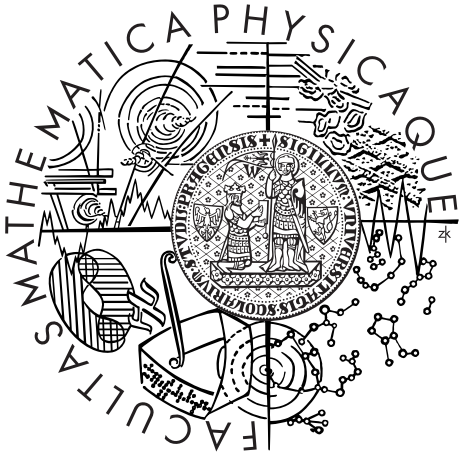
Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

DIPLOMOVÁ PRÁCE



Bc. Radek Strnad

Využití preferencí zájemců při obchodování s nemovitostmi

Katedra softwarového inženýrství

Vedoucí diplomové práce: RNDr. Michal Kopecký, Ph.D.

Studijní program: Informatika

Studijní obor: Softwarové systémy

Praha 2014

Rád bych touto cestou poděkoval RNDr. Michalu Kopeckému, Ph.D. za jeho cenné rady, dohled a trpělivost při vedení mé diplomové práce. Dále bych chtěl poděkovat jednateli realitní společnosti RealExpert s.r.o. panu Martinu Černému za vstřícnost a možnost nasadit obsah práce do ostrého provozu. Také bych rád poděkoval mým rodičům a přítelkyni za podporu během psaní.

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně a výhradně s použitím citovaných pramenů, literatury a dalších odborných zdrojů.

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

V Praze dne 3. prosince 2014 Radek Strnad

Název práce: Využití preferencí zájemců při obchodování s nemovitostmi

Autor: Radek Strnad

Katedra: Katedra softwarového inženýrství

Vedoucí diplomové práce: RNDr. Michal Kopecký, Ph.D., Katedra softwarového inženýrství

Abstrakt: V poslední době se rozdělení hráčů na realitním trhu, aspoň tom českém, příliš nemění. Statistická data potvrzují, že se nevyskytují ani výraznější vzestupné tendence objemu prodaných a pronajatých nemovitostí. Pokud chtějí společnosti obchodující s realitami zaujmout větší podíl na trhu, musí si zajistit konkurenční výhodu nad ostatními. Jednou z možností, jak zaujmout více potenciálních zákazníků, může být zrychlení vyhledávacího procesu u webové prezentace společnosti. V mnohých případech se jedná až o stovky či tisíce různých nabídek, kterými se zájemce musí probrodit, než najde několik vyhovujících.

Cílem diplomové práce je prozkoumat možnosti aplikace preferencí zájemců o obchodování s nemovitostmi. Jedná se zejména o zkoumání algoritmů doporučovacích systémů, jejich charakteristik a omezení. Autor vyhodnocuje použitelnost jednotlivých variant algoritmů a jejich funkčnost nad daty realitní kanceláře.

Kromě teoretické části je v práci elaborován realitní informační systém *RePort*, který je rozšířen o framework pro implementaci algoritmů doporučovacích systémů. Autor má k dispozici provozní data středně velké realitní společnosti, nad kterými si může ověřit správnost svých rozhodnutí. V rámci informačního systému *RePort* a nově vybudovaného frameworku si implementuje vzorový doporučovací algoritmus a zkoumá vliv interakcí zájemců na výsledky doporučení.

Klíčová slova: doporučovací systémy, nemovitosti, uživatelské preference, kolaborativní filtrování, informační systémy

Title: Using customer preferences in property market

Author: Radek Strnad

Department: The Department of Software Engineering

Supervisor: RNDr. Michal Kopecký, Ph.D., The Department of Software Engineering

Abstract: In recent years the market share of major real estate companies, at least the Czech ones, has not changed much. Statistical data don't reflect any significal upward trend in volumes of properties for rent or sale. In case the real estate company would like to access larger market share, they have to secure a competitive advantage over the others. One of the ways how to attract more potential customers might be speeding up the company website's property search process. In many cases the website visitors are facing hundreds or thousands of property offers before finding couple satisfactories.

The aim of the thesis is to explore possibilities of applicating customer preferences in property trading. The focus is put on research of recommender system algorithms, their characteristics and limtations. The author is evaluating usage of each algorithm variant and its suitability for a real world deployment in a real estate area.

Apart from the theoretical part of the work one can find a part, where real estate information system is extended with a framework for implementing recommendation system algorithms. The author is in possesion of production data of a medium sized real estate company. He uses the recommender system framework to build and evaluate example algorithm.

Keywords: recommender systems, properties, user preferences, collaborative filtering, information systems

Obsah

Úvod

Internetový trh s realitami je v současné době přesycen prezentacemi různého druhu. Návštěvník se může setkat s webovými stránkami jednotlivých realitních kanceláří, realitními portály, které agregují data od více společností, až po jednostranně zaměřené servery, nabízející konkrétní developerské projekty či specializující se na tržní niky.

Typická realitní kancelář nabízí nemovitosti hned na několika místech. Kromě tradičních tištěných médií se prezentuje i na Internetu. Exportuje své nabídky, kromě vlastní prezentace, rovnou i na několik inzertních serverů. To vede ke značné duplicitě dat a k prodloužení doby při nalézání vhodné nemovitosti zákazníkem, jelikož drtivá většina stránek řadí nabídky nemovitostí dle stejného vzorce.[1]

Řešením těchto nedostatků může být nasazení doporučovacího systému, který na základě několika vstupů – navštívených nemovitostí, oblíbených nemovitostí, vyplnění vyhledávacích formulářů, ohodnocení nemovitostí, atd. – dokáže vyhodnotit preference daného návštěvníka, nabídnout kvalitnější výsledky hledání a tím i získat konkurenční výhodu. Data získaná při analýze chování návštěvníků se mohou dále zužitkovat při nacenění podobných nemovitostí. Systém v tomto případě může doporučit korekci ceny vzhledem k předchozím případům.

Diplomová práce se zabývá problematikou doporučovacích systémů a jejich nasazení v prostředí prodeje realit. Činí tak po stránce teoretické i praktické. Cílem práce je dostatečně seznámit čtenáře s problematikou doporučovacích systémů, s jejich klady i zápory. Tyto vědomosti se přetaví do stavby frameworku pro doporučovací systémy nad existujícím realitním informačním systémem *RePort*, který vznikl dříve na Matematicko-fyzikální fakultě v rámci předmětu Softwarový projekt. Nad vystavěným frameworkem se na základě analýzy dat konkrétní realitní kanceláře implementuje vzorový, funkční doporučovací systém.

První kapitola práce seznámí čtenáře s principy doporučovacích systémů. Charakteristika je znázorněna na konkrétní internetové službě, která je v dostatečném povědomí uživatelů sítě a eventuální začlenění v prostředí realit. Druhá kapitola seznamuje s realitním informačním systémem *RePort*, jeho stavem po ukončení obhájení v rámci předmětu Softwarový projekt a nutných úpravach. Na projekt se nahlíží nejen z hlediska architektury, ale rovněž i z hlediska čistě funkčního. Debatuje se začlenění doporučovacího systému do právě představeného *RePortu*. Zároveň se čtenář dozvídá o konkurenčních systémech i mimo oblast realit. Třetí kapitola seznamuje s metodikami doporučovacích systémů, s jejich základním rozdělením, výhodami i nevýhodami. Následující, čtvrtá, kapitola zužitkovává nabyté znalosti k vystavění frameworku pro doporučovací systémy nad informačním systémem *RePort*. Zabývá se architekturou, ale i konkrétními aspekty implementace. Zmiňuje se i o implementaci konkrétních doporučovacích algoritmů, definování proměnných či sdílení komponent mezi sebou. Pátá kapitola implementuje v rámci *RePort recommender framework* vzorový algoritmus nad reálnými daty realitní kanceláře. Čtenář si osvojuje postup od zadání požadavků a omezeni, přes analýzu dat, výběru algoritmu až po návrh komponent a konkrétní imple-

[1] Nemovitosti se řadí zpravidla dle některého z parametrů nemovitosti (velikost, cena), případně data vložení

mentaci, včetně optimalizací. Předposlední kapitola představuje obecné metodiky hodnocení algoritmů doporučovacích systémů a aplikuje je na vzorový algoritmus. Seznámí čtenáře s výsledky vyhodnocení. Poslední kapitola sumarizuje dílčí úspěchy i neúspěchy celé práce a navrhuje zlepšení.

1. Doporučovací systémy a jejich uplatnění na trhu s realitami

Kapitola představí problém doporučovacích systémů na reálném příkladu, poukáže na výhody i nevýhody nasazení doporučovacích systémů v praxi, zejména té realitní.

1.1 Doporučovací systémy

Přemíra prezentovaných informací často vede k zahlcení uživatele a snížení uživatelského komfortu. Při vyhledávání určité položky i při poměrně přesném omezení podmínek může nastat situace, kdy je uživatel konfrontován s velkou množinou dat, kterou musí sám analyzovat. Tradiční metody vyhledávání řadí výsledky hledání dle exaktního vzorce, jenž je pro všechny návštěvníky stejný. Je pak pouze dílem náhody či trpělivosti dotyčného, zda nalezne uspokojující položku nebo nikoliv.

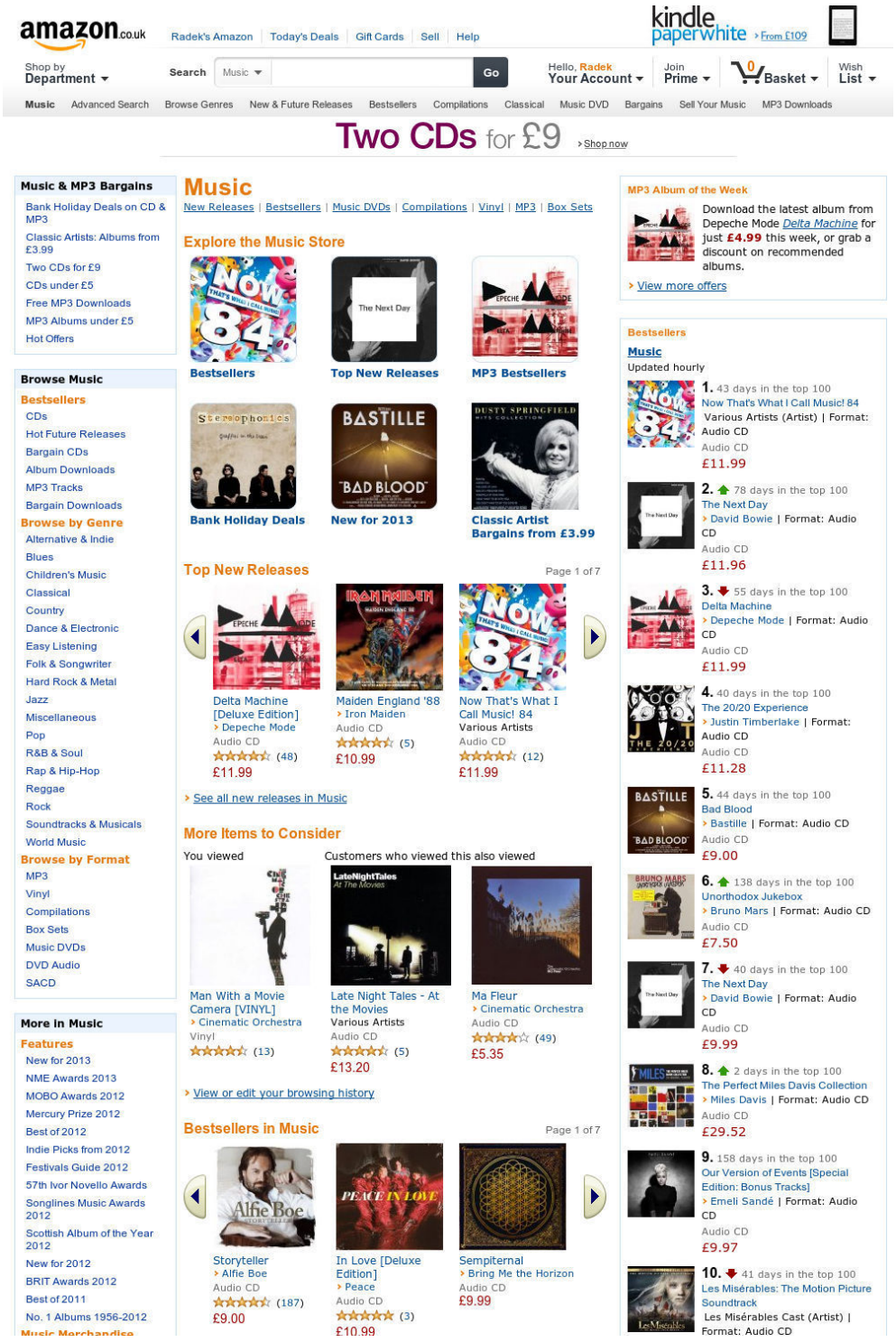
Myšlenka doporučovacích systémů vychází z principu subjektivního doporučení přátel. Pokud se rozhodneme pro zakoupení např. mobilního telefonu, můžeme analyzovat mnoho atributů přístroje jako velikost, cena, paměť, rychlost procesoru, operační systém, atd. Ale až osobní zkušenost dokáže identifikovat subjektivní atributy produktu, jako jsou odevza systému, obtížnost psaní na klávesnici, reálná výdrž baterie a jiné. Podobně i doporučovací systémy sbírají data o předchozím chování uživatelů a na základě analýzy jejich podobnosti s návštěvníkem, dohromady s jeho preferencemi a nastavenými omezeními, doporučí produkty. Výsledkem tohoto procesu jsou individuálně ohodnocené položky, které se pro každého návštěvníka dynamicky mění dle množství posbíraných dat od všech uživatelů systému.

Pro reálnou ilustraci můžeme využít oddělení hudby *Amazonu*[1], jehož snímek obrazovky je na obrázku 1.1. Zaměříme se na několik oblastí v prostředním sloupci, kombinujících univerazální i peronalizované doporučení:

Explore the Music Store, Top New Releases, Bestsellers In Music. Rozcestníky s obecně nejprodávanějšími kategoriemi, jako jsou bestsellery, novinky, případně výprodejové tituly za zvýhodněnou cenu. Tato oblast neobsahuje žádná personalizovaná doporučení. Výběr je identický pro všechny návštěvníky, těží z pouhé statistiky prodejnosti a žádným způsobem nereflektuje vkus zákazníka.

More Items to Consider. Autor práce si v předchozím týdnu před pořízením snímku obrazovky objednal album od *The Cinematic Orchestra - Late Night Tales* na vinylové desce. Během nákupu si prohlížel i album od stejné formace *Man With a Movie Camera*, což mu Amazon připomněl v sekci „You viewed“. V části „Customers who viewed this also viewed“ je naplno využito doporučovacích systému, který jednak nabídne album *Ma Fleur*, rovněž od *The Cinematic Orchestra*, tak i album *Late Night Tales - At the Movies*, které patří do stejné kompilační série, jako zakoupené album.

[1] `http://www.amazon.co.uk`



Obrázek 1.1: Hlavní strana hudebního oddělení britské pobočky Amazonu.

Related to Items You've Viewed. Tato sekce není už na snímku obrazovky viditelná, nicméně nabízí mnohem překvapivější výsledky než „More Items to Consider". Je založena na doporučení posluchačů se stejným vkusem, které kombinuje s analýzou metadat jednotlivých alb.

Jak již bylo zmíněno, doporučovací systémy pomáhají redukovat uživateli počet kroků nutných k nalezení požadovaných výsledků. Díky sběru dat od ostatních návštěvníků může nastat situace - a ta je velmi žádoucí - kdy je překvapivě nabídnuta položka, kterou uživatel nečekal, tedy položka neodpovídající původním omezením vyhledávání, ale vyhovující uživatelovu vkusu.

Herlocker et al. [1] popisuje kromě zřejmých funkcí doporučovacích systémů, tedy filtrování nevhodných a setřízení vhodných položek, i jiné způsoby užití. Některé systémy, např. v odvětví advokacie, vyžadují nalezení všech vyhovujících položek. Přehlédnutí libovolné z nich je velmi nežádoucí. Návštěvníky ale může zajímat i doporučení sledu několika položek. Konkrétní příklad můžeme hledat u internetových rádií.

Funkčnost doporučovacích systémů vyžaduje sběr informací o interakci s uživatelem. Tato interakce může být aktivní (uživatel vytváří obsah) či pasivní (uživatel konzumuje obsah). Více se o této problematice rozepisuje kapitola 3.1. Doporučovací systémy, umožňující aktivní interakci, se setkávají s návštěvníky, kteří je cíleně navštěvují za účelem vyjádření svého názoru, případně se snaží pomoci s rozhodováním a úsudkem ostatních návštěvníků. Je nutno podotknout, že ovlivňování úsudku může být i cíleně negativní.

Přístup ke shromážděným informacím o uživateli bývá zpravidla neveřejný. Výjimkou je např. hudební portál *last.fm*[2], který sbírá díky pluginu instalovanému do hudebního přehrávače statistiky poslechu skladeb. Každý aktivní uživatel zde vlastní veřejný profil, kde jsou zobrazeny seznamy oblíbených skladeb, interpretů, přátel a vzájemná hudební kompatibilita. Těchto dat je využito pro sestavení personalizovaného rádia, doporučení nových skladeb a inzerci hudebních událostí[3].

Implementace doporučovacích systémů není výhodná pouze pro návštěvníky. Výše uvedený *last.fm* díky propojenosti s *iTunes*, *7digital* a *Amazonem* dostává provize z prodejů hudebních skladeb.

Zveřejněním informací se proslavila i *Netflix Prize* [19]. Mezi lety 2006 až 2009 probíhala soutěž o nejlepší algoritmus doporučovacího systému nad poskytnutými daty společnosti *Netflix*[4], která poskytuje video-on-demand[5] služby. Za překonání algoritmu *Cinematch* této společnosti o 10% a poskytnutí zdrojového kódu byla udělena výhra $1 000 000. I přes splnění těchto podmínek nebyl nakonec výherní algoritmus implementován kvůli přílišné složitosti a náročnosti [20]. Nicméně už samotné vypsání soutěže strhlo obrovskou pozornost na problematiku doporučovacích systému a rozšířilo všeobecné povědomí o ní.

Personalizovaná reklama je obecně úspěšnější a vede k vyšším CTR[6]. Jak uvá-

[2] `http://www.last.fm`

[3] Rozsah funkcí je v každém státu omezen rozdílnými licenčními podmínkami, proto některé funkce nemusí být k dispozici.

[4] `http://www.netflix.com`

[5] Služba nabízející zhlédnutí video dle vlastního výběru bez předem stanoveného časového harmonogramu vysílání.

[6] Click-through rate. Míra prokliku, tedy poměr mezi počtem prokliků a počtem zobrazení.

dí případová studie společnosti *Hulu*[7] [22], rovněž poskytovatele on-demand videí, razantní rozdíly mezi CTR způsobují doporučení založené na odlišných vzorcích chování. Zdroj potvrzuje obecnou tezi, že nejvyšší úspěšnost mají položky příbuzné nejvýše explicitně ohodnoceným a nedávno shlédnutým. Lambrecht [3] ve své práci dokonce porovnává i úspěšnost znovuinzerování dříve navštívených položek oproti obecné, nepersonalizované inzerci. Pokud nemá uživatel vyhraněný profil chování, je ve většině případů obecná inzerce úspěšnější. Obě studie nedoporučují pracovat s příliš historickými daty, kdy uživatelé po čase ztrácejí zájem.

1.2 Nasazení doporučovacích systémů v realitním prostředí

Trh s realitami je v České republice plně rozvinutý. S nástupem ekonomické krize v roce 2008 došlo k útlumu v obchodování s realitami. Konkrétně od roku 2009 objem obchodů v oblasti nemovitostí setrvale klesá a v době psaní práce tento trend pokračoval. Studie *MindBridge Consulting a.s.*[35] hovoří o roce 2013 jako o roce, kdy byl pokles trhu zastaven a statistika ČSÚ zaměstnaných osob a aktivních ekonomických subjektů v tomto odvětví jí dává za pravdu (viz příloha B). Stávající subjekty jsou nuceny zvyšovat svou efektivitu, nabízet kvalitnější služby, intenzivně pracovat na propagaci společnosti či se jakýmkoliv způsobem odlišovat od konkurence a tím být lépe zapamatovatelným pro zákazníka. Jednou z možností, jak zvýšit zákaznický komfort, je zavedení doporučovacího systému k webové prezentaci nabídky nemovitostí.

Webové prezentace realit můžeme dělit do dvou základních kategorií:

Agregátory. Slouží jako sdružené katalogy několika realitních kanceláří, kde se dle různých kritérií (čerstvost nabídky, typ placené služby, abecední pořadí...) míchají všechny nemovitosti dohromady a prezentují zájemci. Největším zástupcem kategorie agregátorů na českém trhu je server *Sreality.cz* [8] s řádově statisíci unikátními návštěvami za měsíc.[9]. Existují i alternativní agregátory, propojující zájemce přímo s majitelem, kde se nepočítá se službami realitních kanceláří *Bezrealitky.cz* [10]

Prezentace realitních kanceláří. Výstavní skříní realitní společností na Internetu je její webová prezentace. Kromě prezentace nabídky nemovitostí, která není narozdíl od agregátorů omezená obsahem na pevně definovanou množinu atributů, jde především o představení společnosti. Zájemce má šanci dozvědět se podrobnosti o způsobu obchodování dané kanceláře. Společnosti si také budují svou značku publikacemi odborných článků nebo zprávami poodkrývající život realitní kanceláře.

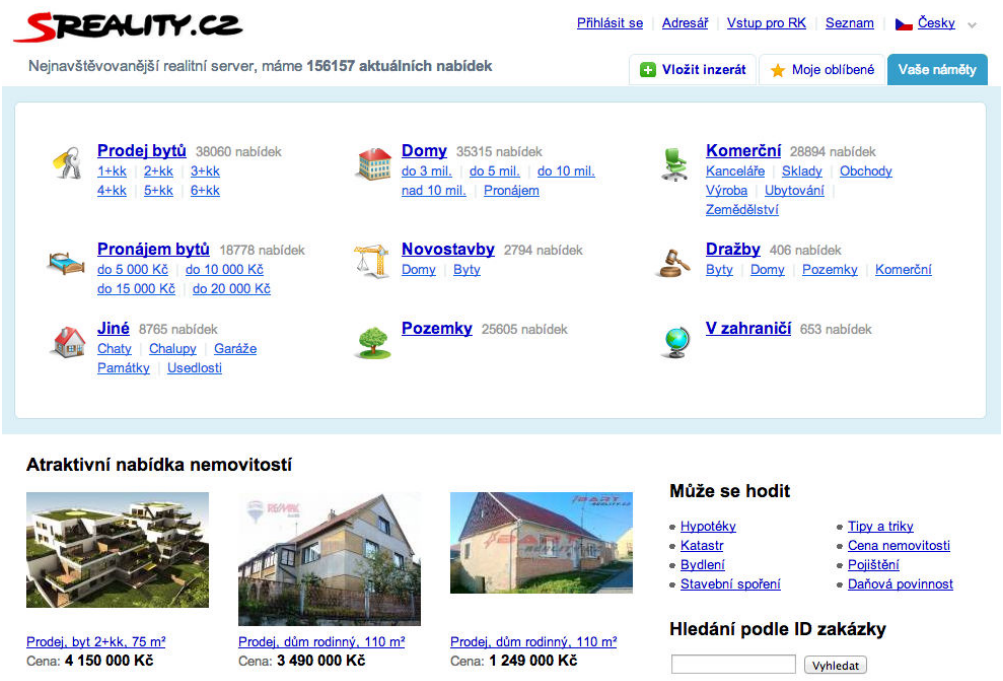
Na obrázcích 1.2 a 1.3 vidíme, že graficky ani členěním nemusí být mezi agregátorem realit a prezentací konkrétní realitní kanceláře příliš velký rozdíl. Na

[7] `http://www.hulu.com`

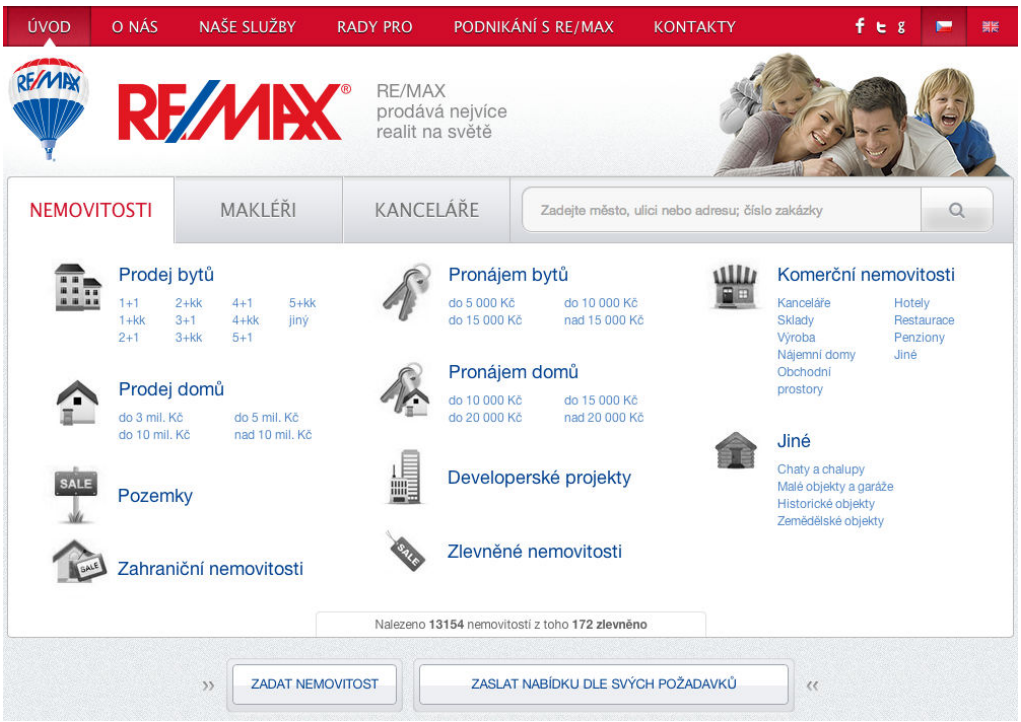
[8] `http://www.sreality.cz`

[9] Server *Sreality.cz* navštívilo v květnu 2013 zhruba 590 tisíc unikátních návštěvníků.

[10] `http://www.bezrealitky.cz`



Obrázek 1.2: Úvodní stránka agregátoru *Sreality.cz*.



Obrázek 1.3: Úvodní stránka realitní prezentace kanceláře *RE/MAX*.

českém trhu se nejedná o výjimku, spíše o pravidlo. Drtivá většina kanceláří začíná vyhledávání realit specifikací typu nemovitosti, poté se přejde ke specifikaci lokality a dalších parametrů nemovitosti. Pokud budeme zkoumat prezentace zahraničních realitních kanceláří či agregátorů, setkáme se i s alternativními pojetími, která v současné době v Čechách chybí či jsou marginálně zastoupeny. Jedná se především o geolokační vyhledávání - ať už přímé vyhledávání v mapě, případně dle občanského vybavení (MHD, školy, zdravotnická zařízení...).

Tato jistá lokální unifikace všech webů usnadňuje prvotní orientaci nově příchozího návštěvníka. Je potřeba si uvědomit, že prodejci se snaží zasáhnout co nejširší možné publikum, proto se zájemcům nesmí klást do cesty překážky. Pokud se budeme zajímat o profil zájemce, můžeme očekávat muže či ženu starší 18ti let se zájmem o koupi či pronájem nemovitosti. Nelze očekávat žádné speciální znalosti a dovednosti.

Zastavme se na chvíli u prostředí, ve kterém by plánovaný doporučovací systém mohl běžet. Lze předpokládat, že realitní kanceláře v případě nasazení budou chtít co nejméně měnit své zaběhlé zvyklosti či infrastrukturu. Při průzkumu trhu během přípravy *RePortu* (dále představen v kapitole 2) se zjistilo, že realitní kanceláře v České republice mají nasazeno zhruba deset různých informačních systémů. Kromě udržování databáze nemovitostí jejich úlohy přesahují do oblasti CRM[11], plánování času, evidence majetku a účetnictví. Součástí výbavy bývá větší množství exportních modulů na různé agregační servery a minimálně jeden pro účely vlastní webové prezentace společnosti. Odtud můžeme čerpat data o makléřích a nabídkách nemovitostí pro účely doporučovacího systému. Zaměstnanci realitních kanceláří většinou nemají žádné hlubší technické znalosti. Chod doporučovacího systému by měl být z hlediska realitní kanceláře co nejméně zatěžující, ať už finančně či kladením nároků na makléře i zákazníky.

Nemovitosti mají předem jasně definovanou doménu atributů - např. cena, rozloha, lokalita, atd. Tato doména je poměrně ustálená díky nutnosti inzerovat nabídky nemovitostí na agregátorech. Některé atributy jsou navíc hierarchicky řazené nebo mají mezi sebou logické vazby.

Na rozdíl od elektronických obchodů, nabízejících drobné zboží, které si návštěvník rovnou objedná, je koupě či pronájem nemovitosti poměrně zdlouhavou procedurou. Pokud pomineme fakt, že zájemce několikrát navštíví nabízený objekt, probíhá licitace ceny. Do samotného procesu mimo zájemce, prodávajícího a prostředníka (realitní kanceláře) vstupují i další instituce. Měla by proběhnout důkladná, nezávislá inspekce technického stavu objektu a ošetření právní stránky prodeje. Nezřídka se koupě financuje hypotečním úvěrem, tedy probíhá ocenění nemovitosti, zjišťování bonity žadatele, pojištění nemovitosti i žadatele, při kterém se dokonce zkoumá i zdravotní stav zájemce, než se přistoupí k samotnému podepsání smluv.

Díky tomu, že je na trhu velká konkurence, je změna počtu nabídek a poptávek pozvolná. V průběhu roku navíc dochází k útlumu prodejů v období letních měsíců a kolem Vánoc. Kardinalita nabídek nemovitostí je v řádech měsíců poměrně stabilní. To samé bohužel nemůžeme říct o počtu zájemců konkrétní realitní kanceláře, kdy do hry vstupuje mnoho faktorů. Ale i tak se dá z velikosti trhu a počtu nabízených nemovitostí zhruba odhadnout přibližný horní odhad zájemců. Pokud se budeme stále omezovat na Českou republiku, lze očekávat desítky až deseti ti-

[11] Customer relationship management - software pro řízení vztahů se zákazníky.

síce nemovitostí v nabídce na jednu realitní kancelář a návštěvnost maximálně deseti tisíce zájemců za den. Na českém trhu operuje kolem 3000 realitních kanceláří, z toho zhruba 1500 v Praze[12]. Medián počtu nabídek je zhruba 100 na realitní kancelář v Praze, 70 u mimopražských. Je třeba uvažovat fakt, že drtivá většina společností obchoduje s malým počtem nabídek - pouze 14 realitních společností na českém trhu nabízí více než 1000 nemovitostí.

Veškeré obchody v realitách se časem přesunou z elektronického světa do reálného a jsou zakončeny aspoň jedním osobním setkáním během podpisu smluv. Tento fakt velmi ztěžuje, ba vylučuje automatické párování dokončených obchodů s daty získanými při prvotním kontaktu se zájemcem skrz webovou prezentaci, jelikož uživatel není typicky jednoznačně identifikován přihlášením. Tato cenná data se zpracovávají např. u již zmíněného *Amazon*u.

Elektronické obchody si zpravidla mohou dovolit optimalizovat výsledky doporučení na zisk z prodaného zboží. Jejich dodávky jsou omezeny pouze kapacitami výrobců či distributorů. Situace u realitních kanceláří je komplikovanější. Nabídka nemovitostí není teoreticky neomezená, společnosti musí o každou bojovat. Nabídky pocházejí z několika zdrojů - přímá nabídka, doporučení známých, inzerce, atd. Jednou z nejefektivnějších metod je periodické kontaktování starých zákazníků, kteří mohou nabízet nemovitosti k prodeji či pronájmu. Úspěšnost zařazení nabídky do inzerce ovšem závisí na předchozí spokojenosti zákazníka s realitní kanceláří. Nejen z tohoto důvodu se proto snaží společnosti prodat veškeré nabízené nemovitosti, zejména pak ty s výhradním zastoupením.

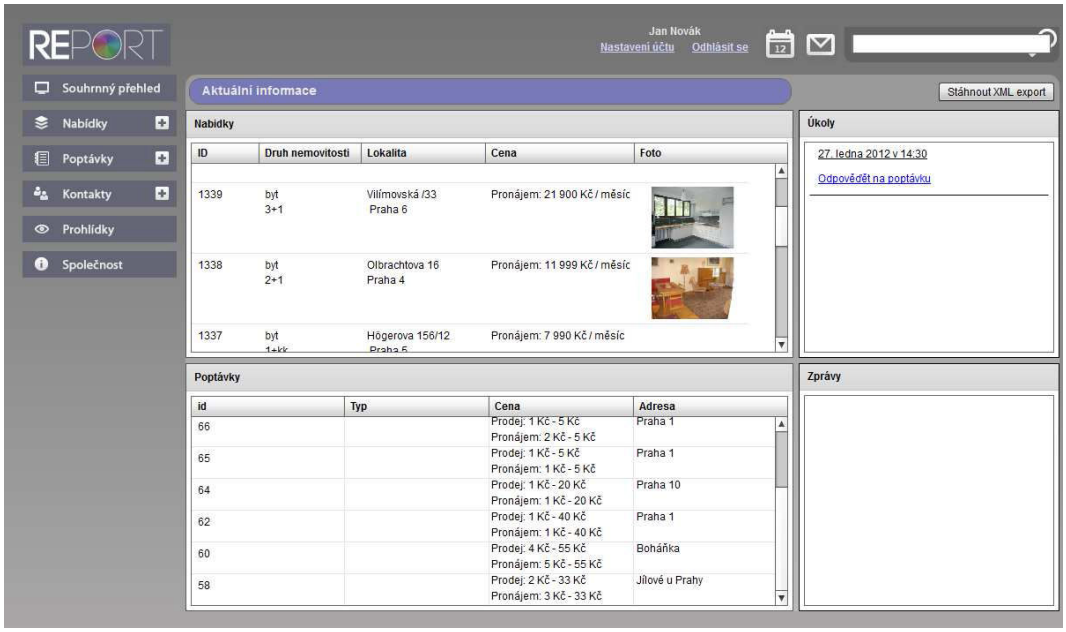
[12] Analýza inzerentů `www.sreality.cz` v březnu 2013

2. RePort

V této kapitole se představí informační systém *RePort*, pozadí jeho vzniku jako školního předmětu Softwarový projekt. Čtenář je seznámen s použitými technologiemi, jejich výběrem a architekturou systému. Dále se dozví o vývoji systému po obhájení softwarového projektu a provedených změnách - jak z hlediska architektury, tak i z hlediska použitých technologií. Nakonec se provede debata nad možnostmi doplnění *RePortu* o doporučovací systém, představí se existující práce a z nich získaná inspirace pro zamýšlené rozšíření.

2.1 Představní RePortu

„*RePort* (Realitní Portál) je informační systém, určený pro realitní firmy a realitní makléře. Slouží k centrálnímu zadávání realitních poptávek, nabídek, správě klientů a poboček, importu a exportu těchto dat, k plánování úkolů a komunikaci mezi makléři“, jak je uvedeno v dokumentaci k projektu [8]. Tento informační systém byl obhájen roku 2012 v rámci předmětu NPRG023 Softwarový projekt na Matematicko-fyzikální fakultě ve složení řešitelů Juan Rodrigo Baquero Forero, Jakub Húska a Radek Strnad.

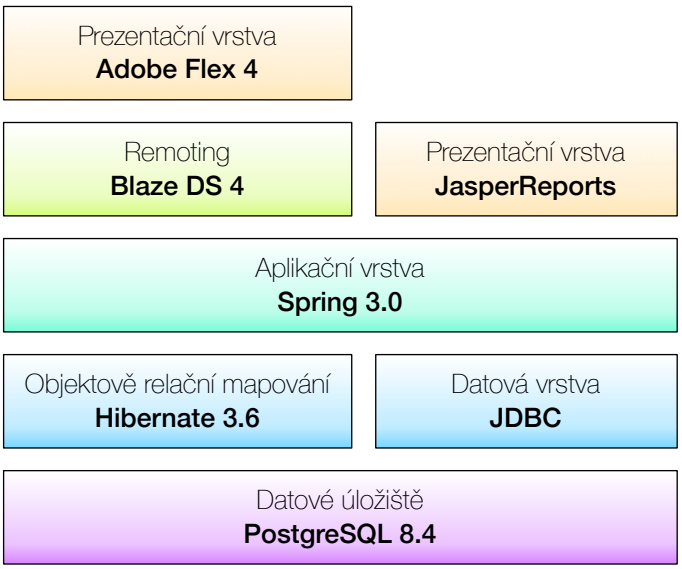


Obrázek 2.1: Úkazka makléřského rozhraní v Adobe Flexu. Stav RePortu v době obhájení softwarového projektu.

RePort svou funkčností umožňuje správu několika realitních kanceláří, jejich poboček, uživatelů, zasílání zpráv uživatelům, správu nabídek nemovitostí (včetně obsazenosti nemovitostí, plánování prohlídek), poptávek nemovitostí a jejich vzájemného párování. Data do *RePortu* lze importovat a lze je i exportovat.

2.2 Architektura aplikace

RePort je postaven jako vícevrstvá, modulární aplikace, využívající několik rozdílných technologií. Celý informační systém běží na *Java Servlet* serveru *Apache Tomcat* (testováno ve verzi 6.0.29), ale s úspěchem jej lze provozovat i jinde. Během vývoje byl například používán server *Jetty*.



Obrázek 2.2: Vícevrstvá architektura *RePortu* v době obhájení softwarového projektu.

2.2.1 Datové úložiště a datová vrstva

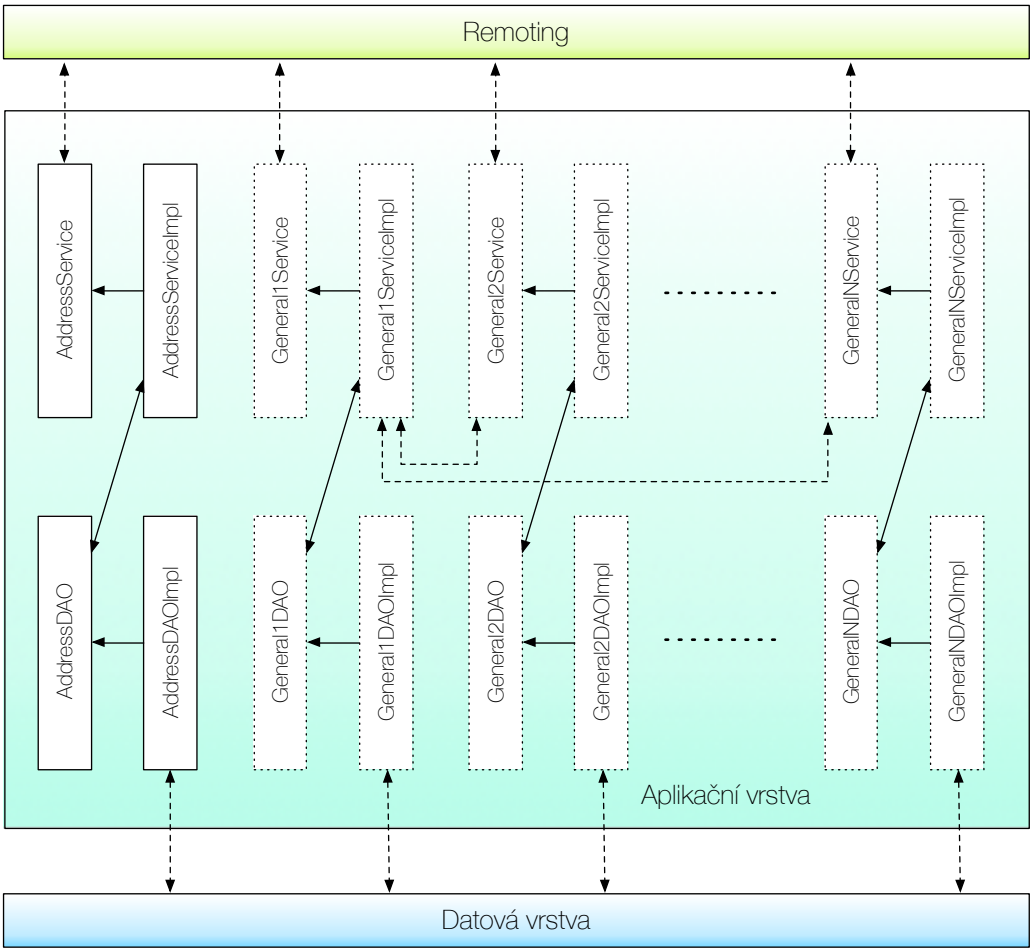
Datové úložiště je postaveno na databázi *PostgreSQL*. Testována byla verze 8.4, nicméne vzhledem k dodržování SQL standardů by neměl být problém s nasazením pozdějších verzí databázového stroje. Objektově-relační mapování a transakce zajišťuje *Hibernate*, nicméně k databázi se v některých případech přistupuje i přímo pomocí *JDBC*. Vývoj *RePortu* původně odstartoval pouze s JDBC, kde se mapování výsledků na POJO[1] provádí manuálně. Postupem času se zjistilo, že by vývoj urychlila automatizace tohoto procesu, proto se přešlo k *Hibernatu*.

2.2.2 Aplikační vrstva

Aplikačni vrstva hojně využívá *Spring* frameworku. Jedním z nejzákladnějších aspektů tohoto frameworku je IoC[2], což nutí programátory vyvíjet jednotlivé funkční prvky jako oddělené moduly. Pro správu uživatelských účtů a omezení viditelnosti dat byla implementována podpora *Spring Security*.

[1] Plain old Java object. Označení obyčejného, nekomplikovaného Java objektu - entity.

[2] Inversion of Control - návrhový vzor objektově orientovaného programování, který rozbíjí závislosti instancí jednotlivých tříd při vývoji. Programuje se oproti interface a objektový graf tříd se vytváří až za běhu.



Obrázek 2.3: Zjednodušený náhled na strukturu aplikační vrstvy.

RePort využívá v aplikační vrstvě Entity, DAO a Service návrhový vzor, který je rovněž typickým rysem *Spring* frameworku. Je vhodné, aby si jej čtenář osvojil, protože mu usnadní pozdější pochopení. Základní stavební jednotkou je **entita** - např. nemovitost, uživatel či lokalita. Každá entita má své ORM mapování na tabulku v SŘBD. Mezi jednotlivými entitami jsou podobné vztahy, jaké známe z databázového světa - 1:1, 1:n, m:n. Vrstva **DAO** odstiňuje operace s SŘBD od ostatního kódu. Definicí interface se stanoví operace nad entitami. Konkrétní implementace potom určí, zda se jedná o dotazy v HQL či SQL. Vrstva služeb (v ustálené terminologii jako **Service**) pro každou logicky oddělitelnou jednotku stanoví atomické operace, které jsou složeny převážně z volání DAO vrstvy. Kombinuje při tom DAO zdroje a již existující služby. Namísto vytváření nových instancí služeb drží *Spring framework* v paměti právě jednu instanci, kterou vytvořil při startu (pokud není explicitně nakonfigurováno jinak) a následně ji zpřístupnil anotací `@Autowired`.

Aplikační vrstva se dále stará o rozsah a druh transakcí. Na jednotlivé služby by se mělo nahlížet jako na funkční moduly, které v drtivé většině případů mají svůj protějšek v uživatelském rozhraní.

Pro představu o rozsahu původní verze *RePortu* si nyní uvedeme seznam všech implementovaných služeb:

AddressService - Stará se o spravování místopisných informací.

AuthorityTemplateService - Uživatelé v *RePortu* mají přidělená práva, která mají poměrně jemnou granularitu. Aby se při přidání nového uživatele nemuselo vybírat mnoho uživatelských práv znovu a znovu, jsou v systému vytvořeny šablony (např. makléř, správce, atd.), o které se tato služba stará.

CompanyService - Stará se o spravování realitních společností v *RePortu*.

DemandService, DemandCustomAttributeService - Starají se o správu poptávek po nemovitostech.

ExportPdfService - Transformuje data poptávek a nabídek nemovitostí na PDF dokumenty pro distribuci emailem či tisk.

ExportXmlService - Provádí export veškerých dat v databázi ve formátu XML.

FileRepositoryService - Společně se správou dat je potřeba uchovávat i fotografie a obecně jakékoliv soubory. `FileRepositoryService` poskytuje prostředky pro ukládání a předávání jakýchkoliv souborů, společně s implementací primitivního verzovacího systému.

FlatOccupancyService - Služba se stará o manipulaci s daty, týkajícími se obsazenosti bytů k pronájmu.

ImportService - Stará se o import dat do *RePortu*. V původní verzi je implementován převodník z XML datového formátu *RealStudia 2008* společnosti BlueWave s.r.o.

MessageService - Zaštituje funkce pořebné k zasílání jednoduchých zpráv mezi uživateli systému.

NoteService - Spravuje krátké textové poznámky, které lze přilepit k nemovitosti či uživateli.

OfferService, OfferCustomAttributeService - Zajišťuje správu nabídek nemovitostí.

OfferDocumentService - S pomocí `FileRepositoryService` umožňuje připojovat k nemovitostem dokumenty, jako např listiny apod.

OfferGalleryService - Obstarává přiřazení fotografií k nabídce nemovitosti a jejich pořadí.

OfferInspectionService - Spravuje prohlídky nemovitostí a výstupy z nich.

OfficeService - Spravuje pobočky realitních kanceláří a přiřazení uživatelů.

TaskService - Ke kontaktům, nabídkám i poptávkám nemovitostí lze přidružit úkoly, o jejichž manipulace se stará tato služba.

UserService - Spravuje kontakty realitních kanceláří i uživatele systému.

WorkgroupService - Spravuje pracovní skupiny realitních kanceláří a přiřazení uživatelů.

2.2.3 Prezentační vrstva a remoting

Prezentační vrstva je napsána v *Adobe Flex* a skriptovacím jazice *ActionScript*, který po překladu běží v *Flash Player* runtimu vnořeném v klasické webové stránce. Pro přístup potřebujeme libovolný webový prohlížeč s pluginem *Flash Player*. Remoting mezi aplikační a prezentační vrstvou zajišťuje v proprietálním binárním formátu *BlazeDS*, který transparentně zpřístupňuje interface všech služeb implementovaných v aplikační vrstvě a provádí konverze datových typů mezi *Javou* a *ActionScriptem*. *BlazeDS* tak nabízí programátorovi téměř identický komfort, jaký by měl při psaní prezentační vrstvy v *Javě*, kde by mohl přímo volat metody všech služeb.

Samotné sestavení uživatelského rozhraní kopíruje objektově komponentový návrh jazyka *Flex*. Vysvětlování jeho principů by přesáhlo rozsah této práce. Pro představu o funkčnosti není nutné zabíhat do podrobností. V případě zájmu, nechť se čtenář odkáže na dokumentaci tohoto programovacího jazyka.

Data uložená v informačním systému můžeme nejen prohlížet na obrazovce, ale i exportovat ve formátu XML do souboru či exportovat do formátu PDF jako tiskovou sestavu pomocí *JasperReports*.

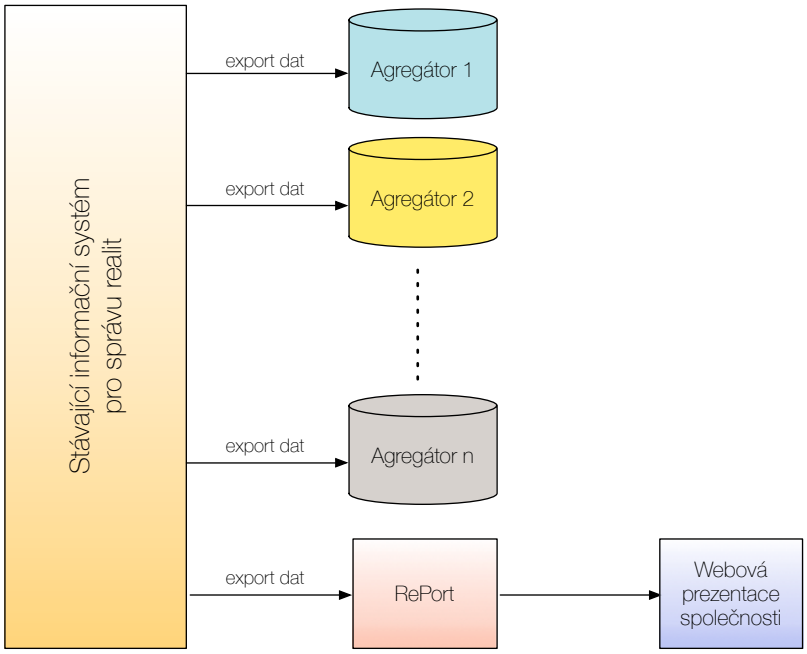
2.3 Vývoj po obhájení softwarovéhu projektu

Jedním z důležitých požadavků při vývoji *RePortu* bylo komfortní ovládání informačního systému podobné desktopové aplikaci, a přesto běh v prohlížeči bez instalace, známé jako RIA[3]. V době návrhu první verze aplikace (podzim 2010), tedy v době, kdy byla aktuální verze Internet Exploreru 8 s neúplnou podporou *HTML 5*, sváděly boj o nadvládu následující RIA technologie: *Adobe Flex*, *Microsoft Silverlight*, *Java FX* a *HTML 5*. Rozhodnutí využít *Adobe Flex* se již během řešení softwarového projektu ukázalo jako značně kontroverzní. Po ukončení podpory Flash Playeru (nezbytný runtime Adobe Flexu) na některých platformách (*Android*, *iOS*) samotným *Adobe*, následovaného darováním celého projektu *Adobe Flex* opensource komunitě, bylo jasné, že budoucnost tohoto řešení je značně nejistá. Podobný osud postihl i *Microsoft Silverlight*, který rovněž i díky rapidnímu nástupu mobilních zařízení skončil v propadlišti dějin.

Úspěšným obhájením softwarového projektu a dokončením první verze *RePortu* byla ukončena spolupráce členů řešitelského týmu. Jelikož i na toto dílo se vztahují autorská práva, bylo nutno zajistit licenci pro další nakládání s kódem. Jeden z řešitelů bohužel neudělil svůj souhlas, proto musel být jeho příspěvek do společného díla, na základě analýzy repozitáře verzovacího systému, odstraněn. Kvůli předejití pozdějším sporům byl vyřazen i kód, který nebyl napsán přímo zmíněným autorem, nicméně úzce na něm závisel (např. kód odvíjející se od předchozích revizí).

V případě aplikační a datové vrstvy je kód slušně logicky rozdělený, proto mohly být odstraněny celé moduly či metody. Jelikož se zmíněný řešitel přímo nepodílel na klíčových částech nabídek a poptávek, zůstaly zachovány. Nový návrh odstranil i některé nedostatky původní verze. Kromě změn v samotném kódu proběhla i aktualizace použitých knihoven na novější verze. Za zmínku stojí zejména samotný *Spring* framework (3.0 → 3.1) a *Hibernate* (3.6 → 4.1).

[3]Rich Internet Application



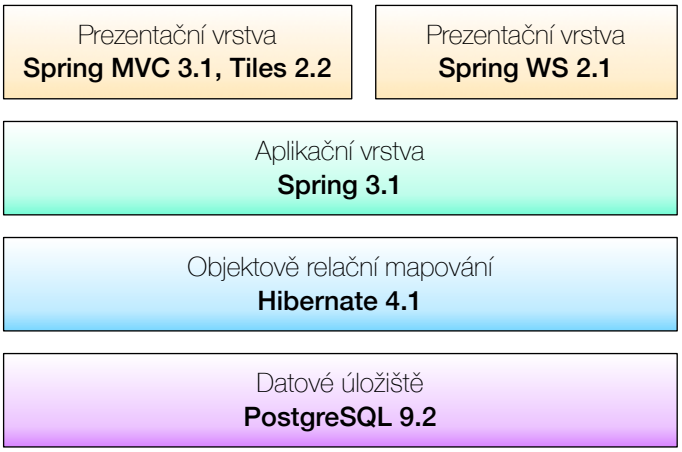
Obrázek 2.4: Přidružení *RePortu* k stávajícímu informačnímu systému realitní kanceláře.

V kapitole 1.2 jsme zjistili, že realitní kanceláře od informačního systému vyžadují daleko více funkcí, než jim *RePort* dokáže v rozsahu řešení softwarového projektu nabídnout. Abychom uvažovali o reálném nasazení *RePortu*, jako základu pro doporučovací systém, museli bychom jej zařadit jako mezičlánek mezi stávající informační systém a webovou prezentaci společnosti (viz obrázek 2.4). Proto se k této skutečnosti během provádění zásadních změn v architektuře přihlíželo. Naopak nebylo nutné nadále udržovat kompletně celé uživatelské rozhraní ve *Flexu*, jelikož nebude používáno. Pojďmě si tedy nyní podrobněji představit změny, kterými *RePort* od konečného stavu při obhájení softwarového projektu prošel.

2.3.1 Datové úložiště a datová vrstva

Nasazení *PostgreSQL* v původní verzi *RePortu* se osvědčilo. Databázový stroj byl aktualizován na novější verzi 9.2. Díky konzervativnímu přístupu komunity k vývoji tohoto produktu nedošlo při upgradu k žádným komplikacím.

Dříve bylo možné k datům přistupovat poněkud schizofrenně, skrz tradiční *JDBC* či *Hibernate*. Pro zjednodušení konfigurace, zvýšení přehlednosti kódu a redukci chyb bylo minoritně zastoupené *JDBC* ve `FileRepositoryService` přepsáno za použití *Hibernate*. Ten byl s malými úpravami konfigurace i částí stávajícího kódu povýšen na novější verzi 4.1. Pokud by přeci jen v budoucnosti vyvstala nutnost dotazovat se přímo v jazyce SQL, implementuje *Hibernate* metodu `Session.createSQLQuery`.



Obrázek 2.5: Pozměněná vícevrstvá architektura *RePortu* před zahájením prací na doporučovacím systému.

2.3.2 Aplikační vrstva

Z hlediska architektury aplikační vrstvy se neudály žádné podstatné změny, odpovídá obrázku 2.3. Závažnější změny proběhly v oblasti jednotlivých služeb. Díky chybějící licenci na užití kódu byly kompletně odstraněny služby (včetně navázaného kódu - DAO, entity, atd.) na zasílání zpráv mezi uživateli `MessageService`, PDF export `ExportPdfService`, XML export `ExportXMLService`, vytváření poznámek `NoteService`, správy obsazenosti bytů `FlatOccupancyService` a evidence prohlídek nemovitostí `OfferInspectionService`. Vzhledem k záměru práce nebylo nutné tyto služby znovu implementovat, proto byly odstraněny bez náhrady. Odstranila se rovněž závislost na *JasperReports*, která pozbyla potřeby.

Naopak služba místopisného zatřídění `AddressService` byla odstraněna a nahrazena `LocationService`. Původní návrh jednoznačně definoval strukturu místopisného zatřídění NUTS dle doporučení EU. V testovacím provozu se ukázalo, že tato klasifikace nemá ve světě realit dostatečnou granularitu. Celou věc ztěžoval fakt, že se ve světě vyskytují lokální klasifikační anomálie. Například Praha má poměrně sofistikované členění na správní obvody, městské části a katastrální území, jejichž názvosloví jsou všeobecně zažitá. Z pohledu členění NUTS se jedná pouze o město s názvem Praha. Nový návrh `LocationService` pro každý stát definuje vlastní hierarchickou strukturu územně správních celků, schopnou se vypořádat i s lokálními zvyklostmi. Tato hierarchie má neomezenou granularitu a např. v České republice je zadefinována až po katastrální území.

Záměr zařadit *RePort* jako jakousi datovou proxy mezi stávající informační systém realitní kanceláře a webovou prezentaci, znázorněnu na obrázku 2.4, vedl k potřebě přepracování importních a exportních služeb. Obhájená verze *RePortu* měla implementovanou importní službu pro informační systém *Real Studio 2008* společnosti Blue Wave, s.r.o. Ta je v současné době odstraněna a nahrazena odlišnou implementací pro *Real Studio 2010* stejné společnosti. Změny proběhly i v samotné `ImportService`. První návrh *RePortu* považoval import za jakýsi přestupní bod, kdy realitní kancelář přelije svá data a nadále bude používat pouze *RePort*. Nově *RePort* udržuje interní číslování i číslování cizího informač-

ního systému tak, aby data zůstala koherentní. Pamatuje i na rozdílné verze dat a kolize. Zlepšila se i modularita importních modulů a přibyl správce spouštění importních úloh.

XML export dat, jak bylo uvedeno výše, byl odstraněn. Náhradou se mu stalo Web services rozhraní, kdy *RePort* pomocí SOAP zpráv komunikuje s protistranou (dále popsáno v kapitole 2.3.3). Autor považoval za přínosné, vzhledem k plánovanému nasazení, doplnit *RePort* o zcela nové služby, orientující se na obsah webové prezentace realitní společnosti. Jmenovitě se jedná o `ArticleService` a `ArticleContentService`, přidávající podporu pro vytváření vícejazyčných článků a `NewsletterSubscriptionService` spravující databázi kontaktů odebírajících newslettery.

2.3.3 Prezentační vrstva

Makléřské rozhraní

Prezentační vrstva, určená pro vykreslování UI pro realitní makléře, byla napsána v *Adobe Flex*. Byla to jedna z technologií, se kterou řešitelé softwarového projektu neměli předchozí zkušenosti. Kvůli nedostatečné modularitě kódu i důvodům uvedeným výše se autor práce rozhodl platformu *Flexu* opustit a makléřské rozhraní zcela přepsat. Odstraněn byl i navázaný *Blaze DS*, starající se o remoting AMF[4] zpráv. Tentokrát bylo rozhodování o výběru RIA technologie jednoduché. Dřívější obavy z adopce *HTML 5* ustoupily díky jednoznačnému vývoji situace do pozadí a naopak penetrace ostatně jmenovaných technologií s příchodem nových platforem poklesla.

Kód *HTML 5* je generován kombinací komponenty *Spring* frameworku - *Spring MVC* a *Apache Tiles*. *Spring MVC* rozšiřuje stávající architekturu o **model - view - controller** návrhový vzor. Třídy controllerů jsou nově umístěny v balíku `cz.cuni.mff.report.web.controller` a rozděleny dle funkcí na logické celky.

ArticleController - Stará se o správu vícejazyčných článků.

DashboardController - Odbavuje požadavky na souhrnné informace (např. po přihlášení).

FileRepositoryController - Zpracovává požadavky na zobrazení souboru (typicky obrázky).

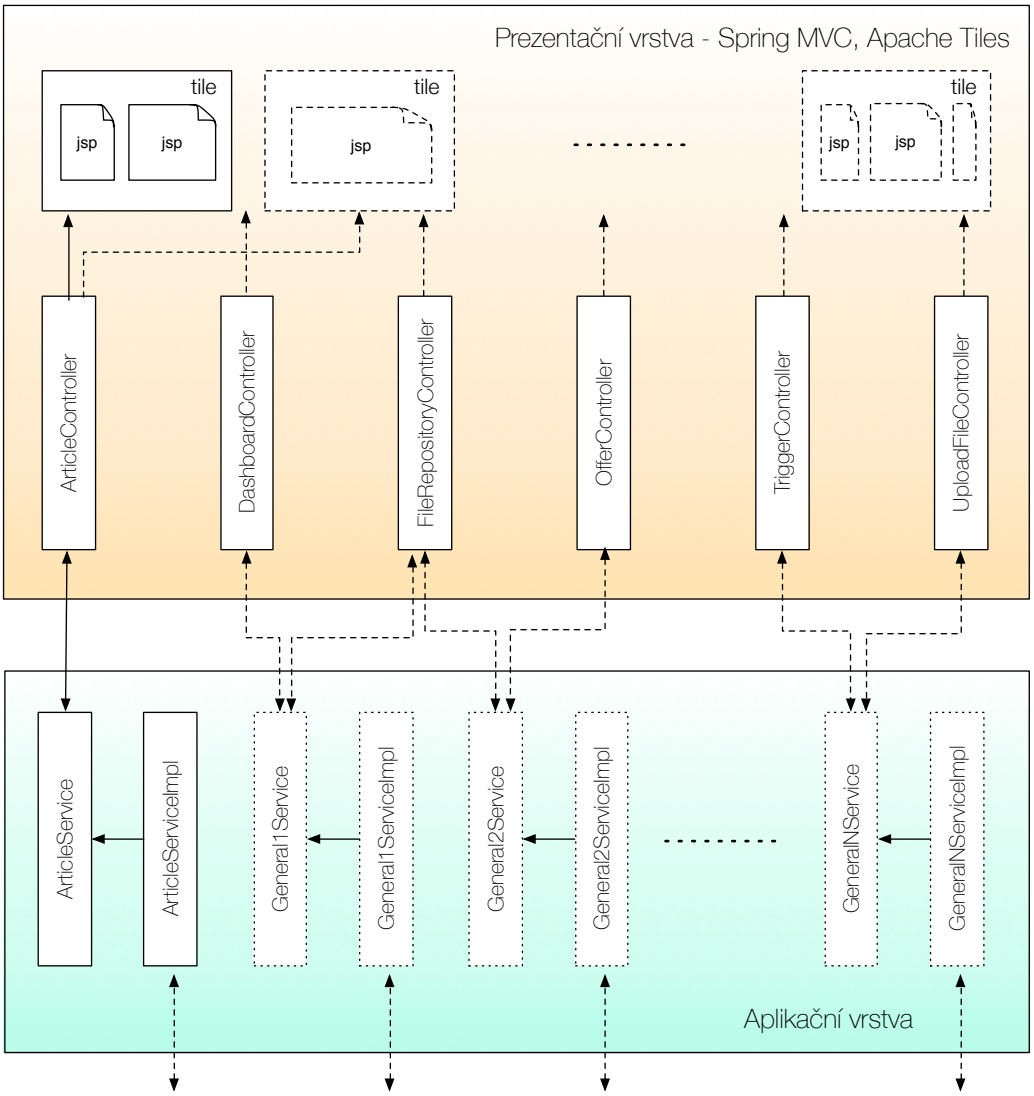
OfferController - Shlukuje všechny požadavky na správu nabídek nemovitostí.

TriggerController - Obsluhuje požadavky na spuštění importu.

UploadFileController - Umožňuje nahrávat soubory do *RePortu*.

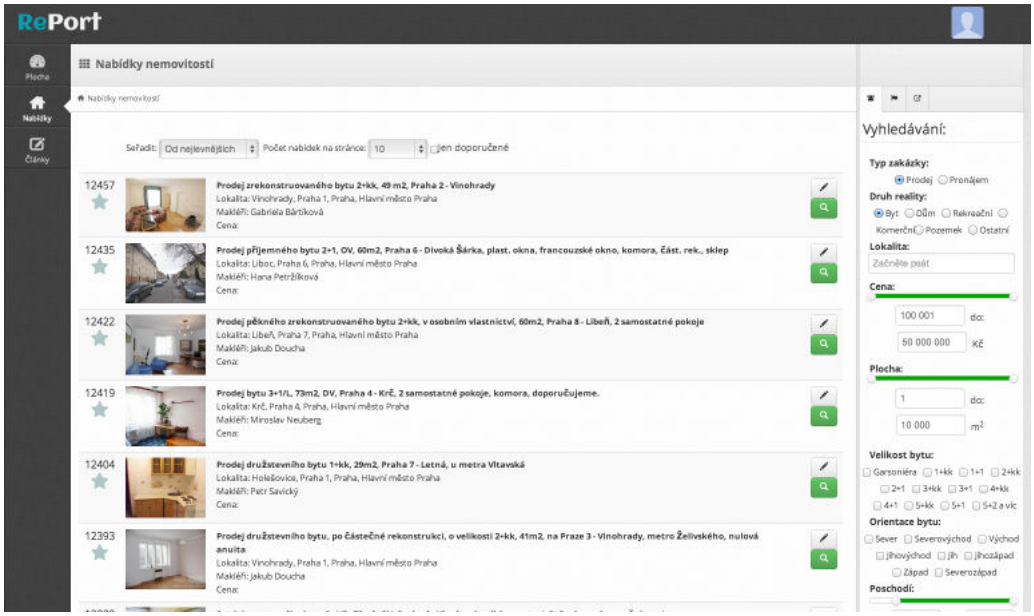
Každé volání controlleru odbavuje právě jeden požadavek prohlížeče. Začne rozklíčováním URL, ze kterého vypreparuje předané GET či POST parametry. Pomocí principu *IoC* popsaného výše jsou controllery propojeny se službami.

[4]komunikační protokol mezi *Spring*em a *Adobe Flex*



Obrázek 2.6: Zjednodušený náhled na strukturu MVC prezentační vrstvy.

Služby v návrhovém vzoru MVC suplují model a provádí se skrz ně veškére databázové dotazy. Výsledné stránky předané prohlížeči jsou sestaveny ve view komponentě díky *Apache Tiles* z dílčích JSP souborů. Vzhledu makléřského rozhraní bylo dosaženo zakoupením šablony *Stilearn* nad CSS a JavaScript frameworkem *Twitter Bootstrap*. Výběr byl uskutečněn na základě předchozích pozitivních zkušeností autora, jednoduše dosažitelnou konzistencí ovládacích prvků, sdílenou s dalšími webovými aplikacemi, a také díky vlastnostem a rozšiřitelností frameworku. Za všechny můžeme jmenovat tzv. „responsive design“, nabízející programátorovi při zachování dohodnutých CSS konvencí napsat aplikaci, která se sama bude přizpůsobovat zobrazovacímu zařízení. Podporovány jsou při udržení dobré použitelnosti různá rozlišení u klasických desktopových prohlížečů, mobilních telefonů a tabletů. Bohužel z důvodu pracnosti původního projektu, který byl vypsán pro čtyři řešitele, se pro potřeby diplomové práce nepodařilo implementovat plný rozsah klientského rozhraní. Jak bylo vysvětleno výše, pro účely implementace doporučovacího systému toto ale ničemu nebrání.



Obrázek 2.7: Úkazka nové implementace makléřského rozhraní v HTML 5.

Web services

Zamýšlené nasazení doporučovacího systému, kde *RePort* měl fungovat jako centrální bod, shromažďující veškerá data, i zajišťovat funkce doporučovacího systému, vedlo k potřebě online výměny dat mezi webovou prezentací a *RePortem*. Jako komunikační protokol byl vybrán SOAP, implementovaný modulem *Spring* frameworku *Spring WS*, který pokrývá všechny základní potřeby typické webové prezentace realitní kanceláře. Implementací rozhraní pro volání webových služeb a zadefinováním sady služeb poskytujících následující data byla odbourána nutnost databázového stroje na straně webové prezentace:

- Seznam článků a množina jejich jazykových verzí
- Detail článku včetně textů v různých jazykových mutacích
- Seznam nemovitostí včetně možnosti filtrování
- Poskytnutí seznamu X nemovitostí (vztahujících se ke konkrétní nemovitosti, dle doporučení, náhodných)
- Detailní výpis konkrétní nemovitosti
- Zadání poptávky nemovitosti
- Přihlášení a odhlášení emailové adresy do newsletteru
- Získání základních statististických údajů o nemovitostech pro potřeby vyhledávacích formulářů (rozsah ceny, plochy, ...)
- Podpora pro autocomplete lokalit
- Služby pro sběr dat o chování uživatele (dále popsané v kapitole 4.6)

Každé realitní kanceláři je přidělen unikátní klíč, kterým se identifikuje během vykonání SOAP požadavku a který zamezuje přístupu k datům ostatních realitních kanceláří či náhodnému uživateli. I v případě kompromitace tohoto klíče by se útočník dostal maximálně k datům, jež jsou určené pro publikaci na Internetu, neobsahují citlivé informace a jsou pouze pro čtení. *RePort* již od první verze implementuje vlastní datové úložiště s primitivním verzovacím systémem a přístupem přes HTTP. V současnosti se používá pouze pro fotografie nemovitostí, nicméně v budoucnu nic nebrání ukládat i dokumenty či jiné soubory.

Nová architektura *RePortu* architekturou klient - server rapidně zkracuje dobu vývoje webové prezentace a snižuje nároky na úložný prostor a hostitelský systém webové prezentace. Na drouhou stranu je třeba poukázat i na nedostatky zvoleného řešení:

Single point of failure. *RePort* se při zvolené architektuře stane kritickým místem, které dokáže vyřadit z provozu všechny napojené webové prezentace. Uvedením do provozu další instance *RePortu* s funkční online replikací databáze a oznámením IP adres těchto instancí webovým prezentacím by se dalo výpadkům předejít.

Zpoždění a overhead SOAP protokolu. Při komunikaci se vzdáleným počítačem dochází k násobně většímu zpoždění, než jaké nastává při využití databáze instalované lokálně. Rovněž je třeba brát v potaz neefektivitu SOAP protokolu, který má mnohem vyšší overhead než binární protokol ovladače databáze. Výběrem poskytovatele hostingu s dostatečnou šířkou pásma, případně snahou o co nejbližší připojení k *RePort* serveru, se dá problém minimalizovat. Během pilotního nasazení se v praxi ověřilo, že tyto obavy jsou liché a rozdíl rychlosti odezvy stránek oproti běžnému standardu je neznatelný. Je tomu tak i díky návrhu protokolu webových služeb, kde si každá ze zobrazených stránek vyžádá typicky jedno, v některých případech dvě SOAP volání.

2.4 RePort a doporučovací systém

Zatím jsme si představili *RePort* a jeho schopnosti správy nemovitostí. Máme položený základ, který budeme dále rozvíjet rozšířením o doporučovací systém. Schröder [27] ve své práci upozrňuje na nutnost správné identifikace cíle doporučovacích systémů. I přesto, že tento krok se může na první pohled zdát zřejmý, často dochází pouze ke konstatování: „Chci, aby můj doporučovací systém nabízel uživateli nejoptimálnější výsledky.“ Brzy zjistíme, jak je toto tvrzení velmi vágní. Musíme se rozhodnout, na základě jakého klíče bude probíhat ono určení nejoptimálnějších doporučení. Zda se jedná čistě o doporučovací systém navázaný na obecný informační systém (např. archiv diskusí), kde nabídnuté položky nemají další dopad na poskytovatele, či se bavíme o obchodním systému, kde nabídnuté položky mají vliv na tržby a tedy se nám jedná o porovnávání ekonomických faktorů.

Jak se přesvědčíme dále v kapitole 3, neexistuje univerzální recept na vytvoření doporučovacího systému, fungujícího na libovolných datech a ve všech doménách. Diverzita existuje i mezi obchodními strategiemi realitních kanceláří.

Díky tomu, že jedna instance *RePortu* nemusí obsluhovat pouze jednu kancelář, ale hned několik, bude potřeba k této diverzitě přihlížet a umožnit každé z realitních společností si vybrat vhodný doporučovací systém a jeho parametry. Je potřeba si uvědomit kontext nasazení *RePortu*, kdy na rozdíl od různých agregátorů nabídek nemovitostí, které si mohou diktovat podmínky, tedy i parametry doporučovacího systému, je *RePort* zdrojem veškerých dat pro webovou prezentaci konkrétní společnosti a měl by jí vyjít maximálně vstříc.

Kromě položení infrastruktury pro konfigurovatelné doporučovací systémy v rámci *RePortu* bude jedním z cílů práce i implementace vzorového algoritmu. Autor práce vývoj *RePortu* i obsah diplomové práce konzultoval s jednatelem společnosti *RealExpert s.r.o.*, který svolil k pilotnímu nasazení informačního systému *RePort* v rámci přípravy nové webové prezentace společnosti v módu tzv. datové proxy (viz obrázek 2.4). Vzorový algoritmus by měl být vyvíjen s ohledem na potřeby, velikost a návštěvnost realitní kanceláře. Jednou z podmínek pilotního nasazení byl požadavek na minimální náročnost uživatelské interakce. Výsledky vzorového algoritmu by měly být analyzovány a nad výsledky se provede diskuse.

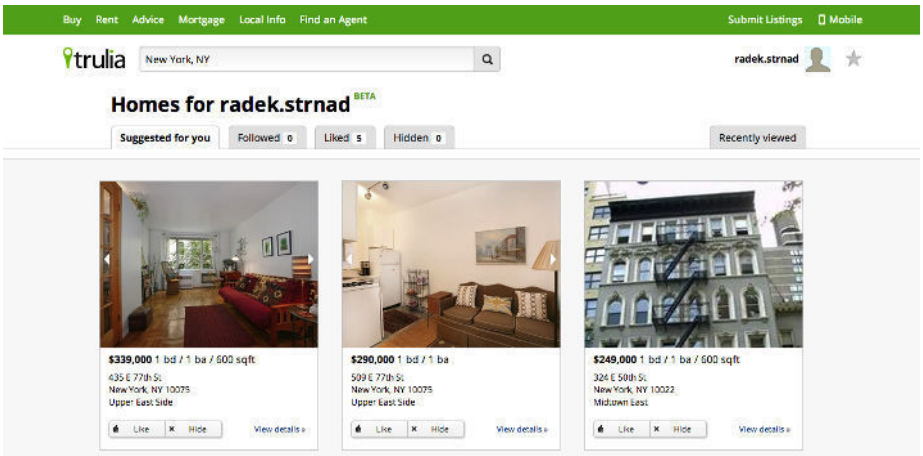
2.5 Existující práce

Použití doporučovacích systému v oblasti realit není nová myšlenka. Zajisté existuje mnoho implementací, které jsou ovšem proprietální a mnoho se o nich nepíše. Přesto lze najít několik světlých výjimek. V rámci interakce se zákazníky a budování značky, některé společnosti prezentují kusé informace o principech fungování svých doporučovacích systémů.

2.5.1 Trulia Suggests

V březnu 2013 společnost Trulia, operující ve Spojených státech amerických s 31 miliónem unikátních návštěvníků za měsíc[5], uvedla na trh beta verzi služby *Trulia*

[5]Data uveřejněná v oznámení o spuštění služby



Obrázek 2.8: Trulia Suggests

Suggests[29] (`http://www.trulia.com/suggestions`, ukázka fungování na obrázku 2.8). *Trulia Suggests* na počátku procesu nabídne širokou nabídku nemovitostí od nejlevnějších po nejdražší v preferované lokalitě návštěvníkem (tu musí zadat před započetím procesu). Následně je uživatel vyzván k ohodnocení dalších aspoň pěti nemovitostí, ze kterých se vytváří personalizovaný seznam. Ten lze dále upřesňovat tlačítky *like* a *hide*. Služba vyžaduje vytvoření uživatelského profilu či propojení s Facebookem. Z oznámení o spuštění služby lze vyčíst, že doporučení se vyvozují z hodnocení uživatelů, tedy že se jedná o systém kolaborativního filtrování, či hybridní doporučovací systém (viz kapitola 3).

Trulia Suggests je ukázkovým příkladem doporučovacího systému pro realitní kanceláře. Z pozorování vyplývá, že tento agregátor na data všech realitních kanceláří používá právě jeden algoritmus. K aktivaci doporučovacího systému musí návštěvník navštívit speciální sekci webové prezentace. Tím nároky na uživatele nekončí. Stále musí ohodnotit prvních 5 nemovitostí, aby se doporučovací systém aktivoval.

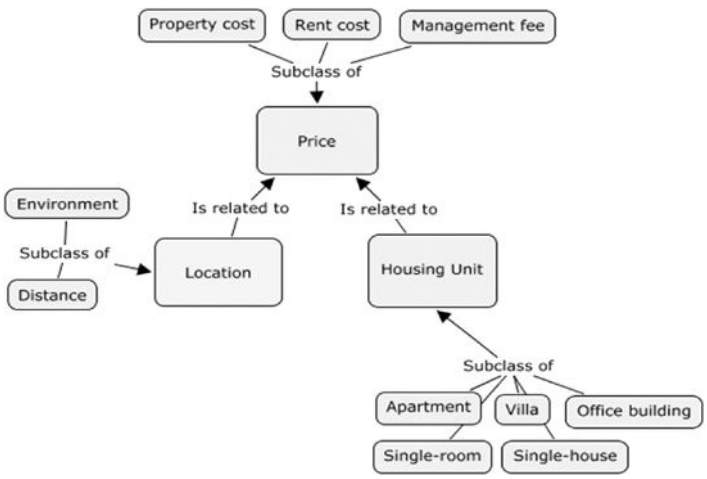
2.5.2 Yuan et al.

Zatímco v České republice se při výběru nemovitosti hledí zejména na cenu, parametry nemovitosti, případně vyhledávače při řazení zohledňují délku setrvání na trhu, jinde, např. ve Spojených státech amerických, se zájemci rozhodují velmi citelně i dle občanské vybavenosti, jelikož výrazně ovlivňuje odváděné daně.

Yuan et al. [5] prezentuje dopručovací systém založený na kombinaci case-based reasoning[6] a související informační ontologie[7]. Pomocí rozpoznávání klíčových slov je pro každou z nemovitostí vytvořen záznam v ontologii (zjednodušenou hierarchii tříd vidíme na obrázku 2.9). Podobně jako nemovitosti jsou i parametry

[6]Case-based reasoning nabízí doporučení dle naučených vztahů. Např.: Pokud mladý bezdětný pár preferuje lokalitu v centru města. Rodiny s malými dětmi vyžadují v rozumné blízkosti školu či školku. Algoritmus odvozování nechápe logické souvztahnosti mezi atributy a na data pohlíží jako na obecné proměnné.

[7]Datová struktura modelující logické vztahy mezi informacemi ze stejné domény. Využívá k tomu koncept tříd, atributů, instancí, vztahů, axiomů a pravidel.

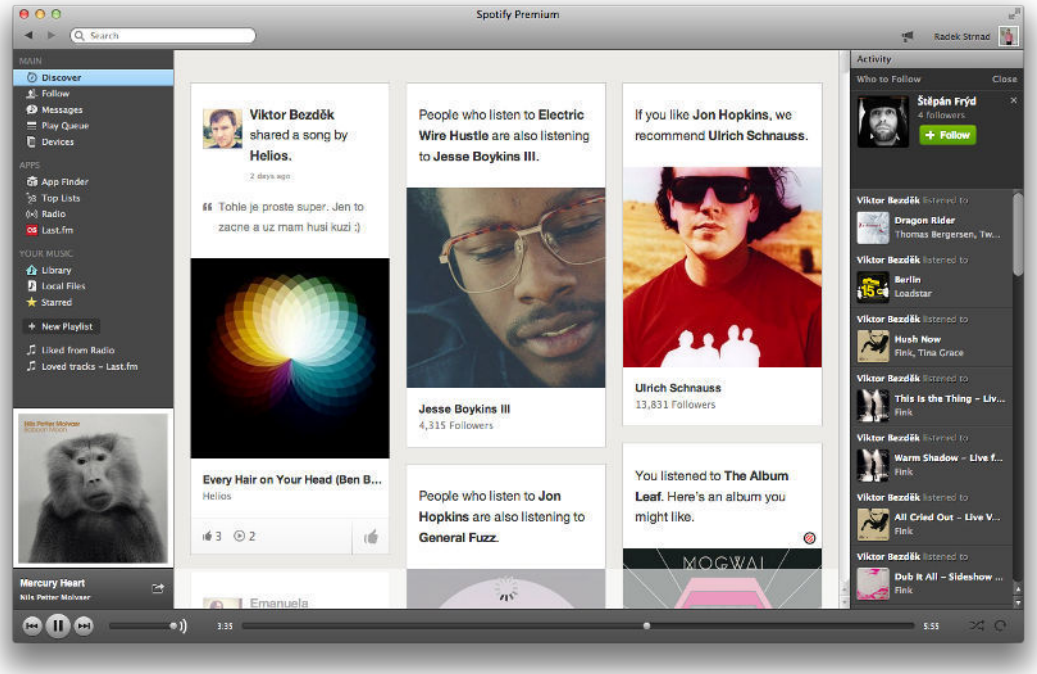


Obrázek 2.9: Yuan et al. - zjednodušená hierarchie ontologie

vyhledávání nemovitostí reflektovány do identické hierarchie. Algoritmem case-based reasoningu jsou vyhodnoceny doporučení a nabídnuty uživateli. Bohužel podrobnosti o fungování algoritmu nejsou v práci uvedeny. Nedostatečné se jeví i vyhodnocení řešení, které je omezeno pouze na dotazník o pěti otázkách, zaměřený převážně na uživatelské rozhraní a jednoduchost jeho používání.

2.5.3 Spotify

Spotify je služba distribující posluchači rozsáhlý hudební katalog za fixní finanční obnos, případně zdarma, ale s vkládanými reklamami mezi jednotlivými skladbami. *Spotify* je samo o sobě sociální sítí (existuje koncept přátel, komentářů, je zde tlačítko „like“), rovněž se propojuje i s ostatními sítěmi (*Facebook*, *last.fm*, aj.) a odtud sbírá data. Kromě zřejmé analýzy poslouchané hudby - „If you like A, we recommend B“, *Spotify* doporučí i dle poslechu ostatních uživatelů s důrazem na interprety vyskytující se v okruhu přátel. Dále se řídí absolutními žebříčky poslechovosti, tedy nabízí aktuální hity, ale doporučí i lokálně významné interprety. Pokud bysme v doporučeních autora na obrázku 2.10 listovali níže, brzy narazíme například na doporučení Jaromíra Nohavicy či Karla Plíhala.



Obrázek 2.10: Spotify

2.5.4 Apache Mahout

Apache Mahout[25] je knihovna algoritmů strojového učení s důrazem na škálovatelnost. Mezi funkční arzenál patří široká paleta algoritmů pro doporučovací systémy - algoritmy kolaborativního filtrování (viz kapitola 3.5), faktorizace matic i pomocné operace pro zjišťování vzdáleností, transformace, prořezávání dat,

shlukování atd. V případě nasazení *Apache Mahout* je potřeba dodržovat konvence kódu (algoritmus *MapReduce*) a zadefinovat *Hadoop* výpočtový cluster. Na druhou stranu programátor nezačíná na zelené louce a mnoho operací (zejména cenné škálování) může kompletně převzít. Existují práce, které se snaží o abstrakci a zjednodušení této knihovny, např. Unresyst[30] Petra Cvengroše.

Apache Mahout byl nasazen v mnoha komerčních i akademických aplikacích. Za všechny jmenujme např. *Amazon*, *Foursquare*, *LinkedIn* a *Twitter*.

2.5.5 Porovnání

Představili jsme si některé existující přístupy k potencionálnímu řešení doporučovacího systému nejen v oblasti realit. Prozkoumali jsme konkrétní řešení agregátoru realit *Trulia Suggest*, kde návštěvník ohodnocuje nemovitosti binárním líbí / nelíbí. Nevýhodou řešení je nutnost doporučovací systém explicitně zapnout a ohodnotit minimální počet nemovitostí. Vzorová aplikace doporučovacího systému nad *RePortem* by se až na zmíněné výtky mohla chovat velice podobně a aktivace by mohla proběhnout až nad určitý počet doporučených nemovitostí. Splnil by se tím jeden ze základních požadavků navrhovaného doporučovacího algoritmu, tedy nenáročnost uživatelské interakce.

U *Yuan et. al.* jsme nahlédli k alternativnímu přístupu za pomocí ontologií. I přes fakt, že v práci nejsou uvedeny konkrétní výsledky, je přínosná svým netradičním pojetím a potencionální inspirací pro ostatní práce.

V případě *Spotify* vidíme na doporučeních konkrétní vysvětlení, proč nám byl daný interpret doporučen. Na této službě je zajímavá rozmanitost použitých doporučovacích algoritmů, pokrývající veškeré kategorie uvedené v kapitole 3.

Nakonec jsme si představili sadu algoritmů *Apache Mahout*, která by dokázala pokrýt požadavky na doporučovací systém nad *RePortem* v případě nutnosti škálování. Jelikož počet uživatelů doporučujícího systému dokážeme řádově odhadnout, budou existovat realitní kanceláře, pro které požadavky na výpočetní prostředky nepřesáhnou výkon jednoho stroje, tedy nebude potřeba škálovat. Autor využije tento případ pro podrobnější seznámení s implementací doporučovacích algoritmů, kdy nebude pouze přejímat již implementované sady. Tím se seznámí s úskalími návrhu a implementace algoritmu doporučovacího systému.

3. Metodiky doporučovacích systémů

V této kapitole si představíme základní principy fungování algoritmů doporučovacích systémů, jejich konvence a značení. Zjistíme, jakým způsobem ovlivňuje chování uživatelů vstupní data. Dále si rozdělíme jednotlivé algoritmy do skupin, nastíníme si jejich příklady a zhodnotíme jejich výhody a nevýhody.

3.1 Interakce uživatelů

Doporučující systémy by nemohly existovat bez interakce s uživateli. Sběr informací o návštěvníkovi pomáhá utvářet detailní profil jeho preferencí, proto stojí za představení několik nejobvyklejších zdrojů. Jako **implicitní** interakce považujeme data o chování návštěvníků v rámci doporučovacího systému, která nejsou spojena s vědomou hodnotící akcí. Typicky se jedná o data sesbíraná z pohybu návštěvníka doporučovacím systémem. Toto chování se snažíme kvantifikovat a přiřadit mu hodnotu na nějaké škále. Jako **explicitní** označujeme vyjádření názoru uživatele, které je až na výjimky už z principu kvantifikované.

Některé z metod interakcí jsou závislé na použité prezentační technologii, proto budeme pro zjednodušení uvažovat prostředí dynamických HTML.

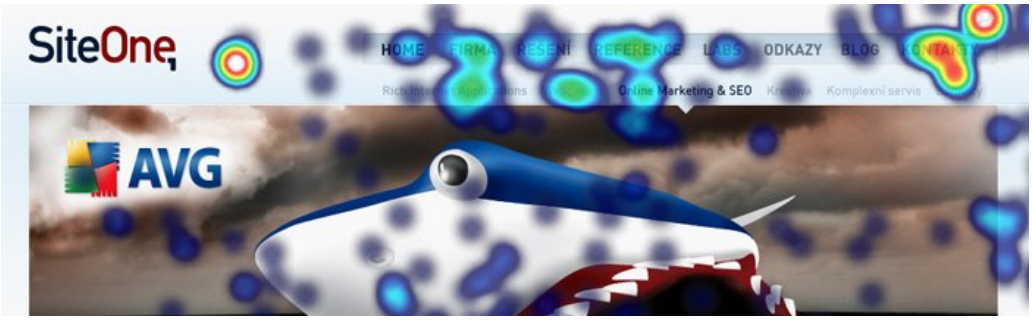
3.1.1 Implicitní

Navštívení položky. Nejprimitivnější implicitní interakcí je samotné navštívení položky např. při listování seznamem. Vyjadřuje jistou míru zájmu o položku.

Akce provedená s položkou. Pokud umožňuje systém provádět s položkami některé specifické akce jako nákup, vložení do košíku nebo přidání mezi oblíbené, lze těmto akcím přiřadit určité pozitivní ohodnocení.

Čas strávený prohlížením. Moderní prohlížeče umožňují sledovat dobu strávenou prohlížením položky. Dokáží i rozpoznat, pokud se uživatel přepnul do jiného okna či záložky a stránku nechal otevřenou. Analýzou doby strávené při aktivním prohlížení položky lze omezeně určit míru zájmu návštěvníka.

Heat mapy. Díky mimovolnému jednání uživatelů, kteří při čtení obsahu přesouvají myš na místo, jež aktuálně čtou, můžeme identifikovat atributy položek, zajímající návětšvníky. Na obrázku 3.1 vidíme nasazení heat map v akci – konkrétně pozice klepnutí myší na stránce. V současné době se tato metoda používá k analýze pohybu návštěvníků na stránce. V oblasti doporučovacích systémů je neprobádaná, nicméně autor práce si myslí, že by se technika dala využít i v této doméně.



Obrázek 3.1: Ukázka heat mapy.

3.1.2 Explicitní

Líbí / nelíbí. Nejjednodušším explicitním ohodnocením položky uživatelem je volba, zda se položka líbí či nelíbí. Obě krajní hodnoty mohou být znázorněny grafickými prvky.

Ohodnocení. Zadefinováním stupnice, např. *zcela souhlasím*, *spíše souhlasím*, *neutrální*, *spíše nesouhlasím*, *nesouhlasím*, případně nabídnutím responsivních grafických prvků (pět hvězdiček, emotikony), zjemníme výrazivo z čistě binárního líbí / nelíbí na různé úrovně libosti či nelibosti položky.

Komentář. Napsáním komentáře může uživatel vyjádřit svou spokojenost či nespokojenost s produktem. Textem lze vyjádřit mnohem jemnější nuance než jen obvyklé ohodnocení. Následná analýza psaného textu a kvantifikace jeho významu je poměrně složitá. Zkoumáním sentimentu psaného textu se zabývá mnoho výzkumů.

3.2 Rozdělení doporučovacích systémů

Díky rozmanitosti nasazení doporučovacích systému, vstupních dat i míře ovlivňování výsledků, nelze jednoduše navrhnout paušální algoritmus. Předtím, než přikročíme k samotnému návrhu doporučovacího systému pro obchodování s realitami, si představíme nejčastěji užívané metodiky, včetně nástinů typicky užívaných algoritmů. Analyzujeme jejich pozitiva i negativa. Doporučující systémy se dělí do tří hlavních rodin dle přístupu:

Systémy založené na analýze obsahu. Obsah každé položky je analyzován pomocí několika předem určených tříd vlastností (atributů). Doporučující systém zkoumá vztah uživatele k těmto atributům a na základě získaných informací vyhodnotí doporučení.

Kolaborativní filtrování. Analýzou matice vztahů *uživatel* × *položka* nabízí doporučení odvíjející se od ohodnocení učiněných dříve jinými uživateli.

Hybridní systémy. Využívá kombinace obou předešlých přístupů k odstranění nežádoucích vlastností některého z nich.

3.2.1 Problémy doporučovacích systémů

Doporučující systémy se již od návrhu potýkají s mnoho omezeními, vyplývajícími z prostředí, ve kterých se nacházejí ([1], [7]).

Řídkost dat. Doporučující systémy operují s obrovskými množstvími dat. Jednotlivý uživatel ale navštíví pouze velmi omezenou množinu dat a ještě menší ohodnotí. Matice ohodnocení *uživatel* × *položka* je proto velmi řídká. Řídkost dat může nastat i v případě nedostatečné transformace textů na vektorovou reprezentaci u analýzy obsahu dat (dále představena v kapitole 3.7).

Studený start. S řídkostí dat souvisí i problém tzv. *studeného startu*, což je označení pro stav, kdy je do doporučovacího systému zavedena nová položka či nový uživatel. Jelikož o nich není sesbírán dostatek dat, položky často nejsou doporučovány a uživatelé neobdrží rozumné výsledky. Pokud si vzpomenete na představený doporučovací systém *Trulia Suggests*2.5.1, ten začal fungovat až po ohodnocení minimálního počtu nemovitostí. Jednalo se právě o zmírnění problému *studeného startu*.

Bezpečnost uživatelů a ochrana dat. Data poskytnutá uživatelem v rámci procesu ohodnocení mohou být považována jako citlivá. V případě propojení s uživatelským účtem se stanou i neanonymními a lze je přiřadit ke konkrétní osobě. Případné zveřejnění informací může být zneužito sociálním inženýringem. V Evropské unii je doporučeno omezení sběru citlivých informací o uživateli nařízením 2009/136/EC Evropského parlamentu [11], které ku příkladu ve Velké Británii vešlo v platnost zákonem 2011 No. 1208 o elektronických komunikacích [12]. Do doby dokončení této práce nebyl v České republice přijat žádný zákon reflektující zmíněnou směrnici.

Kvůli žalobě na ochranu osobních dat nepokračovala v roce 2009 prestižní soutěž *Netflix Prize* [21] i přes to, že publikovaná data neuváděla konkrétnější údaje o uživatelích a ti byli identifikování pouze pod identifikačním číslem.

Škálovatelnost. U rozsáhlých systémů je téměř jisté, že objem dat přesáhne rozumné hodnoty. Stejně tak se i *O(n)* online ohodnocení nových položek dostane za únosnou mez. V tom případě je nutné provést kroky k navrácení do přijatelných hodnot. Typickým řešením bývá předzpracování dat.

Šedé ovce, černé ovce. Jako šedé ovce jsou označovány osoby se značně nekonzistentním názorem. Černými ovcemi pak nazýváme osoby, které cíleně útočí na vlastnosti doporučovacích systémů a cíleně se snaží znevýhodnit některé z položek. Oba případy značně snižují kvalitu doporučení a celkovou věrohodnost systému.

3.3 Značení

Uvažujme doporučovací systémy, kde platí následující zvyklosti:

- *Množina uživatelů* je označena jako U. Uživatelem je např. návštěvník webové prezentace nebo zájemce o koupi nemovitosti.
- *Množina hodnocených položek* je označena jako I. Jako hodnocenou položku označujeme např. konkrétní nabídku realit.
- Libovolný uživatel $u \in U$ může ohodnotit každou položku $i \in I$ maximálně jednou hodnotou z množiny ohodnocení $s \in S$ (např. $S = \{1, 2, 3, 4, 5\}$).
- *Množinu všech ohodnocení* položek uživatelem potom označíme jako $R \subseteq \{r_{ui} : U \times I \times S\}$.
- Od již definovaných množin si zadefinujeme i odvozené množiny pro lepší identifikaci. Jako U_i označíme množinu uživatelů, kteří ohodnotili položku i. Obdobným způsobem zadefinujeme I_u jako množinu položek ohodnocených uživatelem u.
- Kvůli nutnosti častého používání průniku množin budeme místo $I_u \cap I_v$, tedy místo průniku množin položek ohodnocených uživateli u a v, používat značení I_{uv}. Stejným stylem budeme využívat i značení U_{ij} pro průnik množin uživatelů hodnotících položky i a j.

3.4 Systémy založené na analýze obsahu

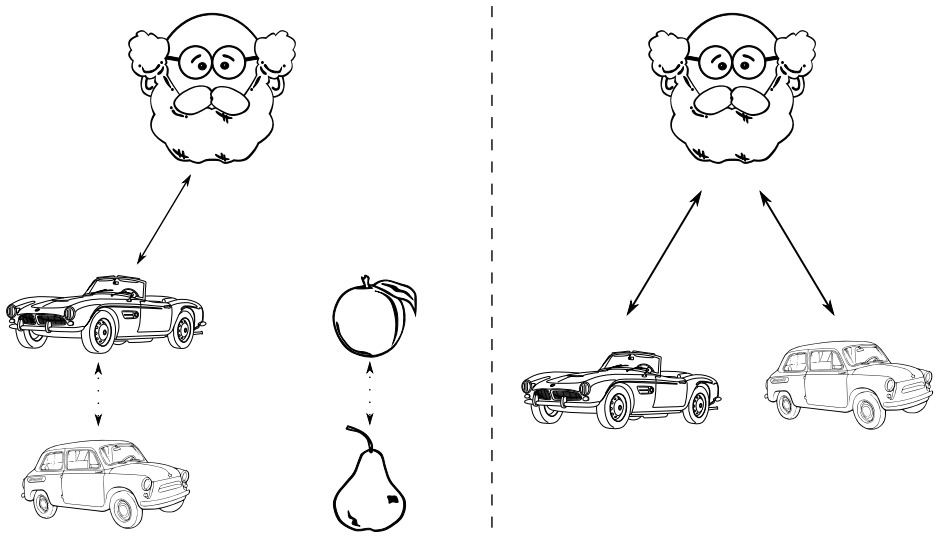
Doporučující systémy založené na analýze obsahu (CBRS[1]) zkoumají vztahy mezi uživateli a položkami [4]. Uživateli se automaticky vytváří profil zájmů na základě interakce s doporučovacím systémem. Analýzou charakteristik položky jsou vytvořeny množiny atributů, které kvantifikují obsah a jsou dále využity k samotnému výpočtu. Pro uživatele se hledají takové položky, které mají podobné obsahové rysy. Doporučující systémy založené na analýze obsahu mohou být nasazené v široké škále aplikací - za obecný pojem položky lze substituovat zboží, webové stránky, instituce, emaily, novinové články, atd.

Doporučující systémy založené na analýze obsahu vykonávají zpravidla tři základní operace nad získanými daty.

Analýza obsahu. V závislosti na zdrojových datech a jejich struktuře se provádí jednodušší či složitější operace v oblasti předzpracování dat. Pokud se pracuje s daty získanými z databázových tabulek, jedná se o výběr sloupců reprezentujících porovnávatelné atributy položek, případně o jejich normalizaci. V případě textového vstupu je třeba provést některou z metod analýzy a extrakce informací z textu, jako je lematizace, stemming a následná vektorová reprezentace textu.

Učení profilu. Analyzovaná a předzpracovaná data jsou vstupem pro učení se profilů a filtrování položek. K syntéze dat se používá některá z metod strojového učení - typicky generalizace (více viz. kapitola 2 v [6]). Společně s kvantifikovanými interakcemi uživatele na jednorozměrné stupnici je sestaven vektor uživatelského profilu.

[1] V anglické literatuře jako Content-Based Recommender Systems



Obrázek 3.2: Systém založený na analýze obsahu

Filtrování položek. Z utvořeného uživatelského profilu se porovnáním s identicky analyzovanými položkami vytvoří seznam doporučení. Výsledek ohodnocení může být binární, či výsledek ohodnocení relevance vypočtené z kosinové míry dvou vektorů reprezentující uživatelský profil a položku.

3.4.1 Keyword-based Vector Space Model

Ač se v mnoha implementacích doporučovacích systémů založených na analýze obsahu využívá sofistikovaných postupů pro předpřípravu dat, nebývá samotný výpočet výsledku doporučení příliš komplikovaný. Principielně jde o reprezentaci všech položek $i \in I$ (v některých literaturách se vyskytuje označení *reprezentace dokumentů*) doporučovacího systému pomocí n-dimenzionálních vektorů.

$I = \{i_1, i_2, ..., i_N\}$ se označí množina všech položek (někdy také *množina všech dokumentů* či *korpus*). $T = \{t_1, t_2, ..., t_n\}$ se označí množina všech slov ve slovníku. V případě, že se každá položka sestává z několika předem určených typů, včetně všech hodnot, které mohou nabývat, je možné určit fixní množinu slov ve slovníku. Zajímavější situace se řeší v případě delších, souvislých textů. Pokud by se text nijak neupravil, pouze by proběhla tokenizace jednotlivých slov, dimenze slovníku T by nepatřičně narostla, čímž by utrpěla použitelnost doporučovací metody. V takovém případě přichází na řadu operace, často používané při sémantickém zpracování textu, která vede k redukci dimenze.

Odstranění stop slov. Stop slovo je velmi často používáné slovo, které nenese dostatečnou sémantickou důležitost, proto může být odstraněno. Jde o předložky, pomocná slovesa, příslovce aj. Ačkoliv neexistuje univerzální seznam stop slov fungujících ve všech prostředích, některé databázové stroje [13] mají implementovány seznam pro usnadnění operací s textem.

Stemming. Díky rozmanitosti jazyka se slova objevují v textu v různých tvarech, mající podobný sémantický význam. Ořezáním slova až na kořen (anglicky *stem*) se sice ztratí některé informace (např. *ryba* $\rightarrow$ *ryb*, *rybář* $\rightarrow$

ryb), nicméně jde o velmi efektivní reduktivní metodu. Podobně jako u odstranění stop slov, ani stemming nemá univerzální algoritmus, ale rovnou několik [14], [15].

Využití thesaurů a encyklopedických znalostí. V redukci dimenzí vektorového prostoru se dá pokračovat aplikací některé z pokročilých technik identifikující synonyma (např. *WordNet*) či totožné encyklopedické kategorie (např. využití *Wikipedie*).

Po redukci definici a redukci klíčových slov nastává vektorizace položek. Každá položka doporučovacího systému je určena n-dimenzionálním vektorem $i_j = \{w_{1j}, w_{2j}, w_{3j}, ..., w_{nj}\}$, kde w_{kj} je váhová příslušnost položky i_j ke slovu t_k.

3.4.2 TF-IDF

Jedna z nejběžnějších metod na výpočet váhové příslušnosti slova t_k k položce i_j a určení příbuzných vektorů je **TF-IDF** (term frequency - inverse document frequency, popsána např. v [16]).

$$TF - IDF(t_k, i_j) = \overbrace{TF(t_k, i_j)}^{TF} \times \overbrace{log\frac{|I|}{n_k}}^{IDF} \tag{3.1}$$

Jak už z názvu vyplývá, výpočet je rozdělen na dvě části - na výpočet výskytu atributu (TF - term frequency) a na následnou normalizaci (IDF - inverse document frequency), kde $|I|$ je celkový počet položek (kardinalita) a n_k je počet položek, kde se slovo t_k vyskytlo aspoň jednou. K určení frekvence četnosti slova t_k a položky i_j TF existuje mnoho přístupů od primitivních booleanových frekvencí, až po např. vážené frekvence, které určuje následující rovnice. Funkce $f(k, j)$ v této rovnici určuje počet výskytů slova t_k v položce i_j.

$$TF(t_k, i_j) = 0.5 + \frac{0.5 \times f_{k,j}}{max_z f_{z,j}} \tag{3.2}$$

Pokud je potřeba, následující rovnicí normalizujeme váhové hodnoty do intervalu $\langle 0, 1\rangle$.

$$w_{k,j} = \frac{TF - IDF(t_k, i_j)}{\sqrt{\sum_{s=1}^{|T|} TF - IDF(t_s, i_j)^2}} \tag{3.3}$$

3.4.3 Výpočet vzdálenosti vektorů

Ačkoliv pro výpočet vzdálenosti vektorů z n-dimenzionálního prostoru existuje několik metod, nejčastěji používánou je kosinova míra:

$$sim(d_i, d_j) = \frac{\sum_k w_{ki} \cdot w_{kj}}{\sqrt{\sum_k w_{ki}^2} \cdot \sqrt{\sum_k w_{kj}^2}} \tag{3.4}$$

3.4.4 Výhody a nevýhody

Analýza obsahu díky svému návrhu stanoví některé zajímavé vlastnosti. Doporučení nezávisí na předchozích doporučeních jiných uživatelů, proto se množina doporučených objektů odvíjí čistě od vkusu konkrétního uživatele. Je mnohem jednodušší odhadnout chování algoritmu. Ten může nabídnout i položky, které jsou nově přidané do systému za předpokladu, že jsou zanalyzovány. Algoritmus proto netrpí z hlediska položek problémem tzv. studeného startu.

Při zavedení nového uživatele do doporučovacího systému se ale problém studeného startu projeví. Nedostatek informací o uživateli a chybějící identifikace klíčových atributů pro konkrétního uživatele vede k nepřesným, či přímo chybným doporučením.

Během popisu algoritmu jsme si definovali slovník T. Množina slov $\{t_1, t_2, ..., t_n\}$, ať už určena pevně předem, či generována automaticky za běhu, je vždy konečná. Dozvěděli jsme se o nutnosti zachování rozumně malé dimenze slovníku. Omezený počet slov zpravidla nedokáže kvantifikovat veškeré nuance obsahu. Právě chybějící slovo může být rozhodující pro správné doporučení.

Poslední problém doporučovacích systémů založených na analýze obsahu se týká oddělenosti a nezávislosti všech uživatelských profilů. Vezměme v úvahu obchod s hudebními nosiči. Pokud si autor oblíbí a pozitivně ohodnotí všechny desky od jednoho interpreta, doporučovací systém mu bude obtížně hledat alternativy mimo žánr zmíněného umělce. Přesto můžeme předpokládat, že hudební vkus uživatele není takto omezený a rád si poslechne i něco jiného.

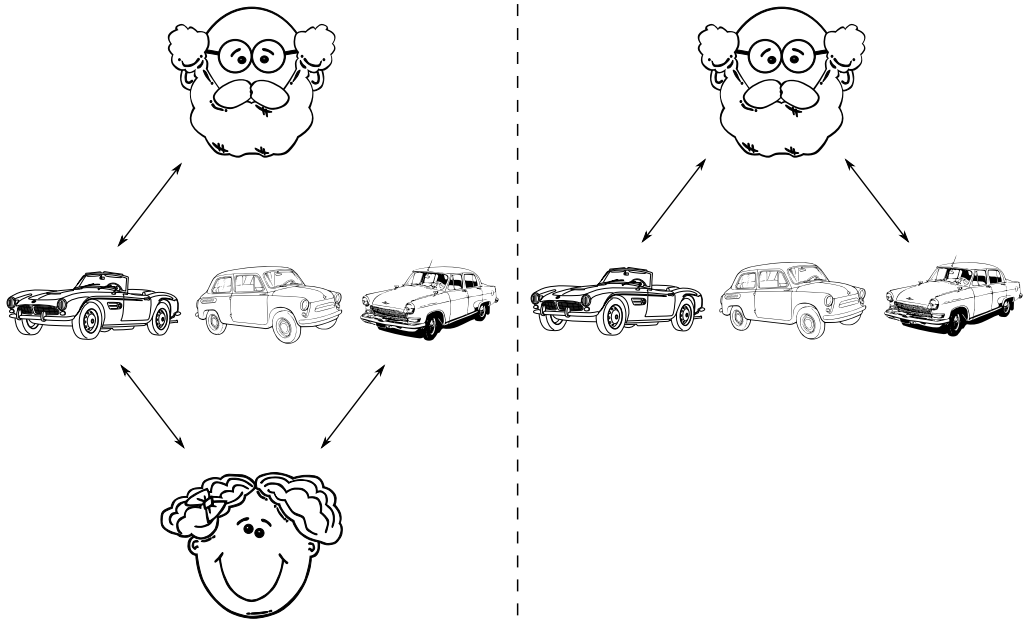
3.5 Kolaborativní filtrování

Metodika kolaborativního filtrování (CF[2]) využívá znalostí předchozích preferencí skupiny uživatelů, jež přetváří v doporučení dosud neznámých položek [7]. Fundamentální myšlenkou kolaborativního filtrování je předpoklad, že pokud uživatel A sdílí s uživatelem B vysoký počet shodných ohodnocení, lze se domnívat, že s vysokou pravděpodobností uživatel A ohodnotí položku x stejně jako uživatel B, ikdyž se tak ještě nestalo.

Názorný příklad ohodnocení několika filmů nalezneme v matici *uživatel* × *položka*, reprezentované tabulkou 3.1. Uvažujeme binární stupnici ohodnocení - líbí / nelíbí pro čtyři uživatele a čtyři filmy. Úkolem je zjistit, zda se bude film *The Iron Lady* líbít Daniele. Analýzou shod ohodnocení společně shlédnutých a ohodnocených filmů zjistíme, že vkus Daniely se s Anežkou zcela neshoduje, s Bedřichem se shoduje v polovině případů a s Cyrilem ve všech společně shlédnutých filmech - v tomto případě pouze v jednom. Intuitivně je „nejbližším“ sousedem Daniely Cyril, tedy film *The Iron Lady* se bude Daniele pravděpodobně líbit.

S nalezením ohodnocení položky úzce souvisí i další problémy - nalezení nejvhodnější položky pro uživatele, či nalezení N nejvhodnějších položek. Problematika kolaborativního filtrování je oproti analýze obsahu komplexnější, existuje více náhledu na řešení, z nichž si některé představíme. Jedná se zejména o *memory-based* a *model-based* algoritmy.

[2]V anglické literatuře jako Collaborative Filtering



Obrázek 3.3: Systém založený na kolaborativním filtrování

	Skyfall	**Lincoln**	**Django Unchained**	**The Iron Lady**
Anežka	✓		✓	×
Bedřich	✓	✓	×	×
Cyril		✓	×	✓
Daniela	×		×	?

Tabulka 3.1: Matice ohodnocení filmů uživateli

3.5.1 Memory-based CF

Algoritmy *memory-based CF* (v některé literatuře také jako *neighborhood-based CF*) jsou velice efektivní a poměrně jednoduché na implementaci. Pro výpočet využívají přímo celou množinu dat, či její reprezentativní vzorek. Cílem algoritmu je identifikovat nejbližšího souseda či množinu sousedů a z jejich uskutečněných ohodnocení vyřešit úlohu kolaborativního filtrování - předpovědět *ohodnocení položky* $\bar{r}_{ui}$, nebo doporučit aktivnímu uživateli položky k prozkoumání. Vyřešení problému lze provést dvěma odlišnými přístupy.

User-based recommendation

Můžeme na něj nahlížet z hlediska uživatele, tedy snažíme se nalézt k nejbližších uživatelů (k-NN[3]) k aktivnímu uživateli u, kteří už položku i ohodnotili (množinu označíme $\mathcal{N}_i(u)$). Míru totožnosti dvou uživatelů $u \neq v$ označíme jako váhu w_{uv}. K určení váhy se da použít např. již představená kosinova míra (rovnice 3.4.3), Pearsonova korelace, případně jiná míra. Samotný výpočet předpovědi ohodnocení položky $\bar{r}_{ui}$ se počítá buď jako jednoduchý vážený průměr z provedených

[3] v anglické literatuře jako k-nearest-neighbors

ohodnocení a vah uživatelů

$$\bar{r}_{ui} = \frac{\sum_{v \in \mathcal{N}_i(u)} w_{uv} r_{vi}}{\sum_{v \in \mathcal{N}_i(u)} \mid w_{uv} \mid}, \tag{3.5}$$

případně se bere v potaz úvaha, že i při fixní stupnici ohodnocení boduje každý uživatel odlišným způsobem. Například některý z uživatelů nejvyšším ohodnocením označí pouze výjimečné položky, zatímco jiný označí nejvyšším ohodnocením klidně i většinu nabídnutých položek. Doporučeným postupem, popsaným v [17], je opět výběr některé z normalizační funkce h, převod ohodnocení $h(r_{ui})$ do normalizované stupnice a převod váženého průměru inverzní funkcí zpět do původní stupnice:

$$\bar{r}_{ui} = h^{-1}\left(\frac{\sum_{v \in \mathcal{N}_i(u)} w_{uv} h(r_{vi})}{\sum_{v \in \mathcal{N}_i(u)} \mid w_{uv} \mid}\right) \tag{3.6}$$

Item-based recommendation

Duálním přístupem k řešení problematiky *memory-based CF* je nalezení k nejbližších položek namísto uživatelů. Podbně jako u *user-based* přístupu i zde si zadefinujeme $\mathcal{N}_u(i)$ jako množinu položek, ohodnocených uživatelem u, nejvíce podobných položce i. Míru totožnosti dvou položek i a j (váhu) pak značíme jako w_{ij}. Vzorec pro výpočet předpovědi ohodnocení by neměl čtenáře ničím překvapit.

$$\bar{r}_{ui} = \frac{\sum_{j \in \mathcal{N}_u(i)} w_{ij} r_{uj}}{\sum_{j \in \mathcal{N}_u(i)} \mid w_{ij} \mid} \tag{3.7}$$

Pro úplnost uvedeme i normalizovanou verzi.

$$\bar{r}_{ui} = h^{-1}\left(\frac{\sum_{v \in \mathcal{N}_i(u)} w_{uv} h(r_{vi})}{\sum_{v \in \mathcal{N}_i(u)} \mid w_{uv} \mid}\right) \tag{3.8}$$

Vidíme, že optimalizace výpočtu je prováděna zejména výběrem způsobu výpočtu vah w a předpřípravou dat (filtrování, normalizace). Algoritmy *memory-based CF* mají pouze jednu proměnnou a to je k, tedy počet sousedních uživatelů či položek. Rozhodování mezi *user-based* a *item-based* metodikou by měla proběhnout na základě poměru mezi uživateli a položkami. V případě, že počet aktivních uživatelů převyšuje násobně počet položek (jako např. již zmíněný *Amazon*), je vhodné hledat podobné položky, namísto uživatelů [31]. V opačném případě (převaha položek nad uživateli) se doporučuje hledat podobné uživatele. Existují i experimentální metody, kdy se oba přístupy kombinují [18] a zvyšují tím robustnost algoritmu.

3.5.2 Model-based CF

Model-based CF algoritmy se vyznačují podobnými rysy jako metodiky používáné při dolování informací. Systém se naučí model na testovacích datech rozpoznávat složité vzorce chování. Aplikuje kategorizaci na reálných či testovacích datech. Obdobně jako u *memory-based CF* i *model-based CF* mají mnoho různých algoritmů, které lze do této rodiny řadit. Ať už se jedná o modely založené

na Bayesovských sítích, neuronových síťích, SVM [4], rozhodovacích stromech, náhodné procházce, Markovových řetězcích, případně jiných pravděpodobnostních metodách, či, v poslední době oblíbená, faktorizace matic. Jelikož hloubkové představení všech důležitých metod jde nad rámec diplomové práce, je na čtenáři, aby si v tomto odvětví prohloubil své znalosti. Přesto si pro názornost představíme jeden z modelů založený na bayesově teorému podmíněné pravděpodobnosti.

Jednoduchý Bayesův model

Jeden z primitivnějších modelů CF je jednoduchý Bayesův model [23], vycházející z Bayesovy podmíněné pravděpodobnosti. Právě pro svou rychlost a nenáročnost (učící fáze roste lineárně s počtem testovacích případů) je často ideální v případě, kdy se dá předpokládat (odtud také také další označení "naivní"), že ohodnocení položek jsou na sobě nezávislé (proměnné F). Ačkoliv se jedná o poměrně silný předpoklad a v praxi zřídkakdy garantovaný, mnohdy se zanedbává a přitom podává dobré výsledky. Pak platí následující rovnosti:

$$p(S \mid F_1, \ldots, F_n) = p(S)p(F_1 \mid C) \ldots p(F_n \mid S) = p(S) \prod_{i=1}^{n} (F_i \mid S) \tag{3.9}$$

Klasifikace neznámé položky se provede výpočtem podmíněné pravděpodobnosti a lze ji aplikovat jak na user-based (za F_i dosadíme $u_j = s_k$), tak i na item-based (za F_i dosadíme $i_j = s_k$) filtrování. Díky tomu, že matice *uživatel* × *položka* nemá ohodnocené všechny kombinace, probíhá výpočet součinu pravděpodobností přes ohodnocené (index o):

$$s = \arg\max_{s_j \in S} p(s_j) \prod_{o} p(X_o = s_o | s_j) \tag{3.10}$$

K výpočtu dílčích pravděpodobností se doporučuje *Laplaceův odhad*:

$$p(X_i = x_i \mid Y = y) \approx \frac{\mid X_i = x_i \mid + \mid Y = y \mid +1}{\mid Y = y \mid + \mid \{X_j \in X : X_j = X_i\} \mid} \tag{3.11}$$

Ačkoliv se právě představený algoritmus může řadit mezi *memory-based CF*, tedy celý výpočet lze udržet v paměti, k složitějším variantám Bayesových modelů takto přistupovat nelze. Jedná se o pravé *model-based CF*.

3.5.3 Výhody a nevýhody

Představené algoritmy kolaborativního filtrování jsou z principu fungování doménově nezávislé. Nehledí na kategorizaci položek ani uživatelů, tedy umožňují míchat různá data, a přesto by systém měl podávat korektní výsledky. Odpadá analýza během přidání položky do systému, která v některých případech může být komplikovaná (existují atributy, které se velmi obtížně kvantifikují). Absence rozboru a nezávislost na atributech položek při rozhodování vede k obecně příznivějším, někdy i pro uživatele překvapivým výsledkům, v porovnání se systémy založenými na analýze obsahu. Tedy uživateli mohou být nabídnuty položky, které

[4] Support Vector Machine

sice svým obsahem neodpovídají očekávání, nicméně jsou pro uživatele hodnotné. Na druhou stranu jakékoliv chybějící informace o položce těsně po zavedení znemožňují její doporučení. Problém studeného startu se u metod kolaborativního filtrování týká nejen nových uživatelů, ale i nových položek.

3.6 Hybridní systémy

Každá ze dvou hlavních metodik doporučovacích systémů, tedy systémy založené na analýze obsahu a kolaborativní filtrování mají již z návrhu značné nedostatky. Nasazení systému založeném na analýze obsahu znamená pro uživatele značnou předvídatelnost, která může být na škodu, kolaborativní filtrování zase trpí studenými starty. Pod skupinu hybridních systémů řadíme kombinace různých přístupů, které zmírňují omezení vybraných metodik či vylepšují výsledky doporučení. Dají se kombinovat nejen systémy založené na anlýze obsahu a kolaborativního filtrování, ale obecně i libovolné další metodiky. Příkladem hybridního systému je *content-boosted collaborative filtering* [32] - metodika, která vychází z *kolaborativního filtrování*. Ta, jak víme, trpí problémem studeného startu, tedy řídkostí matice doporučení. Místo klasické matice doporučení algoritmus využívá pseudomatici, ve které jsou některé hodnoty doplněny na základě zpracování obsahu a matice zhuštěna.

3.6.1 Výhody a nevýhody

Hybridní doporučovací systémy při správném návrhu odstraňují nedostatky výchozích kategorií doporučovacích systémů. V problematických situacích doručují lepší výsledky. Problémy s touto metodikou rostou spolu s komplikovaností algoritmu. Pokud se bavíme o již zmíněných *content-boosted collaborative filtering* nebo o *kaskádových hybridních systémech*[5], setkáme se s problémem identifikace bodu, kdy se přechází od jednoho algoritmu k druhému.

3.7 Doporučující systémy zohledňující kontext

Výsledky doporučovacích systému se nemusí nutně odvíjet pouze od sledování a vyhodnocování chování uživatele. V některých případech je vhodné znát kontext uživatele[33]. Uvažme systém nabízející vstupenky na kulturní akce. Pokud budeme znát lokalitu návštěvníka (např. dle IP adresy nebo GPS modulu v mobilním telefonu), můžeme zaměřit výsledky na představení v blízkém okolí, případně rádius se zvyšujícím se časovým odstupem rozšiřovat. S nástupem sociálních sítí a zveřejňování osobních informací se lze k profilům uživatelů dostat snadněji, než dříve. Nabízejí totiž API pro provázání uživatelských účtů a přístupu k osobním údajům. Základní údaje o uživateli - věk, pohlaví, najdou uplatnění nejen u kulturních akcí. Kontext se nemusí nutně vztahovat vzhledem k uživateli. Doporučující systémy cestovních kanceláří by měly zohledňovat sezóny roku. V případě hudebního a filmového průmyslu je vhodné sledovat výroční ceny, programy festivalů, úmrtí umělců atd.

[5]Doporučující systém, kde je za sebou zřetězeno několik doporučovacích systémů. Výsledkem je doporučení prvního úspěšného algoritmu v řadě.

4. RePort recommender framework

Kapitola představí návrh a základní technická řešení frameworku doporučovacích systémů v rámci *RePortu*. Čtenář se seznámí s požadavky na tento framework a omezeními vyplývajícími z nasazením nad existujícím informačním systémem *RePort* . Následně je představena architektura frameworku, logika jednotlivých komponent frameworku, vzájemná komunikace mezi komponentami i externími prezentacemi realitních kanceláří. Dále je rozvedeno úložiště dat skladovaných pro potřeby algoritmů.

4.1 Rozšíření RePortu o doporučovací systémy

RePort je navržen jako informační systém pro více než jednu realitní kancelář. Každá společnost s sebou přináší různý způsob obchodování a jiné požadavky na informační systém. V kapitole 2.4 jsme zjistili, že různé realitní kanceláře mohou mít odlišné potřeby, co se týče doporučovacích systémů. Naším cílem bude postavit dostatečně robustní základnu pro stavbu na míru šitých doporučovacích systémů. Framework *Report RF (Report Recommender Framework)*, který si nyní představíme a který je funkčním rozšířením *Reportu* o platformu pro implementaci doporučovacích systémů s následujícími požadavky:

Využití existující architektury RePortu. *RePort RF* by měl co nejvíce využít stavájící architektury *RePortu*. Aplikace je navržena poměrně slušně modulárně a v nastoleném trendu by měla pokračovat.

Neuniformnost algoritmů. Každá z realitních kanceláří bude mít v rámci *RePortu* možnost volby odlišného algoritmu.

Logické oddělení jednotlivých algoritmů. Doporučující systémy jednotlivých realitních kanceláří by se neměly vzájemně ovlivňovat, proto je nezbytné je logicky oddělit. Jako idealní řešení se jeví přiřazení dedikovaného doporučovacího systém každé realitní kanceláři, který bude přesně odpovídat zadaným požadavkům.

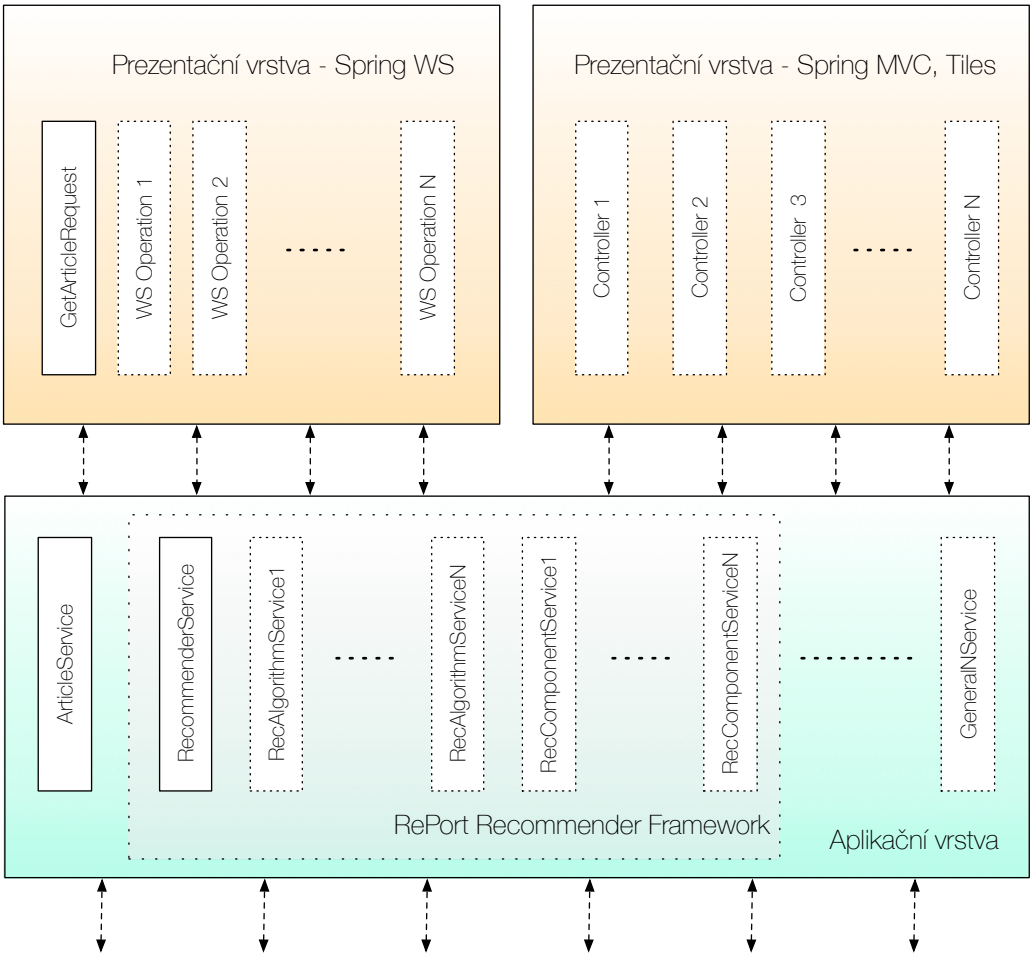
Automatický sběr dat. V kapitole 3.1 jsou uvedeny způsoby interakce uživatele s doporučovacím systémem. Abychom ulehčili implementaci konkrétního doporučovacího algoritmu, je vhodné hned od začátku sbírat maximální množství informací o uživatelských interakcích.

Standardní rozhraní algoritmů. Dá se předpokládat, že realitní kanceláře budou experimentovat s jednotlivými implementacemi doporučovacích algoritmů. Pro snadnou výměnu by měly všechny algoritmy implementovat jednotné rozhraní.

Napojení algoritmů na události. Jedním z častých problémů doporučovacích systémů je problém studeného startu (viz kapitola 3.2.1). Algoritmy se s nimi vyrovnávají rozdílnými způsoby, např. doplněním matice doporučení (kapitola 3.6). Pro tyto účely by měl nově navržený *RePort RF* v případě předem daných událostí (přidání nového uživatele / položky) automaticky notifikovat doporučovací algoritmus.

4.2 Začlenění RePort Recommender Frameworku do RePortu

Design *RePort RF* byl do jisté míry ovlivněn použitým *Spring framework*em. Doporučenými praktikami při vytváření aplikace v tomto prostředí je zapouzdřovat jednotlivé funkční logické celky do tzv. služeb (services) a jejich propojení pomocí Inversion of Control. Každá služba je navíc automaticky implementována jako návrhový vzor *singleton*, pokud není explicitně nakonfigurováno jinak. To vedlo k myšlence rozdělit doporučovací systém na několik nezávislých komponent, reprezentovaných jako jednotlivé služby, vždy se starající o jeden logický okruh. Připomeňme si, že doporučovací systém je primárně určen pro klientské prezentace nemovitostí, které s *RePortem* komunikují přes *Web Services* (viz architektura informačního systému na obrázku 2.5).



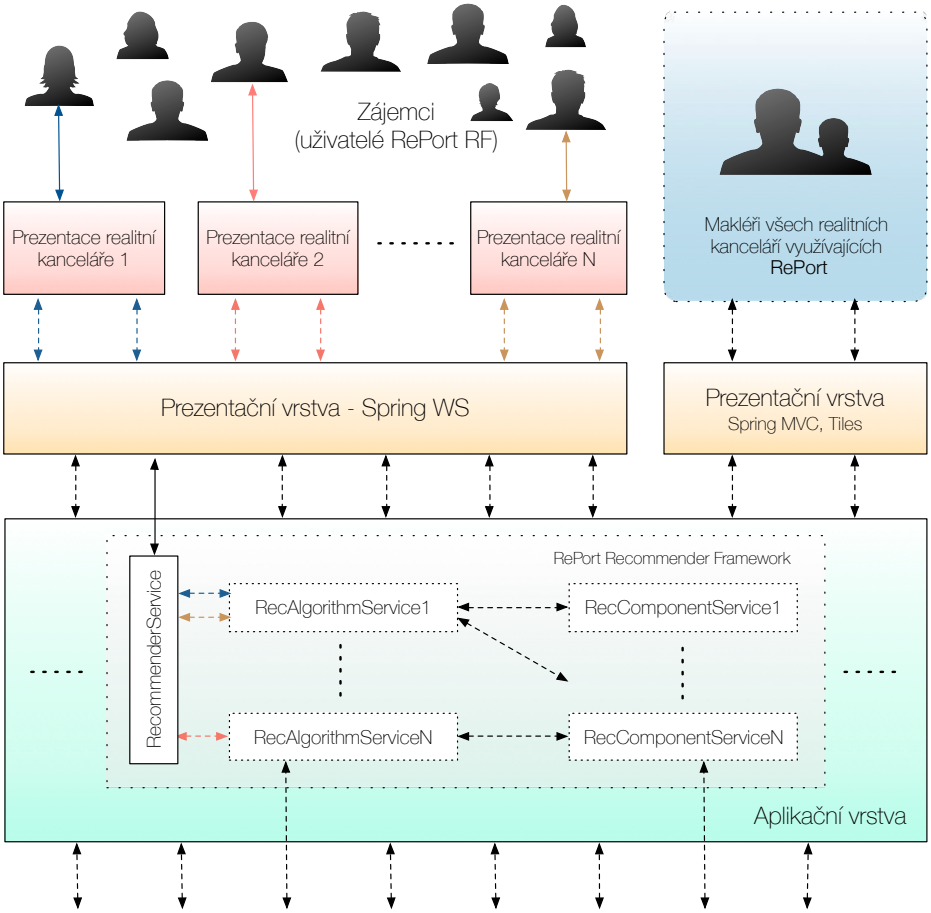
Obrázek 4.1: Zjednodušený diagram začlenění RePort Recommender Frameworku.

Diagram 4.1 zobrazuje zjednodušené začlenění *RePort RF* do stávajícího konceptu *RePortu*. *RePort RF* je soubor služeb a přidružených DAO, implementovaných pomocí IoC obdobým způsobem tak, jak je znázorněno u obrázku 2.3. Je rozděleno na tři logické okruhy, které budou představeny podrobněji v následu-

jících odstavcích. Jedná se o hlavní logiku frameworku, implementaci algoritmů a o sdílené komponenty.

4.3 Hlavní logika frameworku

Výměna dat *RePortu* s webovými prezentacemi realitních kanceláří probíhá přes Web Services. Nastavení profilu realitních kanceláří *RePortu* je doplněno o konfiguraci doporučovacího systému. Web services endpoint komunikuje s hlavní službou doporučovacího systému - `RecommenderService` [1]. Její interface je totožný s rozsahem metod, poskytovaným jednotlivými implementacemi algoritmů. Služba funguje jako transparentní výhybka ke všem implementovaným algoritmům (viz obrázek 4.2. Data přicházející z Web Services endpointu jsou obohacena o identifikační informace uživatele i zdrojové realitní kanceláře, a to stačí k spolehlivému roztřídění. Návrh hlavní logiky *RePort RF* elegantně řeší požadavky vytyčené v kapitole 4.1.



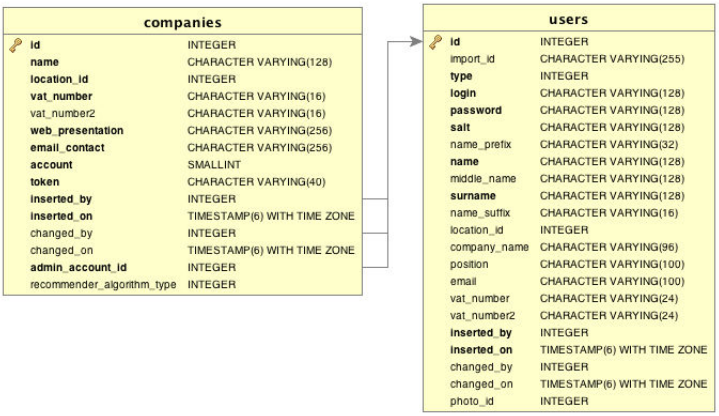
Obrázek 4.2: Diagram toku dat mezi uživateli a RePort RF.

[1] Veškeré zdrojové kódy týkající se *RePort RF* se nacházejí v balících obsahující název „`recommender`“. Tedy Java beans se nacházejí v `cz.cuni.mff.report.bean.recommender`, služby v `cz.cuni.mff.report.service.recommender`, atd.

4.3.1 Identifikace doporučovacích systémů a asociace algoritmu

RePort RF může implementovat neomezené množství doporučovacích algoritmů. Implementace, ladění a správa algoritmů je plánována striktně v režii provozovatele *RePort* , proto se v práci nepočítá se zavedením skriptovacího (meta) jazyka pro implementaci samotného algoritmu, jehož kód by se zaváděl za běhu aplikace. Implementace je provedena v nativním programovacím jazyku *RePortu*, tedy v Javě.

Každý algoritmus má přiřazeno unikátní číslo, jehož mapování na konkrétní třídu je provedeno ve výčtovém typu `AlgorithmType`. Jednotlivé algoritmy jsou implementovány jako separátní služba v balíku `recommender.algorithm`. Přiřazení algoritmu k realitní kanceláři je identifikováno ve sloupci `recommender_algorithm_type` v tabulce `companies`. Ta reprezentuje všechny realitní společnosti v systému. Díky kardinalitě 1:n lze jeden algoritmus aplikovat pro více kanceláří. V praxi by to mohlo vypadat tak, že bude implementována odladěná sada algoritmů pro různé typy společností (malé, velké, obchodující pouze v Praze, celé ČR či zahraničí...) a *RePort* nejen že srazí dobu vývoje webové prezentace, ale i rovnou poskytne funkční doporučovací systém.



Obrázek 4.3: Tabulka companies

4.4 Implementace algoritmů

Z předchozího textu známe veškeré požadavky na implementaci algoritmu doporučovacího systému v rámci *RePort RF* . Jedná se o vhodně pojmenovanou službu umístěnou do balíku `recommender.algorithm`, jejíž metody implementují interface `RecommenderService`. Kromě spřažených událostí, dále rozvedených v 4.4.1, se jedná o klíčové metody doporučování nemovitostí, konkrétně:

Seřazení nemovitostí dle doporučovacího algoritmu. Metoda `sortOffers(List<Offer> offers, RecommenderVisitor visitor)` setřídí pro návštěvníka `visitor` seznam nemovitostí `offers`, předaný parametrem, pomocí algoritmu příslušnému realitní společnosti.

Top-N příbuzných nemovitostí. Pokud se uživatel zajímá o konkrétní nemovitost, zpravidla by rád dostal informace o podobných nabídkách realitní společnosti. K domu slouží metoda `findTopNRelatedOffers(Offer offer, RecommenderVisitor visitor, int n)`, která k dané nemovitosti a uživateli doporučovacího systému navrátí n příbuzných nabídek.

Ohodnocení konkrétní nemovitosti. *RePort RF* dokáže ohodnotit nemovitost hodnotou typu `Double` dvěma různými způsoby. Jedním je ohodnocení z hlediska konkrétního uživatele doporučovacího systému, kdy metoda `estimateOfferRating(Offer offer, RecommenderVisitor visitor)` vrací doporučení vzhledem k vytvořenému uživatelskému profilu. Druhou možností je metoda `estimateOfferRating(Offer offer)`, kdy se globálně analyzuje celková oblíbenost nemovitosti. Funkčnost druhé varianty a její využití osvětlí kapitola 5.5.4.

V kapitole 3 se popisovaly různé metodiky doporučovacích systémů. Zjistili jsme, že zejména v oblasti hybridních doporučovacích systému se části algoritmů recyklují z jiných oblastí. Případně se kaskádově spojují do funkčních celků, kdy samotný hybridní algoritmus je jakousi skládankou existujících dílků. *RePort RF* i *Spring framework* je navržen tak, aby co nejvíce sdílel komponenty, které mohou fungovat jako jednotlivé funkční celky. Více se tomuto tématu věnuje kapitola 4.5.

4.4.1 Spřažené události

U systému založených na analýze obsahu (kapitola 3.4) jsme zjistili, že je potřeba analyzovat obsah všech položek v doporučovacím systému. V našem případě se jedná o nabízené nemovitosti. Ty zůstávají téměř po celou dobu setrvání v databázi statické. Mění se pouze na pokyn makléře, např. v případě úpravy ceny. Z důvodu uspoření výpočetního výkonu a zrychlení odezvy systému by se měla data zpracovat v momentě vložení do databáze či její změny. Na službu starající se o správu nabídek (`MinisiteService`) jsou v případě uložení (změny) či vymazání nemovitosti spřaženy metody `void offerPersistedHook (Offer offer)` a `void offerRemovedHook (Offer offer)`. Chování spřažených funkcí bysme mohli přirovnat k databázovým triggerům.

Podobný trigger můžeme vyžadovat i u návštěvníků. V kapitole 3.2.1 jsme narazili na problém nových uživatelů a absenci dat. Zavoláním metody `void visitorPersistedHook (RecommenderVisitor visitor)` při změně dat o návštěvníkovi můžeme docílit předvyplnění matice doporučení, případně opět provést předzpracování dat. Podobně funguje i metoda pro odstranění návštěvníka (např. po vypršení časového limitu) `visitorRemovedHook`.

4.5 Sdílené komponenty

Různé algoritmy kolaborativního filtrování vychází z několika mála základních přístupů. Zjistili jsme, že např. algoritmy kolaborativního filtrování si potřebují ukládat matici ohodnocení uživatelů. Algoritmy založené na analýze obsahu vyžadují komponentu pro rozbor obsahu položek, aj. Při vhodném, dostatečně volném designu, můžeme navrhnout jednoúčelové komponenty, jejichž funkcionalitu lze sdílet ve více implementacích algoritmů. Algoritmy mohou sdílet libovolný

počet pomocných komponent k provedení výpočtu či obdržení dodatečných dat. V *RePort RF* je každá komponenta reprezentována jednou službou v balíku `service.recommender.component`. Pamatujme, že díky singletonovým návrhovým vzorům se v paměti drží právě jedna instance každé služby. Je na místě zdůraznit, že každá komponenta musí explicitně řešit oddělení dat každé z realitních kanceláří, aby nedošlo k vzájemnému ovlivňování. I z toho důvodu mezi základní výbavu *RePort RF* patří následující komponenta.

4.5.1 Komponenta runtime dat

Při návrhu doporučovacího algoritmu se dříve či později řešitel setká s problémem odstínění dat jednotlivých realitních kanceláří. Jelikož jsou algoritmy implementovány jako singleton service, není možné pro každou z kanceláří vytvořit zvláštní instanci bez rekonfigurace *Spring frameworku*. Řešením může být používání komponenty `RecommenderRuntimeDataService`, která se stará o dočasné uložení a načtení běhových dat do paměti. V principu jsou dvě vnořené mapy, z nichž ta vnější má za klíč identifikátor realitní kanceláře, vnitřní klíč může být volen obecně, např. řetězcová hodnota. `RecommenderRuntimeDataService` je funguje jako jednoduchý wrapper kolem datové struktury `Map<Company, Map<String, Object»`, do které si může programátor po dobu běhu programu pomocí getteru a setteru uložit libovolné datové struktury. Při používání je třeba dbát zvýšené opatrnosti na přetypování objektů, jelikož programátor přistupuje k obecnému objektu a nefungují kontroly překladače.

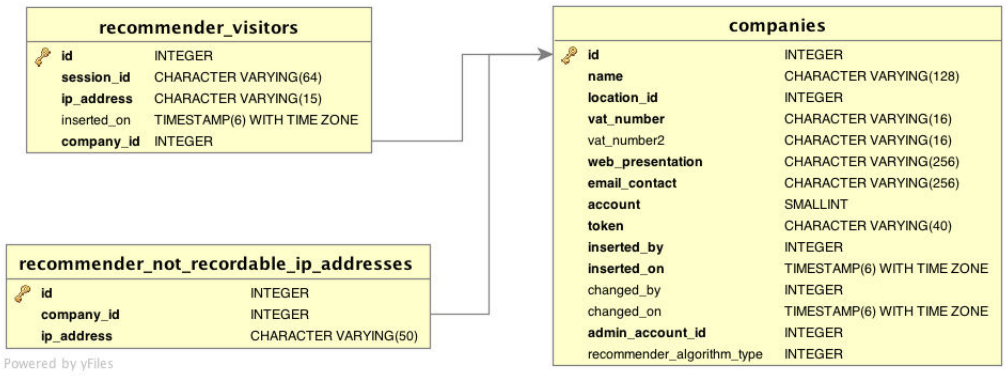
4.6 Sběr dat

Analýzou algoritmů v kapitole 3 byly definovány požadavky na sběr informací od uživatele doporučovacího systému. Důraz by měl být kladen na zaznamenání maxima implicitních informací o chování návštěvníka, zejména IP adresy, času tráveného prohlížením jednotlivých nemovitostí, parametrů hledání atd. Jednotlivým webovým prezentacím realitních kanceláří by dále mělo být umožněno předávat a zpracovávat libovolné parametry doporučovacího systému. Korektní funkčnost sběru uživatelských interakcí bohužel není zajištěna implicitně. Je proto nutné dodržovat určité konvence komunikačního protoklu (viz A), zejména je třeba klást zvýšený důraz na identifikaci uživatele, kdy by mohlo docházet k vytváření duplicit.

4.6.1 Identifikace uživatele

RePort RF pro identifikaci uživatelů využívá kombinaci IP adresy a session ID generované webovým serverem. Doporučuje se tuto session ID při první návštěvě uložit na straně uživatele do cookies a předat ji zpět při pozdějším návratu zpět. Webový server většinou udržuje otevřenou session pouze po dobu několika minut od ukončení poslední komunikace. Nedodržování postupu by vedlo k nárůstu počtu uživatelů a řídkosti hodnotící matice. Je důležité mít na paměti, že na správnou funkčnost doporučovacího systému nemá vliv pouze odladění algoritmu, ale i dodržování doporučení při návrhu prezentace realitní kanceláře. Výše uvedné hodnoty, tedy IP adresy, session ID a časové ra-

zítko záznamu se ukládají do tabulky `recommender_visitors`, která je mapována na Java bean `RecommenderVisitor`. O manipulaci se záznamy se stará `RecommenderVisitorService` a `RecommenderVisitorDAO`.



Obrázek 4.4: Tabulky recommender_visitors, recommender_not_recordable_-ip_addresses

Jelikož se makléři chovají diametrálně odlišně od uživatele, na kterého realitní kanceláře cílí, je potřeba je vyloučit ze sběru dat. Mnoho realitních kanceláří má pevnou IP adresu z důvodu bezpečnosti omezení přihlášení do informačního systému mimo kancelář. Rozsah IP adres byl použit i jako dostatečná identifikace uživatelů přicházejících z realitních kanceláří a data z těchto zdrojů nejsou zaznamenávána. K tomuto účelu slouží tabulka `recommender_not_recordable_ip_addresses` s referencí `company_id` na tabulku společností a `ip_address` IP adresou.

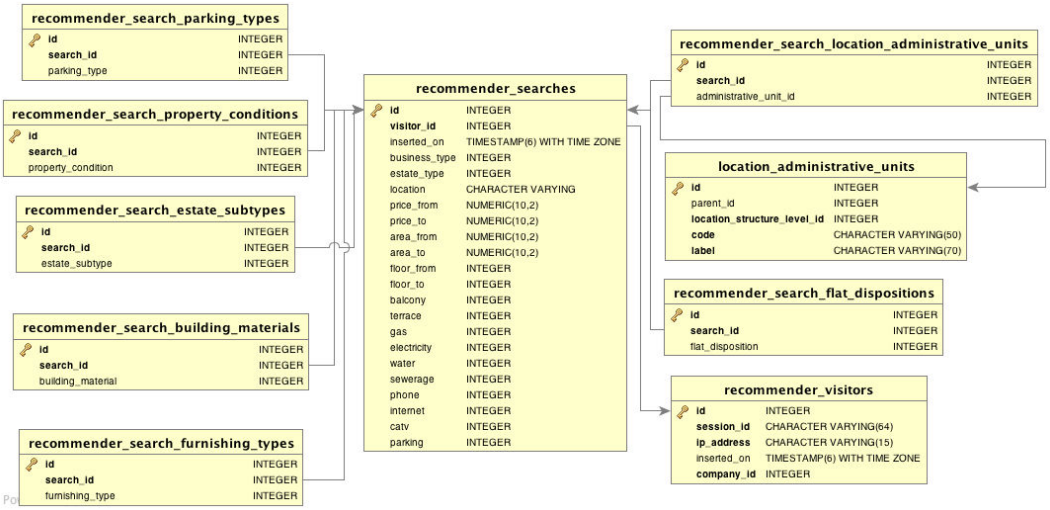
4.6.2 Kontext vyhledávání

RePort RF počítá s možností nasazení doporučovacích systémů zohledňujících kontext uživatele (viz. kapitola 3.7). Pokud budeme uvažovat kontexty v rámci realitní kanceláře, lze uvažovat jako jeden z nich vyplnění vyhledávacího formuláře. Uživatel se může rozhodovat mezi několika typy nemovitostí - např. pronájem bytu či koupě bytu na hypotéku (poměrně častý případ). Pak by bylo vhodné, aby programátor píšící algoritmus doporučovacího systém měl přístup k datům definujicím specifický kontext vyhledávání. *RePort RF* ukládá pro každé uskutečněné filtrování nabídky nemovitostí vyplněné hodnoty vyhledávacího formuláře do tabulky `recommender_searches` (mapována na `RecommenderSearch` Java bean). Některé parametry vyhledávání, převážně výčtové typy a celočíselné hodnoty, mají kardinalitu 1:N. Konkrétně se jedná o

- `RecommenderSearchEstateSubtype` - kategorické zatřízení nemovitostí (poddruhy), např.: chata, rodinný dům
- `RecommenderSearchBuildingMaterial` - konstrukční materiál budov
- `RecommenderSearchFlatDisposition` - dispoziční zatřízení bytu, např.: garsonka, 2+kk

- `RecommenderSearchFurnishingType` - dostupné vybavení bytu, např.: nezařízený, částečně zařízený
- `RecommenderSearchParkingType` - možnosti parkování u objektu
- `RecommenderSearchPropertyCondition` - technický stav nemovitosti
- `RecommenderSearchAdministrativeUnit` - lokalita nemovitosti. *RePort* uchovává databázi místopisných jednotek ve stromové struktuře. V případě České republiky je granularita nastavena na úroveň katastrálních území, nicméně pro každou zemi lze zadefinovat vlastní členění. Filtrování dle administrativní jednotky (zde uloženo jako reference na jednotku v místopisném číselníku) pak probíhá nad daty nacházejícími se v sjednocení podstromů.

Je třeba brát v potaz, že míra zaplnění výše uvedených parametrů závisí na poskytnutých detailech vyhledávání uživatelem.

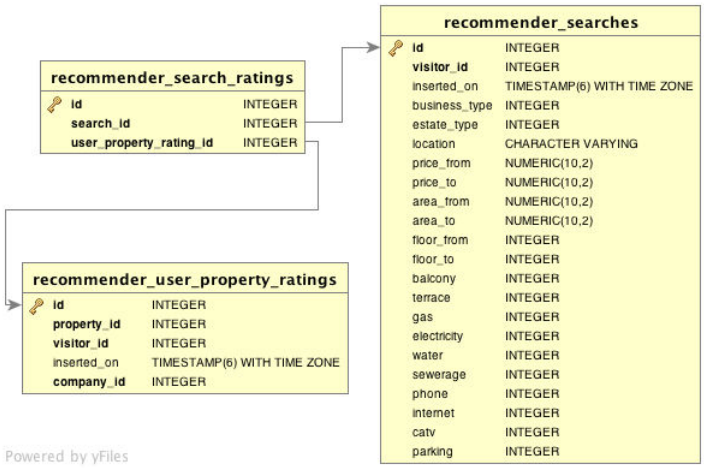


Obrázek 4.5: Tabulka recommender_searches a přidružené

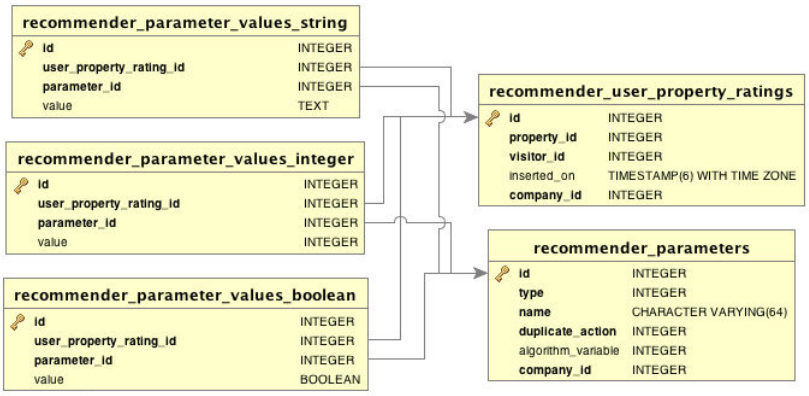
RePort RF jde dokonce ve sledování kontextu odvíjejících se od vyhledávání dále. Zaznamenává veškerý pohyb návštěvníka, vztahující se ke konkrétní konfiguraci vyhledávacího filtru. Programátor doporučovacího algoritmu má k dispozici i časovou posloupnost zhlédnutých nemovitostí. Tabulka `recommender_search_ratings` (viz obrázek 4.6) mapuje závislosti mezi provedenými vyhledáváními (nastaveným kontextem) a konkrétním návštěvníkem prezentace.

4.6.3 Definovatelné proměnné

Aby měl programátor doporučovacího systému k dispozici potřebnou volnost v návrhu algoritmu, nabízí mu *RePort RF* úložiště několika datových typů, ze kterých si může pro své potřeby zadefinovat libovolný počet proměnných. *RePort RF* disponuje v základu třemi datovými typy - *boolean*, *integer*, *string*, ale díky návrhu systému proměnných není problém IS rozšířit o další datové typy. Definice proměnných se opět vztahuje ke konkrétní realitní kanceláři, ne k algoritmu doporučovacího systému. Díky tomu mohou různé algoritmy sdílet data.



Obrázek 4.6: Tabulka recommender_search_ratings a přidružené



Obrázek 4.7: Tabulky definovaných proměnných doporučovacího systému

Klíčovou tabulkou v definici proměnných je `recommender_parameters`, která se mapuje na POJO `RatingParameter`. Každá proměnná má přiřazeno unikátní jméno (`name`) v rámci realitní kanceláře (reference `company_id`). Toto jméno slouží k přiřazení dat předaných přes *Web Services*.

Sloupec `type` určuje datový typ proměnné pomocí mapování výčtového typu textttRatingParameterType na *integer*. Díky tomu nebude v budoucnu problém doplnit další datové typy. Případně se dá uvažovat o konverzích datových typů do existujících - datum a čas se konverzí na *unix timestamp* dá uložit do datového typu *integer*, složitější struktury lze uložit do datového typu *string* ve formátu JSON či XML.

Během sběru dat může nastat situace, kdy uživatel jednu nemovitost ohodnotí vícekrát. Sloupec `duplicate_action` stanoví akci, která v takovéto situaci nastane. Definována je výčtovým typem `RatingParameterDuplicateAction` a nabývá hodnot `add` - sečíst hodnotu (pro `boolean` bitový OR, text se řetězí) či `replace` - za hodnotu proměnné se dosadí poslední příchozí hodnota.

Sloupec `algorithm_variable` ukládá mapování na konkrétní proměnnou algoritmu, vybranou z výčtového typu `UserItemRatingType`. Může nabývat hod-

noty `NULL`, kdy se jedná o volnou proměnnou, která se dále nezpracovává. V opačném případě přiřazujeme definici proměnné z hlediska *RePort RF* sémantickou hodnotu.

Data proměnných *RePort RF* se ukládaji do tabulek s prefixem `recommender_parameter_values_` a příponou konkrétního datového typu. Jak si ukážeme v kapitole 5, implementované datové typy bohatě stačí pro potřeby vzorového algoritmu doporučovacího systému. V případě rozšíření rozsahu se doporučuje čtenáři prostudovat balík `service.recommender.rating`, kde nalezne veškeré služby starající se o definici a ukládání proměnných.

4.6.4 Implicitní a explicitní interakce

RePort RF automaticky ukládá některé implicitní interakce. V kapitole 4.6.2 jsme zjistili, že se jedná o nastavení vyhledávacího formuláře nemovitostí a dále o zobrazení konkrétních nemovitostí. Tyto interakce lze vydedukovat z požadavků provedených webovou prezentací realitní kanceláře a není nutné provádět explicitní volání pro sdělení skutečnosti. Jak jsme se dozvěděli v kapitole 3.1, měly by nás zajímat i akce provedené při prohlížení konkrétní nemovitosti či čas strávený prohlížením nabídky. Pro tyto implicitní interakce už nutně potřebujeme součinnost ze strany prezentace realitní kanceláře. Stejným způsobem je třeba přistupovat i k explicitním interakcím. Za pomocí volání funkce `SubmitRecommenderParameter` *Web Services* rozhraní předáme *RePort RF* hodnoty proměnných zadefinovaných v kapitole 4.6.3.

Příklad explicitní interakce - ohodnocení nemovitosti

Typickou explicitní interakcí je tzv. palec nahoru či dolů, tedy jednorázové vyjádření pozitivního či negativního ohodnocení nemovitosti. Pro tento účel si zadefinujeme instanci třídy `RatingParameter` s parametry datový typ `type` nastavený jako *boolean*, jméno `name` *boolean_rating*, akce při duplicitním zadání `duplicateAction` *REPLACE*, přiřadíme sémantické hodnotě algoritmu `userItemRatingType` výčtovou hodnotu *PROPERTY_RATING_BOOLEAN* (pokud již ve výčtovém typu `UserItemRatingType` neexistuje) a na konec přiřadíme proměnné `company` referenci na instanci existující realitní kanceláře, u které chceme interakci zaznamenávat. Instanci třídy `RatingParameter` uložíme (alternativně pouze přidáme řádek do tabulky `recommender_parameters`) a od této chvíle *RePort RF* ukládá hodnoty interakcí uživatelů předaných skrz volání *Web Service* funkce `GetRecommenderParameterValue`. V případě návratu k hodnocení lze uživateli zobrazit předchozí hodnotu ohodnocení pomocí *Web Service* funkce `GetRecommenderParameterValue`.

Příklad implicitní interakce - délka prohlížení detailu nemovitosti

Měření času na webové prezentaci prohlížením konkrétní nabídky nemovitosti je jednou z typických implicitních interakcí. Z hlediska návrhu doporučovacího algoritmu nám čas strávený čtením informací o nemovitosti dává cennou informaci o míře zájmu návštěvníka. *RePort RF* bohužel nemá jinou možnost než spoléhat na sběr dat a dodržování konvencí programátorem realitní prezentace.

Moderní prohlížeče umožňují za pomoci JavaScriptu sledovat dobu strávenou při prohlížení webové stranky. Na pozadí prohlížení nabídky běží počítadlo, které se zastaví při akci události `window.onblur`, pokud okno či záložka prohlížeče ztratí focus. Při navrácení focusu (událost `window.onfocus`) se počítadlo opět rozběhne a začne přičítat čas. Je několik variant, jak se vypořádat s odesláním konečného součtu. Buď prohlížeč reaguje na událost `window.onbeforeunload`, která nastane těsně před uzavřením záložky či celého prohlížeče. AJAX volání provede s finální hodnotou, případně průběžně ukládá mezivýsledky do *cookies* či *HTML5 Web Storage* a výsledky odesílá v určitých časových intervalech či při otevření nové stránky.

Postup definice proměnné je obdobný jako v případě příkladu explicitní interakce. Pro úplnost se jedná o následující parametry - datový typ `type` nastavený jako *integer* (počet sekund), jméno `name` *boolean_rating*, akce při duplicitním zadání `duplicateAction` *ADD* (doby opakovaných návštěv se sčítají), sémantická hodnota algoritmu `userItemRatingType` *PROPERTY_DISPLAY_TIME* (je součástí `UserItemRatingType`).

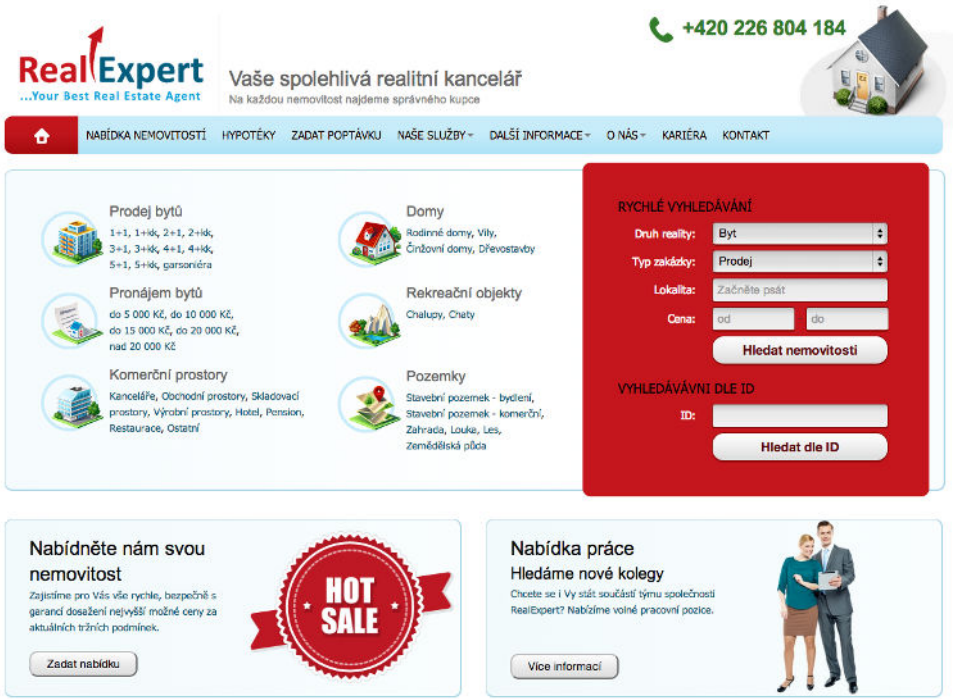
5. Nasazení v praxi

Kapitola se zaměřuje na reálné nasazení informačního systému *RePort* v režimu datové proxy pro konkrétní realitní kancelář. Všechny nové funkce *RePortu*, včetně frameworku pro doporučovací algoritmy *RePort RF*, jsou nasazeny při vývoji nově vznikající webové prezentace společnosti. Na několika následujících stránkách je popsán vznik vzorového doporučovacího algoritmu. Vývoj začíná analýzou požadavků, definicí rozsahu uživatelských interakcí, sběrem dat a jejich analýzou. Nabyté vědomosti jsou přetaveny v konkrétní implementaci doporučovacího systému. V závěru kapitoly jsou zevrubně popsány komponenty nově vzniklého systému.

5.1 Představení prezentace

Do této chvíle práce rozebírala dvě oblasti - doporučovacími systémy v obecné rovině a informační systém *RePort*, případně rozšíření *RePort RF*. Jelikož se shromáždil dostatek poznatků ohledně doporučovacích systémů, přichází chvíle, kdy se nad vybudovanými základy vytvoří vzorový algoritmus. *RePort RF* k tomuto cíli vyžaduje napojení na prezentaci realitní kanceláře, která zatím nebyla zmíněna.

Ve spolupráci se společností *RealExpert s.r.o.* byla v roce 2013 vyvinuta nová webová prezentace nad systémem *RePort*. Tato prezentace slouží k pilotnímu nasazení informačního systému *RePort*, včetně *RePort RF*. Společnost *RealEx-*



Obrázek 5.1: Výřez domovské stránky vzorové prezentace společnosti RealExpert s.r.o.

pert s.r.o. operovala v roce 2013 v jeden okamžik v průměru se zhruba 400 - 500 aktivními nemovitostmi, o které se staralo 20 - 25 makléřů. Svou velikostí se na českém trhu řadí mezi středně velké realitní kanceláře. Jako řada společností i tato získává většinu svých zákazníků z online a tištěné inzerce. Převážná část online zájemců pochází z tzv. *agregátorů* nemovitostí, které nabízí širokou nabídku inzerátů. Oproti specializované prezentaci realitní kanceláře postrádají podrobnější informace o způsobu obchodování dané společnosti. Nejnavštěvovanější agregátor s nemovitostmi v ČR - *Sreality* zabraňuje odchodu návštěvníka na jiný server tím, že neposkytuje u inzerátu nemovitosti jiné informace než emailový a telefonický kontakt na makléře. Takové chování přispívá k značnému rozdílu v návštěvnosti webové prezentace realitních kanceláří a agregátorů[1]. Je žadoucí připomenout, že návštěvníků je omezené množství a počet realitních kanceláří je po saturaci trhu na sestupu.

Prezentace společnosti *RealExpert s.r.o.* je technicky řešena velmi obdobně, jako je tomu u rozhraní pro makléře *RePortu*. Aplikace je psaná v *Javě*, využívá *Spring framework*, zejména je třeba upozrnit na *Spring MVC* pro logické dělení prezentace na model, view, controller a modul *Spring WS* komunikaci přes *Web Services* s rozhraním *RePort RF*. Díky tomu, že jsou všechna data předávána pomocí *Web Services*, neobsahuje prezentace žádné datové úložiště.

5.2 Požadavky na algoritmus

Definujme si nyní požadavky nově implementovaného algoritmu - *RePort Recommender Algorithm* (dále jen *RePort RA*). Již dříve, v kapitole 2.4, se určily některé základní vlastnosti algoritmu:

Výběr algoritmu. Kapitola 3 zevrubně analyzovala základní kategorie doporučovacích systémů. Z referenčních materiálů vyplývá, že vhodným kandidátem bude metodika kolaborativního filtrování. Jelikož dokážeme shora odhadnout počet nemovitostí (< 1000 nemovitostí) a můžeme zanedbat škálování nad rámec jednoho průměrného stroje, vhodným omezením počtu uživatelů dokážeme udržet data potřebná k běhu algoritmu v paměti.

Malá náročnost na explicitní interakce. V kapitole 1.2 je popsáno demografické složení návštěvníků a jsou zmíněny důvody, proč by uživatel měl nárážet na co nejméně explicitních interakcí. Algoritmus by měl v krajním případě fungovat bez nutnosti explicitní uživatelské interakce.

Implementace kompletního algoritmu. Autor by se z vlastní zvědavosti rád vyvaroval použití knihoven s předpřipravenými algoritmy.

Rozumná náročnost. Algoritmus by měl fungovat s rozumnou odezvou pro předpokládané nasazení. Není nutné klást důraz na škálování, protože historický vývoj návštěvnosti nepředpokládá dramatický nárůst v návštěvnosti. Pro účely škálování by se využil např. *Apache Mahout* (2.5.4).

[1] Analýzou dat na `www.toplist.cz` byl zjištěn rozdíl v navštěvnosti jednoho až dvou řádů mezi agregátory a prezentacemi realitních kanceláří v neprospěch prezentací konkrétních realitních kanceláří

Iterativní přístup. Jelikož nemá autor práce předchozí zkušenosti s návrhem doporučovacích systémů, rozhodl se postupovat inkrementálně. Tedy vycházet z algoritmu dříve popsaném v kapitole 3, implementovat jej, analyzovat a navrhovat zlepšení. Cílem bude algoritmus vhodný pro ostré nasazení. Součástí problematiky proto budou i zastavení u zajímavých technických detailů, které by čtenáři neměly uniknout. Předpokládá se zejména lazení výkonu algoritmu.

Z výše uvedených požadavků se jako vhodným kandidátem, ze kterých by měl *RePort RA* vycházet, jeví *memory-based collaborative filtering*, konkrétně varianta *item-based recommendation*. Z analýzy víme, že tato varianta je v případě převahy počtu uživatelů nad položkami vhodnější. Výsledky doporučení by měly v závislosti na posbíraných datech poskytovat slušné, místy i překvapivé výsledky. *Memory-based* varianta je jednodušší na implementaci a snadněji vyhoví studijním účelům. Metodika podává při relativní jednoduchosti velmi slušné výsledky. Bohužel již před samotnou implementací je třeba se připravit na případné komplikace, z předchozích kapitol pak vyplývá zejména následující:

Problém studeného startu. Jelikož je plánované vycházet z metodiky kolaborativního filtrování, algoritmus bude trpět nedostatkem dat pro poskytnutí ohodnocení u nově přidané nemovitosti či nově příchozího návštěvníka.

Prostorová omezení dat. Postupem času budou data doporučovacího systému narůstat. Algoritmus musí počítat s variantou odstranění nepotřebných dat tak, aby nedošlo k přeplnění paměti maticí doporučovacího systému, případně jinými daty.

Výkonnost algoritmu v zátěži. S rostoucím objemem zpracovaných dat se dá předpokládat, že poklesne výkonnost algoritmu a naopak naroste odezva.

5.3 Sběr dat

Vzorová prezentace je nakonfigurována tak, aby sbírala a předávala *RePort RF* několik informací o návštěvníkovi. Algoritmus bude záviset jak na implicitních, tak i na explicitních interakcích. Vzhledem k požadavku na nevtíravost bude vzorová prezentace fungovat pouze z implicitních interakcí. Eventuelní explicitní interakce (která je v tomto případě pouze jedna - ohodnocení na škále 1 až 5 hvězdiček) doporučení pouze zpřesní. V rámci *RePort RF* jsou zadefinovány následující proměnné:

TimeSpentOnProperty - ukládá celkový čas strávený návštěvníkem u konkrétní karty detailu nemovitosti. Odpovídá hodnotě `PROPERTY_DISPLAY_TIME(0)` výčtového typu `UserItemRatingType`, je celočíselného typu a při duplicitním výskytu se hodnota sčítá. Při sběru dat se na straně prohlížeče využívá JavaScript volání `window.on-focus`, `window.onblur`, `window.onbeforeunload` a `window.onmousemove` k výpočtu stráveného času u aktivní záložky či okna prohlížeče. V ukázkové prezentaci existuje v tabulce `recommender_parameters` záznam s ID 2.

TimesPropertyOpened - ukládá počet zobrazení konkrétní karty detailu nemovitosti návštěvníkem. Odpovídá hodnotě `PROPERTY_DETAIL_VIEWED_TIMES(1)` výčtového typu `UserItemRatingType`, je celočíselného typu a při duplicitním výskytu se hodnota sčítá. V ukázkové prezentaci existuje v tabulce `recommender_parameters` záznam s ID 3.

Rating1to5 - ukládá explicitní ohodnocení uživatelem na škále 1 - 5 hvězdiček (umístěno pod stručným přehledem, jak je vidět na obrázku 5.2). Jedná se o jedinou explicitní interakcí uživatele (požadavek malé náročnosti na explicitní interakce z kapitoly 5.2). Odpovídá hodnotě `PROPERTY_RATING_ONE_TO_FIVE(2)` výčtového typu `UserItemRatingType`, je celočíselného typu a při duplicitním výskytu se hodnota nahradí. V ukázkové prezentaci existuje v tabulce `recommender_parameters` záznam s ID 4.

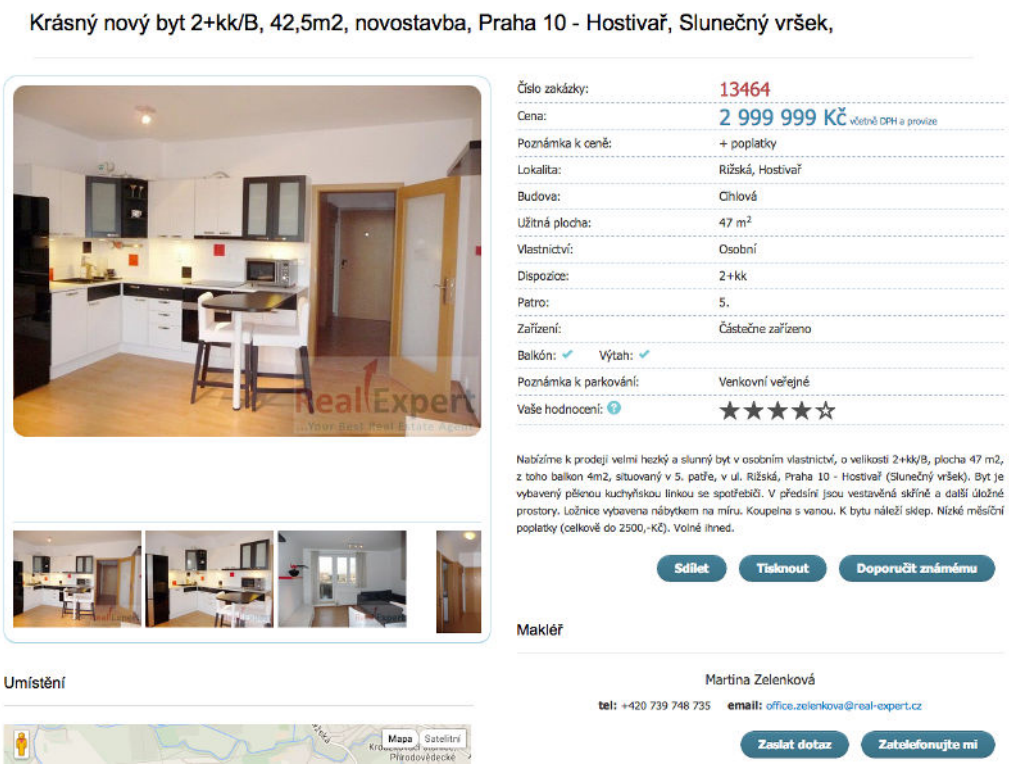
QuestionSubmitted - návštěvník prezentace se může skrz formulář zeptat na podrobnosti o nemovitosti. Tato akce vyjadřuje určitou míru zájmu o nabídku a jedná se o implicitní interakci. V rámcí *RePort RF* odpovídá hodnotě `PROPERTY_QUESTION_SENT(3)` výčtového typu `UserItemRatingType`, je typu boolean a při duplicitním výskytu se hodnota nahradí. V ukázkové prezentaci existuje v tabulce `recommender_parameters` záznam s ID 5.

CallRequested - podobně jako v předchozím bodě, kde může návštěvník u zaslat makléři dotaz, lze požádat o zpětný telefonický kontakt. Opět se jedná o implictní interakci vyjadřující míru zájmu o nemovitost. V rámci *RePort RF* odpovídá hodnotě `PROPERTY_CALL_REQUEST_SENT(4)` výčtového typu `UserItemRatingType`, je celočíselného typu a při duplicitním výskytu se hodnota nahradí. V ukázkové prezentaci existuje v tabulce `recommender_parameters` záznam s ID 6.

TimesShownInSearchResults - počet zobrazení nemovitosti ve výsledcích vyhledávání. V rámci *RePort RF* odpovídá hodnotě `PROPERTY_SHOWN_IN_SEARCH_RESULTS_TIMES(5)` výčtového typu `UserItemRatingType`, je celočíselného typu a při duplicitním výskytu se přičte jednička. V ukázkové prezentaci je definován v tabulce `recommender_parameters` pod ID 7.

Kromě právě nadefinovaných parametrů se u ukázkové implementace sbírají o návštěvnících i následující implicitní interakce:

- IP adresa návštěvníka
- Identifikátor session
- Nastavení vyhledávacího formuláře, z jehož výsledků se uživatel dostal na stránku detailu nemovitosti (pokud je k dispozici)
- Identifikátor nemovitosti, ze které se uživatel dostal na stránku detailu nemovitosti pomocí odkazů na doporučené nemovitosti (pokud je k dispozici)



Obrázek 5.2: Detail nemovitosti u vzorové prezentace.

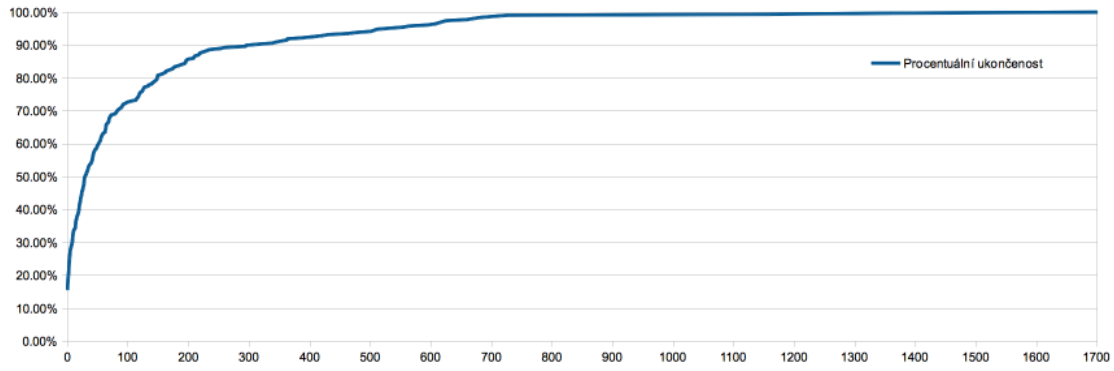
5.4 Analýza dat

Autor před implementací *RePort RA* nechal běžet informační system v ostrém provozu v módu sběru dat po dobu šesti týdnů. Implicitní a explicitní interakce zadefinované v kapitole 5.3 poskytly dostatek dat na to, aby měl konkrétní povědomí o hodnotách jednotlivých proměnných algoritmu a také odstranil některé chyby, které při vývoji vznikly. Jelikož *RePort* během psaní práce obsluhoval pouze jednu realitní kancelář, budou nyní prezentované SQL příkazy zjednodušené o podmínky, které omezují viditelnost dat na konkrétní nemovitost.

5.4.1 Platnost nabídky

V průběhu prodeje či pronájmu nemovitosti dochází ke změně stavu, kdy nemovitost může být zarezervována, realizována či zrušena. Analýzou uložených nemovitostí dostaneme zevrubnou představu o počtu dní, kdy setrvá nemovitost aktivní v nabídce. Je třeba mít na paměti, že datový model nepočítá s variantami, kdy se nemovitosti po nějaké době vracejí znovu do nabídky. Data byla získána následujícím dotazem:

```
SELECT changed_on::date - inserted_on::date AS days,
COUNT(*) AS count FROM offers
WHERE contract_status IN (2, 3, 4)
GROUP BY days ORDER BY days;
```



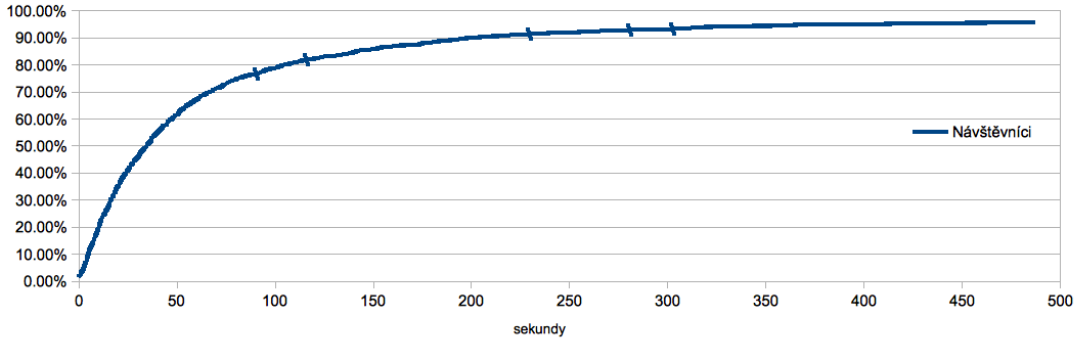
Obrázek 5.3: Délka platnosti nabídky nemovitosti.

Díky aditivnímu grafu procentuálního ukončení zakázek (obrázek 5.3) lze sledovat, že 80% všech zakázek je ukončeno do 150 dní od publikace. Z této hodnoty lze odvodit horní odhad pro ukládání dat pro uživatele. V kapitole 5.2 se zmiňuje obava z nabobtnání dat. Tomu se dá předejít prořezáváním matice doporučení, které algoritmus bude provádět. Z nasbíraných dat víme, že uživatel, který ohodnotil naposledy před 150ti dny, bude mít $\frac{4}{5}$ dat vztahujících se k nemovitostem odstraněným z nabídky.

5.4.2 Celkový čas strávený prohlížením nemovitosti

Čas strávený prohlížením nabídky nemovitosti je základní implicitní interakcí. K detailu nemovitosti se dá v případě vzorové prezentace dostat ze tří různých míst:

- Z úvodní stránky, kde se zobrazí nemovitosti fixně doporučené realitní kanceláří. Pozor, tato doporučení nemají nic společného s doporučovacím systémem. V případě, že realitní kancelář nevybere doporučené nemovitosti, jsou doplněny náhodně.
- Z výsledků vyhledávacího formuláře, které se dají řadit dle stáří nabídky, ceny či dle doporučovacího algoritmu.



Obrázek 5.4: Čas strávený prohlížením nemovitosti.

- Ze spodní částí detailu nemovitosti, kde jsou uvedeny doporučené nemovitosti.

Ve všech třech případech je nutná aktivita uživatele. Pak už záleží pouze na zájmu návštěvníků, jak dlouho stráví čas prohlížením nabídky. Pokud ořežeme posledních 5% návštěvníků, kteří např. nechali otevřený prohlížeč omylem např. po několik dní (i taková data se nasbírala), získáme opět poměrně pravidelnou křivku (obrázek 5.4). Je třeba zdůraznit, že tato interakce není dostupná od každého uživatele. Implementace u vzorové prezentace je totožná s implementací popsané v kapitole 4.6.4. Odlišnosti reprezentace JavaScriptu a nastavení různých prohlížečů způsobují nejednotnost v dostupnosti dat této interakce. Pro úplnost uvedeme SQL dotaz, ze kterého se data získala:

```
SELECT value, count(*)
FROM recommender_parameter_values_integer
WHERE parameter_id = 2
GROUP BY value ORDER BY value;
```

5.4.3 Počet návštěvníků explicitně hodnotících nemovitosti

Jedinou explicitní interakcí, kterou vzorová prezentace dovoluje, je ohodnocení na stupnici od jedné až pěti hvězd, kde jedna hvězda znamená nejnižší ohodnocení a pět hvězd nejvyšší ohodnocení. Realitní kancelář, se kterou se vývoj konzultoval, požadovala co nejmenší počet explicitních interakcí. Při vyhodnocování přijatých hodnot se zjistilo, že nároky na minimální zatížení návštěvníků byly oprávněné. Následující SQL dotaz zjistí procentuální podíl ohodnocených nemovitostí oproti celkovému počtu zobrazených detailů nemovitostí.

```
SELECT
  (SELECT COUNT(*) FROM recommender_parameter_values_integer
  WHERE parameter_id = 4 AND value IS NOT NULL) * 100
/
  (SELECT COUNT(*) FROM recommender_parameter_values_integer
  WHERE parameter_id = 3)
```

Pouze nepatrný zlomek zájemců si najde čas a ohodnotí některou z nemovitostí. Konkrétně se jedná o pouhých 0.1% zhlédnutých nemovitostí. Vzhledem k tomu, že se během období sběru dat neposbíral signifikantní vzorek, nelze analyzovat rozdělení jednotlivých ohodnocení od jedné do pěti hvězd. Proto se bude předpokládat normální rozdělení hodnot.

5.4.4 Počet navštívení totožné nemovitosti

Pokud analyzujeme počet opakujících se návštěv konkrétních nemovitostí, zjistíme, že v 93.08 % případů se návštěva nikdy neopakuje, ve 6.02 % případů se návštěvník vrátí jednou, 0.47 % dvakrát, 0.18 % třikrát. Zbývajících 0.25 % případů připadá na 4 a více návraty na totožnou nemovitost.

```
SELECT value, COUNT(*) AS count
  FROM recommender_parameter_values_integer
  WHERE parameter_id = 3 GROUP BY value ORDER BY value;
```

5.4.5 Počet odeslaných dotazů, počet požadavků na zavolání

Následující dotaz, podobně jako u počtu návštěvníků explicitně hodnotících nemovitost, určí procentuální využití funkce zaslání dotazu makléři, pokud se v prvním vnořeném SELECTu nastaví `parameter_id = 5`. Pro určení procentuálního využití požadavku na telefonický kontakt zájemci se nastaví `parameter_id = 6`.

```
SELECT
  (SELECT COUNT(*) FROM recommender_parameter_values_boolean
  WHERE parameter_id = 5 AND value = TRUE) * 100
/
  (SELECT COUNT(*) FROM recommender_parameter_values_integer
  WHERE parameter_id = 3)
```

Z výsledků dotazů vyplývá. že v obou případech využili návštěvníci uvedené funkce v méně než 0.02% případech.

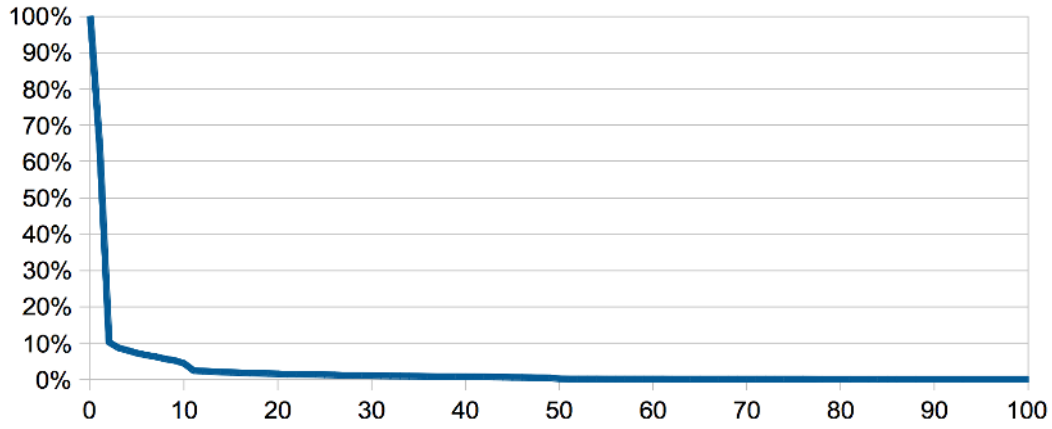
5.4.6 Počet zobrazných nemovitostí na jednoho návštěvníka

Kvůli vytvoření smysluplného uživatelského profilu potřebují doporučovací systémy zpravidla minimální počet položek, které uživatel ohodnotil. V případě *RePort RF* se jedná o návštěvnu minimálního počtu nemovitostí.

```
SELECT c AS pocet_nemovitosti,
COUNT(*) AS pocet_navstevniku
FROM (
  SELECT visitor_id AS v, count(*) AS c
  FROM recommender_user_property_ratings GROUP BY visitor_id
) AS rupr GROUP BY c ORDER BY c;
```

Z kardinality tabulky `recommender_visitors`, jež uchovává záznamy o všech návštěvnících, lze odvodit počet návštěvníků, kteří nezobrazili ani jednu nemovitost, či hodnotu využít pro výpočet procentuálního vyjádření. Graf na obrázku 5.5 zobrazuje procentuální vyjádření počtu uživatelů (osa y) na počtu zhlédnutých nemovitostí (osa x).

Z grafu vyplývá nelichotivý trend, kde drtivá většina návštěvníků zobrazí pouze omezený počet nemovitostí. K vytvoření minimálního profilu uživatele je potřeba určitý počet interakcí, tedy i zobrazených nemovitostí, od kterých se interakce odvíjí. Pokud by se doporučovací systém inspiroval minimálním počtem položek u *Trulia Suggests* (kapitola 2.5.1), pro 5 a více zhlédnutých nemovitostí připadá v úvahu pouhých 7,23% návštěvníků.



Obrázek 5.5: Počet zhlédnutých nemovitostí jedním uživatelem.

Dalším zajímavým zjištěním je několikanásobný nárůst počtu návštěvníků u počtu zobrazených nemovitostí 10 a 50 oproti sousedním hodnotám. Vzhledem k tomu, že při procházení seznamem nemovitostí (vyhledávání) je počet položek v seznamu nastaven na 10, lze přičíst tento výskyt webovým crawlerům. Načtení další desítky nemovitostí je třeba provést AJAXové volání, které nepodporují všichni roboti. Lokální extrém na hodnotě 50 autor přičítá jako omezení hloubky procházení některého z robotů.

5.5 Návrh doporučovacího algoritmu

Z předchozí kapitoly jasně vyplývá, že díky mizivému zastoupení explicitních interakcí, bude nutné během návrhu *RePort RA* vycházet převážně z implicitních interakcí uživatelů.

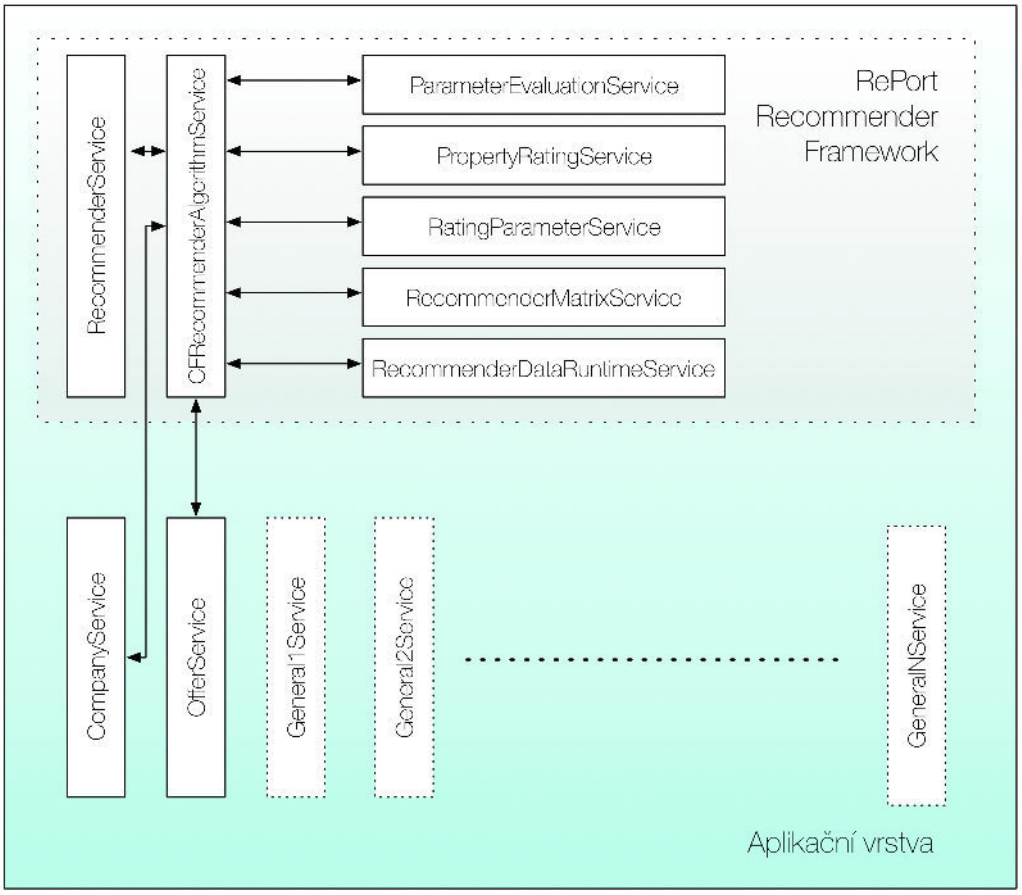
Autor se rozhodl vycházet z memory-based, item-based recommenderu díky doporučením v kapitole 3.5.1. Vzorová prezentace má několikanásobně více aktivních uživatelů než položek. Rozhodnutí dále vyplývá z dobrých uživatelských zkušenosti s doporučovacím systémem *Amazonu*, který má rovněž základ v *item-based CF*.

Algoritmus musí dále počítat s variantou nedostatečného uživatelského profilu, tedy tzv. studeného startu. Bohužel analýza dat v kapitole 5.4.6 ukazuje, že se tento problém týká přes 93% návštěvníků. K setřízení nemovitostí a doporučení podobných nemovitostí se využije kvantifikovaných atributů položek (výčtové a číselné typy). Z hlediska algoritmu nepůjde čistě o kolaborativního filtrování, ale o tzv. *content-boosted* hybridní algoritmus. Konkrétní vzorce výpočtů jsou uvedeny u jednotlivých komponent algoritmu.

5.5.1 Komponenty algoritmu

Věnujme se na chvíli diagramu komponentů na obrázku 5.6. Pro zjednodušení je v sekci *RePort RF* znázorněn právě popisovaný algoritmus (odbavovací služba všech doporučovacích algoritmů `RecommenderService` a hlavní služba algoritmu

`CFRecommenderAlgorithmService` jsou vyobrazeny vertikálně) a jeho komponenty (znázorněny horizontálně). Jednotlivé komponenty odpovídají přesně službám zdrojového kódu.



Obrázek 5.6: Diagram komponent vzorového algoritmu.

Hlavní služba algoritmu `CFRecommenderAlgorithmService` komunikuje s dvěma standardními službami `CompanyService` a `OfferService`, ze které získává informace o nabízených nemovitostech a realitních společnostech. Požadavky na provedení metod doporučovacího algoritmu jsou přeposlány z `RecommenderService`, která se chová jako směrovač (viz 4.3). Vzorový algoritmus využívá pět separátních komponent, z nichž byla komponenta `RecommenderDataRuntimeService` popsána dříve v kapitole 4.5.1. `RatingParameterService` je navíc součástí základní výbavy *RePort RF* a obstarává databázové operace definovatelných proměnných algoritmu (viz kapitoly 4.6.3).

ParameterEvaluationService

Služba normalizuje hodnoty proměnných jednotlivých interakcí do uzavřeného intervalu $< 0; 1 >$ dle následujících pravidel:

- Atributům typu **boolean**, tedy **CallRequested** a **QuestionSubmitted**, bude hodnota *true* převedena na 1 a *false* na 0.

- Atribut **Rating1to5** bude převeden následujícími hodnotami - *null* - 0, *1* - 0.2, *2* - 0.4, *3* - 0.6, *4* - 0.8, *5* - 1.
- U převodu atribut **TimeSpentOnProperty** do intervalu $< 0; 1 >$ se vycházelo z průběhu křivky grafu 5.4. Zjistíme, že se graf nápadně podobá grafu logaritmické funkce. Pro hodnotu 0 definujeme i projekci do intervalu $< 0; 1 >$ jako 0. Pro hodnty $>= 500$ definujeme skóre jako 1. Pro ostatní hodnoty, tedy pro interval $(0; 500)$, definujeme skóre rovnicí, které přibližně kopíruje křivku grafu 5.4.

$$s = \frac{log(x+1)}{log(500)} = log_{500}(x+1) \tag{5.1}$$

- Zbývající atribut **TimesPropertyOpened** je opět odhadnut z křivky analyzovaných dat a počítá se rovnicí

$$s = \frac{n}{n+1} \tag{5.2}$$

- Pokud není hodnota libovolné interakce k dispozici, normalizuje se na 0.

PropertyRatingService

Veškeré implicitní i explicitní interakce zadefinované pro danou nemovitost, reference na návštěvníka, nemovitost, realitní společnost i seznam jsou zapouzdřeny v objektu `UserPropertyRating`. Komponenta `PropertyRatingService` obstarává obvyklé manipulační metody nad zmíněným objektem, zejména inicializaci dle nakonfigurovaných proměnných, nastavení hodnot proměnných (manipulace za pomocí služby `RatingParameterService`), ukládání do databáze a jiné metody pro vyhledání ohodnocení nemovitostí.

RecommenderMatrixService

Právě představené komponenty poskytnou veškeré prostředky nezbytné pro vytvoření matice doporučení. Jakmile je k dispozici matice doporučení, můžeme provést samotný výpočet doporučení. Z hlediska komponenty doporučovací matice se bude zkoumat teoretické sestrojení matice a následně i technické detaily v rámci *RePort RF*.

RePort RA ke každé nemovitosti sbírá hodnoty hned několika proměnných, definovaných v kapitole 5.3, tedy pro každou položku a uživatele vytvoří vektor hodnot, nicméně algoritmy představené v kapitole 3 předpokládají skalární hodnotu. Komponenta `ParameterEvaluationService` normalizuje hodnoty proměnných do intervalu $< 0; 1 >$. Vhodně vybranou projekcí vektoru do skaláru získáme potřebné ohodnocení uživatel $\times$ položka. V případě *RePort RA* se jedná o přiřazení vah následujícím způsobem: **CallRequested** (35%), **QuestionSubmitted** (30%), **Rating1to5** (20%), **TimeSpentOnProperty** (10%), **TimesPropertyOpened** (5%) do intervalu $< 0; 1 >$. Zde navrhované váhy jsou pouze zkušební, vybrané dle autorova úsudku. V kapitole 6.3.1 se navrhne a porovná několik variant, ze které se do reálného provozu nasadí ta nejlepší.

Pro úplnost si připomeneme vzorec pro výpočet předpovědi ohodnocení uživatele u a položky i.

$$\bar{r}_{ui} = \frac{\sum_{j \in \mathcal{N}_u(i)} w_{ij} r_{uj}}{\sum_{j \in \mathcal{N}_u(i)} \mid w_{ij} \mid} \tag{5.3}$$

Z předpokládaného objemu dat je jasné, že pro optimální výkon algoritmu nelze provádět výpočty na úrovni databáze. Počet přístupů na disk a čas jimi strávený dává hodnoty daleko za hranicí systému s okamžitou odezvou. Proto je třeba matici uchovávat v paměti a výpočty provádět zde. Zároveň je nutné vyvinout mechanismus pro paralelní ukládání dat do databáze tak, aby reflektovala stav v paměti a v případě pádu systému nedošlo ke ztrátě dat. Bude potřeba zajistit i mechanismus sestavení matice v paměti z dat v databázi při startu systému.

Pro každou společnost v `RePortu`, s algoritmem využívajícím matici doporučení, je k dispozici instance třídy `VisitorOfferIdMatrix`. Všechny matice jsou uloženy v proměnné `Map<Company, VisitorOfferIdMatrix> matrices` singletonové instance komponentové služby `RecommenderMatrixService`. V *RePort RF* slouží třída `UserPropertyRating` jako implementační abstrakce ohodnocení nemovitosti. Zapouzdřuje hodnoty proměnných (odstavec 4.6.3), reference na realitní společnost, hodnocenou nemovitost a identifikaci návštěvníka. Dále třída obsahuje výchozí nastavení vyhledávacího filtru a časovou značku. Služba `RecommenderMatrixService` z uložených hodnot instancí `UserPropertyRating` v databázi sestaví v paměti datový model matice, který dále aktualizuje a užívá k provedení výpočtů doporučení. Každá matice je v paměti reprezentována následujícím způsobem:

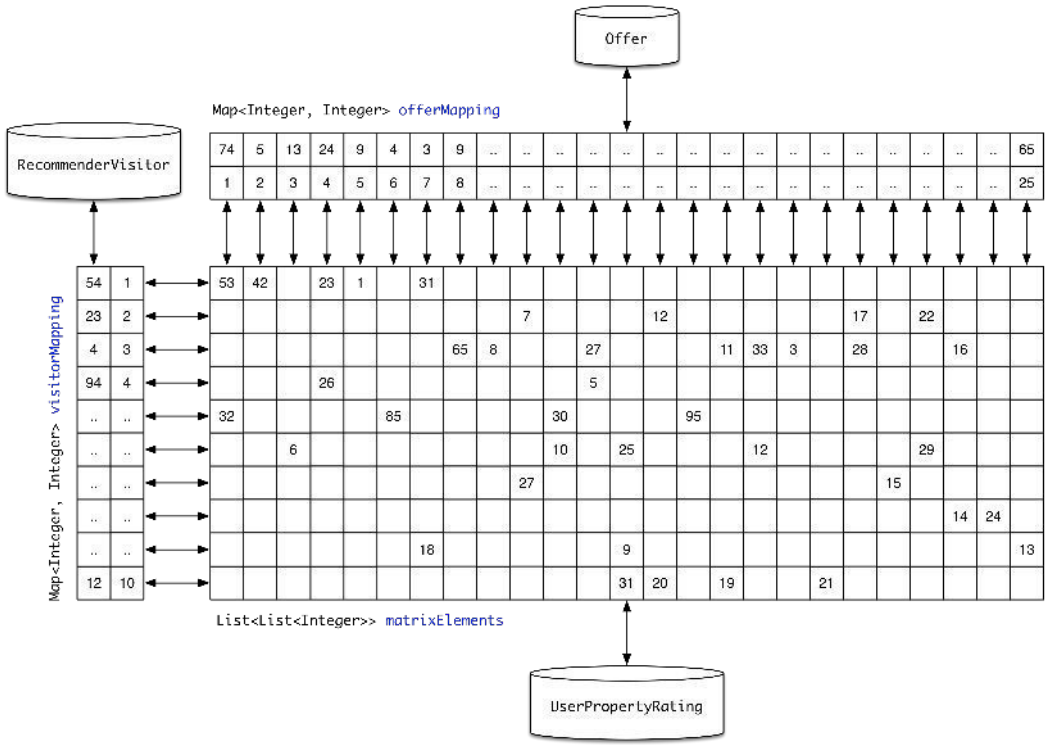
- `List<List<Integer» matrixElements` - Seznam seznamů ID uloženého objektu `UserPropertyRating` do databáze. Jelikož matice neobsahuje absolutně
- `Map<Integer, Integer> visitorMapping` - Mapuje ID uživatelů doporučovacího systému na souřadnice řádku matice doporučení.
- `Map<Integer, Integer> offerMapping` - Mapuje ID nemovitostí na souřadnice sloupce matice doporučení.

Komponenta `RecommenderMatrixService` dále obsahuje veškeré základní metody pro načítání a ukládání dat, kde transparentě provádí operace s databází a zároveň s maticí. Programátor algoritmu se proto nemusí starat o dvojí ukládání.

Zcela první implementace této komponenty počítala s načítáním úplných záznamů tří odkazovaných tabulek místo pouze referenčních ID, nicméně start systému, kdy se matice sestavuje do paměti, probíhal na testovacím stroji[2] zhruba dvacet minut, tedy nepřijatelně dlouho. Zdrojem dat byla kopie získaná z živého systému realitní kanceláře. Zjistilo se, že např. vrstva Hibernate má při načítání velkého počtu malých záznamů vysokou režii. Po úpravě načítání pouze referenčních ID systém v plném nasazení startuje desítky sekund, což je přijatelné.

Během analýzy dat v předchozí sekci této kapitoli jsme vyřkli požadavek na automatické prořezávání matice doporučení. Služba `RecommenderMatrixService`

[2]Testováno na *Intel Core* i5-750, 8MB cache, 2.66 GHz, 4 GB RAM, SSD disk.



Obrázek 5.7: Diagram nejdůležitějších datových struktur matice doporučení a jejich napojení na databázi.

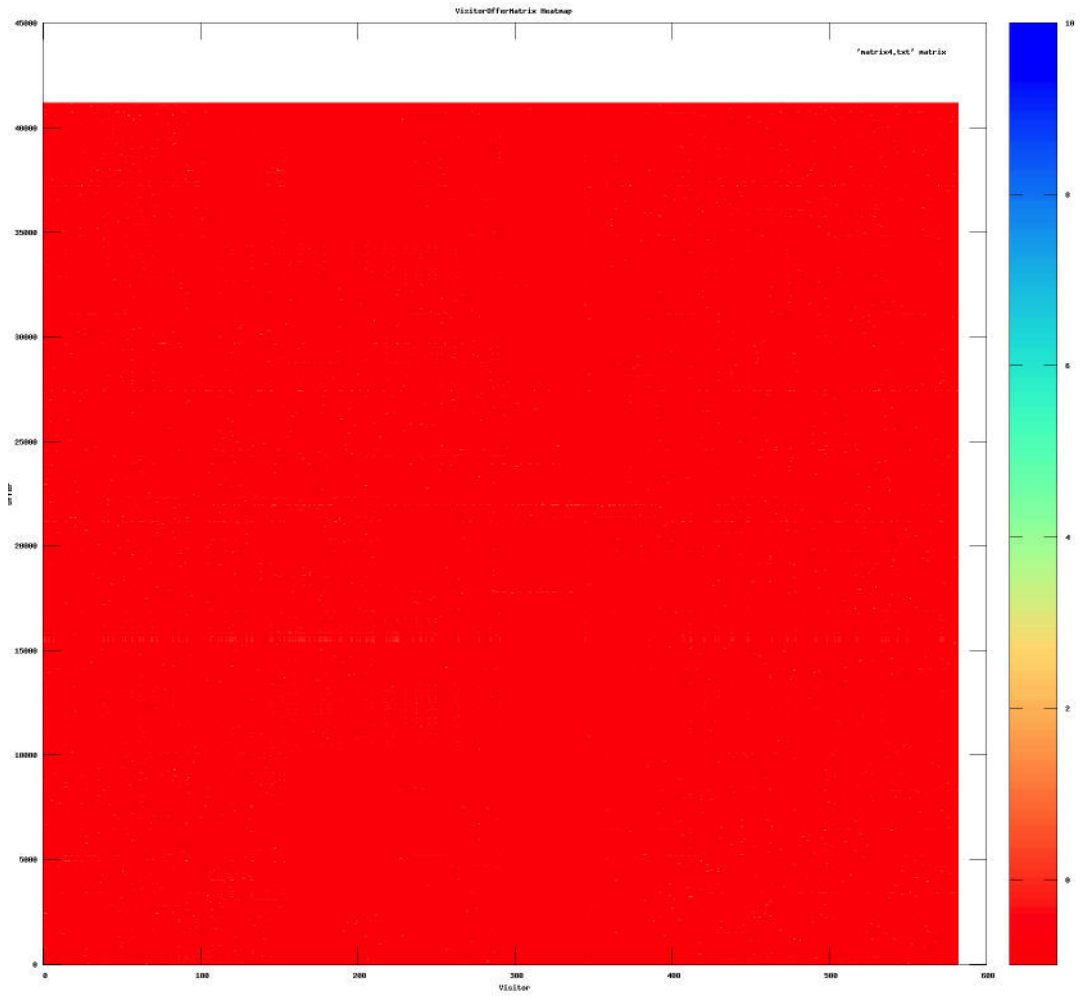
na pozadí periodicky odstraňuje ohodnocení starší 150 dní, která se již nepovažují za relevantní. Při plném běhu systému a po započetí prořezávání se dimenze matice zastavily na zhruba 50000×600. Na obrázku 5.9 vidíme vygenerovanou heatmapu matice doporučení. Ta je dle předpokladu tak řídká, že změny v červené barvě půjdou pravděpodobně vidět v digitální verzi pouze práce po přiblížení.

5.5.2 Doporučení nemovitostí

V kapitole 4.4 byly zadefinovány dvě klíčové metody, které by měl mít implementované každý doporučovací systému v rámci *RePort RF* . Jedná se o **seřazení nemovitostí** dle doporučovacího algoritmu a **nalezení N nejvhodnějších nemovitostí**. Prvně uvedená operace se volá při vyhledávání nemovitostí (WS metoda `getOfferList`). Druhá je součástí návratových dat zobrazení detailu nemovitosti (WS metoda `getOffer`), kdy se k navrácené položce připojí i nejvhodnější nemovitosti pro daného návštěvníka.

Funkčnost algoritmu si názorně ukážeme na nalezení N nejbližších příbuzných nemovitostí. Návratová hodnota detailu nabídky nemovitosti `GetOfferResponse` obsahuje mimo detailní popis nemovitosti i seznam doporučených nemovitostí. Tento seznam se naplní v případě, že je k dispozici doporučovací algoritmus a dostatečný počet ohodnocení k funkčnosti algoritmu. Vzorový algoritmus `CFRecommenderAlgorithmService` vychází primárně z kolaborativního filtrování.

Nejprve se při volání metody `OfferService.findRecommendedOffers` deleguje volání na hlavní službu *RePort RF* - na `RecommenderService.findTopN-RelatedOffers`. Ta dle konfigurace zjistí, která konkrétní implementace algo-



Obrázek 5.8: Heatmapa reálné matice doporučení.

ritmu je přiřazena k dané realitní kanceláři a postoupí požadavek dále. V případě vzorové implementace se zavolá metoda `CFRecommenderAlgorithmService.findTopNRelatedOffers`.

Následuje porovnání počtu ohodnocených nemovitostí uživatele s konstantou `MINIMUM_RATINGS_TO_USE_CF`. Pokud nemá uživatel dostatek záznamů v matici ohodnocení, tedy jedná se o problém studeného startu, není vhodné nabízet doporučení pomocí kolaborativního filtrování. Namísto něj se doporučené nemovitosti vyberou analýzou parametrů nemovitostí. K alternativní metodě doporučení se algoritmus přepne i v případě vypnutí doporučovacího systému jako takového. Pak se doporučení řídí následujícími pravidly:

- *Typ zakázky* (prodej / pronájem), *kategorie i subkategorie nemovitosti*[3] musí souhlasit.
- Získá se seznam X *nejbližších a levnějších* nemovitostí.
- Získá se seznam X *nejbližších a dražších* nemovitostí.

[3]Porovnávají se mezi sebou vždy např. pouze byty 2+kk

- Vzniklý seznam se náhodně promíchá a vybere se prvních X nemovitostí v seznamu.

Uvedená alternativní metoda má na reálných datech překvapivě dobré výsledky. Vzorová realitní kancelář má pro každou subkategorii nemovitostí řádově desítky položek. Ve výsledcích doporučení se díky GPS souřadnicím či číselníkům katastrálních území (pokud nejsou přesné souřadnice k dispozici) vyberou blízké a cenou odpovídající nemovitosti a často jsou návštěvníkovi nabídnuty velice uspokojivé výsledky.

V případě, že návštěvník splní veškeré omezující kritéria popsaná v předchozích odstavcích, vzorový algoritmus `CFRecommenderAlgorithmService` doporučí nemovitosti na základě algoritmu kolaborativního filtrování3.5 a nezávisle na aktuálně prohlížené nemovitosti (i když ta má do budoucnosti na výsledky doporučení také vliv, jelikož je novým ohodnocením). Ze sestavené matice doporučení se vypočítá doporučení N nejbližších nemovitostí od nemovitosti i pro uživatele u v několika krocích.

- Z matice doporučení se vyzdvihnou všechna ohodnocení nemovitostí uživatelem (`RecommenderMatrixService.getRatingsByVisitor`), které označíme $R \subseteq \{r_{ui} : U \times I \times S\}$.

- Připraví se seznam všech aktivních nemovitostí nabízených realitní kanceláří I, z něhož se bude pro každou nemovitost počítat míra doporučení. Povšimněte si faktu, že v tomto případě doporučení se získávají výsledky z celé nabídky nemovitostí, oproti konkrétní kategorii nemovitostí ve výše uvedené alternativní metodě.

- Pro každou nemovitost ze seznamu I se spočíta ohodnocení doporučení dle vzorce

$$\bar{r}_{ui} = \frac{\sum_{j \in \mathcal{N}_u(i)} w_{ij} r_{uj}}{\sum_{j \in \mathcal{N}_u(i)} \mid w_{ij} \mid} \tag{5.4}$$

  [2] a [24] uvádějí, že váhy u kolaborativních algoritmů by se neměly určovat analýzou atributů prvků, nýbrž výpočtem. Nabízí se např. kosinova míra představena v kapitole 3.4.3. V literatuře ([24]) se často zmiňuje i Pearsonova korelace $corr_{ij}$, která je použita v představeném algoritmu pro výpočet váhy w_{ij} mezi dvěma nemovitostmi.

$$w_{ij} = corr_{ij} = \frac{\sum_{u \in U_{ij}} (r_{ui} - \overline{r}_i)(r_{uj} - \overline{r}_j)}{\sqrt{\sum_{u \in U_{ij}} (r_{ui} - \overline{r}_i)^2} \sqrt{\sum_{u \in U_{ij}} (r_{uj} - \overline{r}_j)^2}} \tag{5.5}$$

- Výsledný seznam se setřídí sestupně dle hodnoty $\bar{r}_{ui}$ a prvních N nemovitostí se navrátí jako výsledek doporučení.

Seřazení nemovitostí dle doporučovacícho algoritmu z velké většiny kopíruje algoritmus nalezení **N nejbližších nemovitostí**. Zatímco během nalezení N nejbližších nemovitostí pro daného uživatele se hledalo ohodnocení r_{ui} v rámci nabídky celé realitní kanceláře I, při řazení nemovitostí se r_{ui} počítá pouze nad množinou řazených nemovitostí I_1. Řazení nemovitostí se v *RePort RF* využívá při vyhledávání nemovitostí. Seznam bývá zpravidla filtrovaný, proto platí $|I| \geq |I_1|$.

5.5.3 Optimalizace výpočtu, pomocné datové struktury

V případě, že budeme uvažovat zcela naivní algoritmus bez optimalizací, dotazující se na všechna ohodnocení databáze, který vypočítá doporučení dle vzorců 5.4 a 5.5 na datech sesbíraných vzorovou implementací, obdržíme výsledek za zhruba 15 sekund. Tento čas byl změřen na vývojářském počítači, který slouží jako referenční sestava. Několik následujících odstavců se bude zabývat redukcí výpočetního času na hodnotu přijatelnou u interaktivních webových prezentací, tedy pod jednu sekundu.

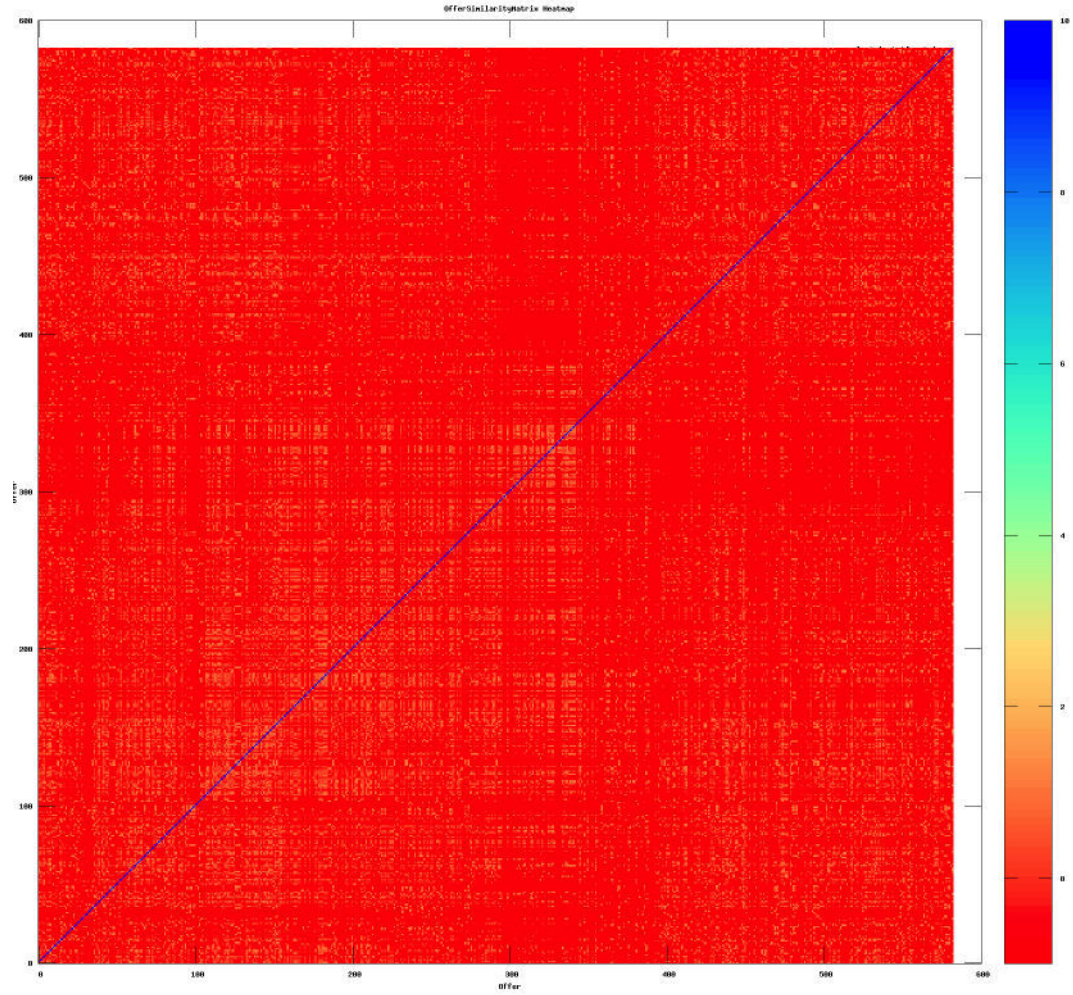
Logickým řešením problému by mohlo být přesunutí všech dat do operační paměti, jelikož drtivá část potřebného k doporučení se stráví načítáním hodnot z databáze. V průběhu vývoje vzorového algoritmu se ale velice rychle narazilo na omezení výpočetních prostředků. Řádově stovky nemovitostí a statitisíce až milióny[4] ohodnocení *návštěvník* × *nemovitost*, kdy se navíc v těchto řádech prováděly několikrát kartézské součiny, by mělo za důsledek nabobtnání datových struktur za hranice udržitelnosti. Připomeňme, že ohodnocení v našem případě je vektor několika definovaných implicitních i explicitních proměnných (viz kapitola 4.6.3), a ne skalární hodnota. Vektor je navíc v paměti reprezentován jako objekt. Pokud uvážíme, že *RePort RF* navíc během vývoje hostil pouze data jediné realitní kanceláře, je zřejmé, že v plánovaném nasazení by nenasytnost algoritmu byla neudržitelná.

Po omezení velikosti datových struktur, z velké části pomocí redukce načtení celých objektů (vektorů ohodnocení) do operační paměti na pouhé klíče záznamů, či omezením rozsahu automaticky načítaných dat v *Hibernate* (`fetch`), se dostavila mírná zlepšení v řádech sekund, která měla za následek posunutí celkového času doporučení do oblasti kolem 10 sekund. Autor dospěl při analýze vzorců 5.4 a 5.5 k názoru, že existují dvě cesty, jak čas výpočtu srazit do únosných mezí - provést částečný výpočet doporučení a materializovat ohodnocení. Poslední provedenou úpravou, která měla za následek zlepšení doporučení bylo vyloučení robotů.

Částečný výpočet doporučení Zaměřme se nyní na vzorec výpočtu doporučení 5.4. Počítá se z ohodnocení *návštěvník* × *nemovitost* r_{ui} a z vah dvou libovolných nemovitostí *nemovitost* × *nemoviost* w_{ij}. Matici doporučení *návštěvník* × *nemovitost*, jak již víme, nemůžeme díky velkým dimenzím držet v paměti, ale matici vah nemovitostí za určitých podmínek už ano. Navíc je tato matice symetrická dle diagonály, tedy $w_{ij} = w_{ji}$, proto stačí držet v paměti hodnoty např. nad diagonálou. Na obrázku 5.9 vidíme symetrickou heatmapu matice vah dvou nemovitostí získanou z reálných dat.

K operacemi s `OfferSimilarityMatrix`, což je reprezentace matice `Offer` × `Offer`, která ukládá váhu dvou nemovitostí dle vzorce Pearsenovy korelace 5.5. Při startu aplikace se matice sestaví dle ohodnocení uložených v databázi s omezením maximálního stáří. V průběhu času jsou přidávány a odebírány nemovitosti dle jejich stavu. Přibývají i nová ohodnocení nemovitostí, které se musí promítnout do výpočtu vah. Pro tyto účely se vytváří a shlukují dávky nemovitostí velikosti `CFOfferSimilarityMatrixService.SIMILARITY_CALCULATION_-`

[4]V tomto případě uvažujeme teoretický nárust návštěvnosti, který může nastat.



Obrázek 5.9: Heatmapa reálné matice podobnosti nemovitostí.

`BATCH_SIZE`[5], které jsou uznačeny jako *dirty* a ty jsou v pravidelných intervalech přepočítávany. Algoritmus nemá vždy k dispozici zcela aktuální hodnoty w_{ij}, pokud je fronta dávek zaplněna, nicméně při desetitisících ohodnocení se jedná pouze o mírnou odchylku od aktuálního stavu. Tuto daň vyvažuje rapidní zkrácení času pod 1 sekundu.

Materializace ohodnocení V kapitole 4.6.3 je popsáno, jak je reprezentováno atomické ohodnocení nemovitosti v databázi. K zjištění všech hodnot zadefinovaných proměnných je třeba spojení $N + 1$ tabulek, kde N je počet proměnných ohodnocení. Jelikož s každým dalším spojením dochází k narůstání času dotazu, jako vhodnou optimalizací se jeví materializace hodnoty ohodnocení dle `ParameterEvaluationService` 5.5.1, tedy projekce proměnných do $< 0, 1 >$. Tato hodnota je uložena v tabulce `recommender_user_property_rating_cf_evaluations` a při změně je vždy automaticky aktualizována. Materializaci ohodnocení doplnila i optimalizace kritických dotazů databáze, které obchází režii `Hibernate` a jsou psány v čistém SQL. Dotaz `SELECT` běží pouze nad jednou tabulkou

[5]Experimenty se ustálila jako vhodná praxe hodnota 200 nemovitostí, kdy dávka neběží příliš dlouho a zároveň nemá velkou režii

a sám o sobě trvá řádově desítky milisekund. V celém výpočtu se jedná o jediný dotaz do databáze. Zbytek dat je přítomen v operační paměti z dřívějška. Optimalizaci pomohla rovněž revize databázových indexů.

Vyloučení robotů Během optimalizací výpočtů, zkoumání matic ohodnocení, zejména pak heatmap, se objevily značné anomálie v ohodnoceních. Pro některé uživatele byly ohodnoceny téměř všechny nemovitosti, zpravidla velice obdobnou hodnotou. Krátké nahlédnutí do logu serveru potvrdilo tezi, že chování způsobují různí roboti.

Aby se do budoucna nesnižovala kvalita doporučení, byla pro tyto účely vytvořena služba `NotRecordableUserAgentService`, která z tabulky `recommender_not_recordable_user_agents` načítá regulární výrazy a zkouší je aplikovat na řetězec user agent prohlížeče. Pokud uspěje, pro daný prohlížeč se doporučovací algoritmus vypíná, podobně jak se tomu děje pro vybrané IP adresy (viz 4.6.1).

5.5.4 Využití ohodnocení pro zjištění oblíbenosti nové nemovitosti

RePort je navržený jako komplexní informační systém, který obsahuje i menší makléřské rozhraní, kde se dá kromě editace článků pro webové prezentace spravovat i nabídky nemovitostí realitní kanceláře. Do současné chvíle se provedená ohodnocení nemovitostí v informačním systému *RePort* využívala pouze zpětně pro účely návštěvníků. Nasbíraná data umožní nahlédnout makléři, do jaké míry je daná nemovitost žádaná, případně jaký má potenciál. Autor se domnívá, že by měly být nemovitosti totožného typu, stejné lokality a podobné cenové hladiny hodnoceny identicky. Bohužel tuto tezi nemá jak otestovat, jelikož makléřské rozhraní není na rozdíl od ostatních komponent *RePortu* v ostrém nasazení. Makléřské rozhraní neobsahuje ani žádné zaznamenávání odpovědi na ohodnocení, proto nebude součástí vyhodnocení v kapitole 6. Čtenář by měl funkci brát jako autorovu ideu, jak s daty netradičně nakládat.

Obrázek 5.10: Grafické zobrazení předpokládané oblíbenosti nemovitosti.

Vhodným vizuálním prvkem, který jednoduše znázorňuje potenciál nemovitosti, je widget z *Twitter Bootstrap* - kruhová stupnice od 0 do 10 na obrázku 5.10. K získání dat *RePort RF* volá metodu `estimateOfferRating(Offer offer)` doporučovacího systému příslušného dané realitní kanceláři. Aby se výpočet provedl, je nutné, aby byla nemovitost aspoň jednou úspěšně uložena v databázi. Výpočet potenciálu probíhá dle následujících kroků:

- Algoritmus vyhledá nemovitosti, které jsou svými parametry podobné nové nemovitosti. Stejný postup se používá v doporučovacím algoritmu před přepnutím na kolaborativní filtrování 5.5.2.

- Algoritmus dohledá vektory všech ohodnocení pro seznam podobných nemovitostí.
- Provede se projekce vektorů do $\mathbf{R}$, vypočítá se aritmetický průměr a výsledek se promítne do intervalu $< 0, 10 >$

Doporučení předpokládané oblíbenosti nabídky není dostupné v několika případech:

- Doporučující algoritmus není inicializovaný (např. probíhá výpočet matice podobnosti nemovitostí).
- K nové nemovitosti neexistuje v nabídce kanceláře podobná nemovitost.
- Doporučující systém neposbíral dostatek ohodnocení podobných nemovitostí.

Pak se makléři zobrazí chybová hláška tak, jak je vyobrazena na obrázku 5.11.

Obrázek 5.11: Nedostupná data oblíbenosti nemovitosti.

6. Vyhodnocení výsledků doporučovacího systémů

Zatím jsme zjistili, že neexistuje univerzální doporučovací systém, který by ideálně fungoval na libovolných datech a ve všech doménách, proto existuje několik doporučovacích metodik. Pokud máme hotový a implementovaný doporučovací systém, je před nás kladen problém s určením úspěšnosti takovéhoto systému. Následující kapitola se věnuje problematice vyhodnocení výsledků doporučovacích systému, vhodnému výběru metrik a jejich aplikaci na vytvořený *RePort RA*. Algoritmus se bude posuzovat z kvalitativního i výkonnostního hlediska. Součástí problematiky vyhodnocení je i ladění dílčích parametrů algoritmu. Jak záhy zjistíme, změny nastavení mohou mít zásadní následky.

6.1 Metodiky vyhodnocení

Existuje několik základních metodik vyhodnocení výsledků doporučovacích systému, lišících se dle přístupu a zdroje hodnotících datových množin. Výběr metodik vychází ze splnění či nesplnění několika klíčových atributů [1], např. zda lze provádět offline analýzu, či je potřeba přistupovat k živým datům. Případně zda měříme metriky na syntetických datech či datech vyprodukovaných návštěvníky. Použité metodiky se mohou lišit dle aktuálního scénáře. Je vhodné testovat výkonnost (časovou náročnost) algoritmu na syntetizovaných datech, nicméně pro určení kvality doporučení se doporučuje využít data posbíraná od uživatelů doporučovacího systému.

6.1.1 Offline vs. online analýza

Vyhodnocení algoritmů se provádí na množině dat, která se získá buď přímo z živého doporučovacího systému (*online*), nebo se staticky sestaví přímo pro účely vyhodnocení (*offline*). Jednodušší variantou z obou přístupů je *offline* analýza. Nad předem přiravenou množinou dat můžeme spouštět různé varianty algoritmů a vhodnou metrikou porovnávat jejich výsledky. Pokud kromě samotných ohodnocení máme i příslušné časové značky, můžeme přehrávat učení doporučovacího systému a sledovat jeho změny. Offline analýza je velmi efektivní metodou, nicméně její zásadní nevýhodou je omezenost datové množiny a s tím spojené problémy. Nelze ku příkladu porovnávat výsledky doporučení s rozhodnutím uživatele, pokud není součástí datové množiny.

Předchozí problém odbourává *online* analýza, která využívá dat sbíraných na živém systému. Principiálně funguje tak, že část uživatelů je přeřazena systémem na alternativní doporučovací algoritmus. Výsledky obou algoritmů pak porovnáváme. Zvýšená náročnost implementace dává provozovateli do rukou vyšší flexibilitu v podobě možnosti výběru profilů uživatelů u testovaného algoritmu.

6.1.2 Syntetická vs. organická data

Pokud se budeme zabývat offline analýzou dat, máme na výběr ze dvou zdrojů, jak data získat - buď si můžeme data vygenerovat, tedy použijeme *syntetická* data, či zaznamenáme reálné chování návštěvníků, tedy *organická* data. Obě skupiny mají své opodstatnění a využití, vycházející z nedokonalosti a nekonzistentnosti organických množin. Tím, jak se návštěvníci chovají nevyzpytatelně, je obtížné simulovat mezní chování algoritmu za určitých podmínek. Opačně mohou nastat problémy, pokud se algoritmus odladí pouze pro syntetická data určitých vlastností, a pak se spustí nad množinou organických dat.

6.2 Metriky vyhodnocení

Obecně se uznávají čtyři základní kategorie metrik [27], [1], které si nyní krátce představíme i s některými zástupci.

Metriky přesnosti predikce. Určují přesnost predikce mezi navrženou hodnotou doporučovacím systémem (proměnná p_i) a ohodnocením uživatele (proměnná r_i). Metriky přesnosti se zpravidla využívají u nebinárních ohodnocení, kde počet zkoumaných ohodnocení je N. Typickým zástupci jsou:

střední hodnota chyby (MAE)

$$MAE = \frac{\sum_{i=1}^{N} |p_i - r_i|}{N} \tag{6.1}$$

rozptyl chyby (MSE)

$$MSE = \frac{\sum_{i=1}^{N} (p_i - r_i)^2}{N} \tag{6.2}$$

střední kvadratická odchylka (RMSE)

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N} (p_i - r_i)^2}{N}} \tag{6.3}$$

Metriky přesnosti klasifikace. Klasifikační metriky se zaměřují na určení relevantnosti výběru položek. Tyto nerozlišují odstupňování vhodnosti doporučení, pouze striktně binárně rozlišují čtyři stavy: *relevantní a predikováno* (t_p), *relevantní a nepredikováno* (f_p), *irelevantní a predikováno* (f_n), *irelevantní a nepredikováno* (t_n). Stavy t_p a t_n jsou žádoucí, naopak stavům f_p a f_n se snažíme vyhnout. Ze součtu naměřených stavů jsou odvozeny základní metriky, které se využívají i v oblasti dobývání informací.

precision - míra pozitivní přesnosti se měří jako poměr korektně doporučených položek oproti součtu všech doporučených položek.

$$precision = \frac{t_p}{t_p + f_p} \tag{6.4}$$

recall - míra pravděpodobnosti, že relevantní položka je doporučena

$$recall = \frac{t_p}{t_p + f_n} \tag{6.5}$$

V obou uvedených případech se snažíme o maximalizaci hodnot. Existuje i metrika nazvaná F_1-*skóre*, která se počítá jako harmonický průměr obou uvedených.

$$F_1 = \frac{2}{\frac{1}{precision} + \frac{1}{recall}} = \frac{2 \cdot precision \cdot recall}{precision + recall} \tag{6.6}$$

Metriky přesnosti ranku. Metriky přesnosti ranku se zaměřují na korektnost pořadí doporučených položek vzhledem k preferencím uživatele. Jde o statistické měření korelace ranku. Metriky se mohou zaměřit na porovnávání celého seznamu doporučených položek či pouze jeho částí. Z tohoto pohledu se dají nastavit kritéria pro důležité pozice v seznamu a zkoumat je. Některé algoritmy nevytváří kompletní seznam položek, protože v danou chvíli není ani potřeba. Prezentační vrstva doporučovacího systému má v drtivé většině případů implementovánu některou z forem stránkování. Příkladem metriky porovnávající dva setřízené seznamy může být *Kendallův τ koeficient* [28].

Ostatní metriky. Metriky této kategorie nespadají do žádné z předchozích a měří jiné parametry doporučovacích systémů. Sem spadají metriky určující rychlost algoritmu, škálovatelnost, aj.

6.3 Vyhodnocení vzorového algoritmu

Vzorový algoritmus představený v kapitole 5 budeme nyní analyzovat ze dvou hlavních hledisek. Jednak se bude hodnotit jak dobře či špatně algoritmus doporučuje. K tomu se využije několik metrik představených výše. Dále se bude zkoumat výkonnost implementace co se časové náročnosti týče.

6.3.1 Úspěšnost doporučení

Pro účely testování úspěšnosti doporučení byla zvolena *offline* metoda vzorku dat ostrého provozu systému. Zkoumá se přesnost predikce dle metrik z kapitoly 6.2. Vzorek dat byl rozdělen do *testovací* a *učící* množiny.

Učící množina. Slouží k „naučení“ doporučovacího systému, jak se chovají vybraní návštěvníci při hodnocení položek. Z dat učící množiny se sestaví matice doporučení, potřebná k výpočtu doporučovacícho skóre.

Testovací množina. Obsahuje hodnocení nemovitostí návštěvníků, kteří nejsou zahrnuti v učící množině. Slouží k porovnání přesnosti doporučovacího algoritmu. Oddělení testovací množiny z hlediska *RePortu* je řešeno velice jednoduše. Objekt `UserPropertyRating` byl rozšířen o atribut `isTesting`, který značí příslušnost testovací množině. Doporučující systém takto označené ohodnocení ignoruje.

Výběr testovací množiny

Pokud se technické řešení rozdělení množin dat na testovací a učící jeví jako snadné, samotný výběr dat už tak jednoznačný není. Výběr náhodných dat z řídké matice ohodnocení není vhodným řešením, jelikož algoritmus nemá v mnoha případech dostatek dat pro výpočet. Proto výběr dat probíhá systematicky v několika krocích. Kód pro separaci dat běží ve smyčce v metodě `CFRecommenderEvaluationService.findTestingData`.

1. Z matice podobnosti mezi nabídkami (`OfferSimilarityMatrix`) se vybere první nemovitost, která již není součástí testovací množiny, není zařazena do množiny ignorovaných nemovitostí a má definovanou podobnost s aspoň N nabídkami, kde N je konstanta `CFRecommenderAlgorithmService.OFFERS_TO_USE_SIMILARITY`[1]. V případě, že takováto nabídka už neexistuje, algoritmus končí.

2. Z matice *VisitorOfferIdMatrix* (obsahující reference ID hodnocení nabídek navštěvníky) se vybere návštěvník, který splňuje kritéria:

   (a) Návštěvník ohodnotil celkem aspoň `CFRecommenderAlgorithmService.MINIMUM_RATINGS_TO_USE_CF` nabídek.

   (b) Návštěvník ohodnotil nabídku z kroku 1.

   (c) Návštěvník ohodnotil aspoň `CFRecommenderAlgorithmService.OFFERS_TO_USE_SIMILARITY` nabídek s definovanou podobností oproti nabídce z kroku 1.

   Pokud se takový návštěvník nenajde, přidá se nabídka z kroku 1 do seznamu ignorovaných nemovitostí a pokračuje se krokem 1.

3. Do testovací množiny se přidá `UserPropertyRating` pro nemovitost a návštěvníka z kroků 1 a 2. Abychom dosáhli větší variability nabídek v testovací množině, omezila se kardinalita ohodnocení `UserPropertyRating` týkající se stejné nabídky na 10. Programová smyčka skončí po označení nejpozději maxTestDataSize[2] objektů `UserPropertyRating` jako testovacích.

Vyhodnocení jednotlivých variant

Po rozdělení dat na učící a testovací množinu nadchází vyhodnocení úspěsnosti doporučovacího algoritmu. Pro každé hodnocení nemovitosti `UserPropertyRating` z testovací množiny získáme návštěvníka u a nemovitost i s hodnocením r. Pomocí doporučovacího algoritmu nad učící množinou vypočítáme predikci hodnocení nemovitosti i návštěvníkem u. Takto získané odhady porovnáme se skutečným ohodnocením nemovitosti i návštěvníkem u z testovací množiny a vypočítáme průměrnou odchylku, rozptyl odchylek a kvadratický průměr odchylek.

Vyhodnocení probíhalo postupně pro různá nastavení projekce vah explicitních a implicitních interakcí (viz kapitola 5.3. Pro osvěžení si je připomeneme:

[1] Během testování byla konstanta `CFRecommenderAlgorithmService.OFFERS_TO_USE_SIMILARITY` nastavena na 5

[2] Nastaví se v metodě `CFRecommenderEvaluationServiceImpl.findTestingData`.

TimeSpentOnProperty (int) - čas strávený návštěvníkem na stránce s detailem nemovitosti.

TimesPropertyOpened (int) - počet zobrazení stránky s detailem nemovitosti.

Rating1to5 (int) - hodnocení 1 - 5 hvězdiček.

QuestionSubmitted (boolean) - `true` pokud návštěvník položil dotaz ohledně nemovitosti.

CallRequested (boolean) - `true` pokud si návštěvník vyžádal telefonát od makléře.

TimesShownInSearchResults (int) - počet zobrazení konkrétní nemovitosti ve výsledcích vyhledávání.

Posledně jmenovaný parametr je využit v případě, že návštěvník neohodnotil dostatečný počet nemovitostí. V projekci objektu`UserPropertyRating` do $\mathbb{R}$ se přímo nepoužívá, avšak v kombinaci s parameterem `TimeSpentOnProperty` vytváří tzv. **penalizaci** nemovitosti, která snižuje doporučení v případě, že se nemovitost zobrazila ve výsledcích vyhledávání, ale nebyla pro uživatele dostatečně atraktivní, aby na ni kliknul. Následuje meta kód výpočtu penalizace. Proměnná `openedValue` je rovna počtu zobrazení detailu nemovitosti. Proměnná `shownInResultsValue` je rovna počtu výskytu ve výsledcích.

```
double score = 0;
if (openedValue > 0 && shownInResultsValue > 0) {
  double ratio = shownInResultsValue / openedValue;
  if (ratio > 1) {
    double numerator = Math.log(ratio);
    double denominator = Math.log(logarithmBase);
    score = numerator / denominator;
  }else {
    return new Double(0);
  }
}else if (openedValue == 0 && shownInResultsValue > 0) {

  double numerator = Math.log(shownInResultsValue);
  double denominator = Math.log(logarithmBase);
  score = numerator / denominator;
}else if (openedValue > 0 && shownInResultsValue == 0) {

  return new Double(0);
}else {

  return new Double(0);
}
if (score > 1) {
   score = 1;
}
return score;
```

Vyhodnocení výsledků probíhalo krokově v několika konfiguracích. Testuje se nejprve výchozí konfigurace tak, jak byla navrhnuta v kapitole 5.5.1. Následuje 8 evaluací, ve kterých se vždy mírně změnily váhy projekce (viz tabulka 6.1), a tím i přesnost doporučení:

1. Ignoruje všechny parametry mimo TimeSpentOnProperty.
2. Rovnoměrně rozděluje váhy mezi TimeSpentOnProperty a TimesPropertyOpened.
3. Bere do úvahy parametry TimeSpentOnProperty a TimesPropertyOpened, velkou váhu dává také vyžádání telefonátu od makléře (CallRequested).
4. Konfigurace je identicky ohodnocená jako předchozí v předchozím případě, až na výjimku v podobě explicitního ohodnocení 1 - 5 hvězdičkami (parametr Rating1to5). Pokud je ohodnocení k dispozici, použije se pouze tento parametr a ostatní se ignorují.
5. Výchozí návrh z kapitoly 5.5.1 určené dle „přirozené důležitosti“. V potaz se berou všechny dostupné parametry kromě penalizace.
6. Identická konfigurace jako předchozí s přidanou penalizací.
7. Rozdělení vah je dané mírně pozměněné směrem více k implicitním interakcím. V případě ohodnocení 1 - 5 hvězdičkami se ostatní ohodnocení překryjí.
8. Konfigurace všech dostupných parametrů bez překrývání s mírně odlišnými vahami směrem k času strávenému prohlížením nemovitosti a počtu otevření.

	1	**2**	**3**	**4**
TimeSpentOnProperty	100 %	50 %	30 %	30 %
TimesPropertyOpened	0 %	50 %	20 %	20 %
Rating1to5	0 %	0 %	0 %	100 %
QuestionSubmitted	0 %	0 %	0 %	0 %
CallRequested	0 %	0 %	50 %	50 %
Penalizace	0 %	0 %	0 %	0 %
	5	**6**	**7**	**8**
TimeSpentOnProperty	10 %	10 %	20 %	20 %
TimesPropertyOpened	5 %	5 %	15 %	20 %
Rating1to5	20 %	20 %	100 %	15 %
QuestionSubmitted	30 %	30 %	30 %	20 %
CallRequested	35 %	35 %	35 %	25 %
Penalizace	0 %	-20 %	-20 %	-20 %

Tabulka 6.1: Váhy testovacích konfigurací

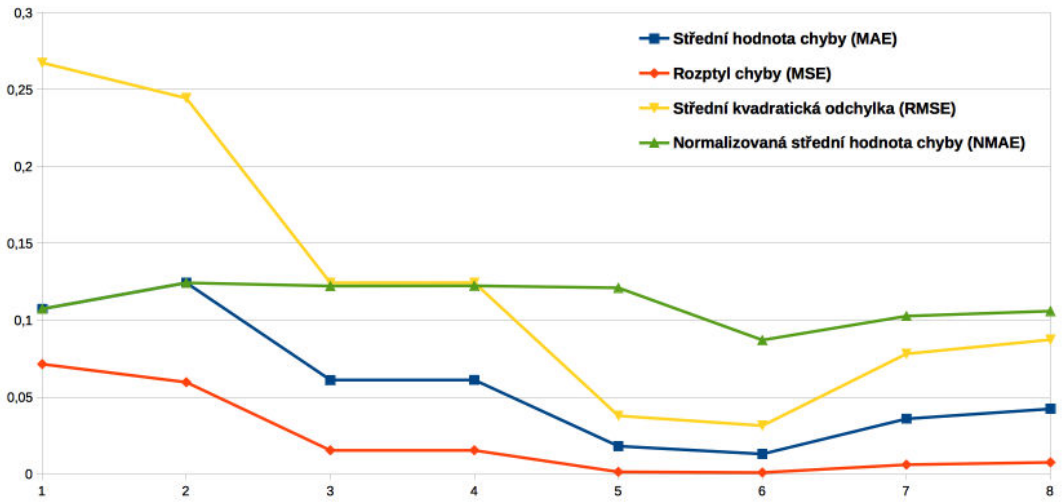
Testování jednotlivých konfigurací probíhalo offline na identických testovacích datech. Protože změna vah vždy změní projekci vektoru parametrů `UserPropertyRating` do $< 0; 1 >$, je nutné přepočítat všechny parciální data, jako např.

matici podobnosti nemovitostí `OfferSimilarityMatrix`. Konkrétně se provedou následující kroky:

1. Určí se testovací a učící množina (viz výše).
2. V kódu se upraví nastavení vah parametrů.
3. Vyprázdní se tabulky `recommender_user_property_rating_cf_evaluations`, `recommender_similarity_matrix_dirty_offers`, `recommender_similarity_matrix_elements`, `recommender_similarity_matrix_actions`, `recommender_similarity_matrices`.
4. Spuštěním JUnit testu `CFRecommenderEvaluationServiceTest.testComputeStatisticalData()` se spustí vyhodnocení dle postupu popsaného výše.

Konfigurace	MAE	MSE	RMSE	NMAE
1	0,107355092	0,071482244	0,267361635	0,107355092
2	0,124311435	0,059698999	0,244333786	0,124311435
3	0,061111194	0,015461884	0,124345826	0,122222388
4	0,061191530	0,015481547	0,124424863	0,122383061
5	0,018158360	0,001437672	0,037916644	0,121055736
6	0,013065305	0,000990405	0,031470712	0,087102030
7	0,035947383	0,006110991	0,078172830	0,102706808
8	0,042380191	0,007630904	0,087355048	0,105950476

Tabulka 6.2: Výsledky vyhodnocení



Obrázek 6.1: Vizualizace výsledků z tabulky 6.2

Výsledky vyhodnocení jednotlivých konfigurací vah jsou uvedeny v tabulce 6.2. Testy probíhaly na identických datech, aby byly porovnatelné. Celkově dávají

konfigurace 6 a 7 poměrně slušné výsledky, pokud si uvědomíme, že možné hodnoty ohodnocení jsou z intervalu $< 0; 1 >$. Bylo by vhodné připomenout poznatky ze sběru dat (kapitola 5.3), kde se zjistilo, že v převážně většině případů má algoritmus k dispozici pouze proměnné TimesSpentOnProperty, TimesPropertyOpened a odvozenou Penalizaci, tedy interval $< 0; 1 >$ se nevyužije celý. Proto se zavedla hodnota **NMAE**, která normalizuje střední hodnotu chyby vzhledem k velikosti projekce vektoru ohodnocení. Tedy v případě konfigurace 1 a 2 se hodnota nijak nemění, protože hodnoty TimeSpentOnProperty a TimesPropertyOpened je typicky k dispozici a pokrývají 100% konfigurace. V případě konfigurace 3 a 4 pokrývají hodnoty TimeSpentOnProperty a TimesPropertyOpened pouze 50%, tedy normalizovaná hodnota je v jejich případě $NMAE = 2 * MAE$.

Z hodnot v tabulce 6.2 a z grafu 6.1 vidíme, že po normalizaci se střední hodnota chyby mění pouze minimálně. Nejlepší nastavení vah bylo zvoleno u konfigurace 6. Po vyhodnocení se příslušným způsobem přenastavily váhy v kódu algoritmu.

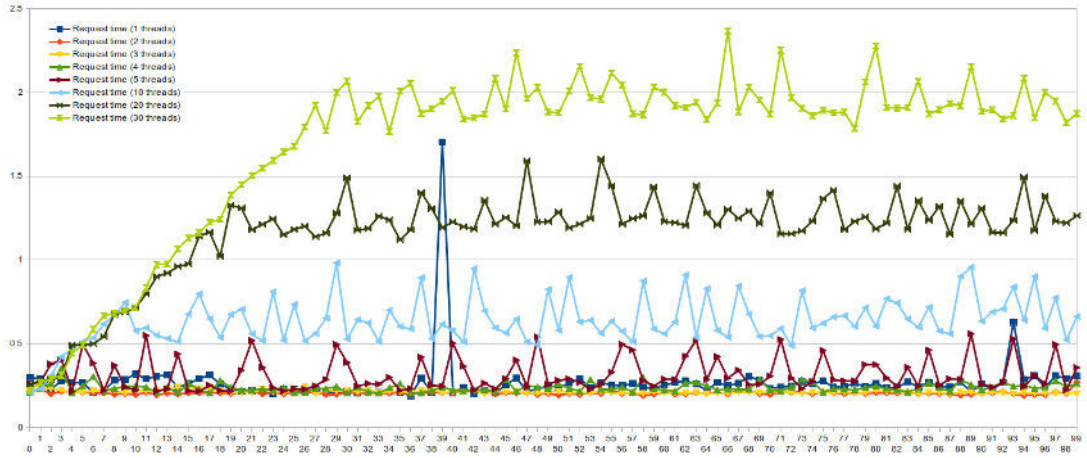
Pokud normalizujeme i hodnoty MSE a RMSE dospějeme k závěru, že zvolený algoritmus se na stupnici $< 0; 1 >$ poměrně slušně trefuje do vkusu návštěvníků. Potvrdil se předpoklad z kapitoly 3.5, kde se předpokládaly dobré výsledky doporučení algoritmu při relativně jednoduchém výpočetním aparátu.

Z výsledků vyhodnocení se dá vyčíst, že při závislosti čistě na implicitních interakcích TimeSpentOnProperty a TimesPropertyOpened se setkáváme s horšími výsledky než v případě zapojení explicitních interakcí. Znatelné zlepšení pozorujeme v případě negativního ohodnocení.

Bohužel je nutné znovu konstatovat, že přes 93% návštěvníků nemá dostatečně naplněný uživatelský profil a trpí problémem studeného startu. Dalším problémem jsou podkladová data pro doporučovací algoritmus. V našem zkoumání jsme zjistili, že celkový počet explicitních interakcí nedá ani 1% z celkového počtu. Musíme se ptát, do jaké míry je doba strávená prohlížením nemovitosti signifikantní z hlediska uživatelského hodnocení. Případně zda není vhodné explicitní interakce vinucovat podobně jako *Trulia Suggests*.

6.3.2 Časová náročnost implementace

V kapitole 5.2 byla rozumná odezva stanovena jako jeden ze základních atributů algoritmu. Později bylo toto číslo upřesněno jako výpočet ukončený za méně než jednu sekundu. Během optimalizací (5.5.3) byl algoritmus testován pouze jednovláknově. V ostrém nasazení je ale nutné simultánně odbavit několik uživatelů naráz. Dle *Google Analytics*, sledujícího návštěvnost webu společnosti *RealExpert s.r.o.*, bývá ve špičce v jednu chvíli do 10 návštěvníků. Neznamená to sice 10 simultánních přístupů v jeden moment, spíš s odstupem několika sekund, nicméně číslo dává dobrou představu o vytíženosti. Pro účely doporučovacího systému pod zátěží provedeme měření. Algoritmus doporučení má dvě větve, jednu pro čerstvé uživatele bez dostatku ohodnocení a druhou pro dostatečně naplněné uživatelské profily. Provedeme měření odezvy obou větví algoritmu a jako horní odhad běhu algoritmu pro daný počet simultálních přístupů vybereme vždy ten horší případ ze dvou větví.



Obrázek 6.2: Naměřený čas pro jednotlivé testovací běhy analýzy atributů nemovitosti

počet vláken	**průměrný čas** (s)
1	0,2699
2	0,2070
3	0,2133
4	0,2389
5	0,3179
10	0,6349
20	1,1536
30	1,1691

Tabulka 6.3: Průměrné časy naměřené u doporučení dle analýzy atributů nemovitosti

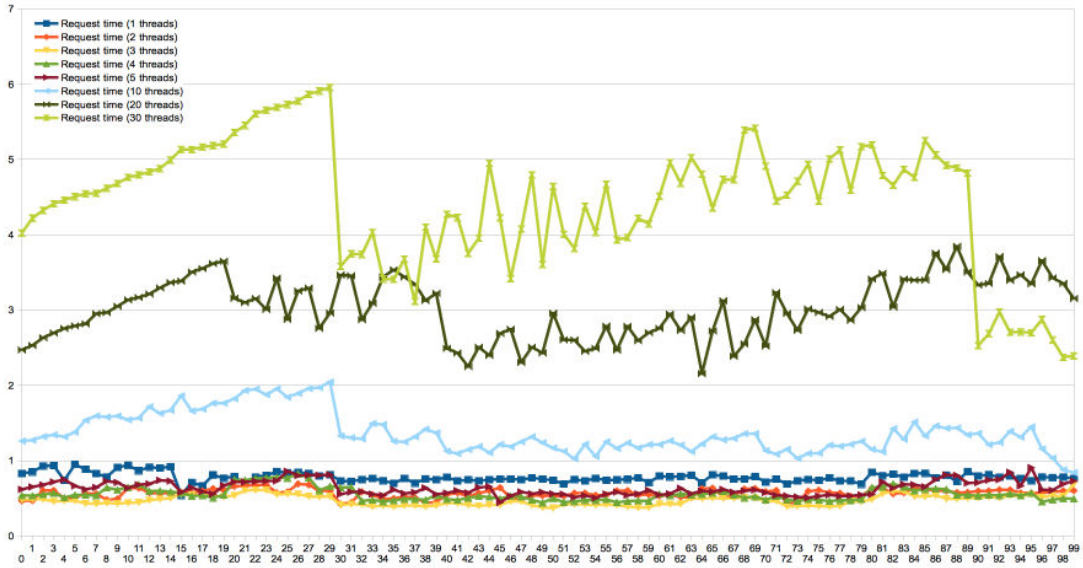
Ohodnocení na základě analýzy atributů nemovitosti

Analýza atributů je jednodušší pro výpočet, jelikož se neprovádí sestavování matice v paměti, ale dotazuje se přímo databáze. Provedeme zátěžový test postupně o 1, 2, 3, 4, 5, 10, 20 a 30 vláknech, kde se budeme nad reálnými daty dotazovat na 100 doporučení. Délky jednotlivých časů pro konkrétní počet vláken je zanesen v grafu na obrázku 6.2 a průměry běhů jsou zaznamenány v tabulce 6.3. Z naměřených dat je patrné, že pokud nepřekročíme počet jader procesoru[3], čas potřebný k výpočtu je téměř identický. Z výsledků vyčteme, že algoritmus v této větvi dokáže ve špičce s přehledem odbavit 10 vláken naráz.

Ohodnocení na základě kolaborativního filtrování

Pro zachování kontinuity budeme měřit stejným způsobem i druhou větev algoritmu. Jelikož se reálná data potýkají s nedostatkem úplných profilů, u kterých dojde k přepnutí k ohodnocení na základě kolaborativního filtrování, testovací procedura vždy vygeneruje seznam uživatelů s náhodným ohodnocením 20 až 50

[3]Procesor *Intel Core* i5-750 testovacího stroje má 4 fyzická jádra



Obrázek 6.3: Naměřený čas pro jednotlivé testovací běhy kolaborativního filtrování

počet vláken	**průměrný čas** (s)
1	0.7788
2	0.5660
3	0.4709
4	0.5528
5	0.6346
10	1.3668
20	3.0205
30	4.4323

Tabulka 6.4: Průměrné časy naměřené u doporučení dle kolaborativního filtrování

nemovitostmi. Zbytek testovací procedury probíhá identickým způsobem jako výše, tedy opět se vytvoří pro konkrétní počet vláken fronty akcí, které se průběžně plní dle ukončení[4]. Průměrné časy běhu pro 1, 2, 3, 4, 5, 10, 20 a 30 vláken jsou opět k dispozici v tabulce 6.4. Časy jednotlivých běhů jsou pak vyneseny v grafu 6.3.

Porovnáním hodnot obou tabulek zjistíme, že doporučení na základě kolaborativního filtrování je násobně časově náročnější, oproti doporučení na základě atributů nemovitostí. V případě kolaborativního filtrování odezva při 10 simultánních požadavcích vyroste mírně nad vytyčenou únosnou mez 1s. Limitním počtem pro navrácení výsledku pod 1s je 7 vláken. **Teoretickým výkonnostním maximem** algoritmu na testovacím stroji je **25 200 doporučení za hodinu**, za předpokladu naplnění uživatelského profilu pro každého dotazovaného uživatele. V případě nedodržení podmínky jsou časy odezvy kratší, proto se výkonnostní maximum posune výše. Jelikož se ale na pozadí periodicky provádí částečná

[4]Konkrétně se jedná o *Executors.newFixedThreadPool()*

materializace matice ohodnocení, reálný výkon algoritmu je naopak stlačen dolů.

7. Závěr

Úvodem práce se vytyčilo několik dílčích cílů, z nichž některé byly na sobě závislé. Aby se mohlo začít pracovat na kódu pro diplomovou práci, bylo potřeba odstranit některé části informačního systému. Jednalo se jak o výběr nových technologií nejen pro makléřskou část `RePortu` (*Spring MVC*, *Twitter Bootstrap*, `Spring WS`), tak i o návrh nových funkčních modulů (modul lokality nemovitostí, importní modul dat, ...). Jelikož měl autor se všemi použitými technologiemi předchozí zkušenosti, rovněž také s použitými návrhovými vzory, dostála implementace jeho očekáváním. Společně s webovou prezentací vzorové realitní kanceláře se implementace stala solidním základním kamenem pro *RePort Recommender Framework* a není jí příliš co vytknout.

Stavbě frameworku pro doporučovací algoritmy *RePort RF* předcházela důkladná analýza metodik doporučovacích systémů, z nichž se podařilo vybrat všechny esenciální požadavky. Framework je dostatečně logicky oddělený od ostatního kódu. Stejně tak jsou stanovena jednoznačná pravidla pro implementaci nových doporučovacích algoritmů, která při dodržovaní povedou k přehlednosti kódu. Jako efektivní se jeví programování algoritmu oproti jednotnému interface, jehož metody vycházejí právě ze zmíněné analýzy. Jedná se převážně o atomické dotazy na doporučení nemovitostí či tzv. logické triggery, které se spouštějí v případě změny stavu databáze (změna v nemovitostech, návštěvnících...). Autor se snažil udržet interface co nejjednodušší, aby se vyhnul nuceným implementacím nepotřebných metod.

Po zkušenosti s laděním doporučovacího algoritmu nad *RePort RF*, zejména pak s vyhodnocením výsledků konfigurací vah jednotlivých uživatelských interakcí, je vhodné do budoucna učinit, případně zvážit následující rozšíření:

Větší konfigurovatelnost doporučovacích algoritmů. *RePort RF* umožňuje každé realitní kanceláři přiřadit nejvýše jeden doporučovací algoritmus, který je pevně nakonfigurovaný. Bylo by vhodné zavést kromě výběru algoritmu i nastavení jeho parametrů. Výzvou by bylo vymyšlení vhodného konfiguračního formátu, který by byl dostatečně volný, vyřešit validaci chyb, přidat navázané akce v případě změny konfigurace, které jsme v akci viděli v kapitole 6.3.1 apod.

AB testování. Ve fázi testování nového algoritmu je užitečné mít možnost souběžného běhu dvou algoritmů zároveň. Majoritní část návštěvníků je zpracována stávajícím algoritmem a pouze zlomek provozu prezentace získá doporučení pomocí nové implementace, která se vyhodnotí.

Propojení se sociálními sítěmi. Sociální sítě jako Facebook [9], Google+ [10], aj. umožňují v omezné míře přistupovat třetím stranám k jejich databázi účtu. Toto propojení by znamenalo přiřazení demografických dat k již získané množině. Bylo by zajímavé sledovat závislosti věku a pohlaví na preferencích nemovitostí. Propojení by vyžadovalo zavedení návštěvnických uživatelských účtů.

Párování uskutečněných obchodů. V případě uzavření obchodu by bylo vhodné spárovat vyhledávání zájemce o nemovitosti, pokud to umožní dostateč-

ný sběr dat a jednoznačná identifikace návštěvníka přihlášením, a zvýšit ocenění doporučení. Je třeba ale brát v potaz skutečnost, že od alokace nemovitosti a samotným obchodem může uběhnout i několik měsíců, a proto řada doporučení už nemusí platit. Opět by bylo nutné zavést návštěvnické uživatelské účty.

Poslední část, kterou se diplomová práce zabývala, se věnovala implementaci vzorového doporučovacího algoritmu v rámci *RePort RF*. Autor zvolil *memory-based* variantu *kolaborativního filtrování*. Vypořádal se se studenými starty uživatelů, kdy algoritmus doplnil o doporučování na základě analýzy položek. Algoritmus prošel několika optimalizacemi, zejména z hlediska rychlosti. V předpokládané zátěži se podařilo v osmi simultánních vláknech dostat pod hranici 1 sekundy. Tuto hodnotu považuje autor za hraniční a v případě navýšení provozu by doporučoval nahrazení algoritmu *model-based* variantou *kolaborativního filtrování*. Další variantou může být nahrazení existujícími knihovnami doporučovacích algoritmů, které nabízejí mnohem širší možnosti experimentování. Rovněž lze předpokládat větší dospělost a optimalizovanost kódu. V neposlední řadě nutno podotknout, že některé dostupné knihovny doporučovacích algoritmů (např. *Apache Mahout*) explicitně řeší škálování. Běží v prostředí *Apache Hadoop* [26], který se transparentně stará o distribuované výpočty v případě, že požadavky na výpočetní výkon přesáhnou prostředky hostitelského počítače.

Pokud budeme zkoumat přesnost doporučení, potvrdilo se, že navržený algoritmus v nejlepší konfiguraci navrací výsledky s průměrnou chybou zhruba 8,7% od předpokládaného ohodnocení, což je slušný výsledek a potvrdil teze z kapitoly 3. Výpočet vycházel z několika implicitních a explicitních uživatelských interakcí. Na výsledcích vyhodnocení se potvrdilo, že explicitní interakce jsou pro doporučení důležitější než implicitní. Bohužel počet vykonaných explicitních interakcí nedosáhl ani 1% podílu. Algoritmus tak v drtivé většině případů počítá doporučení pouze z času stráveného prohlížením nemovitostí, počtu zobrazení nemovitostí a obsahu výsledků hledání.

Návštěvníci vzorové prezentace si během své návštěvy zobrazí malý počet nemovitostí, proto je jich většina ve fázi studeného startu. V té chvíli nemá doporučovací algoritmus dostatek dat pro vytvoření smysluplných doporučení. Autor navrhuje možná zlepšení právě v této počáteční fázi. Mimo jiné se dají aplikovat i nápady uvedené dříve u *RePort RF*

Implicitní specifikace lokality IP. Lze předpokládat, že návštěvníci v případě nespecifikace lokality budou vyhledávat nemovitosti v nejbližší lokalitě. Existují služby[1], specializující se na lokaci uživatele dle IP adresy.

Analýza textových popisu nemovitostí. Stávající algoritmus provádí analýzu obsahu nemovitosti pouze na základě vyčíslitelných hodnot (cena, lokalita, kategorie, atd.). Bylo by zajímave provést analýzu textu inzerátu, vytvořit ontologii a doporučovat na základě hodnocení objektů v ontologii.

Čtenáře během posledních několika odstavců musela napadnout otázka, zda se potvrdila teze z úvodu práce - zda má aplikace doporučovacího systému v oblasti

[1]Služba *MaxMind Geo IP* `http://www.maxmind.com/en/city_accuracy` udává lokalizaci uživatele dle IP do 40 km s přesností 82%.

realit smysl. Na tuto otázku autor nemá jednoznačnou odpověď. V případě menších realitních kanceláří je objem nabízených nemovitostí dostatečně malý na to, aby nepotřeboval doporučovací systém. Při listování výsledky hledání je návštěvník konfrontován typicky s 1 - 3 stranami výsledků. V případě, že je prezentace schopna úspěšně posbírat více uživatelských interakcí, doporučovací systémy mají smysl i u malých objemů nemovitostí a pouze zvyšují uživatelský komfort. Autor by naopak doporučovací systémy nasadil v případě provozování agregátoru nemovitostí, kdy se mění chování návštěvníku a ti jsou schopni prozkoumat i desítky různých nabídek. Matice ohodnocení je v jejich případě hustší.

Seznam použité literatury

[1] Herlocker, et al. *Evaluating Collaborative Filtering Recommender Systems*. ACM Transactions on Information Systems, Vol. 22, No. 1, Leden 2004. ISSN: 1046-8188 EISSN: 1558-2868

[2] Ricci, Rokach, Shapira, Kantor, *Recommender Systems Handbook*. Springer New York Dordrecht Heidelberg London 2011. ISBN: 978-0-387-85819-7 e-ISBN: 978-0-387-85820-3

[3] Lambrecht, Tucker, *When Does Retargeting Work? Information Specificity in Online Advertising* Dostupné online na `http://ssrn.com/abstract=1795105`, navštíveno 31. března 2013

[4] Pazzani, Billsus, *Content-based recommendation systems*, The Adaptive Web: Methods and Strategies of Web Personalization. Series: Lecture Notes in Computer Science, Vol. 4321, Springer, Prosinec 2007. ISBN: 978-3-540-72079-9

[5] Yuan, Lee, Kim, Kim, *Toward a user-oriented recommendation system for realestate websites*. http://dblp.uni-trier.de/rec/bibtex/journals/is/YuanLKK13

[6] Mitchell, *Machine Learning*, McGraw-Hill, 1. vydání, Březen 1997. ISBN: 978-0-070-42807-2

[7] Su, Khoshgoftaar, *A Survey of Collaborative Filtering Techniques*, Advances in Artificial Intelligence, Vol. 2009, Article ID 421425. doi: 10.1155/2009/421425

[8] Baquero, Húska, Strnad, *Report - Dokumentace Softwarového Projektu*. MFF UK Praha, 2012.

[9] *Facebook for Websites*, dokumentace. Dostupné online na `https://developers.facebook.com/docs/guides/web/`, navštíveno 24. března 2013

[10] *Google Accounts Authentication and Authorization*, dokumentace, Dostupné online na `https://developers.google.com/accounts/`, navštíveno 24. března 2013

[11] *Directive 2009/136/EC Of The European Parliament And Of The Council* Dostupné online na `http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=OJ:L:2009:337:0011:0036:En:PDF` , navštíveno 14. dubna 2013

[12] *The Privacy and Electronic Communications (EC Directive) (Amendment) Regulations 2011* Dostupné online na `http://www.legislation.gov.uk/uksi/2011/1208/made`, navštíveno 14. dubna 2013

[13] *MySQL 5.7 Manual - Full-Text Stopwords* Dostupné online na `http://dev.mysql.com/doc/refman/5.7/en/fulltext-stopwords.html`, navštíveno 13. května 2013

[14] Lovins, *Development of a stemming algorithm*, Mechanical Translation and Computational Linguistics, vol.11, nos.1 and 2, Květen, červen 1968.

[15] Paice, *Another stemmer*, SIGIR Forum vol.24, podzim 1990. http://dblp.uni-trier.de/rec/bibtex/journals/sigir/Paice90

[16] Salton, *Automatic Text Processing*, Addison-Wesley Publishing Company, 1989. ISBN: 0-201-12227-8

[17] Breese, Hackerman, Kadle, *Empirical Analysis of Predictive Algorithms for Collaborative Filtering*, Morgan Kaufmann Publishers, květen 1998, Dostupné online na `http://research.microsoft.com/pubs/69656/tr-98-12.pdf`, navštíveno 27. května 2013

[18] Wang, Vries, Reinders, *Unifying User-based and Item-based Collaborative Filtering Approaches by Similarity Fusion*, SIGIR '06 Proceedings of the 29th annual international ACM, ACM New York, 2006. ISBN: 1-59593-369-7

[19] The Netflix Prize Rules , Dostupné online na `http://www.netflixprize.com/rules`, navštíveno 15. dubna 2013

[20] Netflix Recommendations: Beyond the 5 stars (Part 1), Dostupné online na `http://techblog.netflix.com/2012/04/netflix-recommendations-beyond-5-stars.html`, navštíveno 15. dubna 2013

[21] Netflix Prize Update, Dostupné online na `http://blog.netflix.com/2010/03/this-is-neil-hunt-chief-product-officer.html`, navštíveno 15. dubna 2013

[22] Liang Xiang, *Hulu Tech Blog: Hulu's Recommendation System*, Dostupné online na `http://tech.hulu.com/blog/2011/09/19/recommendation-system/`, navštíveno 7. dubna 2013

[23] Miyahara, Pazzani, *Improvement of Collaborative Filtering with the Simple Bayesian Classifier*, PRICAI'00 Proceedings of the 6th Pacific Rim international conference on Artificial intelligence, Springer-Verlag Berlin, Heidelberg, 2000. ISBN: 3-540-67925-1

[24] Sarwar, Karypis, Konstan, Riedl, *Item-based Collaborative Filtering Recommendation Algorithms*, Proceedings of the 10th international conference on World Wide Web, ACM New York, 2001.

[25] Apache Mahout, webová prezentace knihovny, Dostupné online na `http://mahout.apache.org/`, navštíveno 13. srpna 2013

[26] Apache Hadoop, webová prezentace běhového prostředí. Dostupné online na `http://hadoop.apache.org/`, navštíveno 13. srpna 2013

[27] Schröder, Thiele, Lehner, *Setting Goals and Choosing Metrics for Recommender System Evaluations*, 5th ACM Conference on Recommender Systems, Chicago, 2011

[28] Kendall, *A New Measure of Rank Correlation*, Biometrika, Vol. 30, No. 1/2, Biometrika Trust, červen 1938. ISSN: 00063444

[29] Trulia corporate blog - Introducing Trulia Suggests, A New Way to Discover Homes Picked Just For You, blog spoečnosti Trulia, dostupné online na `http://corp.truliablog.com/2013/03/06/introducing-trulia-suggests/`, navštíveno 12. října 2013

[30] Cvengroš, *Univerzální doporučovací systém*, Diplomová práce, Univerzita Karlova v Praze, Matematicko-fyzikální fakulta, 2011

[31] Papagelis, Plexousakis *Qualitative analysis of user-based and item-based prediction algorithms for recommendation agents*, Engineering Applications of Artificial Intelligence 18, 2005

[32] Melville, Mooney, Nagarajan *Content-Boosted Collaborative Filtering*, Proceedings of the SIGIR-2001 Workshop on Recommender Systems, New Orleans, LA, September 2001

[33] Adomavicius, Tuzhilin *Context-Aware Recommender Systems*, AI Magazine, vol. 32, no. 3, 2011. ISSN 0738-4602

[34] Sarwar, Karypis, Konstan, Riedl *Application of Dimensionality Reduction in Recommender System - A Case Study*, ACM WebKDD Workshop, University of Minnesota, 2000

[35] MindBridge Consulting a.s. *Vývoj realitního trhu - pokles se zastavil*, studie residenčního realitního trhu. Publikováno 11. března 2014, dostupné online na `http://www.mindbridge.cz/cs/aktuality/vyvoj-realitniho-trhu-pokles-se-zastavil/`, navštíveno 15. července 2014

[36] Český statistický úřad *Klasifikace ekonomických činností CZ-NACE*, dostupné online na `http://www.czso.cz/csu/klasifik.nsf/i/klasifikace_ekonomickych_cinnosti_(cz_nace)`, navštíveno 15. července 2014

[37] Český statistický úřad *Počet osob v hlavním zaměstnání v sekcích a vybraných oddílech Klasifikace ekonomických činností CZ-NACE*, Veřejná databáze ČSÚ, dostupné online pro jednotlivé roky, navštíveno 15. července 2014

2009 `http://vdb.czso.cz/vdbvo/tabparam.jsp?voa=tabulka&cislotab=BULLE01901R09&vo=null`

2010 `http://vdb.czso.cz/vdbvo/tabparam.jsp?voa=tabulka&cislotab=BULLE01901R10&vo=null`

2011 `http://vdb.czso.cz/vdbvo/tabparam.jsp?voa=tabulka&cislotab=BULLE01901R11&vo=null`

2012 `http://vdb.czso.cz/vdbvo/tabparam.jsp?voa=tabulka&cislotab=BULLE01901R12&vo=null`

2013 http://vdb.czso.cz/vdbvo/tabparam.jsp?voa=tabulka&cislotab=BULLE01901R13&vo=null

[38] Český statistický úřad *Ekonomické subjekty podle odvětví (NACE) a krajů*, Veřejná databáze ČSÚ, dostupné online pro jednotlivé roky na http://vdb.czso.cz/vdbvo/tabparam.jsp?&cislotab=ORG0020UP_KR&voa=tabulka, navštíveno 15. července 2014

Seznam použitých zkratek

AJAX - Asynchronous JavaScript and XML. Asynchronní přenos XML dokumentů v prohlížeči vyvolaný JavaScriptem.

CBRS - Content-based recommender system. Doporučující systém založený na analýze obsahu.

CRM - Customer relationship management. Software pro řízení vztahů se zákazníky.

CF - Collaborative filtering. Kolaborativní filtrování.

CTR - Click-through rate. Míra prokliku. Poměr zobrazených odkazů a kliknutí na ně uživatelem.

DAO - Data access object. Objekt poskytující interface pro operace s datovým úložištěm.

HQL - Hibernate Query Language. Dotazovací jazyk nad frameworkem pro persistenci dat Hibernate.

IoC - Inversion of Control. Návrhový vzor objektově orientovaného programování, který rozbíjí závislosti jednotlivých tříd, programuje se oproti interface a objektový graf tříd se vytváří až za běhu.

JDBC - Java Database Connectivity. Java API pro připojení k databázi, nezávislé na konkrétním databázovém stroji

JSON - JavaScript Object Notation. Jednoduchý a velice obecný způsob zápisu datových struktur do člověkem čitelného řetězce.

ORM - Objektově relační mapování. Technika pro konverzi dat mezi relační a objektovou reprezentací.

POJO - Plain old Java object. Označení obyčejného, nekomplikovaného Java objektu - entity.

RPC - Click-through rate. Míra prokliku, tedy poměr mezi počtem prokliků a počtem zobrazení.

SQL - Structured Query Languge. Dotazovací jazyk nad relačními databázemi.

SŘBD - Systém řízení báze dat.

SOAP - Simple Object Access Protocol. Protokol pro výměnu strukturovaných informací ve formátu XML.

XML - Extensible Markup Language. Obecný značkovací jazyk pro výměnu strukturovaných dat. Dokument ve formátu XML je pro člověka čitelný.

WS - Web service. Komunikační protokol mezi systémy pomocí HTTP.

WSDL - Web Services Description Language. Jazyk pro specifikaci interface (volání a návratových hodnot) ve formátu XML.

A. Přehled interface web services

Web services poskytují množinu metod, které dokáží obstarat drtivou většinu funkcí průměrné prezentace realitní kanceláře. Pro vývoj byla použita metoda shora dolů, tedy nejprve je zadefinováno schéma (soubor `WebContent/WEB-INF/config/WebServiceContract.xsd`), ze kterého se nástrojem `xjc` generují Java beans a který je transformován na WSDL dokument. Spring WS framework se stará o kompletní marshalling / unmarshalling zpráv. Komunikaci obsluhuje služba `EndpointService`, kde vhodnými anotacemi určíme metody plnící SOAP zprávy daty.

Nyní si zevrubně představíme jednotlivé metody, aby měl čtenář představu, čeho se dá nasazením *RePortu* dosáhnout. Pro podrobnou definici datových typů a volání použijte jako referenci výše zmíněný soubor `WebContent/WEB-INF/config/WebServiceContract.xsd`.

GetArticleList (token) - navrací seznam dostupných článků pro realitní kancelář identifikovanou jednoznačným tokenem.

GetArticle (token, articleType, articleKey) - navrací konkrétní článek dle výčtového typu (articleType) a klíče. Součástí návratové hodnoty jsou i metadata a veškeré dostupné jazykové mutace.

GetAllActiveOfferList (token) - navrací seznam titulků aktivních nemovitostí a stránkování pro danou realitní kancelář. Návratová hodnota obsahuje pouze základní informace.

GetOfferList (token, parametry...) - k zadaným parametrům vyhledávání (cena, rozloha, lokalita, dispozice, atd.) vyhledá příslušné nemovitosti. V případě aktivovaného doporučujícího systému a implicitní volby řazení uživatelem probíhá

GetRandomOfferList (token, numberOffers) - navrátí `numberOffers` počet aktivních nemovitostí.

GetRecommendedOfferList (token, numberOffers) - navrátí `numberOffers` nemovitostí, vybraných z označených nemovitostí realitní kanceláří jako doporučené.

SignInToNewsletter (token, userEmail) - přidá emailovou adresu návštěvníka do seznamu pro odebírání novinek.

SignOutOfNewsletter (token, subscriptionId) - odstraní uživatele se zadaným ID ze seznamu pro odebírání novinek.

GetRecommenderParameterValue (sessionId, ratingId, parameter) - pro daného uživatele navrátí hodnotu dříve zadaného parametru (pokud se tak stalo).

SubmitRecommenderParameter (token ,.., parameter*) - uloží hodnoty předdefinovaných parametrů zadané uživatelem.

GetOffer (token, offerId, trackingParameters...) - obdrží nabídku nemovitosti dle ID, včetně podrobností, překladů, fotografií, apod. Součástí volání může být i reference na předchozí ID vyhledávání, které vrátilo nabídku nemovitosti. Tato cesta je uložena.

GetSearchFormBoundaries (token, businessType, estateType) - navrací mezní hodnoty ceny a rozlohy pro účely zobrazení vyhledávacích formulářů.

GetSearchFormLocation (token, ...) - navrací seznam lokalit, shodujících se s řetězcem v parametru volání.

GetDistrictsForRegion (token, region, country) - pro zadaný region a zemi navrací územní jednotky (např. pro Prahu rozdělení na Prahu 1 - Prahu 10).

GetEstateStatistics (token) - navrací sumarizační statistiky o nabídkách realitní kanceláře (počet makléřů, počet jednotlivých nemovitostí, aktuální hodnota obchodovaných nemovitostí).

TestLogin (token) - testuje validitu přihlašovacího tokenu.

GetBrokerEmail (token, offerId) - navrátí emailovou adresu makléře příslušného zadané nabídce.

B. Statistika osob / subjektů činnosti v oblasti nemovitostí

Český statistický úřad zavedl klasifikaci ekonomických činností CZ-NACE (číselník k dispozici v [36]), kde jsou „*Činnosti v oblasti nemovitostí*“ definovány pod kódem číselníku L následujícím způsobem: „Tato sekce zahrnuje činnosti pronajímatelů, agentů nebo makléřů v jedné nebo v několika následujících činnostech: prodej nebo nákup nemovitostí, pronájem nemovitostí, poskytování ostatních služeb v souvislosti s nemovitostmi, např. oceňování nemovitostí nebo vykonávání činností agentů podmíněných smluv o nemovitostech. Činnosti v této sekci mohou být prováděny s vlastním nebo pronajatým majetkem a mohou být vykonávány za úplatu nebo na smluvním základě. Sekce zahrnuje také stavební práce spojené s údržbou vlastních nebo pronajatých objektů. Do této sekce patří také činnosti správců majetku.“

Z veřejné databáze Českého statistického úřadu se dají zkompilovat data o vývoji zaměstnaných osob či počtu ekonomicky aktivních subjektů, které popisují vývoj trhu.

Období	**Celkem**	**Muži**	**Ženy**
Q1 / 2009	38,5	19,8	18,6
Q2 / 2009	40,6	19,5	21,1
Q3 / 2009	42,2	20,7	21,4
Q4 / 2009	41,3	19,2	22,2
Q1 / 2010	39,5	18,3	21,2
Q2 / 2010	38,9	16,9	22,0
Q3 / 2010	39,9	20,1	19,8
Q4 / 2010	41,9	20,7	21,2
Q1 / 2011	39,3	20,8	18,5
Q2 / 2011	42,0	22,6	19,4
Q3 / 2011	43,6	23,6	20,0
Q4 / 2011	43,5	21,0	22,5
Q1 / 2012	42,2	20,8	21,4
Q2 / 2012	44,7	22,9	21,8
Q3 / 2012	46,4	24,2	22,2
Q4 / 2012	49,3	24,6	24,7
Q1 / 2013	51,3	26,8	24,5
Q2 / 2013	47,2	24,4	22,8
Q3 / 2013	49,8	26,0	23,8

Tabulka B.1: Počet osob v hlavním zaměstnání v oblasti nemovitostí v čase (hodnoty uvedeny v tisících), zdroj ČSÚ [37]

Období	Registrované subjekty	Aktivní subjekty
2009	137 865	52 309
2010	146 313	68 422
2011	152 349	70 089
2012	154 621	78 901
2013	152 131	78 373

Tabulka B.2: Počet ekonomických subjekty subjektů v oblasti nemovitostí v čase, zdroj ČSÚ [38]

C. Obsah přiloženého CD

Součástí diplomové práce je i přiložené CD, na kterém se nalézají následující data:

- Zdrojové kódy informačního systému *RePort*, framework pro vývoj doporučujících systémů *Report RF* a vzorový doporučující algoritmus popsaný v této práci.
- Vygenerovaná dokumentace zdrojového kódu *JavaDoc*.
- Elektronická verze práce ve formátu PDF.
- Návod k instalaci a spuštění systému *RePort*.