Sem vložte zadání Vaší práce.

České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra softwarového inženýrství

Diplomová práce

Adaptibilní systém pro doporučování obsahu

Bc. Jan Bouchner

Vedoucí práce: Ing. Jaroslav Kuchař

27. dubna 2014

Poděkování

Chci upřímně poděkovat všem, kteří mi věnovali čas, když jsem potřeboval pomoc při psaní této diplomové práce, především vedoucímu práce Ing. Jaroslavu Kuchaři za správné směrování, celkový vhled do technologií a cenné komentáře. Děkuji také své rodině a přátelům za bezvýhradnou podporu během celých mých studií.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či spracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 27. dubna 2014 .....

Odkaz na tuto práci

Bouchner, Jan. *Adaptibilní systém pro doporučování obsahu.* Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2014.

Abstract

Sem doplňte ekvivalent abstraktu Vaší práce v angličtině.

Keywords Nahraďte seznamem klíčových slov v angličtině oddělených čárkou.

Abstrakt

V několika větách shrňte obsah a přínos této práce v češtině. Po přečtení abstraktu by se čtenář měl mít čtenář dost informací pro rozhodnutí, zda chce Vaši práci číst.

Klíčová slova Nahraďte seznamem klíčových slov v češtině oddělených čárkou.

Obsah

Seznam obrázků

Úvod

> "We are leaving the age of information and entering the age of recommendation."

Hned na úvod práce jsem si dovolil použít citát z článku *The Long Tail* [15] (česky Dlouhý chvost) bývalého šéfredaktora časopisu Wired Chrise Andersona, který je též autorem stejnojmenné teorie. Ta je založena na tom, že díky internetu lze nabízet širokou škálu produktů, které by dříve mohly jen sotva slavit prodejní úspěchy, neboť poptávka po nich je příliš malá.

Co je míněno tvrzením „které by dříve mohly jen sotva slavit prodejní úspěchy"?

Myšlenky se lze dopátrat v následující sekci Lidstvo a informace, po kterémžto vysvětlení bude vše zasazeno do kontextu zmíněné teorie (podsekce Informace a The Long Tail Theory) a její souvislosti s potřebou doporučování informací, jež je popsána v sekci Doporučovací systémy.

Sekce Motivace popisuje vizi systému schopného přizpůsobit se podmínkám a predikovat doporučení na základě předchozích a aktuálních zkušeností.

Popisu nároků na podobný systém, jenž jsem prostřednictvím této práce navrhl a následně implementoval, je věnována sekce Cíle práce. V sekci jsou též stanoveny cíle, kterých by měl nabývat implementovaný produkt.

Struktuře práce vzhledem k vytyčeným cílům se věnuje příslušná sekce Struktura práce.

Lidstvo a informace

Význam informací a dat je pro lidstvo odjakživa nezpochybnitelný. Již před nástupem digitálního věku byl jejich objem značný, takže ke spoustě informací, na základě kterých by si byl jednotlivec utvořil vlastní názor, nebylo snadné se dostat.

Lidé se často spoléhali na obecnou oblíbenost daného produktu (ať už se jednalo o zboží, článek, hudebního interpreta a jiné), která byla ale spíše než

čímkoliv jiným určována aktuálním společenským trendem nebo informace prostě a jednoduše čerpali ze zkušeností svých několika známých na základě jejich doporučení.

Nástup internetu

S masovým rozvojem internetu začalo množství dostupných informací růst obřím tempem (organizace IDC došla v článku *Extracting Value from Chaos* [25] k závěru, že objem světových dat se každé dva roky zdvojnásobuje). Nové informace jsou produkovány takřka každou sekundu a na jejich setřídění máme mnohem méně času než dříve. Není již v lidských silách udržet si o všem přehled.

Takovým vývojem se přirozeně změnil pohled na data. Cílem není shromáždit co největší objem, neboť při tomto pokusu bychom se zanedlouho dostali do stavu informačního zahlcení. Ceněnou schopností je nyní vytěžit maximální množství užitečných informací a ty využít pro získání nových vědomostí a alespoň přibližně správnou odpověď na otázku potřeb konkrétního jedince.

Informace a The Long Tail Theory

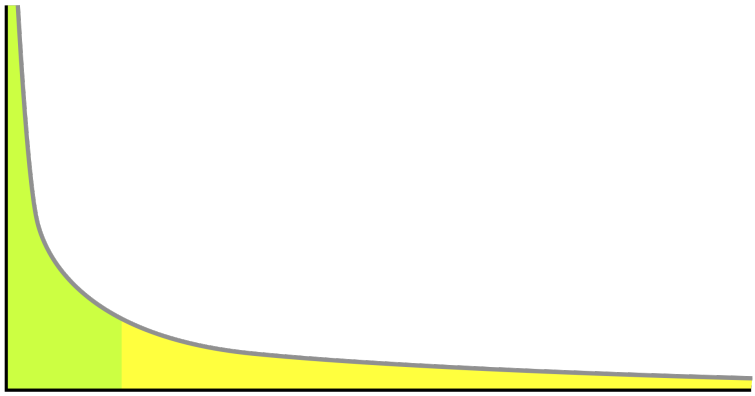
Vrátím se nyní ještě jednou k teorii Dlouhého chvostu. Pro účely této práce ji nemá význam vysvětlovat dopodrobna, ale je vhodné zmínit alespoň hlavní myšlenku, která vychází z úvah výše.

Nejdůležitějším poznatkem je to, že – z pohledu obchodního – s nárůstem informací dochází k fenoménu snižování prodejů dřívějších hitů (tedy těch několika málo produktů, které se na daném trhu dostanou k masovému publiku, a které určuje především trend) ve prospěch prvků nacházejících se v takzvaném dlouhém chvostu. Uživatel se tak může dostat k informacím, ke kterým by se obyčejným hledáním přes jeden z vyhledávačů třeba nikdy nedostal (pokud by tedy opravdu přesně nevěděl, co hledá).

Bližší představu si lze utvořit při pohledu na obrázek 0.1. Na vodorovné ose leží jednotlivé produkty, svislá osa pak znázorňuje objem prodeje (stejně tak se může jednat o popularitu). Žlutá část pod grafem vpravo reprezentuje dlouhý chvost (zhruba 80 procent produktů, po kterých je na trhu poptávka v malých objemech). Zelená část nalevo pak znázorňuje těch několik dominujících produktů (označuje se též jako *hlava*). Tento fenomén lze dobře ilustrovat na příkladu hudebního průmyslu, kde díky obrovským databázím interpretů [1] již nedochází k selekci a prodeji či poslechu jen těch potenciálně nejúspěšnějších, ale uživatelům se dostává mnohem většího výběru a je jen na nich, kterého si zvolí.

Všechny výše zmíněné faktory přirozeně zapříčinily rozvoj přístupu zvaného *informační filtrování*, tedy jakési selekce a redukce informací dle volených

[1] Jmenujme pár nejznámějších: last.fm, Spotify, Grooveshark, Google Music



Obrázek 0.1: Příklad rozdělení zobrazující popularitu hodnocení. Zdroj: `http://en.wikipedia.org/` [33]

parametrů. Odtud už byl jen krok k doporučování na míru, které se v posledních několika letech rozmohlo zaváděním tzv. *doporučovacích systémů*.

Doporučovací systémy

Doporučovací systémy jsou systémy, které mohou být uživateli nápomocné v tom smyslu, že z obrovské množiny informací různými způsoby vyfiltrují relevantní podmnožinu. Například prostým pozorováním toho, jak uživatelé službu využívají (v minulosti zakoupené produkty, seznam všech jimi ohodnocených článků či žebříček preferovaných hudebních interpretů), nebo porovnáváním kontextu uživatele s ostatními lidmi využívajících tu samou službu, lze objevovat pro uživatele nové a jejich vkusu vyhovující položky.

Typy systémů

Tyto systémy lze dle způsobu práce rozdělit do dvou skupin.

Kolaborativní filtrování. První skupinou jsou systémy fungující na principu tzv. kolaborativního filtrování. Tento princip má ještě dvě odnože zahrnující doporučování založené na podobnosti uživatelů (user-based) a doporučení založené na podobnosti položek (item-based). Doporučení z této skupiny nejvíce využívají stránky zabývající se prodejem různých produktů a také stránky nabízející novinové články.

Doporučení založené na obsahu. Druhou skupinou jsou pak doporučení založená na obsahu. Pro tato doporučení se nejčastěji využívá textové podobnosti. Výhodou je rychlé absorbování nových položek do systému

a imunita vůči tzv. *cold start problému* [2]. Již na malé množině dat lze dostávat poměrně uspokojivá doporučení.

Další informace o doporučovacích systémech a aktuální stav na poli doporučovacích systémů včetně technologického pozadí dokumentuje následující kapitola 1. Technické detaily jsou v menší míře k nalezení v sekci 4.2.3 kapitoly 4.

Motivace

Představme si nyní jakéhosi rádce, který dokáže v každém okamžiku rozhodnout, co je za daných okolností nejvhodnější s tím, že to, o čem rádce rozhoduje, je volba algoritmu, kterým by si měl uživatel v danou chvíli nechat doporučit obsah. Pokud by uživatel dbal těchto rad, s největší pravděpodobností by obdržel vhodné doporučení a svou spokojenost by vyjádřil například tím, že by klikl myší na doporučené položky nebo by jiným způsobem vyjádřil svůj zájem o doporučený obsah.

Dokud rádce explicitně nekomunikuje s uživatelem, neprovádí žádné operace a pouze vyčkává na okamžik, kdy bude vyzván k nějaké akci. V okamžiku, kdy je vyzván k tomu, aby uživateli doporučil obsah, je schopen ihned a bez dlouhé analýzy uskutečnit rozhodnutí.

Dokáže též pružně reagovat na situaci, kdy dojde k náhlé změně preferencí nebo vyvstanou jiné nečekané události, které by měli za následek dlouhodobější doporučování nevhodného obsahu.

Cíle práce

Cílem práce je tedy vyvinout rádce popsaného výše, a to ve formě aplikačního software.

Nejdůležitějším cílem této práce je tedy návrh a implementace takového adaptibilního systému, jenž by automaticky a vhodně volil a kombinoval metody pro doporučování obsahu. Metodami zde myslí doporučovací algoritmy. Jejich identifikace a výběr vhodné sady je též jednou ze součástí diplomové práce.

Systém bude využívat zpětnou vazbu ohledně kvality doporučení, která principem odměny za dobré doporučení nebo trestu za špatné ovlivní preference při kombinování. Zároveň bude zodpovědný za podporu většího množství uživatelů zasílajících žádosti na systém.

Nezbytnou součástí diplomové práce je též nastudování základních pojmů a pravidel strojového učení a teorie pravděpodobnosti. Zejména kvůli částem zabývajících se zpětnou vazbou a predikcí vhodného algoritmu.

[2] TODO

Vzhledem k fragmentované povaze projektu bude nutné navrhnout obecné rozhraní jak pro doporučovací algoritmy, tak pro adaptibilní systém zahrnující zpětnou vazbu.

K nastudování jsou nutné i technologie pro zpracování většího množství dat a pro rychlou odezvu (ta je potřebná již ze samotné podstaty problému).

Funkčnost navrženého systému by měla být ověřena implementací prototypu a spuštěním na modelové úloze, jež bude simulovat doporučování obsahu pro skupinu více uživatelů.

Tato diplomová práce dokumentuje návrh a vývoj takového systému a s ním spolupracujících komponent. Následuje seznam modulů, které byly pro potřeby práce vyvinuty:

- **adaptibilní systém pro doporučování obsahu**
- **sada základních algoritmů určených k doporučování**
- **rozhraní pro algoritmy a systém pro kombinování**
- **generátor článků, na kterých bylo prováděno základní měření**

Struktura práce

Tato práce je strukturována do 5 různých kapitol a popisuje celý vývojový cyklus systému. Kapitoly jsou řazeny v tom samém pořadí, v jakém probíhaly fáze vývoje.

Kapitola Aktuální stav na poli doporučování popisuje historické pozadí i aktuální trendy používané současnými doporučovacími systémy. Z této rešerše se snaží vyvodit některé závěry, které by mohly být užitečné pro účely některé z dalších kapitol.

Kapitola Analýza a návrh řešení shromažďuje veškeré požadavky kladené na systém, zabývá se návrhem architektury, detailními návrhy serverové i klientské části, rozhraním zastřešujícím komunikaci těchto dvou částí a výběrem základní sady algoritmů, které budou využity k doporučování obsahu.

Kapitola Adaptibilní systém detailně dokumentuje algoritmické a matematické přístupy použité při návrhu adaptibilního systému.

Kapitola Realizace popisuje vývoj aplikace na základě analýzy a návrhu z předchozích kapitol.

Kapitola Experimenty a vyhodnocení pak shrnuje poznatky nabyté z experimentů se systémem a hodnotí vyvinutou aplikaci.

Kapitola 1

Aktuální stav na poli doporučování

Kamkoliv v dnešní době na internetu zavítáme, máme velkou šanci, že na některý z doporučovacích systémů narazíme. Mohou mít různá jména:

- lidé, které byste mohli znát
- uživatelé kupující produkt X kupují též produkt Y
- položky podobné položce XY

Všechny mají ale stejný význam – zaujmout či upozornit na něco nebo někoho konkrétního. Spousta elektronických obchodů, renomovaných aukčních domů, ale též serverů se zábavou na nich doslova staví svá podnikání, neboť správně navržený a fungující doporučovací systém může firmě přinést výrazné zvýšení zisku.

Sami jejich zákazníci o doporučování stojí. Usnadňují jim navigaci po stránce a existuje vysoká pravděpodobnost, že v závislosti na ní budou přizpůsobovat i své chování. Uživatelská trpělivost ale není neomezená – pokud dostávají špatná doporučení, nenásledují je a ve výsledku odmítají celý systém.

1.1 Příklady systémů

Na internetu je k nalezení samozřejmě několik desítek – možná stovek – běžících systému, záměrně jsem se proto snažil vybral jen zlomek, který ale pokrývá většinu doporučovaného obsahu (zboží, zábava, text). Níže tedy uvádím jako příklady několik do reálného provozu úspěšně nasazených systémů.

1.1.1 Amazon.com

Amazon.com, Inc. [3] je jedním z nejstarších a největších internetových prodejců. Společnost začínala svůj provoz jako online knihkupectví, ale postupem let zařadila do své prodejní nabídky též hudební a filmové nosiče, software, elektroniku, nábytek a spoustu dalšího zboží.

Novinkou posledních let je vlastní spotřební elektronika v podobě čtečky elektronických knih a tabletů Kindle či poskytování služeb z oblasti cloud computingu.

Firmu lze řadit mezi průkopníky doporučování na internetu. Jako jeden z prvních internetových prodejců totiž začala svým zákazníkům doporučovat výrobky na základě nákupů jiných uživatelů.

Doporučovací systém je založen na několika zdrojích informací:

- porovnávání uživatelem prohlížených položek a položek umístěných ve virtuálním nákupním košíku s položkami, které se společně s těmito prohlíženými v minulosti často prodávaly [4].
- udržování informací ohledně hodnocení položek uživateli
- zaznamenávání historie nákupu (pokud uživatel v minulém měsíci zakoupil tři dětské knížky, znamená to, že má dítě?)
- spousta dalších postupů, jako například vyhodnocování demografických informací (dle doručovací adresy), zaznamenávání pohybu po stránce (jaké všechny položky a kolikrát si uživatel prohlédl, než umístil jednu konkrétní do nákupního košíku) nebo sledování prokliků [5] z cílených marketingových e-mailů s odkazy na zboží [4].

Společnost nazývá svou hlavní doporučovací strategii jako *item-to-item kolaborativní filtrování* a používá ji pro přizpůsobení prohlížení webu svým stálým zákazníkům. V tom smyslu, že fanoušek moderních technologií může při své návštěvě stránek nalézt odkazy na technologické novinky všeho druhu, zatímco mladá matka bude mít na těch samých stránkách v nabídce ve větším zastoupení dětské zboží.

Výše jsou popsány pouze základní principy. Doporučovací systém společnosti je samozřejmě velmi komplexní a detailní algoritmus je udržován jako obchodní tajemství. K nahlédnutí jsou ale patenty, např. *Personalized recommendations of items represented within a database* [23] nebo *Collaborative recommendations using item-to-item similarity mappings* [26].

[3] `http://www.amazon.com`

[4] affinity analysis – nacházení spojení mezi odlišnými položkami. Základním příkladem budiž vztah mezi šamponem a kondicionérem. Kupující je většinou používá v ten samý čas [11]. Při nákupu jednoho by mohl mít tedy zájem i o druhý.

[5] Jako proklik se označuje takové kliknutí na odkaz, které uživatele dovede na cílovou stránku [10].

1.1.2 Netflix

Netflix, Inc. [6] je společnost, která začínala nabízet své služby jako internetová videopůjčovna. Během posledních pár let (strategicky významný byl rok 2007, kdy byla nabídka rozšířena o filmy streamované prostřednictvím internetu [8]) se rozrostla v obrovskou mediální společnost nabízející obsah v podobě filmů a seriálů pro většinu v dnešní době používaných platforem jako PC, Mac, PlayStation3, Wii, Xbox a také mobilní telefony a tablety.

Vzhledem k tomu, že firma staví své podnikání na tom, že přicházející uživatelé platí za konzumaci zábavy (dle [7] pochází 2/3 zapůjčených filmů z předchozího doporučení), je v jejím vlastním zájmu, aby těmto uživatelům sledujícím filmy a seriály nabízela automaticky další obsahově či žánrově podobné, zkrátka takové, jenž budou co možná nejvíce lahodit jejich vkusu. Úspěšné podnikání společnosti je tak přímo závislé na tom, jak kvalitním doporučovacím systém společnost disponuje.

1.1.2.1 Netflix Prize

Za účelem zkvalitňování doporučování filmů byl vyvinut vlastní systém s názvem *Cinematch*. Potřeba neustálého zlepšování a zpřesňování doporučení vyústila ale v to, že společnost začátkem října 2006 vypsala soutěž známou jako *Netflix Prize*. Jednalo se o pokus ještě více pokročit na poli doporučování filmů a pro tým, který by dokázal zlepšit dosavadní výsledky systému Cinematch alespoň o 10 procent, byla vypsána odměna ve výši 1 000 000 dolarů.

K tomuto účelu společnost uvolnila testovací data obsahující ID uživatele, ID filmu, hodnocení na intervalu <1,5> a datum uskutečnění hodnocení. Testovací data obsahovala 100 480 507 hodnocení pro 17 770 filmů od 480 189 uživatelů. Uvolněna byla ještě další testovací data obsahující stejné informace, jen byla vynechána uživatelská hodnocení. Cílem úkolu pak bylo předpovědět tato chybějící hodnocení opět na intervalu <1,5>.

Cena byla udělena až v roce 2009 (do té doby docházelo k průběžnému zlepšování, ale nebylo dosaženo stanoveného zlepšení o 10 procent) týmu *BellKor's Pragmatic Chaos* (který vznikl spojením tří do té doby samostatných týmů). Vítězný tým použil k dosažení cíle technik strojového učení, aby při tom zjistil několik zásadních poznatků. Například to, že každé hodnocení filmu je silně subjektivní záležitostí, kterou je dopředu obtížné předpovědět. Ukázalo se také, že velmi záleží na tom, zda uživatel hodnotí právě dosledovaný film nebo film, který zhlédl již před delší dobou. Velkou roli též hraje nálada během dne a další faktory [1].

Výsledný algoritmus je navíc směsicí zhruba stovky menších algoritmů, takže by se dalo s trochou nadsázky prohlásit, že jednou z hlavních taktik je použít tolik doporučovacích algoritmů, kolik je jen možné.

[6] `https://www.netflix.com`

1.1.3 Zite

Zite.com [7] je moderní aplikace pro chytré mobilní telefony a tablety.

Mike Klass, CTO společnosti v jednom z článků [12] prozradil, že vizí bylo vyvinout sofistikovaný, na technikách strojového učení postavený systém, jehož účelem nebude pouhé filtrování příchozích novinek vedoucí k úspoře času uživatele. Cílem tohoto systému bylo přizpůsobit se svému uživateli studiem vzorců chování (pozoruje zájmy a návyky při čtení článků) natolik, aby mu dokázal doporučit přesně to, co v danou chvíli hledá, ale kvůli různým okolnostem (například obsah produkovaný neznámým bloggerem) by na to nikdy neměl šanci narazit.

Zite nasazené v systému uživatele je tedy stejně tak unikátní jako jeho uživatel. Sleduje, jaké článků si vybírá ke čtení, jak dlouhé tyto články jsou a jak dlouho stráví uživatel jejich čtením. Systém tedy každý den vyhodnocuje miliony nových článků, přičemž se zaměřuje na typ, klíčové atributy i na to, jakým způsobem je článek sdílen skrze web. Následně využije tyto informace ke spárování vybraných článků s osobním vkusem uživatele a tyto články mu doručí do aplikace.

Tak jako Netflix a Amazon doporučují filmy a produkty, které by na základě podobnosti mohli uživatele zajímat, stejně tak i Zite provádí na pozadí porovnání mezi čtenáři – jak v rámci aplikace, tak obecně na webu.

1.1.3.1 Zite & CNN & Flipboard

V srpnu 2011 o technologii vyjádřila zájem společnost CNN [2] a Zite přešlo pod ni. Poslední novinky ze začátku března 2014 ale naznačují [12], že Zite opouští CNN a rozhodlo se spojit síly s aplikací Flipboard [8] vývojářské společnosti Flipboard, Inc., jejíž funkcionalitou je agregování obsahu ze sociálních sítí a jiných stránek a následné nabízení ke čtení v magazínovém formátu umožňujícím jednoduchým způsobem listovat napříč vzájemně souvisejícími tématy.

Společným cílem je vystavět nejlepší systém pro personalizované čtení novinek.

1.1.4 Mendeley

Mendeley [9] je systém určený k doporučování vědeckých článků využívající jako svou výpočetní vrstvu technologii Apache Mahout [10]. Cílem systému je spojovat dohromady výzkumníky a jejich data. Svým uživatelům tak pomáhá v organizaci výzkumu, umožňuje jim nalézt potenciální spolupráci s dalšími uživateli aplikace a napomáhá též k objevům nových podnětů pro vlastní

[7] `http://zite.com`
[8] https://flipboard.com
[9] `http://www.mendeley.com`
[10] https://mahout.apache.org

práci. Uživateli této aplikace jsou přední světové university jako University of Cambridge, Stanford University, MIT či University of Michigan. Data pro aplikaci pocházejí z vlastních importů svými uživateli i z externích importů skrze různé katalogy prací.

Projekt samotný se v jedné ze svých prezentací [6] přirovnává k největší hudební databázi na internetu – last.fm [11]. Ta funguje na tom principu, že potenciální uživatel si nainstaluje na svůj počítač desktopovou aplikaci, následně s instalovanou aplikací začne poslouchat hudbu a tím je zahájeno automatické odesílání informace o skladbě (interpret, žánr apod.) na server last.fm. Podle těchto odeslaných dat jsou uživateli v budoucnu doporučovány další skladby.

Mendeley tuto analogii vysvětluje tím, že hudební knihovny jsou v jeho případě výzkumné knihovny, roli interpretů zde zastávají jednotliví výzkumníci, hudební skladby jsou pak jimi publikované články a jednotlivé hudební žánry reprezentují vědecké disciplíny.

Doporučení jsou zde generována dvojím způsobem.

Kolaborativní filtrování Používá se pro personalizované doporučení. Podporována je jak user-based, tak i item-based varianta doporučení.

Filtrování založené na obsahu Používá se k nalezení souvisejícího výzkumu, například nalezení článku ze stejné výzkumné kategorie nebo článku s podobným názvem.

Uživatelé mohou svůj zájem či nezájem o každou z doporučených položek vyjádřit zpětnou vazbou, která je dvojího typu:

Accept Vyjadřuje, že uživatel s daným doporučením souhlasí nebo pro něj vylo nějakým způsobem užitečné.

Remove Vyjadřuje nevhodné doporučení. Uživatel touto volbou dává najevo, že podobné doporučení by raději již příště nedostal.

1.1.5 Google News

Vlastní platformu pro doporučování obsahu má jako součást svého vývoje též společnost **Google, Inc.** [12]. Její výzkumníci se ve své práci [27] zabývali vývojem prediktivního systému pro personalizované doporučování zpráv na webu Google News [13]. Přihlášeným uživatelům, kteří mají v prohlížeči explicitně povolen záznam historie, jsou generována doporučení založená na zájmu těchto uživatelů vycházejících z profilů sestavených pozorováním chování na stránce.

K porozumění, jak se mění zájem uživatelů o zprávy v čase, napomohla výzkumníkům analýza logů (záznamy chování anonymních uživatelů na stránce).

[11] http://www.last.fm

[12] `http://www.google.com/about/company`

[13] `http://news.google.com`

Na základě této analýzy byl vyvinut Bayesovský framework pro předvídání zájmu uživatelů o novinky.

Kombinací mechanismu pro filtrování informací, který vzešel z předpovězených uživatelských profilů a již existujícího mechanismu využívajícího principů kolaborativního filtrování vznikl systém pro generování personalizovaných zpráv. Tato kombinace byla nasazena do živého provozu a následné experimenty prokázaly, že kombinováním metod došlo ke zlepšení kvality doporučování, což vedlo k častějším návratům na stránky.

1.1.6 Výzkum

Také současný výzkum v oblasti doporučovacích systémů, zdá se, nezahálí. Zde je výčet několika akcí, jejichž náplní či součástí je problematika doporučování a doporučovacích systémů.

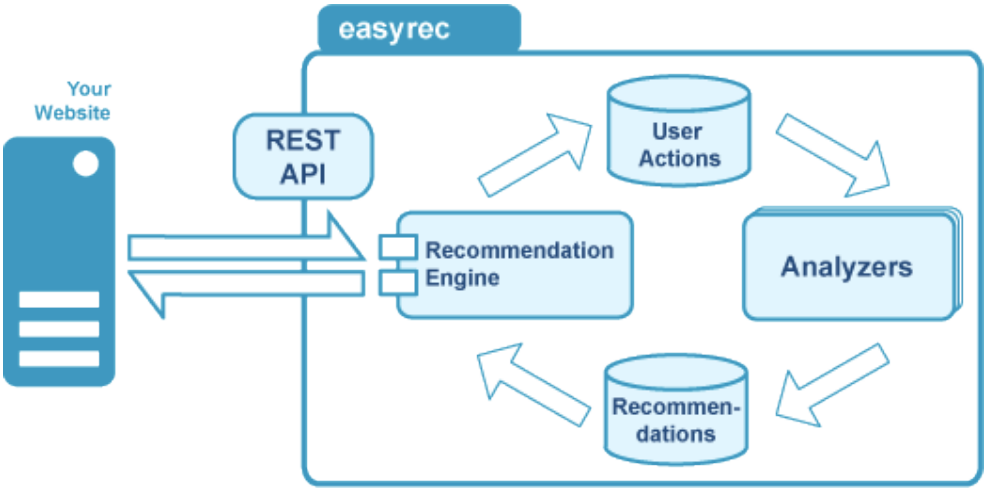
- **ACM RecSys conference.** [14] Tato konference je předním mezinárodním fórem pro prezentaci nových výsledků výzkumu a postupů na poli doporučovacích systémů. RecSys sdružuje hlavní mezinárodní výzkumné skupiny a též mnoho předních světových společností na trhu e-commerce. Nabízí také doprovodný program v podobě zvaných přednášek, konzultace týkající se problematiky RecSys a sympózium studentů doktorských programů.

  Zajímavé prezentace jsou volně dostupné na internetu, dá se tedy načerpat spousta inspirace do vlastního výzkumu. Mně osobně velmi zaujala prezentace [15] z posledního ročníku konference (Hong Kong, 2013), se kterou vystoupil jeden z klíčových řečníků Torben Brodt. V prezentaci popisuje tzv. *Open Recommendation Platform*, která je založena na výběru vždy toho nejlepšího algoritmu pro doporučení v závislosti na daném kontextu.

- **ICWSM: Weblog and Social Media.** [16] The International AAAI Conference on Weblogs and Social Media je mezinárodní konference, na které se střetávají výzkumní pracovníci z oblasti počítačových a společenských věd. Konference je pořádaná za účelem sdílení znalostí, diskutování o nápadech a výměny informací. Probíranými body jsou psychologické a sociální teorie, výpočetní algoritmy pro analýzu sociálních médií a jedním z mnoha témat jsou též doporučovací systémy.

- **ICML: Machine Learning.** [17] The International Conference on Machine Learning je konference s bohatou historií. Její první ročník proběhl

[14] `http://recsys.acm.org`
[15] `http://www.slideshare.net/d0nut/open-recommendation-platform`
[16] `http://www.icwsm.org/2014`
[17] http://icml.cc/2014



Obrázek 1.1: Abstraktní model open source systému easyrec. Zdroj: [3]

již v roce 1980 v Pittsburghu. Jedná se o přední mezinárodní konferenci zabývající se strojovým učením.

O tématu se též píše spousta článků a každým rokem vzniká několik disertačních prací. Dle dostupných informací z ACM RecSys Wiki [18] jich jen za poslední 4 roky bylo přes 50.

Také technická knihovna společnosti IBM obsahuje přehledný seznam [19] několika doporučovacích systémů, z nichž převážná část vznikla jakou součást univerzitního výzkumu.

Z tohoto seznamu se mi jako zajímavý systém jeví *easyrec* [20]. Jedná se o open source webovou aplikaci napsanou v programovacím jazyce Java nabízející personalizovaná doporučení prostřednictvím RESTful webových služeb. Skrz REST API je možné zasílat uživatelské akce jako prohlížení, nákup či hodnocení položky. Tyto uživatelské akce jsou poté ukládány do databáze easyrec, nad kterou je prováděna analýza a z té jsou generována doporučení pro aplikaci uživatele (viz obrázek 1.1).

1.2 Shrnutí poznatků

Z výše uvedených příkladů v sekci 1.1 lze vypozorovat určité společné vlastnosti a zformulovat několik závěrů:

- o uživateli je vedena spousta záznamů, například historie jeho chování na stránkách

[18] `http://www.recsyswiki.com/wiki/List_of_recommender_system_dissertations`
[19] http://www.ibm.com/developerworks/library/os-recommender2/index.html
[20] `http://easyrec.org/recommendation-engine`

- velký důraz je kladen na zpětnou vazbu (vyjádření zájmu uživatele o obsah stránky)
- stále se využívá osvědčených přístupů kolaborativního filtrování a filtrování na základě obsahu
- síla není v použití jednoho konkrétního algoritmu, ale v kombinaci více metod a přístupů dohromady
- většina výpočtů běží na pozadí v tom samém čase, co uživatel tráví prohlížením stránek, a je schopna se přizpůsobit okolnostem
- problematika má silnou závislost na strojovém učení a teorii pravděpodobnosti

Existujících řešení je tedy celá řada. Každé z těchto řešení je ale využíváno ke svému vlastnímu účelu a není vzájemně přenositelné. Navíc se zdá, že době, kdy obchodníkům stačilo na svůj elektronický obchod nasadit jednoduchý algoritmus kolaborativního filtrování, již odzvonilo a budoucnost patří systémům schopným přizpůsobit vývoj doporučování dle nějaké predikce, a to navíc v reálném čase.

Kapitola 2

Analýza a návrh řešení

Zatímco v předchozí kapitole jsem se zabýval popisem existujících řešení a zkoumal, jaké možnosti nabízejí současné doporučovací systémy, tato kapitola je celá věnována analýze požadovaného chování mnou vyvíjeného systému.

Nejprve se budu zabývat požadavky na systém, které vyplynuly z úvodních konzultací s vedoucím práce a vlastním zkoumáním problému. Se znalostí těchto požadavků bude možné provést hrubý návrh architektury systému včetně technického řešení součástí, kterými bude systém disponovat.

Prostřednictvím výstupů získaných v této kapitole budu schopen provést implementaci systému.

2.1 Požadavky

Za účelem vyšší přehlednosti jsem se nejprve rozhodl související požadavky strukturovat do zastřešujících skupin trojího typu (globální požadavky, rozhraní a systém). Nakonec se ale ukázalo, že mezi skupinami *rozhraní* a *systém* existuje velmi tenká hranice (z hlediska požadovaného chování, nikoliv implementačního), proto byly skupiny redukovány na dvě.

2.1.1 Globální požadavky na systém

- Systém bude v pravidelných intervalech ukládat do databáze svůj aktuální stav.
- Systém bude schopen v případě pádu aplikace a po jejím opětovném spuštění načíst naposledy uložený stav.
- Systém bude veškeré své operace týkající se doporučování provádět a vyhodnocovat v reálném čase (live read & live write).
- Systém bude na každou klientskou žádost vracet příslušnou odpověď (i chybovou) – neexistuje nic jako odpověď *null*.

- Systém bude klást důraz na zpětnou vazbu, podle které bude přizpůsobovat své chování.
- Systém bude zpětnou vazbu přijímat na principu odměny za dobré doporučení (přičítání v poměru) a trestu za špatné (odečítání v poměru).
- Systém bude schopen automaticky se přizpůsobovat vývoji v čase normalizováním ukládaných hodnoty (z důvodu ochrany proti přetečení datových typů).
- Bude existovat možnost zobrazit dosavadní průběh doporučování systémem v čase.

2.1.2 Systém & Rozhraní

- Uživatel bude mít možnost vytvářet skrze systémové rozhraní kolekce se seznamem algoritmů určených pro následnou predikci.
- Uživatel bude mít možnost zaslat systému žádost o radu a obdržet informaci o tom, který algoritmus pro doporučení obsahu by měl v danou chvíli zvolit.
- Uživatel bude mít možnost zaslat systému informaci o tom, jaký algoritmus pro doporučení se rozhodl si zvolit.
- Uživatel bude mít možnost zažádat si o doporučení obsahu prostřednictvím algoritmu, jenž mu byl navržen systémem.
- Uživatel bude mít možnost zaslat systému informaci o zpětné vazbě pro daný algoritmus, který si zvolil.
- Aplikace bude umožňovat prostřednictvím svého rozhraní zpracování informace, že jeden uživatel dělá něco s jedním dokumentem (reakce na zpětnou vazbu).
- Aplikace bude umožňovat prostřednictvím svého rozhraní mazání či neviditelnění pro doporučování již neaktuálních informací.

Dle disciplín softwarového inženýrství je výčet uvedený výše označován jako **funkční požadavky**.

Lze definovat i tzv. **nefunkční požadavky** specifikující vlastnosti a omezující podmínky kladené na systém:

- Systém bude postaven na platformě Java.
- Pro uchování informací o uživatelích, položkách a jejich vzájemné interakci bude využita platforma pro vyhledávání v textu Apache Solr.

- Systém bude umožňovat přístup aplikacím třetích stran prostřednictvím RESTful webových služeb (vychází ze zadání).
- Systém bude postaven tak, aby se dal snadno parametrizovat.
- Systém bude připraven na situaci, že jej bude využívat více uživatelů současně.
- Systém poběží jako samostatná aplikace na serveru naslouchající na volném TCP portu. Veškerá komunikace s ní bude probíhat formou zasílaných zpráv.
- Pro snadné nasazení a otestování systému na libovolné pracovní stanici bude nutné vytvořit Chef cookbook a bude jej možno zavést nástrojem Vagrant.

2.2 Adaptibilní systém

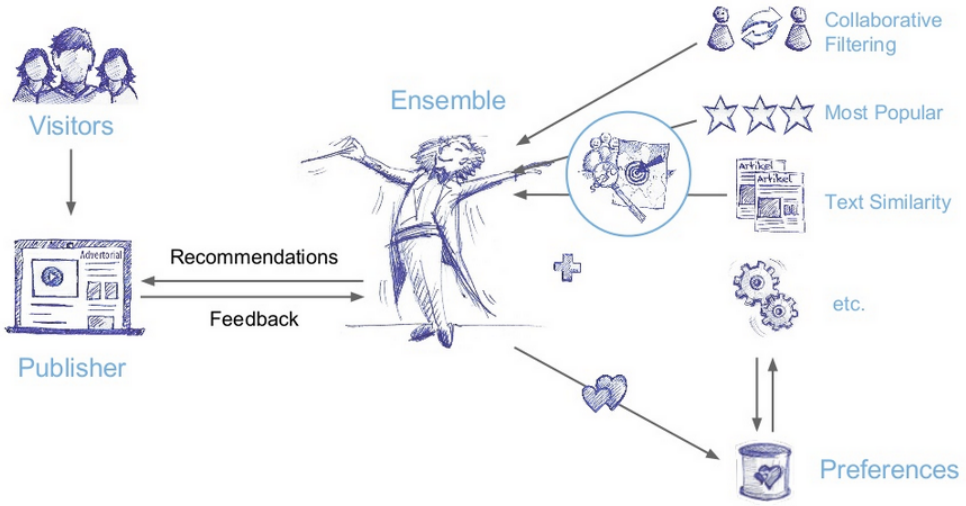
Pro implementaci systému splňujícího požadavky vzešlé z analýzy výše je třeba důkladně zvážit, jaké všechny komponenty bude tento systém obsahovat a případně jaké algoritmické postupy bude vhodné využít. Na základě zkoumání existujících implementací v kapitole 1 mě zaujal projekt *Open recommendation Platorm* (zkr. ORP) společnosti plista [21] prezentovaný na posledním ročníku konference ACM RecSys 2013 v Hong Kongu.

Motivací ORP bylo dosáhnout lepších výsledků kombinací více metod doporučování obsahu společně se zapojením kontextu uživatele (informacemi čerpanými převážně z HTTP hlaviček). Takovými informacemi mohou být:

- IP adresa, která může prozradit geologickou lokaci uživatele
- denní doba, kdy uživatel přistupuje ke stránce
- user agent informující o zařízení, ze kterého bylo k obsahu přistoupeno (mobilní telefon, PC)
- referer pro zjištění způsobu přístupu (přístup z vyhledávání nebo přímý přístup)

Abstraktní pohled na ORP znázorňuje obrázek 2.1, na kterém je systém zachycen jako *Ensemble*. Ensemble je schopen dle uživatelských preferencí vybírat z kolekce doporučovacích algoritmů ten nejvhodnější, jenž použije pro publikování obsahu. Konzumenti doporučeného obsahu (*Visitors*) zasílají poté systému zpětnou vazbu, díky které dojde k rekalkulaci preferencí, což ovlivní výběr budoucího algoritmu.

[21] `https://www.plista.com`



Obrázek 2.1: Abstraktní pohled na systém společnosti plista. Zdroj: [17]

V prezentaci bylo zmíněno několik užitečných rad ohledně toho, co všechno by měl systém podobného typu umět. Zmíněna je potřeba rychlého webového serveru, na kterém bude aplikace běžet, dále nutnost rychlého síťového protokolu a rychlé fronty zpráv. Padla zde též nutnost rychlého úložiště dat. K dosažení funkcionality pro výběr nejlepšího algoritmu za daných podmínek společnost používá tzv. *multi-armed bayesian bandit*, což byla i jedna ze strategií, které mi na úvodní schůzce doporučil vedoucí této práce.

Se spoustou rad z této prezentace jsem se ztotožnil a vyhodnotil, že s jejich pomocí by mohla být uspokojena velká část požadavků na systém stanovených výše.

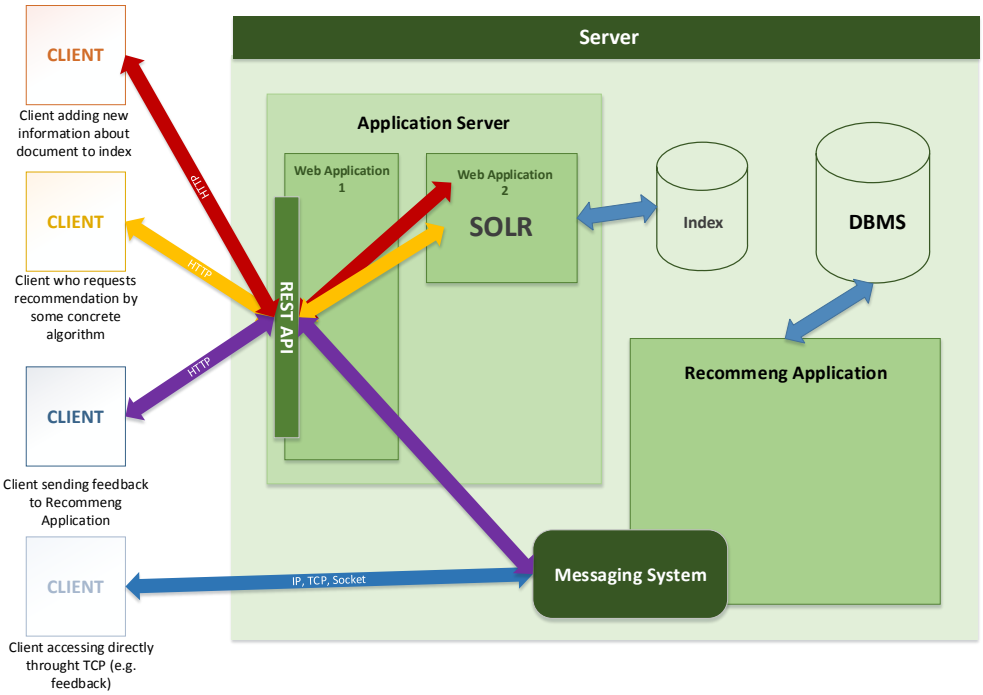
Ukázalo se, že strategie *multi-armed bayesian bandit* je pro mou práci natolik významná, že jsem se rozhodl věnovat jí samostatnou kapitolu 3.

2.3 Architektura systému

2.3.1 Server

Stěžejní serverovou komponentou je doporučovací platforma. Tu jsem pracovně nazval jako **Recommeng** (zkratka pro recommendation engine), abych o ní již nadále nemusel mluvit jako o *adaptibilním systému* či *systému pro kombinování metod*.

Diagram 2.2 znázorňuje abstraktní návrh architektury takové platformy. Mou snahou bylo zachytit přítomnost jednotlivých komponent systému, formu jejich vzájemné komunikace a též základní interakci uživatele se systémem.



Obrázek 2.2: Abstraktní návrh architektury a komponent adaptibilního systému, ve kterém je též vidět základní interakce uživatele se systémem.

V rámci ucelenosti zde uvádím přehled všech komponent nacházející se na serverové straně.

2.3.1.1 Aplikační server

Na aplikačním serveru poběží dvě aplikace.

Webová aplikace s rozhraním pro snadnou komunikaci s klienty. (viz diagram 2.2, komponenta *Web Application 1*) Aplikace bude mít na starosti obsluhu klientských požadavků na systém prováděných prostřednictvím protokolu HTTP. Zároveň bude disponovat sadou algoritmů, které lze použít pro doporučení obsahu. Dále bude aplikace obstarávat komunikační režii s druhou aplikací běžící na aplikačním serveru (*Web Application 2*), kterou je platforma pro vyhledávání v textu – Apache Solr.

Apache Solr (viz diagram 2.2, komponenta *Web Application 2*) Přítomnost této aplikace vychází z nefunkčních požadavků. Apache Solr funguje jako samostatná komponenta umožňující fulltextové vyhledávání (detaily v kapitole 4).

2.3.1.2 Databáze

Kvůli potřebě ukládání průběžného stavu aplikace (časové snímky) a snadného obnovení v případě přerušení běhu systému je nutné zapojit do návrhu databázi. Database Management System (DBMS) hraje důležitou roli i při škálování aplikací. V minulosti tolik používané standardní relační DBMS mohou způsobovat zpoždění při provádění čtení/zápisu a v některých případech hrát roli úzkého hrdla aplikace (bottleneck). Ohledně tohoto problému by možná stálo za úvahu prozkoumat možnosti použití NoSQL databází, které jsou již ze svého principu navrženy pro spolupráci s aplikacemi se zaměřením na výkon a škálovatelnost.

2.3.1.3 Reccomeng Application

Komunikaci s aplikací bude umožňovat systém pro zprávy (Messaging System) s využitím fronty zpráv (Message Queue). Toto řešení je nasnadě kvůli očekávání většího množství žádostí směřujících na systém a snadnější škálovatelnosti aplikace. Obecný příklad takového MQ systému znázorňuje obrázek 2.3

Producenti zpráv (klienti využívající aplikaci) vytvářejí zprávy, které jsou zasílány do fronty. Zprávy jsou z fronty postupně odebírány konzumenty, kteří mají jejich zpracování na starosti. Výhodou je časová nezávislost mezi producenty zpráv a jejich konzumenty – producent tak může zasílat zprávy i tehdy, když žádný z producentů není k dispozici. Zprávy pak existují ve frontě, dokud se o jejich zpracování konzument nepřihlásí.

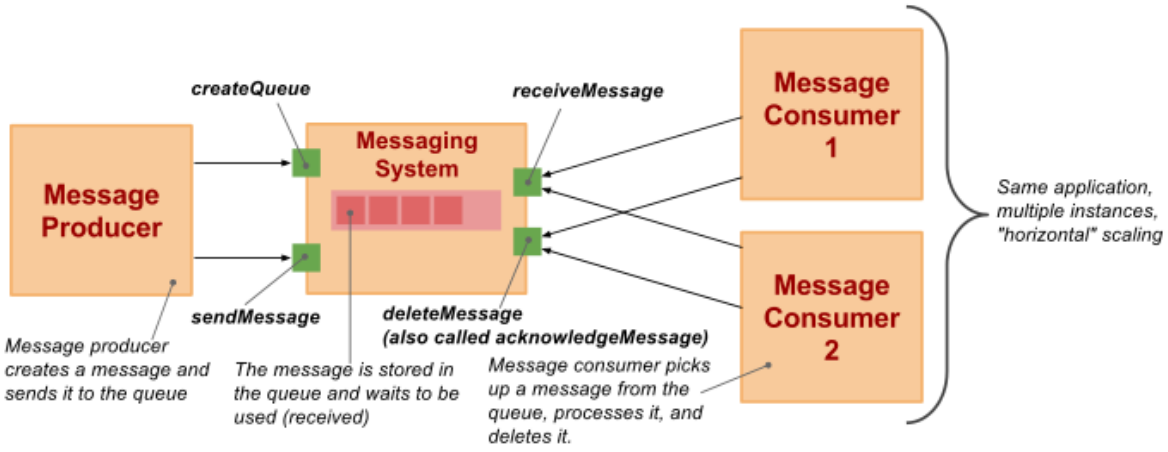
Aplikace taktéž komunikuje s databází, ale využívá ji pouze pro ukládání informací o stavu, které je prováděno v pravidelných časových intervalech, respektive pro načítání posledního uloženého stavu při startu aplikace. Důvodem pro vynechání komunikačního mezikroku s databází je požadavek na rychlost systému – veškerá data pro výpočet a predikci budou udržována přímo ve vnitřní paměti (uvnitř JVM).

2.3.2 Klient

Klientská strana není nikterak složitá. Potenciální uživatel aplikace má v zásadě dvě možnosti, jak s platformou komunikovat:

- jen a pouze zasíláním žádostí na REST API
- kombinací zasílání žádostí na REST API společně s přímou komunikací s Recommeng systémem prostřednictvím fronty zpráv

Některé ze systémových funkcionalit nelze bez komunikace s REST API využívat. Do této kategorie spadá například přidání nového vztahu k položce ve fulltextovém indexu nebo zaslání žádosti o doporučení obsahu některým z podporovaných algoritmů.



Obrázek 2.3: Příklad MQ systému s producentem zpráv a konzumenty. Zdroj: [29]

Přímé spojení s platformou skrze frontu zpráv bez nutnosti zapojení prostředníka v podobě REST API by ale měly umožňovat všechny možnosti použití Recommeng systému. Jmenovitě vytváření kolekcí se seznamem algoritmů, dále zasílání žádostí o radu pro výběr algoritmu a též metody informující systém o uživatelském chování jako zvolení konkrétního algoritmu nebo zaslání zpětné vazby o doporučení.

V diagramu 2.2 jsem se snažil zachytit různé formy komunikace klienta se systémem. Pro lepší názornost jsou jednotlivé žádosti barevně odlišeny. Ty lze považovat za modelové případy užití inspirované systémovými požadavky.

2.3.2.1 Červená varianta

Aneb klient přidávající nový vztah mezi položkou a jejím uživatelem do úložiště.

- klient zašle žádost na rozhraní
- aplikace (Web Application 1) za rozhraním se spojí s indexem
- v případě zdárného průběhu se přidá do indexu informace o tom, že 1 uživatel dělá něco s 1 dokumentem
- klient obdrží odpověď s výsledkem žádosti

2.3.2.2 Fialová varianta

Aneb klient zasílající platformě zpětnou vazbu ohledně doporučení, které dostal.

Fialová varianta může též představovat situaci, kdy klient žádá systém o radu, kterým algoritmem si má nechat doporučit obsah ve své budoucí žádosti o doporučení (viz žlutá varianta 2.3.2.3).

- klient zašle žádost na rozhraní
- aplikace (Web Application 1) za rozhraním se pokusí přistoupit k frontě zpráv
- pokud fronta existuje, žádost je předána do fronty pro zpracování konzumentem (aplikace Recommeng)
- konzument zpracuje žádost a zasílá odpověď, kterou systém zpráv předá zpět do aplikace
- klient obdrží odpověď s výsledkem žádosti

2.3.2.3 Žlutá varianta

Aneb klient poptávající doporučení obsahu konkrétním algoritmem.

Toto doporučení klient poptává na základě odpovědi na žádost, která mu byla udělena systém (viz fialová varianta 2.3.2.2)).

- klient zašle žádost na rozhraní
- aplikace (Web Application 1) za rozhraním zvolí vybraný algoritmus pro doporučení
- algoritmus přistoupí k datům v indexu, nad kterými provede doporučení
- klient obdrží odpověď s výsledkem žádosti

2.3.2.4 Modrá varianta

Aneb klient, který se rozhodl nevyužít možnosti komunikovat prostřednictvím REST API. Svou žádost zasílá přímo do fronty zpráv.

- klient zašle žádost do fronty
- pokud fronta existuje, žádost je předána do fronty pro zpracování konzumentem (aplikace Recommeng)
- konzument zpracuje žádost a zasílá klientovi odpověď

2.3.3 Komunikace

Výše jsme měli možnost poznat dva způsoby komunikace uživatele s doporučovací platformou. Dalším úkolem je navrhnout formát, jakým si budou vyměňovat producent s konzumentem data prostřednictvím fronty zpráv za účelem vzdáleného volání metod Recommeng systému.

Vzhledem k obecné známosti protokolu HTTP jsem se rozhodl napodobit jeho chování a zachovat sémantiku:

- metoda (method)
- cesta (path)
- tělo zprávy (body)

Podobně jako nabízí HTTP, na každou žádost ve tvaru *method, path* a *body* přijde odpověď ve tvaru *status* a *body* s využitím různých návratových kódů pro stav (status).

2.4 Rozhraní

Platforma poskytuje API pro interakci s daty formou RESTful (HTTP implementace REST). Každý uživatel tak může interagovat s rozhraním prostřednictvím veřejných REST endpointů.

Z prováděné analýzy vyplynuly požadavky na aplikační rozhraní takové, že je lze rozdělit do tří skupin dle podstaty úkolu, který mají plnit. K dispozici je diagram 2.4, který by měl celou věc osvětlit.

2.4.1 Rozhraní pro Recommeng

Komunikace s rozhraním by měla umožňovat vytvářet v Recommeng aplikaci kolekce s identifikátory algoritmů, jenž budou zapojeny do predikce.

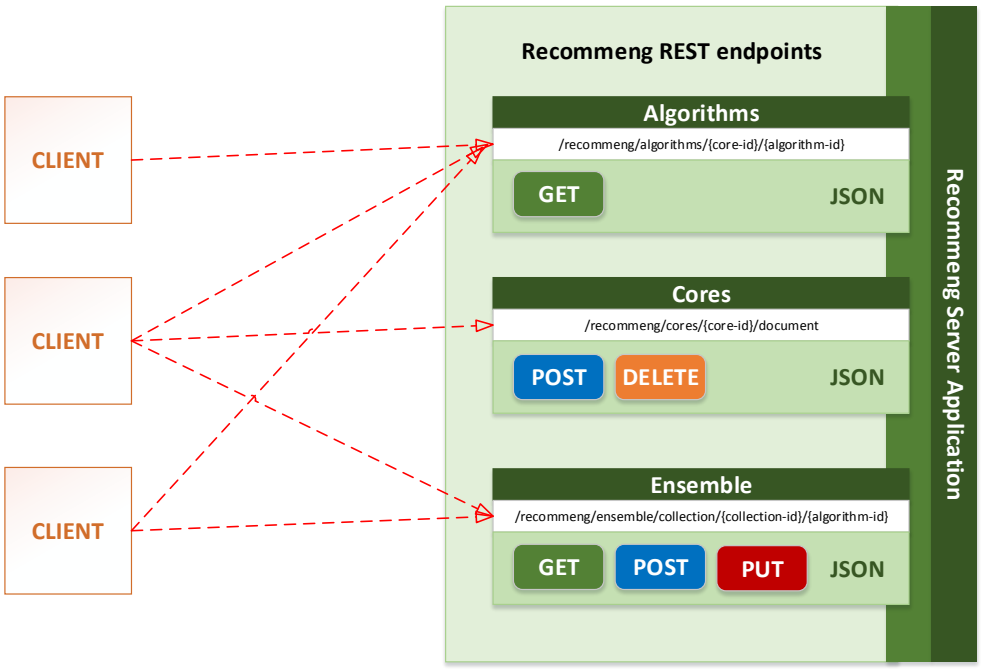
K tomu účelu bude sloužit endpoint:

```
/recommeng/ensemble/collection
```

Pokud uživatel zašle metodou POST žádost na tento endpoint (obsahující identifikátor kolekce a seznam algoritmů), v případě neexistence v systému bude kolekce vytvořena a připravena k predikci.

Dále by měla existovat možnost zaslat na kolekci prosbu o predikci nejlepšího jejího algoritmu pro doporučení.

```
/recommeng/ensemble/collection/{collection-id}
```



Obrázek 2.4: Ukázka REST endpointů navrhovaného adaptibilního systému

Toho dosáhneme žádostí s metodou GET, uvedením identifikátoru kolekce a případným specifikováním požadovaného výstupu (zda chceme vrátit jen nejlepší možnost či všechny možnosti seřazené sestupně od nejlepšího) v parametru dotazu.

Kvůli učení a vývoji znalostí systému je nezbytně nutné mít možnost zaslat informaci o výběru konkrétního algoritmu a též zpětnou vazbu vyjadřující, jak byl žádající uživatel s navrhovanou variantou spokojen.

```
/recommeng/ensemble/collection/{collection-id}/{algorithm-id}
```

Na endpoint výše v takovém případě zašleme metodou PUT upřesňující informaci o operaci, kterou chceme systému sdělit (výběr vs. zpětná vazba).

2.4.2 Rozhraní pro sadu základních algoritmů

Bude se jednat o jednoduchý endpoint, který si při žádosti vystačí s metodou GET.

```
/recommeng/algorithms/{core-id}/{algorithm-id}
```

Úkolem je zpracovat žádost uživatele o doporučení algoritmem, jehož identifikátor je uveden v cestě. Hledání doporučení bude provedeno nad dokumenty

ze specifikovaného indexu (identifikátor indexu je též nutno uvést). V parametrech dotazu lze specifikovat další vstupy pro doporučení jako limit vrácených výsledků, identifikátor pro doporučení článků z té samé skupiny a další.

2.4.3 Rozhraní pro úložiště dat

Důležité rozhraní umožňující zaznamenat chování uživatele vzhledem ke sledovaným položkám.

```
/recommeng/cores/{core-id}/document
```

Prostřednictvím metody POST budou zasílány veškeré informace o položce (u článku např. identifikátor, text, skupina, datum publikace) a jejím uživateli (identifikátor), která má být uložena do fulltextového indexu. Druhou podporovanou metodou je metoda DELETE pro vyřazení položky z doporučování. Metoda je použita v žádosti na týž endpoint jako metoda POST, pouze je nutné v parametru dotazu specifikovat identifikátor článku (z toho důvodu, že identifikátorem článku může být cokoliv, například URL adresa).

2.5 Sada algoritmů

Identifikace základní sady algoritmů určených pro kombinování je jeden z požadavků na systém plynoucí přímo ze zadání.

Uživatel žádající o radu pro výběr nejlepšího algoritmu obdrží od systému identifikátor reprezentující tento algoritmus.

Algoritmy pro doporučování vyhodnocují především informace o uživateli, položkách a hodnocení. Ukládány jsou ale i další údaje, například datum zveřejnění položky či její popis (u článku se může jednat o text zprávy). Také s těmito informacemi mohou doporučovací algoritmy pracovat.

2.5.1 Algoritmus náhodného výběru

Jak je patrné již z názvu, tento algoritmus vybírá položky pro doporučení naprosto náhodným způsobem. Jeho hlavním úkolem je být zde pro srovnání s ostatními algoritmy.

2.5.2 Algoritmus výběru dle nejnovějších položek

Doporučování dle nejnovějších položek je dalším z algoritmů s naivním přístupem k problému. Položky budou v tomto případě doporučovány sestupně dle zveřejněného data.

2.5.3 Algoritmus výběru nejlépe hodnocených položek

Jedná se o první algoritmus založený na složitějším výpočtu. Položky vzešlé z doporučení budou dle určitých parametrů nějakým způsobem lepší než ostatní, které se v doporučení neobjevily. Takovým parametrem může být například hodnocení na škále od 1 do 5, počet pozitivních hodnocení, souhrnné číslo udávající počet přečtení článku nebo jiný z mnoha způsobů vyjádření zájmu o položku.

2.5.4 Algoritmus výběru dle podobnosti obsahu

Algoritmus se snaží na základě podobnosti obsahu nalézt pro položku několik jí podobných položek z databáze. Podobnost se určuje porovnáním jednotlivých parametrů, například tagů, nadpisů nebo celého textu článku.

2.5.5 Algoritmus kolaborativního filtrování

Algoritmus je založen na modelu dřívějšího chování uživatele v systému. Model je většinou konstruován z chování většího množství uživatelů s podobným vkusem. V podstatě lze říci, že doporučení jsou založena na automatické spolupráci více uživatelů a výběru těch, kteří mají co nejpodobnější preference či chování.

Rozlišují se dva hlavní způsoby filtrování.

2.5.5.1 User-based

"You may like it because your friends liked it." [20]

Aneb filtrování založené na uživatelích. Jedná se o starší variantu kolaborativního filtrování. Podstatou je vzít na základě určité podobnosti skupinu (zdroj [20] udává cca 20 až 50) uživatelů s podobných vkusem jako má uživatel, pro něhož je doporučení konstruováno, a poté předpovědět, jak moc zajímavá by pro uživatele byla pro něj neznámá položka, se kterou jsou spojení uživatelé se stejným vkusem.

2.5.5.2 Item-based

"You tend to like that item because you have liked those items." [20]

Aneb filtrování založené na položkách, které použila v roce 2001 jako první společnost Amazon. Myšlenka je taková, že uživatel, který si v minulosti zakoupil nějakou položku, bude v budoucnu při dalším nákupu vyhledávat položku podobnou. Například předpověď toho, co si uživatel zakoupí v budoucnu, lze uskutečnit analýzou uživatelovi historie nákupů [14].

Kapitola 3

Adaptibilní systém

3.1 Minimální teoretický základ

Ještě předtím, než se pustím do popisu strategie, která bude použita v implementaci systému, zopakuji zde některé pojmy týkající se strojového učení, teorie pravděpodobnosti a statistiky. Jedná se o minimální základ potřebný k pochopení postupů popisovaných dále v této sekci.

3.1.1 Strojové učení

Strojové učení je vědecká disciplína (jedna z větví oboru umělé inteligence), jež se zabývá tím, jak se má počítač přizpůsobit určité situaci, aniž by byl pro danou situaci explicitně naprogramován. Detailněji v sekci 3.2.

3.1.2 Agent

Agent je speciální autonomní program, který jedná samostatně a nezávisle bez vedení uživatele. Jeho úkolem je komunikovat s okolím a interagovat v závislosti na okolních podnětech. Dalšími důležitými vlastnostmi jsou schopnost příhodně reagovat na danou situaci a též proaktivně vykonávat činnost a dosahovat cíle prostřednictvím vlastní iniciativy. Jedná se o definici obecnou, ale pro naše potřeby zcela dostačující. Problematika agentů a agentních systémů je jinak samostatný obor spadající do oblasti umělé inteligence a jeho zkoumání by vydalo na další samostatnou závěrečnou práci.

3.1.3 Zpětná vazba

Zpětnou vazbou je označován proces, ve kterém na základě obdržených informací (například při komunikaci s nějakým systémem) a reakcí na ně ovlivňujeme jejich budoucí podobu – část systémového výstupu lze použít jako vstup pro další činnost systému. Rozlišujeme dva typy reakcí, a to kladnou zpětnou

vazbu a zápornou zpětnou vazbu. Téma má blízko k psychologickému zkoumání toho, jakým způsobem dokáže ovlivnit zapojení odměny a trestu lidské chování.

3.1.4 Základy teorie pravděpodobnosti

Nejprve uvedu několik pojmů z teorie pravděpodobnosti, se kterými budu v následujících podsekcích pracovat.

Pravděpodobnostní prostor prováděného náhodného pokusu je tvořen trojicí $(\Omega, \mathcal{F}, P)$, kde:

- Ω je prostor elementárních jevů (např. čísla od jedné do šesti na šestistranné hrací kostce), kde elementárním jevem nazýváme libovolný možný výsledek $\omega \in \Omega$
- $\mathcal{F}$ je množina náhodných jevů (potenční množina [22] množiny Ω)
- P je přiřazení pravděpodobnosti jednotlivým jevům z Ω ($P(\Omega) = 1$)

3.1.4.1 Náhodný jev

Náhodný jev A je popisován jako výsledek náhodného pokusu. Příkladem jevu může být například hod mincí a pozorování, zda padla panna nebo orel. V tomto jevovém prostoru jsou zahrnuty celkem dva elementární jevy (hodnota panna a hodnota orel), jenž souvisejí s pokusem (pozorování výskytu elementárních jevů během házení).

3.1.4.2 Náhodná veličina

Jak vidno, výsledkem náhodného pokusu nemusí být číslo (v příkladu výše jsme měli dvě hodnoty po stranách mince), proto je vhodné těmto výsledkům kvůli matematickému zpracování čísla přiřazovat. Způsob přiřazení čísla výsledku náhodného pokusu se označuje jako *náhodná veličina* X. Pro počítání padlých panen při opakovaném házení mincí by mohlo přiřazení vypadat například takto:

- $X(panna) = 1$
- $X(orel) = 0$

Náhodná veličina na pravděpodobnostním prostoru $(\Omega, \mathcal{F}, P)$ je tedy funkce $X : \Omega \to R$, která každému $\omega \in \Omega$ přiřadí $X(\omega)$ a pro kterou platí podmínka měřitelnosti:

$$\{X \leq x\} = \{\omega \in \Omega : X(\omega) \leq x\} \in \mathcal{F}, \forall x \in R$$

[22] Potenční množina množiny X je množina obsahující všechny podmnožiny množiny X

Dle typu se rozlišují náhodné veličiny diskrétní a spojité. Diskrétní náhodné veličiny mohou nabývat pouze konečného počtu hodnot z R (například ročník studia), zatímco spojité náhodné veličiny nabývají v určitém intervalu libovolné reálné hodnoty (například čas potřebný k dokončení diplomové práce).

Pravděpodobnostní rozdělení náhodné veličiny určuje její distribuční funkce [16].

3.1.5 Distribuční funkce

Distribuční funkce náhodné veličiny X je funkce $F : R \to \langle 0, 1\rangle$ definovaná vztahem

$$F(x) = P(X \leq x) = P(\{\omega \in \Omega : X(\omega) \leq x\}).$$

Vyjadřuje tedy pravděpodobnost, že hodnota náhodné veličiny X nabude hodnoty menší nebo rovné zadané hodnotě (libovolnému $x \in R$). Distribuční funkce určuje rozdělení pravděpodobnosti.

3.1.6 Kvantilová funkce

Podobně jako distribuční funkce, i kvantilová funkce se týká rozdělení pravděpodobnosti. Lze ji považovat za funkci inverzní k distribuční funkci, neboť zatímco distribuční funkce $y = F(x)$ vyjadřuje pravděpodobnost, s jakou bude hodnota náhodné veličiny X menší nebo rovna $x \in R$, výsledkem kvantilové funkce $x = F(y)^{-1}$ je hodnota x, pro jakou je výsledek náhodného pokusu se zadanou pravděpodobností y menší nebo roven x. Jinými slovy hledáme taková x, kterým odpovídá určitá hodnota distribuční funkce $F(x)$. Hodnoty této funkce jsou tedy *kvantily* [5].

3.1.7 Rozdělení pravděpodobnosti

Někdy se též označuje jako distribuce pravděpodobnosti náhodné veličiny X. Rozdělení pravděpodobnosti každému jevu popsanému veličinou X přiřazuje určitou pravděpodobnost. V diskrétním případě přiřazujeme pravděpodobnosti jednotlivým hodnotám (lze si představit jako samostatné body v grafu), ve spojitém případě pak intervalu hodnot náhodné veličiny.

Rozdělení pravděpodobnosti je celá řada, z diskrétních je známé například binomické (n pokusů s rovnocennou pravděpodobností) či geometrické rozdělení. Ze spojitých například normální rozdělení, exponenciální rozdělení nebo beta rozdělení.

3.1.8 Beta rozdělení

Beta rozdělení ($Beta(\alpha, \beta)$) je spojité pravděpodobnostní rozdělení definované na intervalu $\langle 0, 1\rangle$. Rozdělení má dva vstupní parametry α a β určující tvar.

Rozdělení se používá k modelování chování náhodných veličin, které jsou omezené na konečné intervaly.

3.1.9 Bayesovská statistika

Jedná se o moderní větev statistiky pracující s podmíněnou pravděpodobností. Základem bayesovské statistiky je známý Bayesův teorém (často označovaný jako Bayesova věta) vyjadřující pravděpodobnost hypotézy (H_j) v závislosti na datech (D) a případně modelu (M). Lze pomocí ni stanovit pravděpodobnost, aniž bychom měli známá fakta z minulosti (oproti klasickému statistickému přístupu). Oproti klasické statistice taktéž netestujeme hypotézy, ale provádíme odhady.

3.1.9.1 Apriorní pravděpodobnost

Pravděpodobnost $P(H_j|M)$, označována jako *prior*. Značí to, co víme nebo si myslíme předem, ještě před získáním dat (např. výsledky předchozích experimentů) [35]. Lze jí vyjádřit určitou míru nejistoty, například podíl voličů, kteří budou v budoucích volbách hlasovat pro nějakého konkrétního politika.

3.1.9.2 Aposteriorní pravděpodobnost

Pravděpodobnost $P(H_j|D, M)$, označována jako *posterior*. Udává výsledek celého snažení, tedy pravděpodobnost naší hypotézy v závislosti na předchozích znalostech (prior) a současně nových datech [35].

Díky uvedeným pravidlům je možné s každou další objevenou skutečností zpřesňovat pravděpodobnost výchozí hypotézy.

3.1.10 Náhodný výběr

Náhodného výběru se využívá k rozpoznání charakteru rozdělení z toho, že opakované pokusy dávají za stejných podmínek různé výsledky odpovídající hodnotám jednotlivých realizací náhodné veličiny. Jedná se o uspořádanou n-tici $(X_1, X_2, \ldots, X_n)$ náhodných veličin X_i, $1 \leq i \leq n$, které jsou nezávislé a mají stejné rozdělení pravděpodobnosti [9].

Zatímco **náhodným výběrem** označujeme n-prvkovou posloupnost nezávislých náhodných veličin $X_1, X_2, \ldots, X_n$, pojmem **výběr** budeme značit n-prvkovou posloupnost reálných čísel $x_1, X_2, \ldots, X_n$ [31].

3.1.11 Statistická inference

Při statistickém usuzování uvažujeme pojem *populace*, kdy populací myslíme náhodnou veličinu s jejím rozdělením pravděpodobnosti. Úkolem statistické inference je pak s použitím **výběru** z populace *odhadnout parametr*, což je

číselná hodnota platící pro celou populaci (může jí být například střední hodnota, rozptyl a tak podobně). Odhadem je myšleno právě získání číselné hodnoty nebo intervalu hodnot z výběru. Cílem je, aby měl odhad blízko skutečné hodnotě parametru.

Rozlišujeme dva typy odhadů, a to bodový, kdy odhadem je jedna hodnota, a intervalový, kdy je odhadem interval hodnot [31].

3.1.12 Multi-armed Bandits

Jedná se o jeden z klasických učících problémů zasahující do teorie pravděpodobnosti. Herní strategie je podobná filosofii tradičního výherního automatu (představujícího *one-armed strategii*) s tím rozdílem, že multi-armed varianta má více herních pák, a proto lze v každém kole pro jednu hru volit mezi více automaty. Detailněji v sekci 3.3.

S pojmy vysvětlenými výše souvisí strategie v tom smyslu, že pokud je použito bayesovské varianty, prováděním výběru z konečného počtu rozdělení lze maximalizovat průměrnou hodnotu z hodnot, které dostáváme.

Toho lze využít například k cílenému doporučování reklamy nebo výběru personalizované úvodní stránky pro uživatele vstupujícího na naše stránky.

3.2 Význam strojového učení v návrhu systému

Jak vyplynulo z rešerše existujících řešení 1 a analýzy 2, vzhledem k povaze řešeného problému je nutné zaměřit se na metody *strojového učení*.

Těmito metodami učení lze dosáhnout generalizace vstupních instancí na správné výsledky nebo adaptovat existující systém na změny okolí. Používají se ve všech oblastech informačních věd od analýzy snímků, analýzy DNA a textu, až po simulace chování člověka.

Typickým příkladem je vytvořit z dostupných dat model, který dokáže:

- predikovat cenu akcií za 6 měsíců z aktuální výkonnosti společnosti a ekonomických dat
- rozpoznat spam od regulérního e-mailu
- u pacienta hospitalizovaného s infarktem predikovat riziko dalšího infarktu
- napomoci společnostem zabývajících se internetovou reklamou v rozhodování se, kterou reklamní strategii použít k maximalizaci zisků

Algoritmy strojového učení dělíme do taxonomie (nadtříd a podtříd) založené na požadovaném výsledku nebo typu vstupu, jenž je k dispozici během trénování stroje. Algoritmů je celá řada, bude tedy vhodné zmínit zde alespoň ty nejtypičtější.

Supervised learning [23] je typ učení, které se používá v případě, že máme k našim trénovacím instancím korektní výsledky. Pomocí kombinace trénovacích vstupních instancí a jejich požadovaných výsledků lze systém naučit na situaci, aby dokázal automaticky předpovídat výsledek pro každý další platný vstup. Využití nachází například v oblastech rozpoznávání řeči či detekci spamu. [34]

Unsupervised learning [24] je již ze samotné podstaty absence informace od učitele obtížným problémem. Učení bez učitele se používá pro analýzu pozorování (nebo dat), když není k dispozici informace od učitele, tj. trénovací multimnožina. Pozorovaná data se mají vysvětlit pomocí matematického modelu. Používá se v oblasti rozpoznávání vzorů. [22]

Reinforcement learning [25] je oblast informatiky tykající se chování agentů. Jedná se o metodu, při které se agent učí, jakým způsobem má volit akce, aby nalezl optimální strategii pro dané prostředí. Jedná se o učení bez učitele. Agent sice dostává odezvu, ale přímo z prostředí, takže musí experimentovat a zjišťovat, které stavy jsou nějakým způsobem dobré, a kterým stavům je lepší se vyhnout. Průzkum probíhá na principu zpětné vazby v podobě odměny za akce dosahující cíle nebo trestu v opačném případě. Řeší se zde problém explorace vs. exploatace 3.3.2.

Za nejvhodnější algoritmus, který by byl schopen plnit požadavky definované na tuto práci, jsem zvolil algoritmus zpětnovazebního učení.

Rozlišujeme několik typů zpětnovazebního učení, například tzv. *single-stage* (agent se snaží uplatňovat zpětnou vazbu ihned po každé provedené akci), oproti kterému stojí typ *sekvenční* (agent uplatňuje zpětnou vazbu po obdržení série akcí). Dalšími typy jsou pak například *pasivní* a *aktivní* zpětnovazební učení přizpůsobující svůj vývoj na základě pevně dané strategie, respektive učení se a rozhodování o prováděných akcích za chodu systému [24].

Algoritmus zpětnovazebního učení začíná při svém spuštění ve stavu nevědomí, kdy neví nic o daných okolnostech a začíná nabývat své vědomosti postupným testováním systému. Postupující dobou běhu (a tím, jak vstřebává data a vyhodnocuje výsledky) se učí rozpoznat, jaké chování je nejlepší.

3.2.1 Online učení

Navrhovaná strategie je též nazývána jako *online učení*. Nutno zmínit, že slovem online zde není míněno něco ve smyslu internetu, ale ve smyslu neustále se vyvíjející aktualizace dat. Učící algoritmus v každém kole vykoná nějakou akci, přijme zpětnou vazbu a připíše si daný zisk či ztrátu.

[23] učení s učitelem
[24] učení bez učitele
[25] zpětnovazební učení nebo též učení posilováním

Z matematického hlediska má online učení propojení na klasické online algoritmy, teorii (opakovaných) her a teorii pravděpodobnosti. Díky těmto znalostem tak můžeme navrhovat pravděpodobností dynamické systémy, kterými lze modelovat složitá průmyslová zařízení nebo třeba výherní automat známý jako mnohoruký bandita (Multi-Armed Bandit).

3.3 Multi-armed Bandits algoritmus

3.3.1 Princip algoritmu

Základ algoritmu si lze představit tak, že hráč stojí před N výherními automaty (ty jsou podle dle strategie nazývány jako bandité) a v každém kole má možnost vybrat si jeden, na kterém bude hrát.

Strategii je možné formálně popsat jako skupinu výnosových distribučních funkcí $B = \{A_1, A_2, \ldots, A_N\}$, kde N je počet banditů (každý z banditů má tedy přiřazenu právě jednu distribuční funkci vyjadřující pravděpodobnost úspěchu).

Hráč nemá zpočátku žádnou informaci o průběhu hry ani o rozložení pravděpodobnosti úspěchu mezi bandity a maximalizace výhry může dosáhnout pouze tím, že v každém kole vybírá vždy jednoho z banditů.

Kdyby hráč věděl, u kterého z banditů je největší pravděpodobnost výhry, samozřejmě by vždy vybíral právě tohoto. Pravděpodobnosti výher u jednotlivých automatů jsou ale neznámé, úkolem hráče tedy bude nalézt nejlepšího banditu, a to tak nejrychleji, jak jen to je možné.

Návrh strategie je o tom, že jednotliví bandité jsou nejdříve testování za účelem získání nutných znalostí o systému. Poté je už možné zaměřit se na páky, které poskytují díky zužitkovaným znalostem největší odměnu.

Úkol je komplikován stochastickou povahou banditů. Suboptimální bandita může vracet spoustu výher, což by nás mohlo přimět uvěřit, že právě tento štědrý bandita je pro nás tím nejlepším. Podobně ale naopak nejlepší bandita může zpočátku dávat spoustu proher.

Na místě jsou dvě otázky:

- Měli bychom dávat stále šanci i banditům, u kterých často prohráváme, nebo na ně zapomenout a zkoušet štěstí vedle?

- Pokud nalezneme banditu, který vrací *docela dobré* výsledky, měli bychom u něj již zůstat a nadále rozvíjet své skóre, nebo se vyplatí zkoušet i nadále další bandity v naději, že se povede nalézt ještě lepšího?

Toto dilema je odborně nazýváno jako explorace proti exploataci.

3.3.2 Exploration vs. Exploitation

Pokud agent zkouší nové akce, jejichž výsledek nezná, jde o exploraci, pokud provádí akce, o kterých ví, že mu přinášejí užitek, jde o exploataci.

Explorace Nacházení nových oblastí hledání, kdy agent nevyužívá předchozích znalostí.

Exploatace Agent využívá stávajících znalostí, hrozí ale uvíznutí v lokálních extrémech.

Problém je, že optimální strategie nemůže být ani čistě explorační ani čistě exploatační, čili se hledá vyvážený kompromis.

3.4 Bayesian Bandits

Ukazuje se, že nalézt optimální řešení je poměrně obtížné a může trvat i léta, než se k němu systém dopracuje. Existuje ale i spousta *přibližně optimálních* řešení.

Jedním z takových řešení jsou Bayesian Bandits, online algoritmus, o kterém již padla řeč v sekci 3.2.1. Algoritmus má přímou souvislost s učením založeným na zpětné vazbě.

Bayesovské řešení začíná stanovením pravděpodobností výhry z předchozích zkušeností pro každého banditu. Hodnoty jsou v rozmezí $\langle 0, 1\rangle$. Jak bylo již řečeno, každý bandita je reprezentován jednou distribuční funkcí.

V každém kole, kterých je v případě mého návrhu nekonečně (standardní použití Multi-armed Bandits pracuje s konečným počtem stavů) probíhá následující proces:

pro každého z banditů proveď výběr náhodné veličiny X_b z apriorní pravděpodobnosti (počty pokusů, výher, ...) bandity b

ze získaných dat vyber banditu s největší hodnotou, tedy $B = argmax X_b$

pozoruj výsledek dosažený vybraným banditou B a proveď mu aktualizaci apriorní pravděpodobnosti

vrať se k prvnímu kroku

Počáteční apriorní pravděpodobností je u každého bandity rozdělení $Beta(\alpha = 1, \beta = 1)$ (uniformní rozdělení).

Pozorovaná náhodná veličina X (výhra či prohra, tedy 1 nebo 0) je binomická. Aposteriorní pravděpodobnost se po provedení pokusu přizpůsobuje novému rozdělení

$$Beta(\alpha = 1 + X, \beta = 1 + 1 - X).$$

Pokud tedy chceme odpovědět na dřívější otázku, zda bychom měli dávat stále šanci i banditům, u kterých často prohráváme, nebo na ně zapomenout a zkoušet štěstí vedle, tento algoritmus navrhuje, abychom prohrávající bandity přímo nevyřazovali, ale vybírali stále méně častěji, jakmile získáme dostatek jistoty, že existují i lepší bandité. Opravdu tedy existuje nenulová šance, že prohrávající dosáhne statusu B, ale pravděpodobnost této šance se snižuje s rostoucím počtem odehraných kol.

TODO možná nějaký obrázky (grafy) jak se to pulls vyvíjí (všechno je v článku na campdb)

Všimněme si, že se zase až tolik nestaráme o nějaké závěry nad skrytými pravděpodonostmi, spíše se pro tento problém zabýváme o výběr nejlepšího bandity (nebo přesnějšími slovy, stále jistějšího během vybírání banditů)

To je třeba vidět v tom grafu níže, že distribuce červeného bandity je velice široká (což představuje neznalost o tom, jaká by skrytá pravděpodobnost vůbec mohla být), ale jsme si docela dobře jistí, že není nejlepší, algoritmus se tedy rozhodne tohoto banditu ignorovat.

ZPETNA VAZBA

Zajímavé poznámky V případě jakéhokoliv úspěchu doporučení by se měla navýšit někde hodnota pravděpodobnosti, se kterou bude algoritmus znovu vybrán. V případě neúspěchu pak by se tato pravděpodobnost měla exponenciálně snížit. V další iteraci už se bude systém rozhodovat s touto pravděpodobnsotí mezí objevováním a zužitkováním. V případě zužitkování se vybírá z algoritmů, co již předtím něco vynesly. To je ta strategie learnable - u banditů to funguje tak, že poměr mezi objevováním a zužitkováním se mění v čase a ne v závislosti na předchozích výsledcích. Algoritmy jsou rozděleni ve svých dimenzích a každé té dimenzi je přiřazena určitá pravděpodbnost definující míru úspěšnosti nalezení toho, že tenhle algoritmus je fakt nejlepší. V případě, že uživatelé zužitkovávají tyto znalosti, tak si poté vybírají algoritmus, který má maximální pravděpodobnost z dané množiny. epsylog greedy!!!!! - varianta epsylon decreasing Jak to má kybernář: Je li pirát ve stavu objevování, tak v případě útoku na loď nalezne přísslušný prostor, který obsahuje místo útoku, a zvýší mu hodntou pravděpodbnosti. Když je ve stavu zužitkování znalostí, tak pluje do prostoru, kde hledá transportní lodě. V případě, že za celý cyklus nenalezne žádnou loď, tak se exponenciálně sníží hodnota pravděpodobnosti daného prostoru. Nalezne li loď,tak zvýšení pravděpodobnosti proběhne jako ve stavu objevování.

ZAVEREM Algoritmus je velice jednoduchý, proto je také jednoduché jej rozšířit. V mém případě bylo potřeba přidat learning rates - předpokládejme, že to prostředí se prostě může měnit v čase. Technicky by se tedy standardní Bayesian Bandit algoritmus self-updatoval tím, že by se učil tím, že to, co si myslel, že se zdá jako nejlepší, selhává nejčastěji. Také lze docílit toho, aby se algoritmus učil měnícím se prostředím rychleji. Potřebuje pro to jedinou věc - přidat rate napříč updatováním.

Pokud rate menší než 1, algoritmus bude zapomínat předchozí výsledky

rychleji a tlak na neznalost bude směrem dolů. naopak rate větší než 1 implikuje to, že náě algoritmus se bude chovat více riskantně a bude vsázet na dřívější výhry častěji a bude tedy více odolný proti změnám měnícího se prostředí.

Kapitola 4

Realizace

Tato kapitola se zaměřuje na popis programátorských principů následovaných při realizaci systému, stejně tak popisuje technologie použité při implementaci a vzniklé výsledné řešení.

4.1 Principy a technologie

K implementaci aplikace byly použity principy a technologie uvedené níže.

4.1.1 RESTful API

4.1.1.1 REST

REpresentational State Transfer (REST) je architektonický styl definující určitá pravidla a vlastnosti návrhu API webových služeb orientovaných na zdroje. REST je silně založen na architektuře Klient-Server (server poskytuje přístup ke zdrojům, klient k nim může přistupovat a modifikovat je) a k jeho realizaci je možné využít protokolu HTTP (takovou realizaci pak nazýváme jako RESTful). Role protokolu HTTP zde není nikterak náhodná, neboť autorem REST není nikdo jiný než Roy Fielding, jenž je u protokolu HTTP podepsaný jako spoluautor [21].

Díky protokolu HTTP lze stanovit mnoho vlastností návrhu, například přítomnost adresovatelných zdrojů, kdy je každý zdroj adresovatelný pomocí Uniform Resource Identifier (URI). RESTful pro manipulaci se svými zdroji používá HTTP metody (v základu GET, POST, PUT, DELETE, ale i další, například OPTIONS či PATCH). Další vlastností je užívání standardních stavových kódů [26] HTTP (typicky 2xx, 3xx, 4xx, 5xx) v odpovědi na žádost či bezestavová komunikace.

[26] Definici všech stavových kódů viz `http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html`.

Data mohou být reprezentována v rozličných formátech jako XML, JSON či YAML.

Padla zde zmínka o bezestavosti. Aplikační servery určené pro běh aplikací jsou dvojího typu – stavové a bezestavové. Rozdíl je v uchovávání stavu aplikace během komunikace a mezi jednotlivými připojeními. V případě REST uvažujeme bezestavovost a k vyjádření přechodů mezi stavy aplikace používáme odkazy. Tento princip je nazývána jako Hypertext as the Engine of Application State (HATEOAS).

4.1.1.2 Jersey (JAX-RS)

Java API for RESTful Web Services (JAX-RS) [27] je standard definovaný v Java Specification Request 311 [28]. Jedná se o specifikaci pro RESTful webové služby implementované v programovacím jazyce Java. Pomocí JAX-RS lze s využíváním anotací jednoduše a přehledně definovat sémantiku jednotlivých tříd a jejich metod z hlediska využití v architektuře REST. Příklady anotací jsou *@Path(relativni_cesta)* pro specifikaci relativní cesty zdroje, *@GET* či *@POST* specifikující typ žádosti nebo třeba *@QueryParam* pro přiřazení parametru HTTP dotazu k hodnotě parametru příslušné metody třídy.

Pro účely implementace rozhraní systému Recommeng jsem zvolil vzhledem k předchozím zkušenostem referenční implementaci tohoto standardu v podobě frameworku **Jersey 2.x**.

4.1.2 Fronta zpráv a síťová komunikace

4.1.2.1 ØMQ

ØMQ (ZeroMQ) [29] je vysoce výkonná [30] síťová knihovna napsaná v jazyce C++ vhodná k nasazení v distribuovaných a vícevláknových aplikacích vyžadujících velkou škálovatelnost. S jejím využitím lze poměrně snadno navrhnout komplexní komunikační systém. Ke komunikaci užívá socketů [31]. Duchovním otcem a spoluautorem knihovny je slovenský expert na oblast messaging middleware Martin Sústrik [32].

Hned na úvod je nutné sdělit, že se nejedná o klasický messaging system (message-oriented middleware) typu Apache ActiveMQ [33] a další jemu podobné. Takové systémy jsou většinou hotová řešení připravená k okamžitému nasazení a integraci s dalšími službami, zatímco filosofie ZeroMQ je jiná.

[27] https://jax-rs-spec.java.net

[28] https://jcp.org/en/jsr/detail?id=311

[29] http://zeromq.org

[30] Viz výkonnostní testy na oficiální stránce `http://zeromq.org/results:_start`.

[31] Socket je mechanismus, kterým je možno zprostředkovat lokální čí vzdálenou komunikaci dvou uzlů, která má charakter klient/server [30].

[32] `http://250bpm.com/contact`

[33] `http://activemq.apache.org`

ZeroMQ je tedy multiplatformní knihovna určená k programovému využití (nabízí podporu více než 30 programovacích jazyků [34]). Pomocí jednoduchého socketového API umožňuje programátorovi sestavit si vlastní messaging system dle svého nejlepšího uvážen. Programátor využívá ze strany API veškerou podporu usnadňující práci se sítí a s trochou nadsázky lze prohlásit, že se stará pouze o zasílání zpráv. Sama socketová komunikace je výrazně zjednodušena od komunikace přes raw sockety (kde se musí průběžně krmit socketový buffer). Navíc nám dává kompletní svobodu v tom, jakým způsobem zakódujeme naši zprávu (JSON, BSON nebo jakýkoliv vlastní navržený formát).

Podporovány jsou čtyři protokoly pro komunikaci [19]:

tcp jako síťově založený přenos (procesy na jedné síti)

inproc alias model komunikace vláken uvnitř jednoho procesu

ipc alias model komunikace mezi procesy (procesy out-of-box)

multicast komunikující skrze PGM [35].

Knihovna také nabízí základní messaging patterns nabízející různé formy práce, ať už se jedná o doručování zpráv na jednotlivé uzly, mapování uzlů na vlákna, procesy, filosofii zpracování zasílaných zpráv či umisťování zpráv do fronty [36]. Každý vzor určuje jistou síťovou topologii.

Vestavěnými vzory jsou:

- Request-reply
- Pub-sub
- Pipeline
- Exclusive pair

Nejvhodnějším vzorem pro mou práci je vzor Request-reply, jehož konkrétní použití a vysvětlení, proč byl zvolen právě on, následuje záhy v sekci Popis realizace adaptibilního systému. Ostatní vzory nemá smysl v rámci této práce popisovat.

Já osobně jsem se pro účely Recommeng systému rozhodl využít čistou Java implementaci této knihovny v podobě knihovny jeroMQ [37], která z výkonnostního hlediska za původním řešením zaostává jen nepatrně [38].

[34] TODO

[35] `http://tools.ietf.org/html/rfc3208`

[36] `http://zguide.zeromq.org/page:all#Messaging-Patterns`

[37] `https://github.com/zeromq/jeromq`

[38] Viz srovnávací testy `https://github.com/zeromq/jeromq/wiki/Perfomance`.

4.1.3 Úložiště dat

4.1.3.1 Apache Solr

Apache Solr [39] je populární [40] open-source platforma pro vyhledávání napsaná v programovacím jazyce Java. Jejími charakteristickými vlastnosti jsou podpora pro fulltextové vyhledávání, fasetové vyhledávání (analogie ke konstrukci GROUP BY v RDBMS), dobrá škálovatelnost pomocí kešování a distribuovaného vyhledávání, využívání vyhledávací konstrukce *more like this*, o které bude řeč v sekci 4.2.3, a také například tzv. *near real-time indexing* [41] (dokumenty je možné vyhledávat téměř ihned po jejich zaindexování).

Z hlediska architektury programu jde o samostatný server pro fulltextové vyhledávání běžící v servletovém kontejneru (například Apache Tomcat). K indexaci a fulltextovému vyhledávání využívá ve svém jádru knihovnu Apache Lucene.

Apache Lucene [42] je vysoce výkonná knihovna pro účely vyhledávání v textu a indexování.

Vstupem pro indexaci jsou dokumenty. Každý takový dokument obsahuje množinu elementů, kde je tento element nazvaný jako *field*). Každý field má své jméno, datový typ a případně další atributy.

Vstupem pro vyhledávání jsou textové řetězce (viz syntax [43], případně dotazované objekty.

Index je uložen na disku ve formě souborů ve struktuře invertovaného indexu dokumentů [32].

Ukázka definice několika field dokumentu ve schématu Solr:

```
<field name="userId" type="int" indexed="true" stored="true"
   multiValued="true"/>
<field name="time" type="date" indexed="true" stored="true"/>
<field name="usedInRec" type="boolean" indexed="true"
   stored="true"/>
```

Ukázka reprezentace dokumentu ve výsledku vyhledávání pomocí Apache Solr ve formátu XML:

```
<doc>
   <int name="id">1</int>
   <str name="articleId">http://somedomain.org/somearticle.html</str>
   <str name="articleText">Hello Bob and Alice!</str>
```

[39] https://lucene.apache.org/solr

[40] Viz seznam serverů využívajících služeb Solr https://wiki.apache.org/solr/PublicServers

[41] https://cwiki.apache.org/confluence/display/solr/Near+Real+Time+Searching

[42] http://lucene.apache.org/core

[43] http://lucene.apache.org/core/2_9_4/queryparsersyntax.html

```
      <int name="group">123</int>
      <bool name="usedInRecommendation">true</bool>
      <date name="time">2009-04-12T20:44:55Z</date>
      <float name="1_rating">0.1</float>
      <float name="2_rating">0.5</float>
      <float name="3_rating">0.5</float>
      <arr name="userId">
         <int>1</int>
         <int>2</int>
         <int>3</int>
      </arr>
      <long name="_version_">1465487644804775936</long>
</doc>
```

Formu jejich spolupráce lze popsat tak, že Apache Solr nabízí k vyhledávání RESTful API, za kterým je skryto a voláno JAVA API knihovny Lucene. Díky tomu je možné pomocí protokolu HTTP komunikovat se Solr z jakékoliv platformy napsané v jakémkoliv programovacím jazyce.

K integraci Apache Solr s dalšími aplikacemi je možné vybírat ze spousty nástrojů a knihoven [44]. Pro účely mé aplikace psané v programovacím jazyce Java jsem si zvolil klientskou aplikaci **SolrJ** [45] s rozhraním pro vyhledávání, přidávání a aktualizaci indexu.

4.1.3.2 Apache Cassandra 2.0

Apache Cassandra 2.0 [46] je open-source distribuovaný DBMS navržený pro obsluhu velkého množství dat. Z hlediska datového modelu je Cassandra jakýmsi hybridem mezi key-value (pod 1 klíčem je uložena 1 hodnota) a column-oriented databázemi. V dokumentaci [18] se lze dočíst, že jde o row-oriented databázi.

Základem modelu je *column family* (analogie tabulky v RDBMS), jež je složena z řádků a sloupců. Každý řádek má unikátní identifikátor ve formě klíče – každý řádek obsahuje více sloupců. Sloupce mají jméno, hodnotu a časovou značku. Výhodou oproti RDBMS přístupu je to, že rozdílné řádky ze stejné column family nemusí sdílet stejnou množinu sloupců – do jedné nebo více řádek lze totiž v libovolný čas zapsat jakýkoliv sloupec.

Vzhledem k tomu, že jedním z vyzdvihovaných případů užití databáze je uchovávání časových snímků, rozhodl jsem se ji experimentálně zapojit do vytvářeného Recommeng systému. Ještě předtím jsem však detailněji zkoumal možnost použití jiné NoSQL databáze, a to **Redis**.

[44] `http://wiki.apache.org/solr/IntegratingSolr`
[45] `http://wiki.apache.org/solr/Solrj`
[46] `http://cassandra.apache.org`

Redis je klasickou key-value databází uchovávající data primárně v paměti. Postupným vývojem se jeho funkcionalita propracovala k tomu, že pod jeden klíč je nyní možné uložit několik datových struktur (např. množiny a asociativní pole). Vzhledem k ukládání dat do paměti disponuje značnou rychlostí, navíc jej lze dle konfigurace nastavit tak, aby se obsah paměti průběžně ukládal na disk pro potřeby snadného obnovení dat v případě pádu aplikace.

Jeho zapojení do aplikace jsem zvažoval ve fázi zkoumání, jakým způsobem bude v adaptibilním systému řešen failover dat. Po následném návrhu datového modelu pro ukládání časových snímků stavu aplikace jsem však sáhl po použití Apache Cassandra jako po lepší z nabízených variant pro budoucí potřeby práce (rozsahové dotazy, vizualizace vývoje systému apod.).

Velkým benefitem je podpora Cassandra Query Language (CQL), dotazovacího jazyka umožňujícího vytvářet podobné konstrukty, jaké nabízí jazyk SQL. CQL je nyní k dispozici ve verzi 3.1 a s pomocí **DataStax Java Driver 2.0** mohu snadno manipulovat s databází přímo ze své aplikace.

4.1.4 Ostatní použité technologie

4.1.4.1 Apache Mahout

Apache Mahout [47] je knihovna napsaná v programovacím jazyce Java, jež poskytuje implementaci rozličných technik z oblasti strojového učení:

- **shlukování (clustering)** – položky nacházejících se v určitých třídách (například webové stránky či novinové články) jsou organizovány do skupin tak, že položky v těchto skupinách jsou si vzájemně podobné
- **klasifikace (classification)** – učení se ze stávajících kategorizací a zařazování neklasifikovaných položek do nejvhodnější kategorie
- **doporučování (recommendation)**
- **často se vyskytující skupiny položek (frequent itemset mining)** – analýza položek v rámci nějaké skupiny (například nákupní košík) a identifikace, které položky se nejčastěji vyskytují pohromadě

Pro tyto techniky realizuje příslušné algoritmy jako například kolaborativní filtrování, k-means, náhodné lesy, skryté markovské modely a další. Některé algoritmy jsou připraveny pro běh v distribuovaném módu s využitím paradigmatu Map/Reduce, některé pak v lokálním módu (samotný Mahout je založen na Apache Hadoop, ale lze jej pohodlně využívat i bez něj) [28]. Mahout poskytuje též knihovny pro obecné matematické operace (zaměřené hlavně na oblast statistiky) a kolekce [48].

[47] `https://mahout.apache.org`

[48] `https://mahout.apache.org/users/basics/mahout-collections.html`

Využít jej pro potřeby své práce jsem se rozhodl poté, co jsem na něj narazil v projektu Mendeley během zkoumání existujících řešení doporučovacích systémů.

4.1.4.2 Spring Framework

Při tvorbě každého nového Java projektu od základu je dobré zamyslet se nad možností využití některého z mnoha frameworků a dalších užitečných nástrojů, s jejichž pomocí si lze do značné míry usnadnit proces vývoje. Díky dobrým zkušenostem z dřívějších projektů jsem zvolil open-source framework pro tvorbu moderních enterprise aplikací **Spring** [49].

Jeho výhodami jsou snadná konfigurovatelnost, podpora dependency injection, rozšiřitelnost a také integrace s jinými frameworky. V mém případě to byla integrace s frameworkem Jersey, kterou jsem využil při implementaci RESTful API.

4.1.4.3 Apache Tomcat

Apache Tomcat je známý open-source webový server a servletový kontejner. Jedná se o oficiální referenční implementaci technologií Java Servlet a Java Server Pages (JSP). Na serveru mohou běžet uživatelské servlety (programy napsané v Javě), které umí zpracovávat požadavky zasílané pomocí HTTP protokolu a tímtéž protokolem na ně odpovídat. Apache Tomcat zde slouží jako zásobník servletů starající se o jejich spouštění, běh, ukončování a podobně.

4.1.4.4 Správa závislostí

Často slýchaným pojmem z úst mnoha vývojářů v programovacím jazyce Java je tzv. *classpath hell*. V podstatě jde o problémy spojené s načítáním programových tříd. V dnešní době existuje spousta nástrojů schopných tento problém efektivně řešit používáním správně projektové struktury, sestavovacích nástrojů a nástrojů pro správu závislostí. Jmenujme například Apache Ant společně s Apache Ivy, Gradle nebo třeba Apache Maven.

Nástroj **Apache Maven** jsem využil při implementaci všech komponent aplikace Recommeng.

4.2 Popis realizace adaptibilního systému

4.2.1 Vrstva pro obsluhu zpráv

Následující text popisuje postup, jakým jsem realizoval vrstvu pro obsluhu zpráv pomocí síťové knihovny ZeroMQ. Postup by se dal shrnout do třech bodů:

[49] `http://projects.spring.io/spring-framework`

- rozhodnutí o komunikačním protokolu
- definice síťové infrastruktury
- výběr vzoru pro zasílání zpráv (messaging pattern)

4.2.1.1 Komunikační protokol

V prvním kroku bylo nutné rozhodnout o tom, jakým způsobem bude probíhat přenos dat směrem k aplikaci a naopak. Má volba padla na síťově založený přenos přes **TCP** z toho důvodu, že dle návrhu architektury má adaptibilní systém běžet na vlastním serveru a uživatelé navazovat spojení zvenčí mimo server.

4.2.1.2 Infrastruktura

Dále bylo nutné definovat, jakým způsobem budou vzájemně propojeny odlišné komponenty systému. Vyšel jsem z nativní povahy Klient-Server architektury, kde server zastupuje roli stabilnější komponenty v síti. Bude tedy zastávat roli BIND a přijímat spojení na socketu na specifikovaném portu. Klienti jsou dynamičtější, přiřadíme jim proto roli CONNECT a budou se připojovat k socketu.

TODO obrázek z visio KLIENT - fronta - SERVER

Zároveň jsem mezi ně umístil prostředníka, ke kterému se mohou jak klient, tak i server připojit. Tímto krokem jsem si navíc ulehčil práci, neboť používáním této komponenty odpadá potřeba řešit další logiku aplikace starající se o seznam připojených peerů.

V případě messaging patternu REQ/RESP se prostředník nazývá fronta.

4.2.1.3 Zasílání zpráv

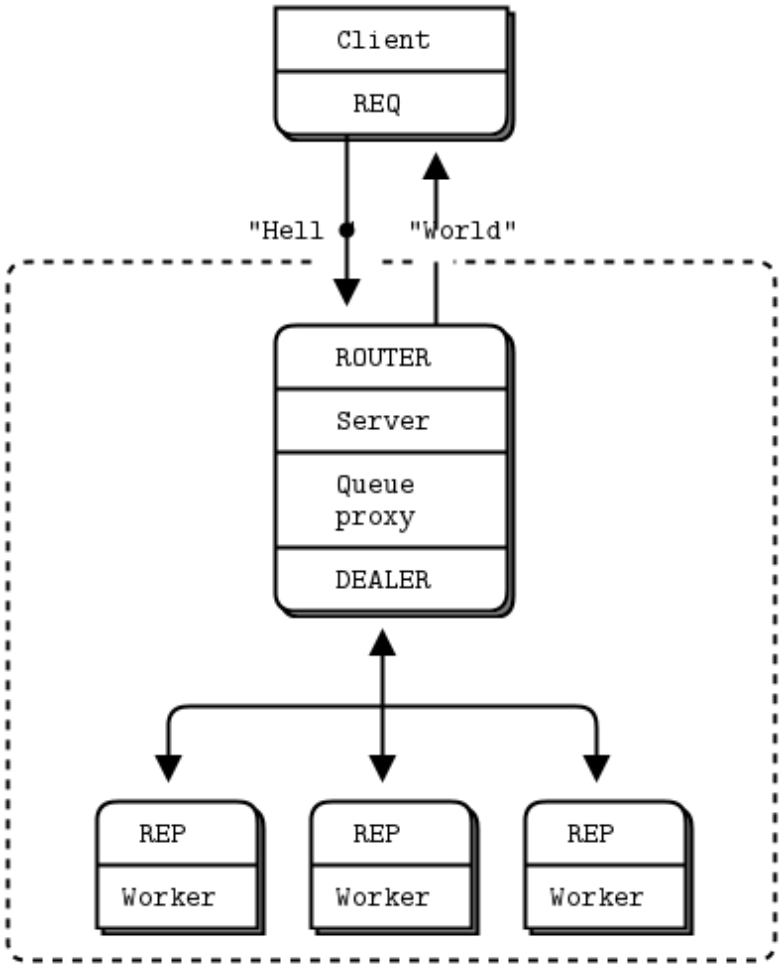
Zbývá ještě specifikovat tok zpráv. Z požadavků na systém vyplývala v podstatě jediná možnost volby vzoru, a to obousměrný, stavově založený *Request Reply* vzor s podporou vyvažování zátěže (load balancing). Toto paradigma je známé z většiny typů serverů

TODO POPIS Z

`http://nichol.as/zeromq-an-introduction` a `https://docs.google.com/document/d/1Aqpz4JjIsz_YXll8RPzItv4CAbugCnIvyicnZc5Qq8s/edit#`

a tady je multithreading a ZMQ `http://zguide.zeromq.org/page:all#Multithreading-with-MQ`

TODO format zasilani zprav



Obrázek 4.1: Vícevláknový server s využitím ROUTER a DEALER. Zdroj: [13]

4.2.1.4 Vícevláknový server

Ja budu pouzivat vzor: XREQ/XREP dela asynchronni komunikaci XREQ je historicky jmeno pro dealera, dealer asynchrone odesila to *REP sockets, muze odeslat na XREP (router) sockety take

XREP je historicke jmeno pro router router asynchrone obsluhuje *REQ sockets, muze obsluhovat take dealera sockety

```
// Prepare our context and socket
ZMQ.Context context = ZMQ.context(1);
// Socket to talk to clients
ZMQ.Socket clients = context.socket(ZMQ.ROUTER);
clients.bind("tcp://" + host + ":" + port);

// Socket to talk to workers
ZMQ.Socket workers = context.socket(ZMQ.DEALER);
workers.bind("inproc://workers");

// Create worker threads pool
Thread threads[] = new Thread[10];

// Launch worker threads
for (int i = 0; i < threads.length; i++) {
 threads[i] = new MultiThreadServer.WorkerThread(
      i, context, api, new RequestHandlerJson()
 );
    threads[i].start();
}
// Connect work threads to client threads via a queue
ZMQQueue queue = new ZMQQueue(context, clients, workers);

logger.info("Application started successfully!");

//Forwards messages from router to dealer and vice versa.
 new Thread(queue).start();
```

4.2.2 Adaptibilní systém

4.2.2.1 Parametrizace

volba parametrů, ať už těch, co ovlivňují tresty nebo třeba úložiště (redis...)

4.2.2.2 Pravidelné ukládání časových snímků

http://www.petrikainulainen.net/programming/spring-framework/spring-from-the-trenches-using-environment-specific-cron-expressions-with-

`the-scheduled-annotation/`

4.2.3 Sada algoritmů

4.2.4 RESTful API

Kapitola 5

Experimenty a vyhodnocení

Jednou z mnoha výzev pro někoho, kdo se snaží vybudovat doporučovací systém, je to, že je velice těžké dopředu říct, zda budou naše předpovědi dost přesné. Alespoň do té doby, dokud je nezačneme dělat a nebudeme pozorovat, jak často naši uživatelé přijímají naše návrhy. Je zde obrovský prostor možností (možných metod), z čeho vybírat.

kecy o testování

5.1 Testování různých způsobů chování

viz jak jarda vymyslel těch zhruba 5 příkladů, co mohou nastat

5.2 Experimenty

`http://contest.plista.com/wiki/example`

5.2.1 Vyhodnocovací technologie

Výpočty. kvůli kombinování budeme počítat s floaty

5.3 Zhodnocení aplikace

Slovní zhodnocení

5.4 Budoucí práce

bude li nějaká

Závěr

Literatura

[1] BellKor, AT&T Labs, Inc. – Research. [online], stav ze dne 21.4.2012. Dostupné z: http://www2.research.att.com/~volinsky/netflix/

[2] CNN Acquires Zite. [online], stav ze dne 21.4.2012. Dostupné z: http://cnnpressroom.blogs.cnn.com/2011/08/30/cnnzite/

[3] easyrec: Recommendation Engine. [online], stav ze dne 24.4.2012. Dostupné z: http://easyrec.org/recommendation-engine

[4] How does the Amazon Recommendation feature work? [online], stav ze dne 21.4.2012. Dostupné z: http://stackoverflow.com/questions/2323768/how-does-the-amazon-recommendation-feature-work

[5] Kvantily. [online], stav ze dne 26.4.2012. Dostupné z: http://www-troja.fjfi.cvut.cz/~limpouch/sigdat/pravdh/node8.html

[6] Mendeley: Recommendation Systems for Academic Literature. [online], stav ze dne 21.4.2012. Dostupné z: http://www.slideshare.net/KrisJack/mendeley-recommendation-systems-for-academic-literature

[7] Netflix Contest: 1 Million Dollars for Better Recommendations. [online], stav ze dne 21.4.2012. Dostupné z: http://www.uie.com/brainsparks/2006/10/02/netflix-contest-1-million-dollars-for-better-recommendations/

[8] Netflix offers streaming movies to subscribers. [online], stav ze dne 21.4.2012. Dostupné z: http://arstechnica.com/uncategorized/2007/01/8627/

[9] Náhodný výběr. [online], stav ze dne 26.4.2012. Dostupné z: ftp://math.feld.cvut.cz/pub/prucha/ubmi/predn/u12.pdf

[10] Proklik. [online], stav ze dne 21.4.2012. Dostupné z: http://www.adaptic.cz/znalosti/slovnicek/proklik/

[11] What Is Affinity Analysis? [online], stav ze dne 21.4.2012. Dostupné z: http://www.wisegeek.com/what-is-affinity-analysis.htm

[12] Zite is Flipping out. [online], stav ze dne 21.4.2012. Dostupné z: http://blog.zite.com/2014/03/05/zite-is-flipping-out/

[13] ØMQ - The Guide. [online], stav ze dne 26.4.2012. Dostupné z: http://zguide.zeromq.org/page:all

[14] Almazro, D.; Shahatah, G.; Albdulkarim, L.; aj.: A Survey Paper on Recommender Systems. [online], stav ze dne 26.4.2012. Dostupné z: http://arxiv.org/pdf/1006.5278v4.pdf

[15] Anderson, C.: The Long Tail. [online], říjen 2004. Dostupné z: http://archive.wired.com/wired/archive/12.10/tail.html

[16] Blažek, R. B.; Kotecký, R.; Hrabáková, J.; aj.: BI-PST – Pravděpodobnost a statistika, přednáška 3. [online], stav ze dne 26.4.2012. Dostupné z: https://edux.fit.cvut.cz/courses/BI-PST/_media/lectures/3_handout_pst-v2.pdf

[17] Brodt, T.: Open Recommendation Platform. 2013, [Online; stav z 23. dubna 2014]. Dostupné z: http://www.slideshare.net/d0nut/open-recommendation-platform

[18] DataStax: Architecture in brief | DataStax Cassandra 2.0 Documentation. [online], stav ze dne 27.4.2012. Dostupné z: http://www.datastax.com/documentation/cassandra/2.0/cassandra/architecture/architectureIntro_c.html

[19] Dennis, J.: ZeroMQ: Super Sockets. [online], stav ze dne 27.4.2012. Dostupné z: http://www.slideshare.net/j2d2/zeromq-super-sockets-by-j2-labs

[20] Drachsler, H.: Recommender Systems and Learning Analytics in TEL. University Lecture, 2014, stav ze dne 24.4.2012. Dostupné z: http://www.slideshare.net/Drachsler/rec-sys-mupplelecturekmi

[21] Fielding, R. T.: *Architectural Styles and the Design of Network-based Software Architectures*. Dizertační práce, 2000, aAI9980887.

[22] Hlaváč, V.: Učení bez učitele. University Lecture, 2014, stav ze dne 24.4.2012. Dostupné z: http://cmp.felk.cvut.cz/~hlavac/Public/TeachingLectures/UceniBezUcitele.pdf

[23] Jacobi, J.; Benson, E.; Linden, G.: Personalized recommendations of items represented within a database. Září 26 2006, uS Patent 7,113,917. Dostupné z: http://www.google.com/patents/US7113917

[24] Jakob, M.: Reinforcement Learning. University Lecture, 2010, stav ze dne 24.4.2012. Dostupné z: https://cw.felk.cvut.cz/wiki/_media/courses/a3m33ui/prednasky/files/ui-2010-p11-reinforcement_learning.pdf

[25] John Gantz, D. R.: Extracting Value from Chaos. [online], červen 2011. Dostupné z: http://www.emc.com/collateral/analyst-reports/idc-extracting-value-from-chaos-ar.pdf

[26] Linden, G.; Jacobi, J.; Benson, E.: Collaborative recommendations using item-to-item similarity mappings. Červenec 24 2001, uS Patent 6,266,649. Dostupné z: https://www.google.com/patents/US6266649

[27] Liu, J.; Pedersen, E.; Dolan, P.: Personalized News Recommendation Based on Click Behavior. In *2010 International Conference on Intelligent User Interfaces*, 2010.

[28] Musto, C.: Apache Mahout – Tutorial (2014). [online], stav ze dne 27.4.2012. Dostupné z: http://www.slideshare.net/Cataldo/apache-mahout-tutorial-recommendation-20132014

[29] Vitvar, T.: Lecture 5: Application Server Services. University Lecture, 2014, stav ze dne 24.4.2012. Dostupné z: http://humla.vitvar.com/slides/mdw/lecture5-1p.pdf

[30] Vychodil, V.: Komunikace pomocí Socketu. [online], stav ze dne 27.4.2012. Dostupné z: http://vychodil.inf.upol.cz/publications/white-papers/socket-referat.pdf

[31] Vychodil, V.: Pravděpodobnost a statistika: Normální rozdělení a centrální limitní věta. [online], stav ze dne 26.4.2012. Dostupné z: http://vychodil.inf.upol.cz/kmi/pras/pr09.pdf

[32] Wikipedia: Inverted index — Wikipedia, The Free Encyclopedia. 2014, [Online; stav z 27. dubna 2014]. Dostupné z: http://en.wikipedia.org/w/index.php?title=Inverted_index&oldid=591814302

[33] Wikipedia: Long tail — Wikipedia, The Free Encyclopedia. 2014, [Online; stav z 19. dubna 2014]. Dostupné z: http://en.wikipedia.org/w/index.php?title=Long_tail&oldid=603431399

[34] aihorizon: Your Online Artificial Intelligence Resource: Machine Learning, Part I: Supervised and Unsupervised Learning. [online], stav ze dne

24.4.2012. Dostupné z: http://www.aihorizon.com/essays/generalai/supervised_unsupervised_machine_learning.htm

[35] Černý, D.: Bayesovská statistika: klíč k porozumění vesmíru? [online], stav ze dne 26.4.2012. Dostupné z: http://gchd.cz/fygyz/2012_2013/david_cerny-bayesovska_statistika.pdf

Příloha **A**

Seznam použitých zkratek

IDC

CTO

MIT

ACM Association for Computing Machinery

ICWSM

ICML

IBM

REST

API

TCP

DBMS

NoSQL

MQ

JVM

JSON

URL

ORP

HTTP

Příloha **B**

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `exe` ..... adresář se spustitelnou formou implementace
- `src`
  - `impl` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF
  - `thesis.ps` ..... text práce ve formátu PS