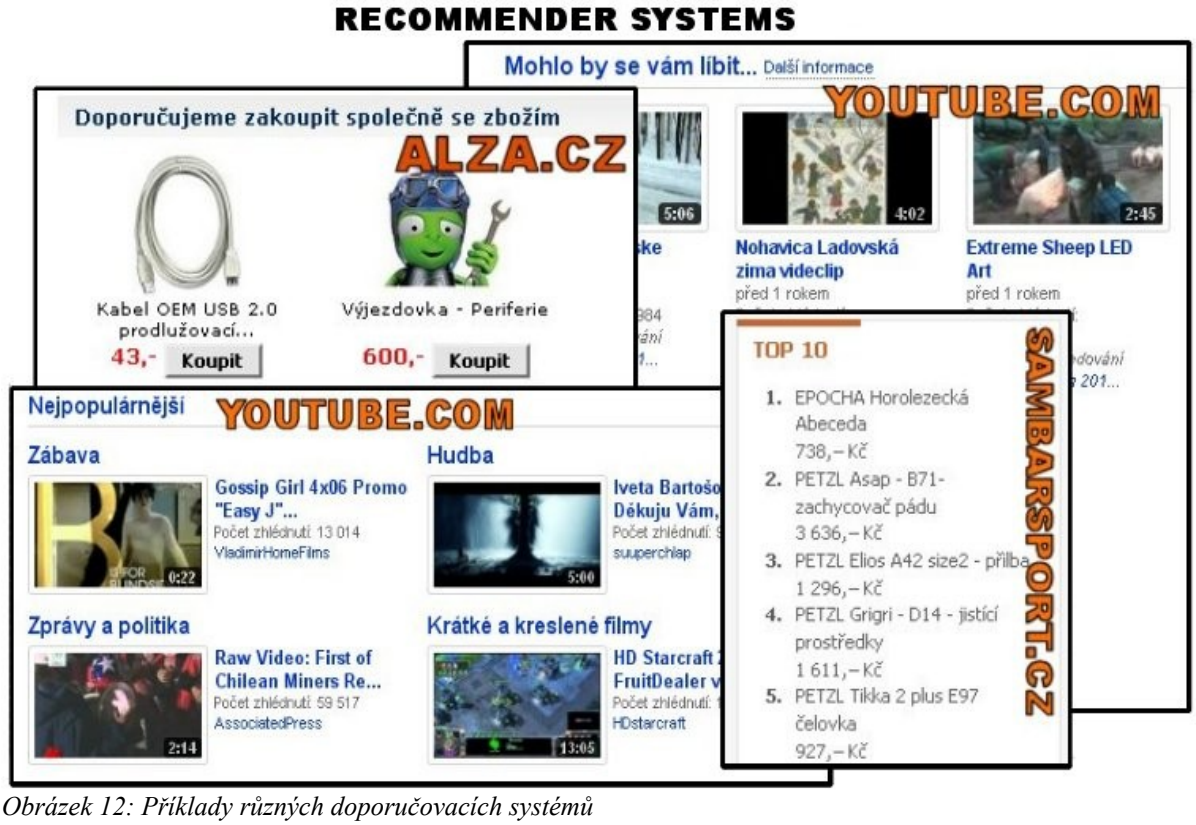
5.3 Co UPComp umí?

UPComp bylo vyvíjeno s požadavkem „řešit pouze to, co provozovatel webu neumí sám“. Nepřináší proto žádné nové uživatelské GUI, nebo jiné způsoby prezentace objektů, neboť si myslíme, že to nejlépe zvládnete Vy - tvůrci daného webu.

UPComp naopak umí dobře 2 věci:

- získávat informace o uživatelském chování
- odpovídat na Vaše dotazy = počítat preferenci uživatele k objektům

Typicky je tedy možné použít UPComp jako sofistikovaný nástroj pro návrh nejrůznějších tipů na objekty, navrhování podobných objektů, nebo na vylepšení Vašeho vyhledávání doplněním uživatelských preferencí.



Obrázek 12: Příklady různých doporučovacích systémů

5.4 Před instalací - požadavky k instalaci UPComp

Před instalací UPComp ověřte, že Váš web splňuje následující požadavky (první 4 požadavky jsou nezbytné pro správný UPComp jako takové, další 3 jsou doporučené – bez nich by UPComp mohla provádět jen velmi omezené předpovědi):

- Váš web je naprogramován v jazyku PHP (UPComp požaduje alespoň PHP5.2.0) a využívá

databázi MySQL

- všechny **objekty** Vašeho webu **lze jednoznačně identifikovat** pomocí jednoho atributu ID
- existuje alespoň jedna tabulka databáze, která obsahuje (mimo jiné) jako řádky všechny ID objektů
- na Vašem webu existuje „**Katalog objektů**“ – tj. výpis seznamu objektů, případně filtrovaný podle některých atributů.
- Ke každému objektu existuje „**Detail objektu**“ – webová stránka/stránky prezentující atributy objektu
- Umožňujete uživateli, aby si objekt **objednal** nebo **zakoupil.**

Výkon většiny metod, které UPComp používá k výpočtu uživatelské preference, je závislý na množství uživatelů Vašeho webu a na množství získaných uživatelských dat. Je tedy lepší (i když ne nezbytné), aby Váš web měl již v době nasazení UPComp stabilní návštěvnost. Instalaci UPComp je možné provést ve 2 fázích - nejprve spustit pouze sběr uživatelských dat a až po získání dostatečného množství zpětných vazeb spustit samotné dotazování. Jako dobrý orientační bod pro spuštění dotazování se jeví stav, kdy má systém více uživatelů, než objektů.

Dále je dobré si ještě před nasazením UPComp vytipovat místa jejího budoucího použití (případně i použité metody – ***viz 5.9***). Obvykle jsou to místa, kde již máte vlastní dotazy do databáze, které slouží k prezentaci top-k objektů (top-10, nejprodávanější produkty v kategorii, podobné objekty, společně nakupované objekty atp.).

Obdobně je vhodné znát i webové stránky (a příslušné zdrojové soubory), ze kterých bude prováděn sběr uživatelských dat. Pro více informací o sběru uživatelských dat ***viz 5.7.1*** **,** pro více informací o dotazování UPComp ***viz 5.7.2*** **.**

V neposlední řadě je třeba si zvolit, co přesně představují objekty systému (*obvykle je tato volba jasná, nicméně v některých specifických případech může být složitější – například otázka, zda jsou stejné parfémy o různém objemu jedním objektem, nebo ne*).

Před začátkem instalace doporučujeme **vytvořit zálohu** současného stavu Vašeho webu – tj. provést export databáze a vytvořit kopii zdrojových souborů.

5.5 Instalace UPComp

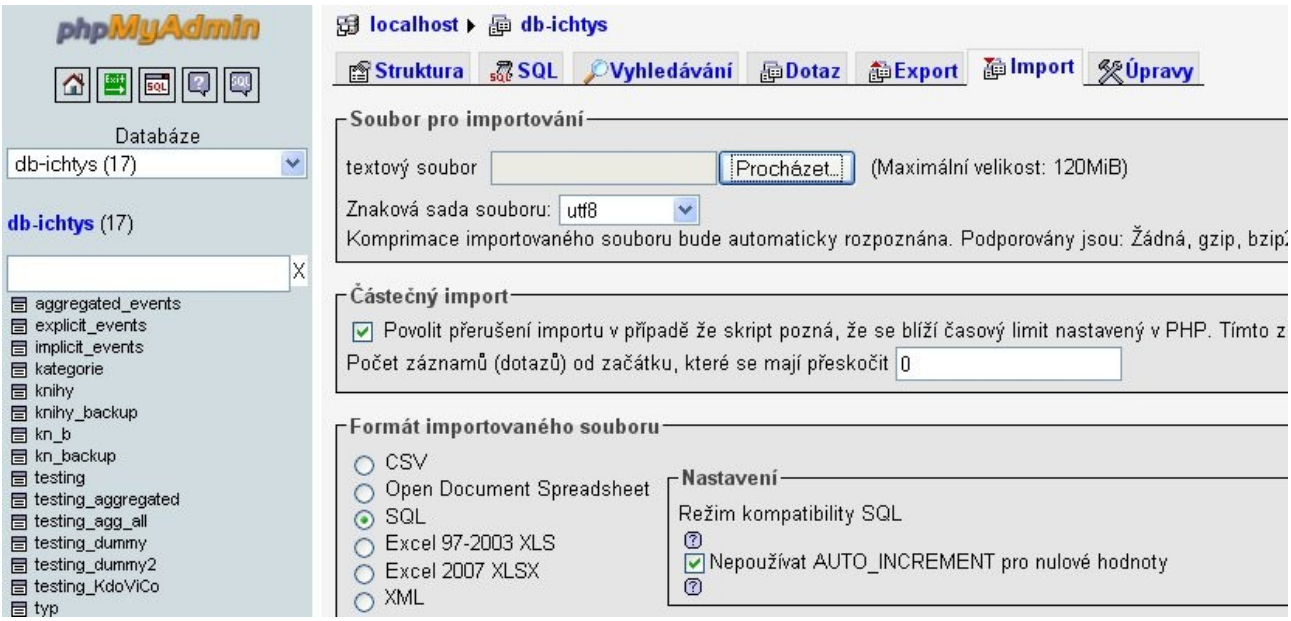
Instalace UPComp probíhá v několika krocích:

1. **Nahrát zdrojové soubory UPComp**

   Do kořenového adresáře Vašeho webu nahrajte složku DP/UPComp z přiloženého CD. Tato složka obsahuje veškeré zdrojové soubory komponenty.

2. **Vytvořit tabulky v MySQL**

   Ve Vaší MySQL databázi spusťte SQL dotazy ze souboru DP/createUPCompDb.sql. Tyto dotazy vytvoří tabulky nutné pro práci UPComp. Dotazy můžete spustit buď jednotlivě, nebo například v PhpMyAdmin pomocí menu „import“. ***Je třeba, aby vytvořené tabulky byly ve stejné databázi, jako tabulky uživatelů a objektů Vašeho webu.***



Obrázek 13: Screenshot - nabídka Import v PhpMyAdmin

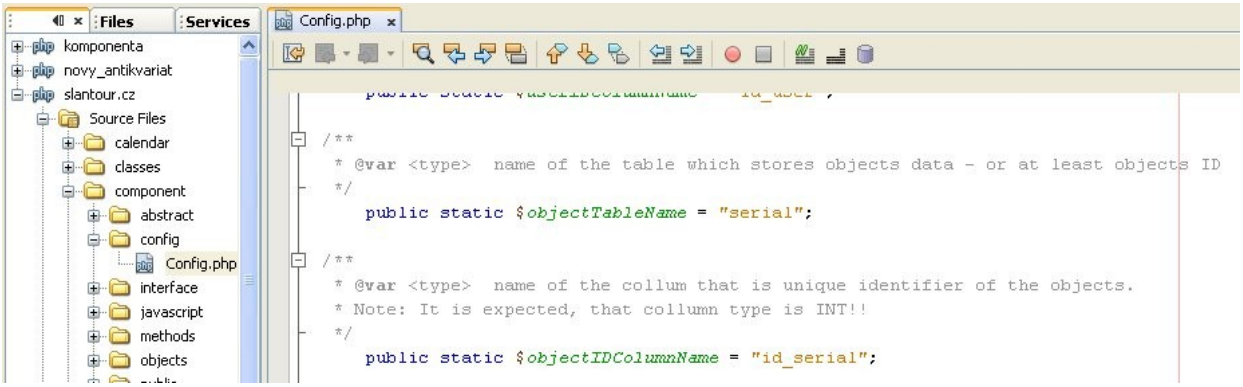
3. Upravit konfigurační soubor

Otevřete ve vhodném editoru php skriptů soubor DP/UPComp/Config/config.php

Zde je třeba nastavit správné hodnoty pro 2 proměnné:

$objectIDColumnName = název sloupce, který jednoznačně identifikuje objekty Vašeho webu (např. „id_object“)

$objectTableName = název tabulky, která obsahuje všechny objekty (např. „objekty“)



Obrázek 14: Screenshot - editace souboru Config.php v Netbeans IDE

Po úpravě soubor uložte a nahrajte do složky /UPComp/Config Vašeho webu.

4. Úprava připojení do databáze

Otevřete soubor DP/UPComp/Objects/MySQLDatabase.php

Zde je třeba nastavit správné hodnoty pro přípojení k Vaší MySQL databázi:

$dbServer = <adresa databázoveho serveru>

$dbName = <přihlašovací jméno k databázi>

$dbPasswd = <heslo k databázi>

$dbNameOfDatabase = <název databáze>

Po úpravě soubor uložte a nahrajte do složky /UPComp/Objects Vašeho webu.

5. **Úprava zdrojových kódů**

   Upravte zdrojové soubory Vašeho webu následujícím způsobem:

   - Na začátek každé webové stránky, u které chcete provádět sběr uživatelských dat nebo doporučování objektů, vložte následující kód:

```
<?php
require_once("./UPComp/public/UPCompCore.php");
UPCompCore::loadCore();
?>
```

   - Do každé stránky, kde chcete provádět sběr uživatelských dat, navíc vložte těsně před tag </head> následující kód:

```
<?php
   UPCompCore::loadJavaScripts($objectID);
?>
```

     $objectID je proměnná obsahující id aktuálního objektu. To je typicky známo, pokud se pohybujeme na stránce s detailem objektu. V ostatních případech zvolte $objectID=0. Zjištění ID objektu je zásadní pro správné fungování sběru uživatelské zpětné vazby. UPComp nemá prostředky pro rozlišení stránky s detailem objektu od ostatních stránek, proto je nutné tuto informaci poskytnout externě.

5. **Spuštění sběru uživatelské zpětné vazby**

   Zpětná vazba je sbírána pomocí aktivace příslušných JavaScriptových funkcí. Ty jsou typicky použity jako zpracování událostí u konkrétních HTML elementů stránky. Například

   `<body onload="objectOpen();">` vyvolá po načtení stránky událost `objectOpen()`, která uloží implicitní zpětnou vazbu typu „uživatel U otevřel objekt O“.

   Kompletní přehled uvažovaných zpětných vazeb a způsobu jejich zaznamenávání naleznete v ***kapitole 5.7.1*** **.**

6. **Spuštění doporučování**

   Po získání dostatečného množství zpětných vazeb můžete přikročit k implementaci doporučování.. Předpokládáme, že budete chtít použít UPComp v místě, kde již máte umístěn SQL dotaz do databáze a prezentaci jeho výsledků.

   Původní kód může vypadat zhruba následovně:

```
$sql = „select <atributy objektu> from <tabulky> where <podminky na objekty> … limit
        <limit dotazu>“
$result = mysql_query($sqlQuery);
if (!$result) {
        <zpracování chyby dotazu>
        die('Invalid query: ' . mysql_error() );
} else {
        while($resultRow = mysql_fetch_array($result)){
                <zpracování prezentace objektu>
        }
```

```
}
```

Příslušný zdrojový kód nahradíte následujícím kódem:

```
$sql = „select <atributy objektu> from <tabulky>
        where <podminky na objekty – pravděpodobně zjednodušené> … limit <limit dotazu>“
$attributes = array(<další sql atributy>);
$userExpr = array(<metody hledající top-k uživatelů>);
$objectExpr = array(<metody hledající top-k objektů>);

$qHandler = new ComplexQueryHandler($sql, $attributes, $userExpr, $objectExpr);
$qHandler->sendQuery();
$result = $qHandler->getQueryResponse();

if (!$result->getQueryState()) {
        <zpracování chyby dotazu>
        die('Invalid query: ' . $result->getDtbMessage());
} else {
        while($resultRow = $result->getNextRow()){
                <zpracování prezentace objektu>
        }
 }
```

Porovnáním zdrojových kódů zjistíte, že hlavní rozdíl mezi dotazováním do MySQL databáze a dotazováním do UPComp je v tvorbě dotazu samotného. Odesílání dotazu a zpracování odpovědi již probíhá velmi podobně ($resultRow je v obou případech pole, které obsahuje jednotlivé atributy objektu indexované podle jména a pořadí – prezentace objektu tedy může zůstat beze změny).

Přesnou syntaxi a sémantiku dotazu do UPComp naleznete v ***kapitole 5.7.2*** **.**

5.6 UPComp „Hello World“

V této kapitole ukážeme jednoduchý příklad zdrojového kódu webové stránky s nainstalovanou UPComp, která získává zpětnou vazbu od uživatele a zobrazuje tipy na další objekty.

Předpokládáme, že jsou správně nastaveny hodnoty v **Config.php**, připojení k databázi a databáze obsahuje tabulku „objects“:

id_object	Name	Price
1	Pastelky	50
2	Pravítko	10
3	Kružítko	85
4	Penál	120
5	Tužka	2

Tabulka 12: obsah tabulky "objects"

Následující kód pak generuje HTML stránku s detailem objektu č. 3.

```
<?php
        require_once("./UPComp/public/UPCompCore.php");
        UPCompCore::loadCore();
?>
<html>
<head>
        <title>Detail objektu Pravítko</title>
<?php
        $objectID = 3;
        UPCompCore::loadJavaScripts($objectID);
?>
</head>
<?php /*získávání informace o otevřeném objektu*/ ?>
<body onload="objectOpen()">
        <h1>Pravítko</h1>
        <div id="cena">10 Kč</div>
        <div id="podobne">
                <h2>Podobné objekty v naší nabídce</h2>
<?php
$sql="select * from objects limit 0,2";
$attributes = "";
$userExpr = "";
$objectExpr = array(
    //ObjectExpression(MethodType, MethodImportance, MethodName, MethodParameters)
    new ObjectExpression("ObjectRating", "1", "Dummy", array("noOfObjects"=>8) )
);
$qHandler = new ComplexQueryHandler($sql, $attributes,$userExpr, $objectExpr);
$qHandler->sendQuery();
$qResponse = $qHandler->getQueryResponse();
if( $qResponse->getQueryState() ) {
    while($qRow = $qResponse->getNextRow()) {
```

```
            echo "<h3>".$qRow["Name"]."</h3>";
            echo "<i> Cena: ".$qRow["Price"]." Kč</i>";
        }
    }
    ?>
    </div>
    </body>
    </html>
```

Obrázek 15: Screensho - Hello World stránka v prohlížeči

5.7 Rozhraní UPComp

5.7.1 Uvažovaná zpětná vazba

V UPComp dělíme uživatelskou zpětnou vazbu do 3 typů:

- **Explicitní zpětná vazba**, kterou nám uživatel poskytl vědomě .
- **Implicitní zpětná vazba**, která je poskytnuta bez úsilí uživatele.
- **Agregovaná zpětná vazba**, která je vlastně implicitní zpětnou vazbou, ale uchováváme u ní pouze informace o objektu – ne o uživateli (především se hodí u velmi četných zpětných vazeb, které by jinak zabíraly příliš mnoho místa v databázi).

Jednotlivé typy zpětné vazby jsou uloženy v separovaných tabulkách databáze. Typy se dále dělí podle toho, jaká konkrétní zpětná vazba byla poskytnuta.

Název zpětné vazby	Popis, implementace
pageview	Otevření stránky s detailem objektu. **Implementace**: <body onload="objectOpen()">
deep_pageview	Důkladné prohlédnutí stránky s detailem objektu (případně víceúrovňové). **Implementace**: buď podle času stráveném na stránce (použití timeru), nebo podle počtu akcí uživatele na stránce (např. přidáním onmouseover/onmouseclick="objectOperation(<váha>)" na klíčové součásti stránky).
order	Objednávka objektu uživatelem (případně víceúrovňové). **Implementace**: po dokončení objednávky vyvolání funkce objectOrder(<váha>).
print	Tisk stránky s detailem objektu
comment	Odeslání komentáře/poznámky k objektu
bookmark	Přidání stránky s detailem objektu do záložek

Tabulka 13: Typy implicitní zpětné vazby

Pro implicitní preference print, comment a bookmark nejsou aktuálně zpracované JavaScriptové funkce pro jejich zjišťování – v případě print a bookmark neexistují standardní DOM události (ale podporují je některé prohlížeče – možno doimplementovat), comment není zcela obvyklým prvkem prodejních webů, jeho implementaci proto nechováme na zvážení.

Název zpětné vazby	Popis, implementace
user_rating	Hodnocení objektu uživatelem. Pro implementaci je možno využít buď kompletní příklad obsažený na přiloženém CD: docs/codeSamples/, nebo použijte funkci objectRate(objID, rateObtained)

Tabulka 14: Typy explicitní zpětné vazby

Název zpětné vazby	Popis, implementace
object_shown_in_list	Objekt byl zobrazen v katalogu objektů (resp. počet událostí tohoto typu): při zobrazení objektu zavolejte funkci objectsShownInList(objectIDs) (jako parametr můžete uvést i více id najednou oddělených čárkami)
object_opened_from_list	Objekt byl z katalogu otevřen (resp. počet událostí tohoto typu) uživatel se tedy na objekt nedostal přímo, ale přes navigaci prodejního webu. Implementace: u odkazu na detail objektu událost onclick="objectOpenedFromList(objectID)"
object_ordered	Objednávka objektu (resp. počet událostí tohoto typu). Implementace shodná jako u implicitní zpětné vazby order

Tabulka 15: Typy agregované zpětné vazby

5.7.2 Dotaz do UPComp

Dotaz do UPComp se skládá ze čtyř částí, každá uložená v separované proměnné. Zpracování dotazu typicky vypadá následovně:

```
$qHandler = new ComplexQueryHandler($sql, $attributes,$userExpr, $objectExpr);
$qHandler->sendQuery();
```

```
$qResponse = $qHandler->getQueryResponse();
```

Výsledkem dotazu je instance třídy **QueryResponse**, třída obsahuje metody

$qResponse->getQueryState(), která informuje, zda byl dotaz úspěšně vykonán a

$qResponse->getNextRow(), která vrací pole odpovídající jednomu řádku výsledku dotazu (stejně jako mysql_fetch_row())

Proměnné $sql, $attributes, $userExpr, $objectExpr mají následující význam:

- **$sql (povinné):** obsahuje kód sql dotazu, který je přímo odeslán do databáze. Dotaz je použit pro definici „základní množiny uvažovaných objektů“ (tedy objekty, u kterých chcete, aby na nich byla vyhodnocena uživatelova preference). Dále slouží $sql k identifikaci, které atributy objektu vyžadujete na výstupu (prvky za „select“ stejně jako při běžném dotazu do databáze) a kolik objektů na výstupu očekáváte. Příklad SQL dotazu:

```
$sql = “select `id`, `name`, `price` from `objects` where `price` < 250 limit 0,5“ ;
```

  - Ve **FROM** části SQL dotazu musí být obsažena tabulka specifikovaná v Config.php v proměnné $objectTableName.
  - Dotaz musí obsahovat část **LIMIT** pro specifikaci o kolik objektů máte zájem

- **$attributes** (volitelné): pole instancí třídy Attribute(). Instance třídy Attribute() je zápis požadovaných vlastností jednoho atributu objektu, jednotlivé atributy se přímo překládají do SQL.

  Celkovou motivací **$attributes** je umožnit setřídit objekty podle kombinace jejich vlastností – ratingu. Například chceme zobrazit uživateli i ty objekty, které sice některou požadovanou vlastnost nemají, ale v ostatních jsou velmi dobré, nebo rozlišit objekty, které jsou v dané vlastnosti jen „přijatelné“ od těch, které jsou v ní „výborné“ (více informací ***viz 2.2.4***).

  Příkladem informace, kterou je možné pomocí třídy Attribute() snadno vyjádřit je: „Nejlepší objekty jsou takové, které mají cenu mezi 1500 a 2000, přípustné jsou ještě ty, které mají cenu od 1250 do 2250. Do výsledku chci zahrnout i objekty, které tuto podmínku nesplňují, ale její priorita je vysoká.“ Informace by se zapsala následujícím způsobem:

```
//IntegerAttribute($name, $valueFrom, $valueTo, $tolerance, $importance);
AttributeFactory::IntegerAttribute("`cena`", 1500, 2000, 250, HI_IMPORTANCE);
```

- **$userExpr** (volitelné): pole instancí třídy UserExpression(). Instance třídy UserExpression() je specifikace jedné metody, která má být použita ke hledání podobných uživatelů. Nalezení uživatelů podobných aktuálnímu uživateli je zásadní pro funkci kolaborativních metod (metody založené na doporučování objektů, které se líbí uživatelům, kteří jsou podobní současnému uživateli – ***viz 4.5***).

- **$objectExpr (volitelné):** pole instancí třídy ObjectExpression(). Instance třídy ObjectExpression() je specifikace jedné metody, která má být použita ke hledání top-k objektů (nejlepších k objektů podle dané metody). Jedná se tedy o hlavní nástroj jak ovlivnit odpověď na Váš dotaz. Je-li použito více metod pro výpočet top-k objektů, jsou jejich výsledky zprůměrovány (vážený průměr podle důležitosti jednotlivých metod). Výsledkem je pak k nejlepších objektů dle váženého průměru. Specifikace metody vypadá následovně:

```
//ObjectExpression($methodType, $importance, $methodName, $methodParameters);
new ObjectExpression("ObjectRating", 1, "Dummy", array("noOfObjects"=>5)  );
```

5.7.3 Rozhraní třídy AttributeFactory

Třída **AttributeFactory** slouží ke tvorbě jednotlivých Atributů UPComp dotazu. UPComp rozeznává následující typy atributů:

- **Numeric** – definuje libovolné číslo. AttributeFactory umožňuje nastavit rozsah hodnot od – do a toleranci.
- **Date** – definuje datum. AttributeFactory umožňuje nastavit rozsah hodnot a toleranci.
- **Bool** – definuje booleovskou proměnnou. AttributeFactory umožňuje nastavit požadovanou hodnotu.
- **String** – definuje textový řetězec. AttributeFactory umožňuje nastavit hodnotu.
- **StringMultival** – definuje textový řetězec s více možnými správnými variantami. AttributeFactory umožňuje definovat pole možných hodnot.
- **StringFulltext** – definuje textový řetězec, vyhodnocovaný pomocí fulltextového vyhledávání. AttributeFactory umožňuje nastavit požadovanou hodnotu.

Mimo výše uvedené umožňuje AttributeFactory nastavit ještě důležitost (importance) w_i každého atributu a to, zda objekt, který nesplnil podmínku daného atributu má být zařazen do výsledku.

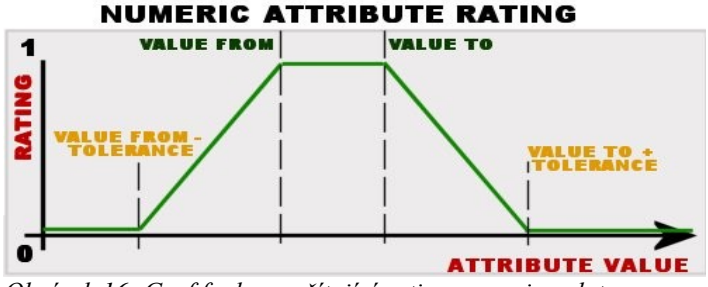
Vyhodnocení Atributů

Jednotlivé atributy A_i jsou přímo přeloženy do SQL jazyka tak, aby databáze vrátila seznam objektů o uspořádaných podle ratingu objektů = vážený průměr ratingu objektu podle jednotlivých uvažovaných atributů.

$$AttributesRating(o) = \frac{\sum_{\forall i} w_i * R_i(A_i(o))}{\sum_{\forall i} w_i}$$

Rating jednotlivých atributů R_i je počítán následovně:

- **Bool:** rating = 1, pokud objekt má požadovanou hodnotu, jinak 0
- **String**: rating je výsledek SQL podmínky AttributeName like“%Value%“
- **StringMultival**: rating je výsledek série podmínek AttributeName like“%Value1%“ or AttributeName like“%Value2%“ or...
- **StringFulltext**: rating je výsledek SQL podmínky Match(AttributeName) Against(“Value“)
- Rating **Numeric** a **Date** je funkce s následujícím grafem (*funkce nabývá hodnoty 1 v rozsahu hodnot od – do, Od nich pak lineárně klesá až k bodům (od - tolerance) a (do + tolerance), kde nabývá hodnoty 0)*:



Obrázek 16: Graf funkce počítající rating numeric a date atributů

5.7.4 Rozhraní třídy UserExpression

Instance třídy **UserExpression** je specifikace jedné metody, která má být použita k nalezení top-k nejpodobnějších uživatelů k současnému. Vytvoření specifikace metody provedete následujícím kódem:

```
//UserExpression($methodType, $importance, $methodName, $methodParameters);
new UserExpression(
        "UserSimilarity",
        $importance,
        $methodName,
        array(
                "userID"=>UPCompCore::getUserId(),
                "noOfUsers"=>$noOfUsers,
                <další parametry dle zvolené $methodName>
        )
)
```

$importance určuje důležitost výpočetní metody,

$methodName definuje, která metoda bude použita k výpočtu podobnosti (***viz 5.9***),

$noOfUsers je požadovaný počet podobných uživatelů

Další parametry jsou závislé na použité metodě, obvykle specifikují, které uživatelské zpětné vazby se mají použít.

Následující kód vrátí na výstupu 10 nejpodobnějších uživatelů k současnému. Výpočet bude proveden **NRMSE** metodou a uvažována bude implicitní zpětná vazba objednávka a zobrazení objektu.

```
new UserExpression(
        "UserSimilarity", 1, "NRmse",
        array(
                "userID"=>UPCompCore::getUserId(),
                "noOfUsers"=>10,
                 "implicitEventsList"=>array("pageview","order")
        )
)
```

5.7.5 Rozhraní třídy ObjectExpression

Instance třídy **ObjectExpression** je specifikace jedné metody, která má být použita k nalezení top-k objektů. Rozhraní ObjectExpression je velmi podobné rozhraní třídy UserExpression popsaném výše.

```
new ObjectExpression(
        $methodType,
        $importance,
        $methodName,
        array(
                "noOfObjects"=>$noOfObjects,
                <další parametry dle zvolené $methodName>
        )
)
```

$methodType určuje typ výpočtu – ***viz 4.4.4*** . Může nabývat hodnot

- **ObjectRating** – hodnocení objektů, bez ohledu na konkrétního uživatele
- **ObjectSimilarity** – podobnost mezi objekty
- **Collaborative** – hodnocení objektů s přihlédnutím ke konkrétnímu uživateli

$importance určuje důležitost výpočetní metody,

$methodName definuje, která metoda bude použita k výpočtu podobnosti (***viz 5.9***),

$noOfObjects je požadovaný počet objektů.

Další parametry jsou závislé na použité metodě a typu výpočtu, obvykle specifikují:

- id aktuálního objektu ("objectID"=>$objectID)
- které zpětné vazby použít ("implicitEventsList"=>array("pageview","order"), "aggregatedEventsList"=>array("opened_vs_shown_fraction")...)
- Následující kód vrátí na výstupu 10 nejlepších objektů dle **Aggregated** metody. Při výpočtu bude uvažována pouze zpětná vazba typu opened_vs_shown_fraction.

```
new ObjectExpression(
        "ObjectRating", 1, "Aggregated",
        array(
                "noOfObjects"=>10,
                "aggregatedEventsList"=>array("opened_vs_shown_fraction")
        )
)
```

5.7.6 Chybové hlášky (třída ErrorLog)

Většina chyb, ke kterým může v UPComp dojít, jsou zaznamenávány pomocí standardních PHP chybových hlášení. Nicméně některé specifické chyby, ke kterým došlo v rámci výpočtu dílčích metod PHP chybové hlášení nevyvolají a jsou zaznamenávány a zpřístupněny pomocí singleton třídy **ErrorLog** (jedná se především o chyby typu nedostatek dat pro predikci a chybné parametry metody). Idea je pokračovat ve výpočtu objektů i přes nalezenou chybu – pokud je to možné. Předpokládejme, že mám dotaz s následujícími object expressions:

```
$objectExpr = array(
```

```
            new ObjectExpression( <first method>),
            new ObjectExpression( <second method>),
            new ObjectExpression( <third method>),
);
```

Pokud dojde k chybě v první metodě, vrátí prázdnou množinu objektů. Nicméně druhá a třetí metoda stále mohou přinést užitečný výsledek. To může být i očekávané chování (první metoda dává nejlepší výsledky, ale je dostupná jen pro některé objekty), proto není vhodné vyvolávat standardní PHP chybové hlášení, které by přinejmenším rušilo uživatele.

Instanci třídy **ErrorLog** nahrajete do proměnné $errorLog pomocí následujícího kódu:

```
$errorLog = ErrorLog::getInstance();
```

Informace o chybách pak můžete získat pomocí následujících metod:

Funkce	Návratová hodnota
componentState()	TRUE, pokud k žádné chybě nedošlo, jinak FALSE
getErrorMessages()	Vrací pole chybových hlášek
getErrors()	Vrací pole instancí třídy Error, která obsahuje úplné informace o chybě

Tabulka 16: Metody třídy ErrorLog

5.7.7 Informace o stavu UPComp (třída ComponentInfo)

UPComp podává různé informace o svém stavu – především o množství obsažených dat prostřednictvím statické třídy **ComponentInfo**. Tyto informace můžete použít především pro rozhodnutí, kterou (které) metody pro výpočet preference použít.

Funkce	Návratová hodnota
getInteractedObjects($eventType)	Metoda vrátí objekty, o kterých existuje od aktuálního uživatele zpětná vazba zvoleného typu
getInteractingUsersCount($eventType, $objectID)	Metoda vrátí počet uživatelů, kteří mají zpětnou vazbu zvoleného typu o vybraném objektu
getEligibleUsers(...)	Metoda vrací ID uživatelů, kteří jsou vhodnými kandidáty do kolaborativního filtrování
getEventsCount($eventType, $eventName="")	Metoda vrací počet zpětných vazeb daného typu a názvu

Tabulka 17: Metody třídy ComponentInfo

5.7.8 Mazání zpětných vazeb (třída HalftimeDecay)

Významnost či spolehlivost zaznamenané zpětné vazby uživatele k objektu obvykle v čase klesá (pro více informací ***viz 2.4***). Navíc - bez ohledu na velikost prodejního webu - může množství zpětných vazeb dříve či později překročit únosnou mez. Problémem může být jak nedostatečná kapacita databáze, tak i prodlužující se doba výpočtu.

Z dlouhodobého hlediska je proto nutné pravidelně mazat nejstarší či nejméně významné uživatelské zpětné vazby. Interval mazání a množství smazaných dat je třeba zvolit dle situace jednotlivých prodejních webů. Důležitými faktory jsou především:

- množství získaných zpětných vazeb za jednotku času
- celková skladovací a výpočetní kapacita prodejního webu
- průměrná doba mezi 2 nákupy či 2 návštěvami jednoho uživatele

UPComp obsahuje statickou třídu **HalfTimeDecay**, která umožňuje mazání uživatelských zpětných vazeb.

V případě implicitních a explicitních dat maže vždy nejstarší uložená data, v případě agregované preference používá tzv. exponenciální zapomínání (vynásobí příslušnou zpětnou vazbu koeficientem k element (0,1)).

Třídu **HalftimeDecay** použijte následujícím způsobem:

```
HalftimeDecay::removeEvents($table, $eventNames, $decayCoeficient);
```

Parametr	Význam
$table	Název tabulky (implicit_events, explicit_events, nebo aggregated_events)
$eventNames	Pole názvů zpětných vazeb, které chceme mazat
$decayCoeficient	Číslo (0,1), specifikuje jakou část dat chceme smazat

Tabulka 18: Parametry funkce removeEvents()

Následující kód smaže 50% nejstarších implicitních uživatelských dat typu pageview a order.

```
HalftimeDecay::removeEvents("implicit_events",array("pageview","order"), 0.5);
```

Třídu HalftimeDecay doporučujeme používat nejlépe v kombinaci s nějakým automatickým spouštěčem skriptů, například **Cron [Wiki10j]**.

5.8 Typické příklady použití

V následující kapitole uvedeme příklady několika typických dotazů do UPComp. Předpokládáme vždy, že $sql obsahuje SQL dotaz pro určení základní množiny objektů a $attributes je prázdné.

Obecně doporučujeme, abyste při dotazu do UPComp používali dvou nebo více metod pro výpočet top-k objektů. Předpokládáme, že jedna metoda je „hlavní" a další metody „doplňkové" - mají výrazně menší důležitost a jsou potřebné především v situaci, kdy hlavní metoda nevrátí dostatečné množství výsledků nebo je preference jednotlivých objektů velmi podobná.

5.8.1 Nejprodávanější objekty

Nejprodávanější objekty jsou běžnou součástí prodejních webů, zobrazují objekty s nejvyšším počtem objednávek.

```
$userExpr ="";
$objectExpr=array(
    new ObjectExpression(
        "ObjectRating", "2", "Aggregated",
        array("noOfObjects"=>10, "aggregatedEventsList"=>array("object_ordered") )
         ),
    new ObjectExpression(
```

```
            "ObjectRating", "1", "Dummy", array("noOfObjects"=>5)
             )
);
$UPQuery = new ComplexQueryHandler($sql, $attributes, $userExpr, $objectExpr);
```

5.8.2 Nejlepší objekty

Nejlepší objekty jsou rozšířením nejprodávanějších objektů – preferenci jednotlivých objektů počítají nejen podle objednávek, ale i dalších uživatelských zpětných vazeb.

```
$userExpr ="";
$objectExpr=array(
     new ObjectExpression(
            "ObjectRating", "6", "Aggregated",
            array(
                        "noOfObjects"=>10,
                        "aggregatedEventsList"=>
                        array("opened_vs_shown_fraction","object_ordered")
             )
    ),
    new ObjectExpression(
            "ObjectRating", "3", "Standard",
            array(
                        "noOfObjects"=>10,
                        "implicitEventsList"=>array("deep_pageview")
            )
     ),
     new ObjectExpression(
            "ObjectRating", "1", "Dummy",
            array("noOfObjects"=>5)
     )
);
$UPQuery = new ComplexQueryHandler($sql, $attributes, $userExpr, $objectExpr);
```

5.8.3 Objekty podobné právě zobrazenému

Podobnost objektů v tomto případě nedefinujeme jako podobnost jejich atributů, ale jako podobnost zpětných vazeb uživatelů, kteří interagovali s oběma objekty.

```
$userExpr ="";
$objectExpr=array(
      new ObjectExpression(
```

```
            "ObjectSimilarity", "6", "Standard",
            array(
                        "objectID"=>$id_object,
                        "noOfObjects"=>10,
                         "implicitEventsList"=>array("pageview","deep_pageview","order")
            )
      ),
      new ObjectExpression(
            "ObjectRating", "1", "Dummy",
            array("noOfObjects"=>5)
      )
);
$UPQuery = new ComplexQueryHandler($sql, $attributes, $userExpr, $objectExpr);
```

5.8.4 Kolaborativní filtrování

Kolaborativní filtrování je oblíbený postup pro doporučování objektů konkrétnímu uživateli. Doporučují se objekty, které jsou nejvíce preferované podobnými uživateli.

```
$userExpr =array(
       new UserExpression(
            "UserSimilarity", "3", "Standard",
            array(
                        "userID"=>ComponentCore::getUserId(),
                        "noOfUsers"=>10,
                         "implicitEventsList"=>array("pageview","deep_pageview","order")
            )
      ),
);
$objectExpr=array(
      new ObjectExpression(
            "Collaborative", "6", "Standard",
            array(
                        "noOfObjects"=>10,
                         "implicitEventsList"=>array("pageview","deep_pageview","order")
            )
      ),
      new ObjectExpression(
            "ObjectRating", "1", "Dummy",
            array("noOfObjects"=>5)
      )
```

```
);
$UPQuery = new ComplexQueryHandler($sql, $attributes, $userExpr, $objectExpr);
```

5.9 Popis výpočetních metod

UPComp rozděluje jednotlivé výpočetní metody jednak podle základního principu (algoritmu), na kterém fungují (metody se stejným principem jsou obsaženy v jedné třídě), a poté podle typu úlohy, kterou počítají (jednotlivé funkce v rámci třídy). Kombinace typ úlohy + název metody pak jednoznačně určuje použitou funkci pro výpočet preference (dále jí nazýváme dílčí výpočetní metodou). Více ***viz 4.4.4 .***

Typy úloh jsou následující:

- **ObjectRating:** hodnocení objektů nezávisle na kontextu aktuálního uživatele nebo aktuálního objektu.
- **ObjectSimilarity:** výpočet „podobnosti“ objektů k aktuálnímu objektu. Podobnost nemusí nutně znamenat podobnost na základě atributů objektu, ale i společné objednávky jedním uživatelem atp.
- **UserSimilarity:** výpočet „podobnosti“ uživatelů k aktuálně přihlášenému uživateli.
- **Collaborative:** hodnocení objektů závislé na aktuálním uživateli. Obvykle je Collaborative výsledkem nějaké agregace hodnocení od top-k uživatelů nejpodobnějších aktuálně přihlášenému.

UPComp má v současnosti implementováno 7 různých metod, přičemž většina metod ovšem neimplementuje všechny typy úloh. Podrobný popis algoritmů jednotlivých metod naleznete v ***kapitole 4.5*** .

- **Dummy:** Metoda Dummy provádí náhodný výběr ze zvolených objektů. Dummy vyhodnocuje všechny 4 typy úloh. Jediným parametrem všech výpočetních funkcí metody Dummy je počet vrácených objektů.
- **Standard**: Metoda Standard obsahuje základní varianty výpočtu preference na základě implicitních a explicitních dat. Standard metoda počítá rating objektů podle jednotlivých uživatelů na základě získaných zpětných vazeb a ten buď agreguje pro jednotlivé objekty, nebo na základě něj počítá podobnost uživatelů nebo objektů.
- **Randomized:** Randomized je obdoba Standard metody. Používá stejné algoritmy pro výpočet, ale množství zpracovávaných zpětných vazeb, objektů a uživatelů shora omezuje číslem K. Pokud je potenciálních dat více, vybere z nich náhodně K.
- **PearsonCorrelation:** Metoda PearsonCorrelation se zabývá pouze ObjectSimilarity a UserSimilarity. Metoda určuje podobnost na základě Pearsonovy korelace mezi ratingem objektů od jednotlivých uživatelů.
- **NRMSE:** Metoda NRMSE se zabývá pouze ObjectSimilarity a UserSimilarity. Metoda určuje podobnost na základě normalizované odmocniny ze střední čtvercové chyby (Normalized Root Mean Squared Error) mezi ratingem objektů od jednotlivých uživatelů.
- **Aggregated:** Metoda Aggregated je založená na agregované zpětné vazbě. Z tohoto důvodu umožňuje pouze výpočet ObjectRating. Parametry Aggregated jsou počet vracených objektů a uvažované typy agregované zpětné vazby.
- **Attributes:** Metoda Attributes počítá podobnost mezi objekty (ObjectSimilarity) na základě uvažovaných atributů objektu. Pro funkčnost metody Attributes je třeba jí jako parametr předat

- ID aktuálního objektu
- pole uvažovaných atributů (zadávané obdobně jako $attributes)
- SQL dotaz, kterým je možné atributy získat

Následující příklad obsahuje použití metody Attributes na prodejním webu Antikvariátu Ichtys:

```
$query="SELECT * FROM `knihy` natural join `kategorie` WHERE `prodano`<>1  limit 8 ";
$attributes = "";
$userExprs= "";
        $attributesQuery = "select * from `knihy` WHERE `prodano`<>1 ";
        $attributeMethodAttributes = array(
           AttributeFactory::MethodIntegerAttribute("`knihy`.`id_typ`", 0, 3),
           AttributeFactory::MethodIntegerAttribute("`knihy`.`cena`", 300, 1),
           AttributeFactory::MethodStringMatchAttribute("`knihy`.`nazev`", 3),
           AttributeFactory::MethodStringMatchAttribute("`knihy`.`autor`", 7)
        );

$objectExprs= array(
  new ObjectExpression( "ObjectSimilarity", "6", "Attributes",
          array(
                    "objectID"=>$zaznam["id_knihy"], "noOfObjects"=>8,
                    "attributesSelectionSQL"=>$attributesQuery,
                    "attributesList"=>$attributeMethodAttributes
           )
   ),
);
$qHandler = new ComplexQueryHandler($query, $attributes,$userExprs, $objectExprs);
...
```

Následující tabulka obsahuje přehled, které metody umožňují provést který typ úlohy:

Třída metod	ObjectRating	ObjectSimilarity	UserSimilarity	Collaborative
Dummy	OK	OK	OK	OK
Standard	OK	OK	OK	OK
Randomized	OK	OK	OK	OK
PearsonCor.		OK	OK	
NRMSE		OK	OK	
Aggregated	OK			
Attributes		OK		

Tabulka 19: Metody a typy výpočtu, které umožňují

6 Programátorská dokumentace

Programátorská dokumentace je určena především programátorům, jejichž cílem je systém UPComp nějak upravit, případně dále rozšiřovat. Dokumentace obsahuje detailní popis použitých technologií, popis funkce klíčových částí systému a postup řešení několika typických rozšíření systému.

Programátorská dokumentace může být distribuována i samostatně bez ostatních součástí diplomové práce, ale předpokládáme, že čtenář programátorské dokumentace se seznámil s požadavky a návrhem UPComp (***kapitola 4***), nebo alespoň s odpovídajícími částmi uživatelské příručky (***5.1 – 5.3***, ***5.7***).

6.1 Použité technologie

UPComp byla napsána v programovacím jazyku **PHP**, požadovaná verze alespoň **5.2.0**. Ke své práci systém dále využívá databázi **MySQL**, verze alespoň **5.0**. Vzhledem k povaze komponenty se předpokládá její instalace jako součást prodejního webu spuštěném na nějakém webovém serveru. Na použitý webový server však UPComp žádné další požadavky neklade.

K vývoji UPComp bylo použito univerzální IDE **NetBeans 6.7.1.** NetBeans doporučujeme i k případné editaci zdrojových kódů.

Veškeré technologie a aplikace potřebné k instalaci, běhu a dalšímu vývoji UPComp jsou volně šiřitelné a vzhledem k jejich povaze je možné UPComp používat nezávisle na operačním systému.

6.2 Dokumentace

Kompletní programátorská dokumentace byla vygenerovaná pomocí programu doxygen a nachází se na přiloženém CD v adresáři **docs/dokumentace**.

6.3 Databáze systému

Jako databáze systému UPComp byla vybrána MySQL. Hlavním důvodem je její rozšíření v rámci prodejních webů, měla by tedy vyhovovat maximálnímu množství uživatelů. MySQL databázi je možné vyměnit za jinou SQL databázi, ***viz 6.7.4*** . UPComp vyžaduje pro svojí funkci následující vlastní tabulky:

implicit_events	Ukládání implicitní zpětné vazby
explicit_events	Ukládání explicitní zpětné vazby
aggregated_events	Ukládání agregované zpětné vazby

Tabulka 20: Databázové tabulky používané UPComp

Názvy jednotlivých tabulek jsou konfigurovatelné v souboru **config/Config.php**.

Vzhledem k tomu, že UPComp může kombinovat dotazy pro získání specifických zpětných vazeb s dotazy pro získání dat o uživatelích a objektech, je třeba, aby tabulky pro ukládání zpětných vazeb byly přímo součástí databáze prodejního webu (nebo alespoň té části databáze prodejního webu, na kterou je možné se dotazovat v rámci SQL části dotazu do UPComp). Vzhledem k používání některých specifických SQL konstrukcí v rámci zpracovávání dotazu do UPComp předpokládáme, že verze MySQL databáze je alespoň 5.0.

Veškerá práce se zpětnou vazbou uživatele je ze své podstaty nekritická (v nejhorším případě podáme nepřesné doporučení), nepovažujeme proto za nutné použití transakcí nebo zámků tabulek. U zpětných vazeb se nepředpokládá ani použití fulltextového vyhledávání. Z těchto důvodů není důležité, jaké úložiště bude pro tabulky zpětných vazeb zvoleno.

6.4 Zdrojové kódy UPComp

Veškeré zdrojové kódy k systému UPComp jsou obsaženy ve složce **DP/UPComp**. Dále složka

- **abstract**: obsahuje abstraktní třídy, ze kterých dědí jednotlivé výpočetní metody
- **config**: obsahuje konfigurační soubor(y).
- **interface**: obsahuje interface pro jednotlivé typy metody ***(viz 4.4.4)***, databázi a třídy zpracovávající jednotlivé typy zpětných vazeb
- **javascript:** obsahuje JavaSriptové soubory s funkcemi pro zaznamenávání uživatelské zpětné vazby a její odeslání pomocí XMLHttpRequest
- **methods**: obsahuje jednotlivé výpočetní metody (Dummy, Standard, Aggregated...)
- **objects**: obsahuje další objekty systému (třídy Attribute, ObjectExpression, Query,...)
- **public**: obsahuje třídy a zdrojové kódy, o kterých se předpokládá, že s nimi přímo přijde do styku uživatel systému (ComplexQueryHandler, UPCompCore, HalftimeDecay...)

6.5 Architektura systému

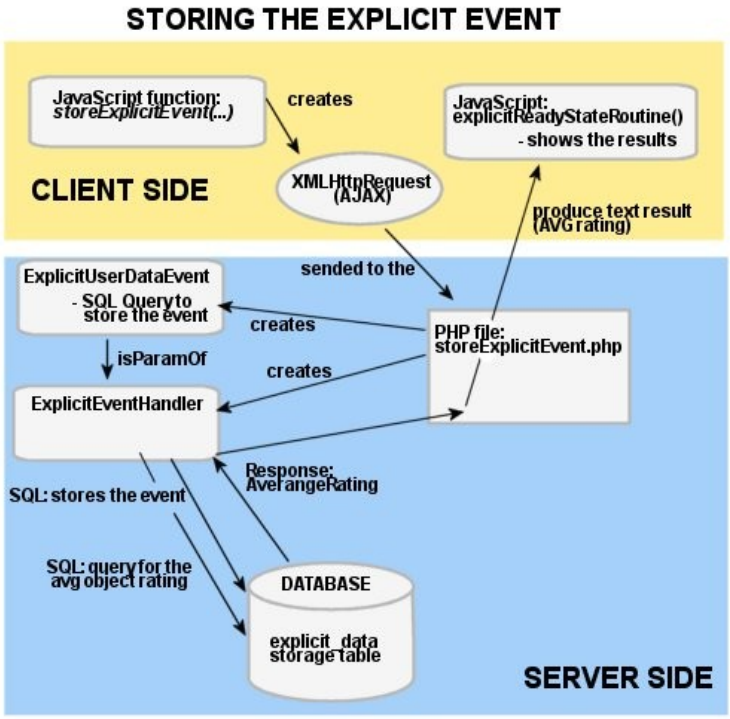
Systém UPComp se skládá ze 2 nezávisle pracujících součástí:

- Zaznamenávání uživatelské zpětné vazby
- Tvorba doporučení (výpočet top-k objektů)

Obě součásti využívají některé společné třídy, ale mohou fungovat nezávisle na sobě (při doporučování nás nezajímá původ dat a naopak).

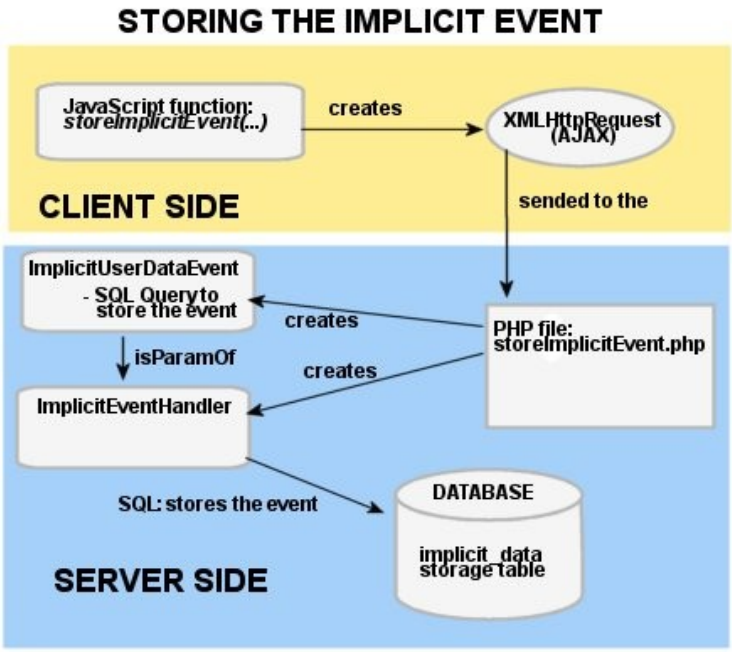
6.5.1 Zaznamenávání uživatelské zpětné vazby

Způsob zaznamenávání byl navrhnut již v ***kapitole 4.4.2*** – s použitím client-side skriptovacího jazyku JavaScript. Diagram pro zpracování explicitní zpětné vazby vypadá následovně:



Obrázek 17: Průběh ukládání explicitní zpětné vazby

U implicitních a agregovaných zpětných vazeb je zpracování obdobné, pouze – vzhledem k tomu, že uživatel o zaznamenávání události obvykle nic neví, není mu prezentován žádný výsledek – a tedy není požadována žádná odpověď z databáze.



Obrázek 18: Průběh ukládání implicitní zpětné vazby

6.5.2 Tvorba doporučení (výpočet top-k objektů)

Tvorba dotazu

Dotaz do UPComp se skládá ze 4 částí:

- SQL dotazu
- Atributů SQL
- specifikace metod pro výpočet top-k uživatelů UserExpressions
- specifikace metod pro výpočet top-k objektů ObjectExpressions

SQL dotaz je textová proměnná, ze které je následně vytvořena třída **Query**. Třída **Query** se stará především o rozparsování SQL dotazu na jednotlivé složky (select, from,...limit) a jejich zpřístupnění. Atributy jsou instance třídy **Attribute** (obvykle generované pomocí factory třídy **AttributeFactory**). UserExpressions a ObjectExpressions jsou instance tříd **UserExpression** a **ObjectExpression**. Parametry jednotlivých tříd a příklady dotazů do UPComp ***viz 5.7.2 – 5.7.5*** **.**

Postup výpočtu

Výpočet top-k objektů je prováděn třídou **ComplexQueryHandler**. Třída na vstupu dostane jednotlivé části UPComp dotazu. Samotný výpočet se děje v metodě **ComplexQueryHandler ->sendQuery()**. Nejprve je vytvořen na základě **SQL** a **Attributes** části dotazu SQL dotaz do databáze, který vrátí dvojici ID objektu, rating vypočtený SQL dotazem. O zpracování SQL dotazu se stará třída **ExtendedQueryToDatabaseInteraction**, nebo **BasicQueryToDatabaseInteraction** (Basic v případě, kdy pole Attributes je prázdné).

Dvojce (**objectID, Rating**) pak slouží jako základní množina uvažovaných objektů pro další metody. Následně jsou vyhodnocovány pomocí funkce **ComplexQueryHandler ->evaluateExpression()**

jednotlivé **UserExpressions**. Zde (a stejně tak u vyhodnocování **ObjectExpressions**) se uplatňují PHP třídy ReflectionClass a ReflectionMethod, které zajistí spuštění příslušné výpočetní funkce vybrané metody se zvolenými parametry, více ***viz 6.6.1 .***

Výsledky jednotlivých výrazů jsou normovány funkcí **ComplexQueryHandler ->normalize()** (nejlepší objekt/uživatel má vždy hodnotu 1).

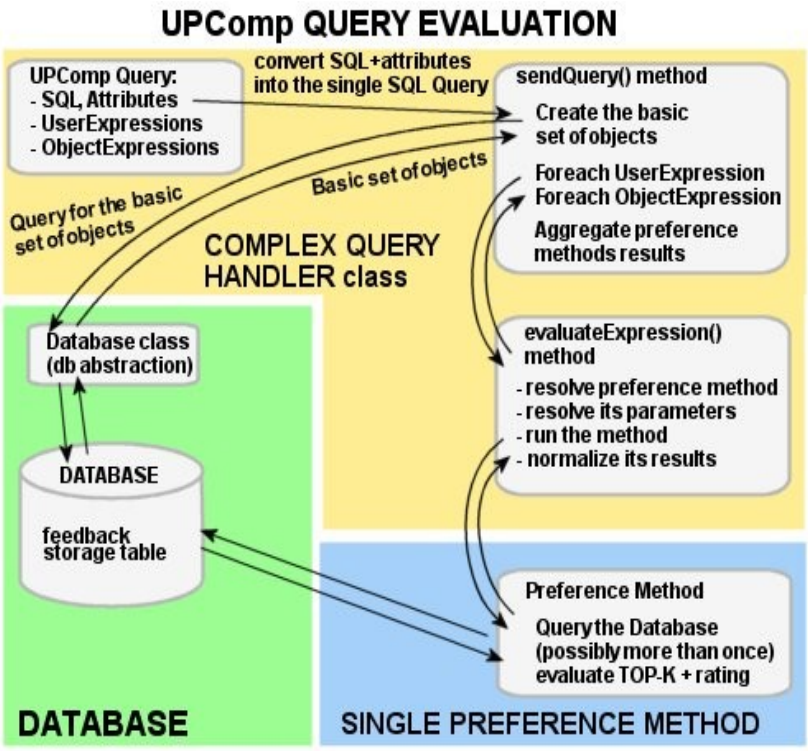
Po vypočtení všech UserExpressions jsou jednotlivé výsledky agregovány funkcí **ComplexQueryHandler ->aggregateExprResults()**.

Po ukončení výpočtu **UserExpressions** se provádí stejným způsobem výpočet **ObjectExpressions** (vyžaduje-li některá metoda specifikovaná v ObjectExpression na vstupu pole uživatelů, použije se výstup agregace přes všechny UserExpressions).

Po agregaci všech **ObjectExpressions** je z nich vybrána část odpovídající složce **limit** SQL části dotazu (funkce **ComplexQueryHandler->getDemandedObjectsFromList()**). Tyto objekty jsou pak použity jako restrikce původní SQL části dotazu:

```
Select <atributy objektu> from <tabulky objektu>
where <objectID> in( <výsledek ObjectExpressions> ) and <původní podmínky na objekt>...
```

Přepracovaný dotaz je opět odeslán třídě **BasicQueryToDatabaseInteraction** a její výsledek (instance třídy **QueryResponse**) poskytnut jako výsledek celého zpracování dotazu.



Obrázek 19: Průběh zpracování UPComp dotazu (zjednodušené)

6.5.3 Rozšiřitelnost

Systém UPComp byl navrhnut tak, aby umožňoval především snadnou rozšiřitelnost o další metody pro výpočet top-k – ***viz 6.7.2 .*** Dále předpokládáme možnost rozšíření o nové uživatelské zpětné vazby – ***viz 6.7.1*** , případně i přidání typu výpočtu ***viz 6.7.3*** .

6.6 Zajímavé části implementace

V následující kapitole jsou popsána neobvyklá, či jinak zajímavá řešení UPComp.

6.6.1 ReflectionClass a ReflectionFunction

ReflectionClass a ReflectionMethod jsou součástí PHP Reflection API (viz http://cz.php.net/manual/en/intro.reflection.php). Jedná se o speciální třídy, které poskytují informace o třídě či metodě a de facto umožňují v PHP provádět reverse engeneering **[Wiki10k]**.

UPComp ReflectionClass využívá při zpracovávání dotazu, konkrétně při zpracování jednotlivých User a ObjectExpression v metodě **ComplexQueryHandler->evaluateExpression($expr)**. Hlavními výhodami jsou:

- možnost ověřit snadno, že metoda specifikovaná v $expr je schopna zpracovat daný typ výpočtu
- snadné ověřování a případně doplňování parametrů metody

6.6.2 Úprava SQL dotazů

SQL část dotazu do UPComp je použita pouze jako základ výsledného dotazu do databáze. Nejčastěji jsou v průběhu výpočtu měněny rozsah (limit) dotazu, hledané atributy (select), případně přidávány další podmínky.

K manipulaci s UPComp dotazem (a tedy i s jeho SQL částí) slouží třída **Query**, která se stará především o:

- Rozparsování SQL dotazu na jednotlivé složky (část select, from, where...)
- Nahrazování jednotlivých částí dotazu za jiné
- Zpětné složení SQL dotazu z jeho částí

Aplikační logika změn dotazu je obsažena ve třídě **ComplexQueryHandler**, případně v metodě **Attributes**, která také třídu **Query** využívá (***viz 4.5***).

6.6.3 Vyhodnocování dotazu

Vyhodnocování dotazu do UPComp prováděné funkcí **ComplexQueryHandler->sendQuery()** patří nepochybně mezi zajímavé části implementace. Podrobně se mu věnujeme v ***kapitole 6.5.2***.

6.7 Typické příklady rozšíření a úprav UPComp

Následující kapitola obsahuje popis nejobvyklejších možností, jak může být UPComp v budoucnu rozšiřována. Spíše než úplný tutoriál se jedná o posloupnost kroků, na které by programátor neměl zapomenout a několik rad, kterými je vhodné se řídit.

6.7.1 Přidání typu zpětné vazby

Pro přidání nového typu zpětné vazby je třeba jednak vytvořit funkci pro její zachycení a pak informovat UPComp že nová zpětná vazba existuje a jak jí zpracovat.

Nejprve je třeba zvolit jméno zpětné vazby a rozhodnout se, zda jde o implicitní, explicitní nebo agregovanou zpětnou vazbu (***viz 2.2 , 4.4.2***). Je důležité si také rozmyslet, jakým způsobem bude zpětná vazba získávána. Novou zpětnou vazbu je třeba nejprve zaregistrovat.

- otevřete soubor **config/Config.php**

- do proměnné **$recognizedImplicitEvent** / **$recognizedExplicitEvent** / **$recognizedAggregatedEvent** pro implicitní / explicitní / agregované zpětné vazby přidejte jako položku pole jméno nové zpětné vazby.

```
public static $recognizedExplicitEvent = array("user_rating","<jméno nové zpětné vazby>");
```

Dále v konfiguračním souboru nastavíme standardní váhu (důležitost) nové zpětné vazby:

- do proměnné **$eventImportance** přidáme jako položku pole záznam <jméno zpětné vazby>=><integer – důležitost>

```
public static $eventImportance = array( "<jméno nové zpětné vazby>"=>2, "order"=>10,...);
```

Nyní je třeba vytvořit JavaScriptovou funkci, která bude zvolenou zpětnou vazbu odchytávat. Otevřete soubor **javascript/storeFunctions.js**. Zde jsou uloženy všechny funkce, které se starají o zachycení zpětných vazeb. Zde naprogramujte novou funkci, která se bude starato o zachycení Vaší zpětné vazby. Samotná konstrukce funkce (funkcí) je již velmi závislá na zvolené zpětné vazbě. Je pouze třeba v rámci Vaší funkce zavolat s příslušnými parametry funkci **storeImplicitEvent** / **storeExplicitEvent** / **storeAggregatedEvent** – podle typu zpětné vazby.

Posledním krokem je vyvolání Vaší funkce jako reakci na událost uživatele. Zde opět přesný postup závisí na povaze zpětné vazby, obvykle však využijete některou z DOM událostí na konkrétní stránce. Například implicitní zpětná vazba „pageview“ je zpracována tak, že u každé stránky s detailem objektu je v elementu **<body>** přidána událost **onload="objectOpen();"**.

```
...
<body onload="objectOpen();">
...
```

6.7.2 Přidání výpočetní metody

Pro přidání nové výpočetní metody je nutné metodu naprogramovat, zaregistrovat jí do UPComp (a případně začít používat v dotazech do UPComp).

Nejprve vytvořte nový PHP soubor ve složce **methods/** (nejlépe pojmenovaný podle použité metody) a v něm vytvořte PHP třídu <název_metody>. Nastavte, že třída je potomkem abstraktní třídy AbstractMethod. Nyní si rozmyslete, které typy výpočtu bude metoda implementovat. Každému typu výpočtu (ObjectRating, ObjectSimilarity, UserSimilarity, Collaborative) odpovídá jeden interface s metodami, které musíte implementovat a jejich povinnými parametry. Názvy jednotlivých interface jsou stejné jako názvy typů výpočtu. Na příkladu je vidět správně definovaná třída NewMethod, která obsahuje metody pro výpočet ObjectRating a ObjectSimilarity.

```
class NewMethod extends AbstractMethod implements ObjectRating,ObjectSimilarity{
…
}
```

Nyní můžete přikročit k samotné implementaci: každý interface obsahuje 1 nebo 2 metody, které je třeba implementovat (jedna vracející top-k objektů/uživatelů, druhá objekty/uživatele na pozici od k do k+j). Jednoduchou kostru všech typů výpočtu (především s ohledem na povinné vstupní parametry a formát výstupu) obsahuje třída **Dummy** (methods/Dummy.php).

Po implementaci metody můžete přistoupit k její registraci do systému UPComp – stačí pouze includovat metodu spolu s ostatními metodami při zavádění UPComp.

- Otevřete pro editaci soubor **public/UPCompCore.php**
- ve funkci **loadCore()** přidejte za řádek
  `require_once(ComponentCore::$baseDir."/methods/Dummy.php");` následující řádek
  `require_once(ComponentCore::$baseDir."/methods/<název metody>.php");`

Posledním krokem je samozřejmě použití metody v dotazu do UPComp. Předpokládáme, že jsme vytvořili metodu **NewMethod** implementující ObjectRating, kde vyžaduje oproti povinným vstupním parametrům z interface ještě parametr $test[integer]. Metodu v dotazu použijeme následovně:

```
$sql="...";
$attributes="";
$userExpr="";
$objectExpr=array(
        new ObjectExpression(
            "ObjectRating", "3", "NewMethod", array("noOfObjects"=>10, "test"=>15 )
        ),
);
$qHandler = new ComplexQueryHandler($sql, $attributes,$userExpr, $objectExpr);
```

6.7.3 Přidání typu výpočtu

UPComp v současnosti uvažuje o 4 možných typech výpočtu (ObjectRating, ObjectSimilarity, UserSimilarity a Collaborative). Více o typech výpočtu a důvodech pro výběr současných typů **viz.4.4**.

Pro vytvoření nového typu výpočtu je třeba vytvořit jeho interface, zaregistrovat jej v UPComp a případně vytvořit alespoň jednu metodu, která typ výpočtu implementuje.

Nejprve ve složce **interface/** vytvořte nový soubor a nazvěte jej stejně jako uvažovaný typ výpočtu a v něm vytvořte stejnojmenný PHP interface. Obsahem interface bude alespoň jedna metoda, jejímž výstupem je top-k uživatelů, nebo top-k objektů. Název metody se nesmí shodovat s názvem jakékoli metody použité v ostatních interface. Důležité je metodu a její parametry dobře dokumentovat (obdobně jako u ostatních interface). Rozmyslete si také, které parametry budete u metod vyžadovat a které jsou už závislé na konkrétní implementaci. Následující příklad obsahuje možný interface pro hodnocení uživatele - UserRating:

```
interface UserRating {
    /** returns $noOfUsers of the best rated users
     * rating method depends on implementation
     * @param <type> $noOfUsers number of users, we search for
     * @param <type> $userList array of allowed users (basic set)
     * @return array( userID => rating) )
     */
    public function getBestUsers($noOfUsers, $userList="");
}
```

Po vytvoření interface je třeba jej includovat metodu spolu s ostatními interface při zavádění

UPComp:

- Otevřete pro editaci soubor **public/UPCompCore.php**
- ve funkci **loadCore()** přidejte za řádek

  require_once(UPCompCore::$baseDir."/interface/ObjectRating.php"); následující řádek

  require_once(UPCompCore::$baseDir."/interface/<název interface>.php");

Dále je třeba upravit mapování mezi typem výpočtu a konkrétní použitou metodou

- Otevřete pro editaci soubor **config/Config.php**
- Upravte funkci **getMethodForExpression($expr)** tak, aby pro nový typ výpočtu vracela funkci, která výpočet provádí. Vyhodnocení použité funkce nemusí být tak přímočaré jako v případě ostatních typů, můžete použít i komplexnější konstrukce (parametr **$expr** je instance třídy UserExpression nebo ObjectExpression). Pro výše zmíněný příklad UserRating by upravená funkce getMethodForExpression($expr) vypadala následovně:

```
public static function getMethodForExpression($expr){
        switch ($expr->getExpressionType()) {
                case "UserRating":
                   return "getBestUsers";
                   break;
                case...
        }
}
```

Posledním krokem je pak vytvoření metody, která typ výpočtu implementuje: vytváření nových metod ***viz 6.7.2*** .

6.7.4 Změna databáze

UPComp v současné době používá MySQL databázi, ke které ovšem přistupuje přes definované rozhraní. Povinné prvky rozhraní jsou definované v interface **UPCompDatabaseInterface**. Současná implementace příslušného interface se nachází v singleton třídě **UPCompDatabase,** v souboru **objects/MySQLDatabase.php**. Toto poněkud matoucí uspořádání bylo zvoleno z důvodu, aby nebylo třeba při změně databáze měnit třídu pro práci s databází, ale zároveň aby byla uchována informace o jakou databázi se jedná.

Pro změnu databáze je nejprve třeba ověřit, zda Vámi zvolená databáze podporuje všechny SQL konstrukce použité v UPComp (v opačném případě je třeba vybrané dotazy upravit tak, aby odpovídali funkčnosti Vaší databáze). Mimo podporu běžných konstrukcí select/insert into/delete/update... ověřte především:

- **ON DUPLICATE KEY UPDATE**
- **<collumn name> IN (<list of values>)**
- **LIMIT <from>, <count>**
- výpočet hodnoty (ratingu) v části **SELECT** a vrácení jako jeden ze sloupců výsledku

v souboru **objects/<název databáze>.php** vytvořte třídu **UPCompDatabase**, která implementuje interface **UPCompDatabaseInterface**. Třída musí být singleton, kde static metoda **get_instance()** slouží k získání instance třídy (jako kostru je možné použít implementaci rozhraní MySQL databáze).

Po vytvoření rozhraní k Vaší databáze je již nutné jen vyměnit include MySQL databáze za Vaší databázi:

- Otevřete pro editaci soubor **public/UPCompCore.php**
- ve funkci **loadCore()** nahraďte řádek

  require_once(UPCompCore::$baseDir."/objects/MySQLDatabase.php"); za řádek

  require_once(ComponentCore::$baseDir."/objects/<název databáze>.php");

- ve **funkci loadCoreEvents()** nahraďte řádek

  require_once(ComponentCore::$baseDirEvents."/objects/MySQLDatabase.php"); za řádek

  require_once(ComponentCore::$baseDirEvents."/objects/<název databáze>.php");

6.7.5 Změna agregace výsledků

Agregace výsledků jednotlivých výpočetních metod se děje ve funkci

ComplexQueryHandler->aggregateExprResults($exprResults).

$exprResults je pole polí: array(array(objectID=>rating)), kde vnitřní pole odpovídá výsledku jedné metody pro výpočet top-k.

Změnu agregační funkce provedete změnou funkce **aggregateExprResults()**. Důležité je dodržet:

- formát vstupních dat: pravděpodobně bude třeba data procházet data dvěma for-cykly stejně jako současná implementace
- monotónnost agregace: pro objekty A a B, kde rating objektu A >= rating objektu B pro každou dílčí výpočetní metodu p je vhodné dodržet, že výsledný rating $A >= B$

  (**pro každé i**: $p_i(A) >= p_i(B)$) => $@(p_i(A)) >= @(p_i(B))$
- normalizace dat: je vhodné, aby výsledný rating každého objektu byl **element** [0,1]. Můžete využít funkci **normalize()** stejně jako současná implementace.

V současné době funkce počítá vážený průměr hodnocení jednotlivých objektů. Můžete buď navrhnout vlastní agregační funkci, nebo využít některou z funkcí obvyklých ve fuzzy logice – především fuzzy konjunkci nebo fuzzy disjunkci (konjunkce a disjunkce pro vícehodnotovou – fuzzy – logiku). Další informace o fuzzy logice je možné nalézt na **[Wiki10l]**, příklady fuzzy logických spojek lze například na *http://www.ksi.mff.cuni.cz/~vojtas/vyuka/NDBI021PrincipyUzivatelskychPreferenci/1011/NDBI021.html*.

Obecně mají fuzzy konjunkce spíše diskriminační charakter (špatný výsledek v jedné metodě znamená špatný výsledek celkově), fuzzy disjunkce právě naopak (dobrý výsledek byť jen v jediné metodě umožňuje celkově dobrý výsledek). Z tohoto pohledu je současná implementace agregační funkce – vážený průměr – zhruba uprostřed mezi konjunkcí a disjunkcí.

6.7.6 Spuštění UPComp s jinou konfigurací

UPComp umožňuje snadnou změnu své konfigurace: veškeré nastavitelné vlastnosti UPComp obsahuje třída Config v souboru **config/Config.php**. Chcete-li konfiguraci pouze změnit, můžete editovat přímo soubor **Config.php**. Změny v konfiguraci se projeví okamžitě.

Druhou možností je uchovávat více variant konfigurace a spustit vždy některou z nich. V takovém případě soubor **Config.php** uložte do stejného adresáře pod jiným jménem (například

ConfigNew.php) a proveďte editaci požadovaných vlastností.

UPComp obsahuje 2 metody pro includování svého obsahu:

- **UPCompCore::loadCore();**
- **UPCompCore::loadEventsCore();**

Obě metody mají volitelný parametr **$configName** (název konfiguračního souboru) s default hodnotou „**Config.php**“. Pokud tedy chcete UPComp spustit například s konfigurací obsaženou v souboru **config/ConfigNew.php**, nahraďte výskyty:

- UPCompCore::loadCore(); za UPCompCore::loadCore(“ConfigNew.php“); a
- UPCompCore::loadEventsCore(); za UPCompCore::loadEventsCore(“ConfigNew.php“);

7 Testování komponenty pro podporu dotazování

V následující kapitole se budeme zabývat testováním systému UPComp. Testování komponenty provedeme na dvou existujících prodejních webech ***www.slantour.cz*** a ***www.antikvariat-ichtys.cz***. Budeme tedy pracovat s reálnými uživateli. Cíle testování jsou následující:

1. prokázat možnost implementovat UPComp na prodejní weby
2. zjistit výkon použitých metod (zda nějak zlepšily výsledky prodejního webu)
3. zjistit časovou náročnost použitých metod

První bod bude (alespoň částečně) prokázán samotnou implementací komponenty na testovací weby.

Třetí bod byl zjišťován především porovnáváním času běhu PHP skriptů. Zde se jako zásadní problém ukázalo zjišťování top-k uživatelů. Jako reakce na výsledky testu rychlosti byl navrhnut postup, který odstraní problémy s příliš pomalým výpočtem top-k uživatelů – **více *viz 7.2*** .

Druhý bod se týká samotného doporučování UPComp: po svém nasazení systém uživateli zobrazuje tzv. Tipy na objekty. Tip na objekty je zkrácený výpis top-k objektů zobrazený uživateli. Úspěšnost metody posuzujeme jednak podle počtu otevřených/zobrazených tipů na objekty a pak dle poměru objednaných/zobrazených tipů na objekty **(více *viz 7.3*)**.

7.1 Testovací prostředí

Systém UPComp byl pro účely testování implementován na 2 prodejní weby:

- ***www.slantour.cz*** - rezervační systém zájezdů cestovní kanceláře SLAN tour
- ***www.antikvariat-ichtys.cz*** – on-line prodej knih antikvariátu Ichtys

Oba weby patří spíše do kategorie netypického použití UPComp, proto by bylo vhodné v budoucnu provést jako rozšíření diplomové práce implementaci UPComp na nějaký e-shop.

7.1.1 www.slantour.cz

Cestovní kancelář SLAN tour vznikla v roce 1990. Průběžně nabízí cca 300-700 zájezdů. K zájezdu je obvykle přiřazeno více termínů, kdy je možné jej absolvovat. Zájezdy většinou zůstávají (s mírnými obměnami) v nabídce CK několik let. Atributy zájezdu jsou například typ zájezdu, země, destinace, název a popis zájezdu, typ ubytování, stravování, dopravy, ceny a termíny zájezdu atd.

Dle údajů měřící služby Google Analytics za září a říjen má web následující vlastnosti:

Počet zobrazených stránek za sledované období	239548

Počet zobrazených detailů zájezdu za sledované období	112203
Počet objednávek za sledované období	182
Průměrný počet zobrazených stránek / den	3927
Průměrný počet unikátních uživatelů / den	767
Průměrný počet stránek / uživatele / sledované období	5,1
Průměrný počet zobrazení detailu zájezdu / den	1839
Průměrná doba, kterou uživatel stráví na webu / jedna návštěva	3min 19sec
Průměrný počet objednávek / den	3

Rezervační systém CK SLAN tour vznikl rozšířením bakalářské práce **[Pes08]**. Systém:

- Obsahuje katalog zájezdů řazený dle typu zájezdu (pobytový, poznávací...), cílové země zájezdu a cílové destinace.
- Obsahuje vyhledávání dle parametrů zájezdu
- Umožňuje uživateli vyplnit objednávku, předběžnou poptávku a dotaz k zájezdu
- Obsahuje další součásti, které již nejsou příliš důležité pro UPComp (informace o destinacích, PDF katalog atd.)
- Již před implementací UPComp systém obsahoval v rámci katalogu tzv. Tipy na zájezdy, které však jen náhodně zobrazovaly zájezdy ze zvolené kategorie katalogu.

Implementace UPComp

Pro potřeby UPComp jsme zvolili za objekty systému jednotlivé zájezdy. Explicitní zpětná vazba nebyla v původním systému zahrnuta a (na přání provozovatele) nebyla ani implementována. Za implicitní zpětnou vazbu jsou považovány následující události:

- **pageview**: zobrazení stránky s detailem objektu (pouze boolovská proměnná ano/ne)
- **deep_pageview**: dostatečné množství akcí uživatele na stránce (přejetí myší přes objekt, kliknutí na odkaz atp.)
- **order**: objednávka zájezdu, 3-úrovňová dle druhu objednávky (objednávka, předběžná poptávka, dotaz)

Dále byla ještě uvažována agregovaná implicitní zpětná vazba:

- počet objednávek zájezdu
- počet zobrazení zájezdu v seznamu (zobrazení v katalogu, výsledcích vyhledávání nebo v tipech na objekty)
- počet otevření zájezdu ze seznamu (kliknutí na odkaz na detail zájezdu)

UPComp byla použita pro generování „tipů na zájezdy“ – jednak v rámci katalogu zájezdů a poté u detailu konkrétního zájezdu. Obě varianty jsou – vzhledem k jiné sémantice doporučování (recommended x related) vyhodnocovány zvlášť.

7.1.2 www.antikvariat-ichtys.cz

Antikvariát Ichtys funguje od roku 2005 a nabízí k on-line prodeji cca 5000 knih. Atributy knih jsou autor, název knihy, popis, cena a přiřazená kategorie. Knihy se obvykle prodávají v jediném exempláři – po uskutečnění prodeje je tedy kniha stažena z nabídky.

Podle měřící služby *Navrcholu.cz* měl antikvariát od 1.října do 31. října následující návštěvnost:

Počet zobrazených stránek za sledované období	8305
Počet objednávek za sledované období	34
Průměrný počet zobrazených stránek / den	268
Průměrný počet unikátních uživatelů / den	51
Průměrný počet stránek / uživatele / sledované období	5,25
Průměrný počet objednávek / den	1,1

Systém antikvariátu odpovídá definici prodejního webu. Obsahuje katalog knih, parametrické vyhledávání a možnost knihu objednat.

Implementace UPComp

Pro potřeby UPComp byly za objekty zvoleny jednotlivé knihy. Implicitní a agregované události byly zvoleny stejně jako u *www.slantour.cz* – pouze s tím rozdílem, že objednávka knihy je pouze jednoúrovňová.

Dále byla ještě oproti původnímu stavu systému přidána možnost uživatelského hodnocení jednotlivých knih a na ní navázaný sběr explicitní zpětné vazby. Zde se bohužel extrémním způsobem projevil problém neochoty uživatelů explicitní data poskytovat (***viz 2.2.3***). Za celou dobu nasazení a testování UPComp (30.10.2010 – 7.12.2010) byly získány celkem 4 hodnocení knih. Důvodem pro tak malý zájem byla pravděpodobně jednak nižší návštěvnost webu jako taková a jednak asi i jistá nekoncepčnost použití explicitní zpětné vazby (absence benefitů pro uživatele). V dalším případném rozšíření diplomové práce by bylo vhodné UPComp instalovat na prodejní web, který již hodnocení objektů úspěšně využívá.

UPComp byla použita pro generování „tipů na knihy“ – jednak v rámci katalogu knih a poté u detailu konkrétní knihy. Obě varianty jsou – vzhledem k jiné sémantice doporučování (recommended x related) vyhodnocovány zvlášť.

7.2 Testování rychlosti odpovědí na dotaz do UPComp

Testování rychlosti odpovědí UPComp na položený dotaz bylo prováděno jak na webu Antikvariát-ichtys.cz, tak na slantour.cz. Výsledkem testu je rozdíl času ve vteřinách mezi okamžikem těsně před začátkem konstrukce dotazu do UPComp a okamžikem těsně po získání odpovědi na dotaz. V testu rychlosti odpovědi byla testována každá metoda použitá v testu výkonnosti metod. Test byl rozdělen do 3 částí:

- testování doporučování na hlavní straně
- testování doporučování v rámci katalogu objektů
- testování doporučování podobných (related) objektů.

Pro testování doporučování v rámci katalogu bylo náhodně vybráno několik kategorií katalogu, na kterých pak byla testována rychlost odpovědi každé metody. Obdobně pro testování podobných objektů bylo vybráno náhodně několik objektů, na kterých pak byly testovány všechny metody.

Kompletní výsledky testů obsahuje ***příloha E*** .

7.2.1 Testování rychlosti metod na antikvariat-ichtys.cz

Testování všech metod použitých v *Antikvariát-ichtys.cz* dopadlo vcelku uspokojivě – při žádném testu nebyl zjištěn čas na odpověď vyšší než 0.3 vteřiny. Je však možné, že dobré výsledky

(především metod založených na podobnosti uživatelů) byly dosaženy především díky celkově menšímu množství uložených uživatelských dat. Rozsáhlejší testování provedené po delším používání systému UPComp by mohlo odpovědět na tuto otázku.

7.2.2 Testování rychlosti metod na slantour.cz

Metody na *Slantour.cz* jsme testovali z pohledu 3 různých uživatelů:

- Nový uživatel (bez uložené zpětné vazby)
- Běžný uživatel (7 uložených zpětných vazeb o 5 objektech – o vybraných objektech existovalo v systému 92 zpětných vazeb od 60 uživatelů)
- Častý uživatel (50 uložených zpětných vazeb o 39 objektech – o vybraných objektech existovalo v systému celkem 5939 zpětných vazeb od 4409 uživatelů)

U testování metod použitých na Slantour.cz se objevily problémy především u metod založených na podobnosti uživatele. Doba běhu metod byla pro Častého uživatele nepřijatelná (doba odpovědi až 11 vteřin). Žádná z variant metod založených na podobnosti uživatelů nedosáhla výrazně lepších výsledků než ostatní. Pro zbylé 2 uživatele byl čas pro výpočet všech metod přijatelný (doba odpovědi průměrně <1 vteřina).

Ostatní metody (Object rating, ObjectSimilarity, Attributes a Dummy) dosahovaly dobrých výsledků pro všechny uživatele (doba odpovědi průměrně <0.4 vteřiny).

Jako reakce na problémy metod založených na podobnosti uživatelů byly jednak navrženy úpravy metod pro výpočet podobnosti uživatelů, které vedly ke snížení času odpovědi na cca 6-7 vteřin, což je ovšem stále ještě nepřijatelný výsledek. Následně byla navržena metoda Randomized UserSimilarity with Attributes, která:

- K výpočtu top-k uživatelů (UserSimilarity) využívá Randomized metodu s omezením maximálního počtu řádků v odpovědi na SQL dotaz na 1500 (=maximální počet zpětných vazeb, nad kterými se počítá podobnost uživatelů).
- Jinak je shodná se Standard UserSimilarity with Attributes.

Tato metoda již i pro Častého uživatele dosahovala přijatelných výsledků (doba odpovědi průměrně 1.4 vteřiny). Je však otázkou dalšího zkoumání, zda může dávat dostatečně dobré odpovědi.

Další možností, jak problém řešit je pomocí nějakých heuristik omezit uživatele, mezi kterými hledáme top-k nejpodobnějších (například pouze uživatelé se zpětnými vazbami k alespoň X objektům atd.). Podle nastíněného principu byla navržena metoda Heuristics UserSimilarity with Attributes, která je shodná se Standard metodou s výjimkou, že omezuje prohledávaný prostor uživatelů na takové, o kterých systém uchovává zpětné vazby alespoň u 2 objektů. Tato metoda již dosahovala přijatelných časů odpovědi (průměrná doba odpovědi <1 vteřina).

Jako ideální řešení do budoucna se jeví kombinace použití heuristik na omezení počtu uživatelů spolu s omezením maximálního počtu uvažovaných zpětných vazeb.

7.3 Kritéria testu výkonnosti metod

Cílem testu výkonnosti metod je stanovit, jaký je vztah mezi „vypočítanou“ a „správnou“ uživatelovou preferencí.

V případě tzv. tipů na objekty pro nás nebude příliš důležitá samotná vypočtená preference, ale spíše pouze to zda objekt je/není v top-k zobrazených objektů (zde předpokládáme, že k je u tipů na objekty dostatečně malé tak, aby samotná pozice objektu v top-k měla jen malý vliv na uživatelovo rozhodování – de facto očekáváme, že všechny objekty se uživateli zobrazí na obrazovce najednou bez nutnosti scrollovat).

Správná preference je skutečná hodnota uživatelovy preference k objektu – tedy ochota uživatele objekt koupit. Zde je hlavním problémem jak zjistit skutečnou preferenci. Někteří autoři **[WJR01, CLWB01]** mohli využít například uživatelských dotazníků, upravených prohlížečů atp. k poskytnutí explicitní preference, kterou považovali za správnou preferenci. Je však otázka, zda je tato úvaha správná, nehledě na to, že použití explicitních prvků může přímo ovlivnit chování uživatele – a tedy znehodnotit celé testování.

Vzhledem k popsaným problémům bylo rozhodnuto použít pro zjištění skutečné preference implicitní zpětné vazby. Vycházíme ze 2 předpokladů:

- **Objedná-li si uživatel nějaký objekt, je jeho preference k němu velmi vysoká.**
- **Má-li uživatel zobrazený seznam objektů (s odkazem na detail objektu), otevře pouze ty objekty, které ho dostatečně zaujmou (tedy jejich preference je vyšší, než prahová hodnota X).**

Na základě předpokladů jsme stanovili 3 měřené veličiny:

$$OpenShown = \frac{počet\ otevřených\ objektů\ z\ tipů\ na\ objekty}{počtet\ objektů\ zobrazených\ v\ tipech\ na\ objekty}$$

$$OrderShShown = \frac{počet\ objednaných\ objektů\ ,\ které\ byly\ zobrazeny\ v\ tipech\ na\ objekty}{počtet\ objektů\ zobrazených\ v\ tipech\ na\ objekty}$$

$$OrderOpShown = \frac{počet\ objednaných\ objektů\ ,\ které\ byly\ zobrazeny\ v\ tipech\ na\ objekty\ a\ otevřeny\ uživatelem}{počtet\ objektů\ zobrazených\ v\ tipech\ na\ objekty}$$

U definovaných veličin pak předpokládáme, že čím vyšší je skóre metody v dané veličině, tím správnější jsou její odpovědi. Vzhledem k tomu, že pracujeme s vágními pojmy jako „velmi vysoká“ a „vyšší než“, můžeme dané veličiny použít pouze pro relativní porovnání mezi jednotlivými metodami. Zajímavé pro nás budou především zjištění, zda některá metoda je v některých aspektech statisticky významně lepší než jiná.

7.4 Testované metody

Následující kapitoly obsahují detailní popis metod použitých v testování výkonnosti na Antikvariátu Ichtys a CK SLAN tour.

7.4.1 Metody použité v antikvariat-ichtys.cz

Při implementaci UPComp do prodejního webu antikvariat-ichtys.cz bylo třeba se vyrovnat se zásadním problémem: antikvariát má poměrně malou návštěvnost (cca 50 uživatelů denně) v poměru k počtu objektů (aktuálně cca 5000). Z toho vyplývá, že v krátké době běhu komponenty bylo možné jen obtížně (kvůli cold start problému) využívat kolaborativní metody. Místo nich byly testováno především doporučování na základě ObjectRatingu a na základě podobnosti atributů (nicméně kolaborativní metody byly také vyzkoušeny na konci testovacího období tak, aby měly k dispozici maximální možné množství uživatelských dat).

Testování metod pro výpočet top-k objektů bylo rozděleno do 2 částí: testování doporučování v rámci katalogu knih a testování doporučování na stránkách detailu knihy (v obou případech doporučováno 8 objektů). Rozdíl je jednak v informaci o aktuálním objektu (která umožňuje použití širšího spektra metod), jiném umístění doporučení v rámci stránky (požadavek provozovatele webu) a také v jiné sémantice doporučení (related vs. recommended). Předpokládáme, že výsledky obou částí jsou kvůli zmíněným důvodům vzájemně neporovnatelné.

Obrázek 20: Screenshot - doporučování knih v Antikvariátu Ichtys

Metody pro doporučování objektů v katalogu (recommended)

- **Dummy:** základní metoda, která vybírá doporučené objekty náhodně ze zadané množiny. Zadaná množina jsou právě všechny knihy ze zobrazené kategorie katalogu. Dummy metoda reprezentuje na jedné straně nejlepší jednoduché řešení, které může implementovat provozovatel webu bez znalosti uživatelských preferencí. Na druhé straně představuje Dummy metoda minimální výkon (v měřených veličinách), kterého by měla dosáhnout každá sofistikovanější metoda. **Dummy** metoda byla na webu spuštěna s následujícími parametry:

```
//ObjectExpression($methodType, $importance, $methodName, $methodParameters )
ObjectExpression("ObjectRating", "1", "Dummy", array("noOfObjects"=>8) )
```

- **ObjectRating:** ObjectRating představuje jednoduchou metodu nezávislou na aktuálním uživateli, která ze základní množiny doporučuje nejlépe hodnocené objekty. Základní množina obsahuje stejně jako u Dummy všechny knihy ze zobrazené kategorie katalogu. Hodnocení objektu je závislé na 2 složkách, kde největší váhu má poměr (#otevření objektu ze seznamu / #zobrazení objektu v seznamu), nižší pak #detailních prohlídek objektu uživatelem. Do hodnocení byl zahrnut i náhodný prvek s nízkou váhou (v případě rovnosti hodnocení, nebo nemožnosti předpovědi kvůli nedostatku dat). **ObjectRating** byla na webu spuštěna s následujícími parametry:

```
ObjectExpression("ObjectRating", "6", "Aggregated", array("noOfObjects"=>24,
        "aggregatedEventsList"=>array("opened_vs_shown_fraction") ) ),
ObjectExpression("ObjectRating", "3", "Standard", array("noOfObjects"=>24,
        "implicitEventsList"=>array("deep_pageview") ) ),
ObjectExpression("ObjectRating", "1", "Dummy", array("noOfObjects"=>8) )
```

- **Collaborative**: Základem Collaborative metody je klasické kolaborativní filtrování založené na k-NN (k-nearest neighbours). S menší váhou je pak v celkovém výsledku zastoupen ObjectRating podle poměru (#otevření/#zobrazení objektů) jako v předchozí metodě a

náhodný prvek (obojí především kvůli možnému nedostatečnému množství výsledků z hlavní metody). Základní množina obsahuje stejně jako u předchozích metod všechny knihy ze zobrazené kategorie katalogu.

Pro výpočet k-NN jsou použity 2 metody: Standard založená na podobnosti implicitních zpětných vazeb uživatelů a Dummy, která vybere uživatele náhodně (opět, Dummy metoda je použita především kvůli obavě z možného nedostatečného množství výsledků, její váha je nižší). **Collaborative** byla na webu spuštěna následovně:

Výpočet k-NN:

```
UserExpression("UserSimilarity", "5", "Standard", array("userID"=>UPCompCore::getUserId(),
        "noOfUsers"=>5, "implicitEventsList"=>array("pageview","deep_pageview","order") ) ),
UserExpression("UserSimilarity", "1", "Dummy", array("userID"=>UPCompCore::getUserId(),
        "noOfUsers"=>3 ) )
```

Výpočet top-k objektů:

```
ObjectExpression("Collaborative", "6", "Standard", array( "noOfObjects"=>24,
        "implicitEventsList"=>array("pageview","deep_pageview","order") ) ),
ObjectExpression("ObjectRating", "3", "Aggregated", array("noOfObjects"=>24,
        "aggregatedEventsList"=>array("opened_vs_shown_fraction") ) ),
ObjectExpression("ObjectRating", "1", "Dummy", array("noOfObjects"=>8) )
```

Metody pro doporučování objektů na detailu objektu (related)

- **DummySimilarity:** základní metoda, která vybírá podobné objekty náhodně ze základní množiny. Základní množina jsou knihy ze stejné kategorie, jako aktuální kniha. **DummySimilarity** metoda byla na webu spuštěna s následujícími parametry:

```
ObjectExpression("ObjectSimilarity", "1", "Dummy", array("objectID"=><id_knihy>,
        "noOfObjects"=>8) ),
```

- **AttributesSimilarity**: jednoduchá metoda nezávislá na aktuálním uživateli, která porovnává podobnost jednotlivých atributů knih ze základní množiny. Na výsledném hodnocení se s menší vahou podílí také podobnost objektů na základě podobnosti implicitních akcí uživatele na objektech a náhodný prvek. Zadaná množina jsou knihy ze stejné kategorie, jako aktuální kniha. **AttributesSimilarity** byla použita s následujícími parametry:

Atributy, na kterých byla měřena podobnost:

```
//MethodIntegerAttribute($attributeName, $tolerance, $importance),
        AttributeFactory::MethodIntegerAttribute("`knihy`.`id_typ`", 0, 3),
        AttributeFactory::MethodIntegerAttribute("`knihy`.`cena`", 300, 1),
//MethodStringMatchAttribute($attributeName, $importance),
        AttributeFactory::MethodStringMatchAttribute("`knihy`.`nazev`", 3),
        AttributeFactory::MethodStringMatchAttribute("`knihy`.`autor`", 7)
```

Výpočet top-k objektů:

```
ObjectExpression("ObjectSimilarity", "6", "Attributes", array("objectID"=><id_knihy>,
        "noOfObjects"=>40, "attributesSelectionSQL"=><porovnávané atributy>,
```

```
        "attributesList"=><SQL dotaz pro získání atributů>) ),
ObjectExpression("ObjectSimilarity", "3", "Standard", array("objectID"=><id_knihy>,
        "noOfObjects"=>8, "implicitEventsList"=>array("pageview","deep_pageview") ) ),
ObjectExpression("ObjectSimilarity", "1", "Dummy", array("objectID"=><id_knihy>,
        "noOfObjects"=>8) ),
```

- **Collaborative+ObjectSimilarity**: poslední testovaná metoda je velmi podobná **Collaborative** metodě pro doporučování objektů v katalogu. Jediný rozdíl je, že místo **ObjectRating** jako pomocného kritéria je zvolena **ObjectSimilarity** stejně jako u metody **AttributeSimilarity**.

Výpočet top-k objektů:

```
ObjectExpression("Collaborative", "6", "Standard", array( "noOfObjects"=>24,
        "implicitEventsList"=>array("pageview","deep_pageview","order") ) ),
ObjectExpression("ObjectSimilarity", "3", "Standard",  array( "objectID"=><id_knihy>,
        "noOfObjects"=>24, "implicitEventsList"=>array("pageview","deep_pageview") ) ),
ObjectExpression("ObjectRating", "1", "Dummy", array("noOfObjects"=>8) )
```

7.4.2 Metody použité ve slantour.cz

Při implementaci UPComp do rezervačního systému CK SLAN tour se vyskytl opačný problém, než u Antikvariátu Ichtys: Vysoké množství uživatelů a uživatelských zpětných vazeb způsobilo neúměrně dlouhý běh metod závislých na výpočtu podobnosti uživatelů (k-NN, k-Nearest Neighbours) – více ***viz 7.2*** . Tato skutečnost byla zjištěna až v průběhu testování výkonnosti metod a mohla se negativně promítnout na přesnosti jeho výsledků. Mezi rozšíření diplomové práce by tak mělo patřit i testování upravených metod, které dosahují přijatelné doby výpočtu (***viz návrhy metod na konci kapitoly 7.2***).

Obdobně jako u Antikvariátu Ichtys bylo testování rozděleno do 2 nezávislých částí: testování metod pro doporučování v katalogu zájezdů (recommended, doporučeny 3 zájezdy) a testování metod pro doporučování na stránce s detailem zájezdu (related, doporučeny 4 zájezdy).

Obrázek 21: Screenshot - doporučování objektů v CK SLAN tour

Metody pro doporučování objektů v katalogu (recommended)

- **Dummy**: Dummy je obdobná metoda, jako byla použita u Antikvariátu Ichtys:Provádí náhodný výběr z objektů ze základní množiny, základní množina je tvořena pouze objekty z aktuální kategorie katalogu. Předpokládáme, že každá použitá sofistikovanější metoda by měla být prokazatelně lepší, než **Dummy**.

```
ObjectExpression("ObjectRating", "1", "Dummy", array("noOfObjects"=>3) )
```

- **ObjectRating**: ObjectRating je jednoduchá metoda nezávislá na aktuálním uživateli. Ze základní množiny doporučuje nejlépe hodnocené objekty. Základní množinu tvoří opět objekty aktuální kategorie katalogu. Oproti verzi použité v Antikvariátu je rating objektu počítán i z počtu objednávek (u antikvariátu je kniha po první objednávce typicky prohlášena za prodanou a dále se nenabízí):

```
ObjectExpression("ObjectRating", "6", "Aggregated", array("noOfObjects"=>15,
        "aggregatedEventsList"=>array("opened_vs_shown_fraction","object_ordered") ) ),
ObjectExpression("ObjectRating", "3", "Standard", array("noOfObjects"=>15,
        "implicitEventsList"=>array("deep_pageview") ) ),
ObjectExpression("ObjectRating", "1", "Dummy", array("noOfObjects"=>3) )
```

- **Collaborative**: Collaborative je první varianta metody používající kolaborativní filtrování (a hledání k-nearest neighbours, k-NN). **Collaborative** využívá pro hledání podobných uživatelů metodu Standard (***viz 4.5***). Na výpočtu výpočet top-j objektů se pak s nejvyšší důležitostí podílí vážený rating objektů od k nejbližších sousedů (míra podobnosti = váha, **Collaborative** typ výpočtu v metodě **Standard**), s nižší pak **ObjectRating** na základě počtu objednávek a poměru (#otevření/#zobrazení objektu v seznamu) a také náhodný faktor (metoda **Dummy**). Oproti předchozím příkladům tvoří základní množinu metody Collaborative všechny objekty systému – bez omezení na aktuální kategorii.

Výpočet k-NN:

```
UserExpression("UserSimilarity", "3", "Standard", array("userID"=>ComponentCore::getUserId(),
"noOfUsers"=>10, "implicitEventsList"=>array("pageview","deep_pageview","order") ) ),
```

Výpočet top-j objektů:

```
ObjectExpression( "Collaborative", "6", "Standard", array( "noOfObjects"=>15,
"implicitEventsList"=>array("pageview","deep_pageview","order") ) ),
ObjectExpression("ObjectRating", "3", "Aggregated", array("noOfObjects"=>15,
"aggregatedEventsList"=>array("opened_vs_shown_fraction","object_ordered") ) ),
ObjectExpression("ObjectRating", "1", "Dummy", array("noOfObjects"=>3) )
```

- **Collaborative with Attributes**: Nepříliš dobré výsledky metody **Collaborative** (***viz 7.5***) vedly k návrhu 2 dalších obdobných metod založených na kolaborativním filtrování. Předpokladem bylo, že hlavním problémem **Collaborative** metody je příliš široká základní množina objektů. **Collaborative with Attributes** sice základní množinu jako takovou neomezuje, ale přiřazuje jednotlivým objektům (pomocí **Attributes** rozšíření SQL části dotazu) rating podle toho, zda jsou objekty obsaženy v aktuální kategorii (nebo alespoň v její nadkategorii). Tento rating má pak nejvyšší váhu ve výpočtu výsledných top-j objektů (přednostně tedy budou zobrazovány objekty ze současné kategorie, i když je možné, že některé jiné - velmi dobře hodnocené - objekty je mohou přeskočit). Výpočet k-NN je stejný, jako u **Collaborative**.

Attributes rozšíření SQL:

```
AttributeFactory::StringAttribute("`typ`.`nazev_typ_web`", <aktuální kategorie: typ zájezdu>, 1)
AttributeFactory::StringAttribute("`zeme`.`nazev_zeme_web`", <aktuální kategorie: země>, 1);
```

Výpočet top-j objektů:

```
<do výpočtu top-j objektů je krom zmíněných metod započítán i rating objektů na základě Attributes s váhou 9>
ObjectExpression( "Collaborative", "6", "Standard", array( "noOfObjects"=>15, "implicitEventsList"=>array("pageview","deep_pageview","order") ) ),
ObjectExpression("ObjectRating", "3", "Aggregated", array("noOfObjects"=>15, "aggregatedEventsList"=>array("opened_vs_shown_fraction","object_ordered") ) ),
ObjectExpression("ObjectRating", "1", "Dummy", array("noOfObjects"=>3) )
```

- **Collaborative (Pearson Correlation) with Attributes**: Dalším možným důvodem, proč dosáhla **Collaborative** metoda nepřesvědčivých výsledků je špatný volba algoritmu pro výpočet podobných uživatelů. Z tohoto důvodu byla navržena metoda **Collaborative (Pearson Correlation) with Attributes**. Metoda vychází z **Collaborative with Attributes**, pouze používá pro určení k-NN Pearsonovu korelaci ratingu objektů od aktuálního uživatele a uvažovaných ostatních uživatelů. Zbytek výpočtu je v obou metodách shodný.

Výpočet k-NN:

```
UserExpression("UserSimilarity", "3", "PearsonCorrelation",
        array( "userID"=>ComponentCore::getUserId(), "noOfUsers"=>10,
        "implicitEventsList"=>array("pageview","deep_pageview","order") ) ),
```

Metody pro doporučování objektů na detailu objektu (related)

- **DummySimilarity**: Metoda DummySimilarity je shodná se stejnojmennou metodou použitou u Antikvariátu Ichtys:

```
ObjectExpression("ObjectSimilarity", "1", "Dummy", array("objectID"=><id_objektu>,
        "noOfObjects"=>4) )
```

- **ObjectSimilarity:** Metoda **ObjectSimilarity** je založena na výpočtu podobnosti objektů ze základní množiny k aktuálnímu objektu na základě podobnosti ratingů získaných od jednotlivých uživatelů (k výpočtu podobnosti byla použita **Standard** metoda, ***viz 4.5***). Základní množinu objektů představují všechny objekty v systému. Na výsledném výběru top-k se menší měrou podílely také **ObjectRating** na základě počtu objednávek a poměru (#otevření / #zobrazení objektu v seznamu) a náhodný faktor (metoda **Dummy**):

```
ObjectExpression("ObjectSimilarity", "6", "Standard", array("objectID"=><id_objektu>,
        "noOfObjects"=>20,
        "implicitEventsList"=>array("pageview","deep_pageview","order") ) ),
ObjectExpression("ObjectRating", "2", "Aggregated", array("noOfObjects"=>20,
        "aggregatedEventsList"=>array("opened_vs_shown_fraction","object_ordered") ) ),
ObjectExpression("ObjectRating", "1", "Dummy", array("noOfObjects"=>4) )
```

- **ObjectSimilarity with Attributes**: Výsledky **ObjectSimilarity** byly zklamáním (ve všech měřených hodnotách byla statisticky významně horší, než **DummySimilarity** – ***viz 7.5)***, proto byla (obdobně jako u **Collaborative**) navrhnuta metoda **ObjectSimilarity with Attributes**, která dává rating objektům ze základní množiny na základě toho, zda patří do

stejné kategorie či nadkategorie, jako aktuální objekt. Jinak je metoda shodná s **ObjectSimilarity**.

Attributes rozšíření SQL:

```
AttributeFactory::StringAttribute("`typ`.`nazev_typ_web`", <aktuální kategorie: typ zájezdu>, 1)
AttributeFactory::StringAttribute("`zeme`.`nazev_zeme_web`", <aktuální kategorie: země>, 1);
```

Výpočet top-k objektů:

```
<do výpočtu top-k objektů je krom zmíněných metod započítán i rating objektů na základě
Attributes s váhou 9>
ObjectExpression("ObjectSimilarity", "6", "Standard", array("objectID"=><id_objektu>,
        "noOfObjects"=>20,
        "implicitEventsList"=>array("pageview","deep_pageview","order") ) ),
ObjectExpression("ObjectRating", "2", "Aggregated", array("noOfObjects"=>20,
        "aggregatedEventsList"=>array("opened_vs_shown_fraction","object_ordered") ) ),
ObjectExpression("ObjectRating", "1", "Dummy", array("noOfObjects"=>4) )
```

- **AttributesSimilarity:** AttributesSimilarity doporučuje objekty na základě podobnosti jejich atributů. Na výpočtu top-k objektů se s menší vahou podílí i **ObjectSimilarity** na základě podobnosti ratingů získaných od jednotlivých uživatelů a náhodný faktor (metoda **Dummy**):

Uvažované atributy podobnosti objektů:

```
//MethodIntegerAttribute($atttributeName, $tolerance, $importance)
AttributeFactory::MethodIntegerAttribute("`serial`.`id_typ`", 0, 3),
AttributeFactory::MethodIntegerAttribute("COALESCE(`zeme_serial`.`id_zeme`)", 0, 2),
AttributeFactory::MethodIntegerAttribute("COALESCE(`destinace_serial`.`id_destinace`)", 0, 1),
AttributeFactory::MethodIntegerAttribute("COALESCE(`cena_zajezd`.`castka`)", 5000, 1)
```

Výpočet top-k objektů

```
ObjectExpression("ObjectSimilarity", "6", "Attributes", array("objectID"=><id_objektu>,
        "noOfObjects"=>20, "attributesList"=><atributy podobnosti objektů>,
        "attributesSelectionSQL"=><SQL dotaz pro získání atributů> ) ),
ObjectExpression("ObjectSimilarity", "3", "Standard", array("objectID"=><id_objektu>,
        "noOfObjects"=>20,
        "implicitEventsList"=>array("pageview","deep_pageview","order") ) ),
ObjectExpression("ObjectRating", "1", "Dummy", array("noOfObjects"=>4) )
```

7.5 Vyhodnocení testu výkonnosti metod

Test výkonnosti metod probíhal od 30.10.2010 do 7.12.2010. Každá z výše popsaných metod byla nasazena po dobu cca 1 týdne na příslušném prodejním webu a její výsledky byly zaznamenávány do tabulky testing. Důvod pro testování tímto způsobem byl především fakt, že výsledky jedné metody mohou ovlivnit (a ovlivnily) konstrukci další metody (zvolené parametry, váhy dílčích metod atp.) - průběžné zlepšování výsledků doporučování byl jedním z faktorů, který přesvědčil provozovatele prodejních webů, aby umožnili testování UPComp. Ze stejného důvodu bylo také po celou dobu testování spuštěno zaznamenávání uživatelských zpětných vazeb. Při testování byly nejprve nasazeny metody méně závislé na množství zpětné vazby (Dummy, ObjectRating) a až později metody, pro něž je množství zpětné vazby kritické (Collaborative).

Kompletní výsledky testování výkonnosti metod obsahuje ***příloha D*.**

Vzhledem k různým množinám zpětných vazeb se jako dostatečně průkazné se jeví především srovnání výsledků jednotlivých metod oproti Dummy (které jsou nezávislé na zpětné vazbě), výsledky srovnání ostatních metod mezi sebou by mělo raději ještě potvrdit testování na shodné množině zpětných vazeb.

Porovnávání jednotlivých metod bylo provedeno podle veličin definovaných v ***7.3*** (*OpenShown, OrderShShown, OrderOpShown*) ve statistickém programu R (viz *http://www.r-project.org/*). Pro účely porovnávání byl úspěch metody (např. kliknutí na nabízený objekt) byl interpretován jako „1“, neúspěch jako „0“. Pro každou dvojici porovnávaných metod a každou porovnávanou veličinu byl nejprve proveden F-test pro zjištění shodnosti rozptylu, dále pak dvouvýběrový oboustranný t-test pro určení, zda je mezi výsledky metod v příslušné veličině statisticky významný rozdíl. Pokud byla zamítnuta hypotéza rovnosti veličin, byl ještě proveden jednostranný t-test. Hladina významnosti testů byla stanovena na 95%. *Pokud nebyla zamítnuta nulová hypotéza, je ve výsledcích je uvedena p-hodnota oboustranného t-testu, v opačném případě p-hodnota jednostranného.*

7.5.1 Testování na Antikvariátu Ichtys

U testování na Atikvariátu Ichtys jsme se obecně potýkali s nedostatkem dat – jak množšství zpětných vazeb, tak i počtu objednávek knih. Přesto jsou výsledky porovnávání metod vcelku povzbudivé:

- U žádné pokročilejší metody se v žádné z veličin neprokázalo, že by její výsledky byly horší, než výsledky **Dummy** (**DummySimilarity**)
- Každá z použitých metod dosáhla statisticky významně lepších výsledků než **Dummy** (**DummySimilarity**) alespoň v jedné sledované veličině (metoda **Collaborative** ve všech; p-values: 0.01811, 0.001585 a 0.02376).
- Porovnávání pokročilejších metod mezi sebou nepřineslo statisticky významné výsledky (s výjimkou **ObjectSimilarity** vs. **AttributesSimilarity**: **ObjectSimilarity** je lepší v *OrderShShown,* p-value:0.01426)

Jako nejlepší metodu (z testovaných) pro doporučování objektů v katalogu jsme vyhodnotili **Collaborative**, pro doporučování podobných objektů pak **ObjectSimilarity**.

7.5.2 Testování na CK SLAN tour

Porovnávání metod pro výpočet podobných objektů

Porovnávání metod na CK SLAN tour přineslo překvapivé výsledky především u doporučování podobných objektů: žádná z pokročilých testovaných metod nedosáhla v žádné veličině statisticky významně lepších výsledků než **DummySimilarity**. Naopak **DummySimilarity** každou použitou metodu alespoň v 1 sledované veličině statisticky významně předstihla. Tento výsledek je překvapivý mimo jiné i proto, že obdobné metody nasazené na Antikvariátu Ichtys dopadly v porovnání s Dummy výrazně lépe. Test metod doporučujících podobné objekty zopakujeme v rámci rozšíření práce pro ověření zjištěných výsledků.

Porovnávání metod pro doporučování objektů v katalogu

Metoda **ObjectRating** přinesla statisticky významně lepší výsledky než **Dummy** ve všech sledovaných veličinách (p-values:0.02447, 2.430e-05 a 0.004133). Metody založené na kolaborativním filtrování většinou dopadly hůře než **Dummy** v *OpenShown* (**Collaborative**, **Collaborative with Attributes**; p-values: < 2.2e-16 a 0.008134) a naopak lépe v *OrderShShown* a **Collaborative with Attributes** i v *OrderOpShown* (p-value: 0.003649). Důvodem pro malý počet

objektů otevřených ze seznamu může být i dlouhá doba výpočtu top-k objektů (*objekty se sice po čase zobrazí, pozornost uživatele je už ale obrácena jinam*). Jako rozšíření práce by proto mělo být provedeno testování upravených kolaborativních metod s kratší dobou výpočtu.

Mezi příjemné překvapení naopak patří, že většina objednaných objektů se objevila i v doporučených objektech kolaborativních metod (**Collaborative**: 85%, **Collaborative with Attributes** a **Collaborative (Pearson Correlation)**: 76%).

Při vzájemném porovnávání pokročilejších metod dosáhla **ObjectRating** statisticky významně lepších výsledků v *OpenShown*, než všechny kolaborativní metody. Porovnání kolaborativních metod mezi sebou statisticky významné výsledky nepřineslo.

Jako nejlepší metodu (z testovaných) pro doporučování objektů v katalogu jsme vyhodnotili **ObjectRating**, pro doporučování podobných objektů pak **DummySimilarity**.

8 Závěr

V úvodních kapitolách jsme čtenáře seznámili s pojmem uživatelská preference, zpětná vazba a prodejní web. Uvedené pojmy jsme (s využitím další literatury) popsali z různých aspektů jejich dělení, sběru, ukládání a především možnosti jejich využití v prodejních webech.

V dalších kapitolách jsme představili návrh nezávislé komponenty pro doporučování objektů v prodejních webech na základě uživatelské zpětné vazby.

Tato komponenta byla v rámci diplomové práce implementována a nasazena na dva prodejní weby (*antikvariat-ichtys.cz* a *slantour.cz*). Testování komponenty prokázalo její schopnost doporučovat uživatelům vhodné objekty a přínosnost pro prodejní weby (i když výsledky některých metod nedopadly zcela podle očekávání a bude třeba další práce na zdokonalování použitých metod).

Mezi problémy či nedostatky práce patří špatné výsledky některých testovaných metod v rychlosti výpočtu doporučovaných objektů (byly navrženy varianty, jak dané metody upravit) a nedostatečná podpora UPComp pro tvorbu výsledků vyhledávání (*viz Možná rozšíření práce*). Jako drobný neúspěch je možné chápat i absenci explicitních zpětných vazeb od uživatelů během testování komponenty.

8.1 Možná rozšíření práce

Další vývoj či rozšíření práce se může ubírat několika směry:

- **Implementace sofistikovaných metod pro výpočet uživatelské preference:** všechny implementované a testované metody UPComp patří spíše do kategorie jednoduchých. Zajímavá by proto mohla být implementace některých pokročilejších metod (SVM, Multilayer perceptron...) a jejich porovnání z hlediska rychlosti a přesnosti odpovědí.

  Testování rychlosti současných metod navíc odhalilo problémy metod založených na podobnosti uživatelů. V rámci práce bylo navrhnuto několik možných řešení, jak metody upravit. Implementace a testování upravených metod by mohla být také přínosná.

- **Implementace a testování UPComp na dalších prodejních webech:** vhodné by bylo především nasazení a testování UPComp na klasickém e-shopu. Ideálně takovém, který již implementuje (a jeho uživatelé dostatečně využívají) nějakou formu získávání explicitní zpětné vazby.

- **Rozšíření dotazu do UPComp pro** použití v rámci **vyhledávání podle atributů**, nebo fulltextového vyhledávání. Při aplikaci UPComp jako nástroje pro zpracování vyhledávání bychom narazili na problém především u setřídění (Order by) objektů, zjištění celkového počtu vyhovujících objektů atp. Vhodným rozšířením tříd Query a Database je možné tyto

nedostatky odstranit.

- **Zlepšení povědomí provozovatelů prodejních webů o uživatelských preferencích**: informování široké veřejnosti je vždy během na dlouhé trati. Nicméně věříme, že je to správná cesta pro další vývoj a rozšíření použití doporučovacích systému na prodejních webech.
- **Formulace a ověřování hypotéz o uživatelských preferencích a prodejních webech:** v rámci diplomové práce byla prezentována kolekce reálných uživatelských dat uživatelů CK SLAN tour. Především další výzkum a testování dobrých implicitních faktorů by mohl být do budoucna velmi přínosný. Rozšíření kolekce dat, nebo úprava získávaných informací by mohla při zmíněném výzkumu také pomoci.

9 Použitá literatura

[Bro02] Andrei Broder: A taxonomy of web search. ACM SIGIR Forum Volume 36 Issue 2, Fall 2002, 3 – 10. ACM 2002. URL:*http://portal.acm.org/citation.cfm?id=792550.792552*

[CLWB01] Mark Claypool, Phong Le, Makoto Waseda, David Brown: Implicit interest indicators. IUI '01, 33-40. ACM, 2001. URL:*http://portal.acm.org/citation.cfm?id=359836*

[Cia10] Paolo Ciaccia: Preference Relations, URL: *http://www-db.deis.unibo.it/courses/SI-LS/slides/04_PreferenceRelations.pdf*

[EHV07] A.Eckhardt, T. Horváth, P. Vojtáš: .PHASES: A user Profile Learning Aproach for Web Search. WI '07, 780-783. IEEE Computer Society Washington, 2007. URL:*http://portal.acm.org/citation.cfm?id=1331740.1331829*

[EV09a] Alan Eckhardt, Peter Vojtáš: Combining Various Methods of Automated User Decision and Preferences Modelling. MDAI '09 172-181. Springer-Verlag Berlin, Heidelberg, 2009. URL:*http://portal.acm.org/citation.cfm?id=1695077*

[EV09b] Alan Eckhardt, Peter Vojtáš: How to learn fuzzy user preferences with variable objectives. IFSA/EUSFLAT Conf. 2009: 938-943. URL:*http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.156.4647*

[HB10] Michal Holub, Maria Bielikova: Estimation of User Interest in Visited Web Page. Proc. of Int. Conf. on World Wide Web WWW'10, 1111-1112. ACM Press, 2010. URL:*http://portal.acm.org/citation.cfm?id=1772690.1772829*

[HKV08] Yifan Hu, Yehuda Koren, Chris Volinsky: Collaborative Filtering for Implicit Feedback Datasets. ICDM '08 263-272. IEEE Computer Society Washington, 2008. URL:*http://portal.acm.org/citation.cfm?id=1510528.1511352*

[JSK10a] Gawesh Jawaheer, Martin Szomszor,Patty Kostkova: Characterisation of explicit feedback in an online music recommendation service. RecSys '10, 317-320. ACM, 2010. URL:*http://portal.acm.org/citation.cfm?id=1864708.1864776*

[JSK10b] Gawesh Jawaheer, Martin Szomszor,Patty Kostkova: Comparison of implicit and explicit feedback from an online music recommendation service. HetRec '10, 47-51. ACM, 2010. URL:*http://portal.acm.org/citation.cfm?id=1869446.1869453*

[Kie02] Werner Kießling: Foundations of Preferences in Database Systems. VLDB '02, 311-322. VLDB Endowment, 2002. URL:*http://portal.acm.org/citation.cfm?id=1287369.1287397*

[KT03] Diane Kelly, Jaime Teevan: Implicit feedback for inferring user preference: a bibliography. ACM SIGIR Forum, Volume 37 Issue 2, Fall 2003, 18-28. ACM, 2003. URL:*http://portal.acm.org/citation.cfm?id=959258.959260*

[Pes08] Ladislav Peška: Rezervační a informační systém cestovní kanceláře. Bakalářská práce na MFF UK, Správa Informatické Sítě A Laboratoří, 2008.

[SKKR01] Badrul Sarwar, George Karypis, Joseph Konstan, John Reidl: Item-based collaborative filtering

recommendation algorithms. WWW '01, 285-295. ACM, 2001. URL:http://portal.acm.org/citation.cfm?id=372071

[Vac10] Bronislav Václav: Modely uživatelských preferencí v prostředí webovských obchodů. Diplomová práce na MFF UK, Katedra softwarového inženýrství 2010.

[Wiki10a] Wikipedia: World Wide Web. URL:http://sk.wikipedia.org/wiki/World_Wide_Web

[Wiki10b] Wikipedia: E-shop. URL:http://cs.wikipedia.org/wiki/E-shop

[Wiki10c] Wikipedia: Recommender system. URL:http://en.wikipedia.org/wiki/Recommender_system

[Wiki10d] Wikipedia: Ajax (programming) URL:http://en.wikipedia.org/wiki/Ajax_(programming)

[Wiki10e] Wikipedia: Server side scripting. URL: http://en.wikipedia.org/wiki/Server-side_scripting

[Wiki10f] Wikipedia: Web crawler. URL: http://en.wikipedia.org/wiki/Web_crawler

[Wiki10g] Wikipedia: Client side scripting. URL:http://en.wikipedia.org/wiki/Client_side_scripting

[Wiki10h] Wikipedia: Cold start problem. URL:http://en.wikipedia.org/wiki/Cold_start

[Wiki10i] Wikipedia: Root mean square deviation. URL:http://en.wikipedia.org/wiki/RMSE

[Wiki10j] Wikipedia: Cron. URL: http://en.wikipedia.org/wiki/Cron

[Wiki10k] Wikipedia: Reverse engeneering. URL:http://en.wikipedia.org/wiki/Reverse_engineering

[Wiki10l] Wikipedia: Fuzzy logic. URL:http://en.wikipedia.org/wiki/Fuzzy_logic

[WJR01] Ryen W. White, Joemon M Jose, I. Ruthven: Comparing explicit and implicit feedback techniques for Web retrieval: TREC-10 interactive track report. TREC 2001.
URL:http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.66.4679

[Zak10] Vladimír Žák: Analýza chování uživatele na webových stránkách. Diplomová práce na MFF UK, Katedra softwarového inženýrství 2010.

10 Přílohy

A Obsah CD

Na přiloženém disku se nachází:

- Zdrojový kód aplikace v adresáři *DP/UPComp/*
- Tato práce ve formátu PDF: *docs/dp.pdf*
- Programátorská dokumentace (doxygen) v *docs/doxygen/*
- Soubor pro vytvoření tabulek databáze: *DP/createUPCompDb.sql*
- Podklady pro testování UPComp:
  - získané implicitní / explicitní a agregované zpětné vazby (export SQL tabulek) v *test/feedback/*
  - zaznamenané výsledky jednotlivých metod (export SQL tabulek) v *test/methods/*
  - výsledky testu rychlosti metod v *test/speed/*
  - výsledky testu výkonnosti metod v *test/performance/*
  - Podklady pro testování hypotéz v *test/hypothesis/*
- Ukázky zdrojových kódů v adresáři *samples/*

B Tabulky databáze

UPComp využívá pro svojí práci 3 tabulky pro ukládání zpětné vazby (standardně nazvané *implicit_events*, *explicit_events*, *aggregated_events*), dále (pokud také provádí identifikaci uživatelů) tabulku *users*. Pro testování výkonnosti metod navíc tabulku *testing*.

Sloupec	Typ	Výchozí hodnota	Extra
id	int(11)	*Žádná*	Auto_increment
userID	int(11)	0	
objectID	int(11)	0	
eventName	text	*Žádná*	
eventValue	double	0	

Tabulka 21: tabulky implicit_events = explicit events

Sloupec	Typ	Výchozí hodnota	Extra
objectID	int(11)	0	
eventName	text	*Žádná*	
eventValue	double	0	

Tabulka 22: tabulka aggregated_events

Sloupec	Typ	Výchozí hodnota	Extra
id_user	int(11)		Auto_increment
name	text	*Žádná*	

Tabulka 23: tabulka users

Sloupec	Typ	Výchozí hodnota	Extra
id	int(11)	*Žádná*	Auto_increment
userID	int(11)	0	
objectID	int(11)	0	
eventName	text	*Žádná*	
position	text	*Žádná*	

Tabulka 24: tabulka testing

C Dotazy do UPComp k jednotlivým metodám

Vzhledem k rozsahu textu jsou konkrétní UPComp dotazy použité při testování uloženy na přiloženém CD ve složce *test/queries* v souborech *antikvariat.php* a *slantour.php*.

D Kompletní výsledky testů výkonnosti metod

Antikvariát Ichtys: testování výkonnosti metod

	USERS	SHOWN	OPEN	ORDER	OrderSH	OrderOp	OPEN/ SHOWN	OrderSH/ SHOWN	OrderOP/ SHOWN	OPEN/ SHOWN	OrderSH/ SHOWN	OrderOP/ SHOWN
Doporučené objekty	# Uživatelů	# Zobrazených objektů v seznamu	# Otevřených objektů ze seznamu	#Objednávek	# Objednávek objektů zobrazených v seznamu	# Objednávek objektů otevřených ze seznamu				Ratio against Dummy		
Dummy	188	7744	32	8	2	1	0,004132	0,000258	0,000129			
ObjectRating	621	25173	142	42	26	13	0,005641	0,001033	0,000516	1,37	4	4
Collaborative	199	8565	56	26	15	7	0,006538	0,001751	0,000817	1,58	6,78	6,33
Související objekty												
DummySimilarity	438	6044	33	25	2	0	0,005460	0,000331	0,000000			
AttributesSimilarity	435	6224	22	31	11	5	0,003535	0,001767	0,000803	0,65	5,34	#DIV/0!
ObjectSimilarity	143	4224	22	26	17	3	0,005208	0,004025	0,000710	0,95	12,16	#DIV/0!

Antikvariát Ichtys: výsledky testování hypotéz

Doporučené objekty	**OPEN/Shown**				**OrderSH/SHOWN**				**OrderOP/SHOWN**			
	stejný rozptyl	T-test hypotéza (lepší metoda)	Výsledek t- testu	P-value	stejný rozptyl	T-test hypotéza (lepší metoda)	Výsledek t- testu	P-value	stejný rozptyl	T-test hypotéza (lepší metoda)	Výsledek t- testu	P-value
Dummy X ObjectRat	1	ObjRating	ne	0.1093	1	ObjRating	OK	0.02044	1	ObjRating	Ne	0.1481
Dummy X Coll.	1	Collaborative	OK	0.01811	1	Collaborative	OK	0.001585	1	Collaborative	OK	0.02376
ObjRat X Coll.	1	Collaborative	Ne	0.3477	1	Collaborative	Ne	0.09924	1	Collaborative	Ne	0.3242

Související objekty	**OPEN/Shown**				**OrderSH/SHOWN**				**OrderOP/SHOWN**			
	stejný rozptyl	T-test hypotéza (lepší metoda)	Výsledek t- testu	P-value	stejný rozptyl	T-test hypotéza (lepší metoda)	Výsledek t- testu	P-value	stejný rozptyl	T-test hypotéza (lepší metoda)	Výsledek t- testu	P-value
Dummy X AttrSim	1	Dummy	Ne	0.1106	1	AttrSim	OK	0.007245	1	AttrSim	OK	0.01376
Dummy X ObjSim	0	Dummy	Ne	0.8494	1	ObjSim	OK	9.055e-06	1	ObjSim	OK	0.01913
ObjSim X AttrSim	1	ObjSim	Ne	0.2004	1	ObjSim	OK	0.01426	1	AttrSim	Ne	0.866

CK SLAN tour: testování výkonnosti metod

	USERS	SHOWN	OPEN	ORDER	OrderSH	OrderOp	OPEN/ SHOWN	OrderSH/ SHOWN	OrderOP/ SHOWN	OPEN/ SHOWN	OrderSH/ SHOWN	OrderOP/ SHOWN
Doporučené objekty	# Uživatelů	# Zobrazených objektů v seznamu	# Otevřených objektů ze seznamu	#Objednávek	# Objednávek objektů zobrazených v seznamu	# Objednávek objektů otevřených ze seznamu				Poměr oproti Dummy		
Dummy	2828	40997	836	66	16	6	0,020392	0,000390	0,000146			
ObjectRating	3310	45128	1008	94	53	21	0,022336	0,001174	0,000465	1,1	3,01	3,18
Collaborative	2680	37449	409	67	57	13	0,010922	0,001522	0,000347	0,54	3,9	2,37
Collaborative w. Attr.	2395	34971	629	58	44	17	0,017986	0,001258	0,000486	0,88	3,22	3,32
Coll. Pearson w. Attr.	2892	40176	754	58	44	14	0,018767	0,001095	0,000348	0,92	2,81	2,38
Související objekty												
Dummy	4740	43509	385	88	64	7	0,008849	0,001471	0,000161			
ObjectSimilarity	4618	47756	349	67	22	3	0,007308	0,000461	0,000063	0,83	0,31	0,39
ObjSim. w. Att.	4568	47352	429	58	21	5	0,009060	0,000443	0,000106	1,02	0,3	0,66
AttributesSimilarity	5083	54088	420	58	19	4	0,007765	0,000351	0,000074	0,88	0,24	0,46

CK SLAN tour: výsledky testování hypotéz

Doporučené objekty	**OPEN/Shown**				**OrderSH/SHOWN**				**OrderOP/SHOWN**			
	stejný rozptyl	T-test hypotéza (lepší metoda)	Výsledek t- testu	P-value	stejný rozptyl	T-test hypotéza (lepší metoda)	Výsledek t- testu	P-value	stejný rozptyl	T-test hypotéza (lepší metoda)	Výsledek t- testu	P-value
Dummy X ObjectRat	1	ObjRating	OK	0.02447	1	ObjRating	OK	2.430e-05	1	ObjRating	OK	0.004133
Dummy X Collaborative	1	Dummy	OK	< 2.2e-16	1	Collaborative	OK	1.033e-07	1	Collaborative	ne	0.07106
Dummy X Coll w. Attr.	1	Dummy	OK	0.008134	1	Coll w. Attr.	OK	1.096e-05	1	Coll w. Attr.	OK	0.003649
Dummy X Coll Pearson	1	Dummy	ne	0.09499	1	Coll Pearson	OK	0.0001100	1	Coll Pearson	ne	0.0666
ObjRat X Collaborative	1	ObjRating	OK	< 2.2e-16	1	Collaborative	ne	0.08636	1	ObjRating	ne	0.2023
ObjRat X Coll.w.A.	1	ObjRat	OK	7.953e-06	1	Coll . w. A	ne	0.7354	1	Coll . w. A	ne	0.8935
ObjRat X Coll.P.	1	ObjRat	OK	0.0001269	1	ObjRat	ne	0.7316	1	ObjRat	ne	0.4001
Coll.w.A. X Coll.P.	1	Coll. P	ne	0.4268	1	Coll . w. A	ne	0.5146	1	Coll . w. A	ne	0.354

Související objekty	**OPEN/Shown**				**OrderSH/SHOWN**				**OrderOP/SHOWN**			
	stejný rozptyl	T-test hypotéza (lepší metoda)	Výsledek t- testu	P-value	stejný rozptyl	T-test hypotéza (lepší metoda)	Výsledek t- testu	P-value	stejný rozptyl	T-test hypotéza (lepší metoda)	Výsledek t- testu	P-value
Dummy X ObjSimilarity	1	Dummy	OK	0.004623	1	Dummy	OK	3.373e-07	1	Dummy	ne	0.1575
Dummy X ObjSim w.A.	1	ObjSim w.A.	ne	0.7359	1	Dummy	OK	2.084e-07	1	Dummy	ne	0.4687
Dummy X AttrSim	1	Dummy	ne	0.06282	1	Dummy	OK	1.223e-09	1	Dummy	ne	0.2035
ObjSim w.A. X AttrSim	1	ObjSim w.A.	OK	0.01197	1	ObjSim w.A.	ne	0.4605	1	ObjSim w.A.	ne	0.5935

E Kompletní výsledky testů rychlosti odpovědi

Antikvariát Ichtys: testování rychlosti odpovědi na dotaz do UPComp

Recomended			
	Collaborative	**ObjectRating**	**Dummy**
Hlavní stránka	0.17699694633484	0.26127600669861	0.11003088951111
	0.21022200584412	0.24694395065308	0.098733901977539
	0.22716999053955	0.20924806594849	0.098929882049561
Subkategorie katalogu	0.034975051879883	0.019328117370605	0.0079221725463867
	0.066660165786743	0.027195930480957	0.028495073318481
	0.037806034088135	0.021600008010864	0.017258167266846
	0.034140110015869	0.031559944152832	0.017037153244019

Related			
	Collaborative+ObSim.	**AttributesSimilarity**	**DummySimilarity**
Detail objektu	0.09521484375	0.18849802017212	0.10380601882935
	0.088866949081421	0.10518312454224	0.094153881072998
	0.080010890960693	0.11236500740051	0.11218905448914
	0.12563991546631	0.16940522193909	0.10115599632263

CK SLAN tour: testování rychlosti odpovědi na dotaz do UPComp

Recommended – hlavní strana						
	Coll. Pearson	Coll. w. Attribs.	Collaborative	ObjectRating	Dummy	UserSimilarity ONLY*
Častý uživatel (30 objektů, 6000 zpětných vazeb od podobných uživatelů)	8.34425115585	6.57889604568	8.53555607796	0.213379144669	0.0410079956055	7.20804905891
	7.71568608284	6.92479705811	9.8121778965	0.24786901474	0.0608339309692	6.28719091415
	7.8007569313	7.30650091171	8.987844944	0.305757045746	0.0528440475464	7.07711291313
	7.86661911011	6.56872010231	9.09178900719	0.254482030869	0.0362510681152	6.53430891037
Nový uživatel (žádné informace o zpětné vazbě)	0.0955579280853	0.0989921092987	0.15856385231	0.261390924454	0.0870180130005	
	0.0933599472046	0.0775599479675	0.128443002701	0.238642930984	0.0562651157379	
	0.11189198494	0.091206073761	0.14387178421	0.248774051666	0.053426027298	
	0.0797550678253	0.113656997681	0.149777889252	0.328373908997	0.0407598018646	
Běžný uživatel (5 objektů, 90 zpětných vazeb od podobných uživatelů)	0.496714115143	0.630079984665	0.593150854111	0.236577987671	0.0388040542603	0.483606100082
	0.575428962708	0.714385032654	0.769418001175	0.325412034988	0.0362250804901	0.671722888947
	0.512192964554	0.496795892715	0.564899921417	0.415601968765	0.0606071949005	0.342746019363
	0.595407962799	0.533095836639	0.733111143112	0.245622873306	0.0440490245819	0.597184896469

* testování pouze rychlosti výpočtu top-k uživatelů

Recommended – katalog zájezdů						
	Coll. Pearson	Coll. w. Attribs.	Collaborative	ObjectRating	Dummy	Randomized Coll.**
Častý uživatel	10.869093895	9.74672007561	7.37310504913	0.0526971817017	0.0159459114075	1.4832239151
	11.5313191414	7.77969312668	7.09076285362	0.0923299789429	0.0175759792328	1.32809185982
	10.1874890327	9.25678896904	8.00422811508	0.103750944138	0.0184609889984	1.29313993454
	9.45968198776	7.79130005836	7.6595518589	0.0721700191498	0.0222380161285	1.44844007492
Nový uživatel	0.155372858047	0.153678894043	0.0839228630066	0.020339012146	0.0261659622192	
	0.109629869461	0.181151151657	0.099231004715	0.0850610733032	0.0294070243835	Heuristics Coll.**
	0.190840005875	0.167429924011	0.0677101612091	0.0448160171509	0.0253429412842	0.958287000656
	0.220566987991	0.120564937592	0.0819041728973	0.0255069732666	0.0288290977478	0.653996944427
Běžný uživatel	0.584627866745	0.675673961639	0.672466039658	0.053699016571	0.0241289138794	0.590213060379
	0.913197994232	0.577287197113	0.531569957733	0.0545599460602	0.0261268615723	0.573520898819
	0.513552904129	0.915753126144	0.357557058334	0.057373046875	0.0152580738068	
	0.726919174194	0.559563159943	0.691243886948	0.0416839122772	0.0206990242004	

** testování upravených metod (s omezeným počtem zpětných vazeb a s aplikací heuristik na omezení uživatelů), testování pouze pro častého uživatele

Related				
	AttributesSimilarity	ObjectSim. w. Attr.	ObjectSimilarity	DummySimilarity
Častý uživatel	0.0269258022308	0.209318876266	0.227147102356	0.0705060958862
	0.0319979190826	0.46706199646	0.23979306221	0.0242490768433
	0.13454914093	0.392493963242	0.175736904144	0.0405869483948
	0.22097992897	0.3234770298	1.05586004257	0.0275559425354
Nový uživatel	0.0749089717865	0.275785923004	0.232434988022	0.0244870185852
	0.0892009735107	0.123351097107	0.269226789474	0.015702009201
	0.0765218734741	0.192130088806	0.27575302124	0.0218789577484
	0.162548065186	0.0974218845367	0.812685966492	0.062422990799
Běžný uživatel	0.185716152191	0.0967309474945	0.134080886841	0.0250480175018
	0.105027914047	0.175717115402	0.191926002502	0.0127310752869
	0.0749220848083	0.189150094986	0.110540866852	0.048150062561
	0.0756549835205	0.207953929901	0.86506986618	0.0289900302887