Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií

FIIT-5212-49447

Martin Černák

Dynamické odporúčanie

Bakalárska práca

Študijný program: Informatika
Študijný odbor: 9.2.1 Informatika
Miesto vypracovania: Ústav informatiky a softvérového inžinierstva
Vedúci práce: prof. Ing. Pavol Návrat, PhD.

Máj 2015

Anotácia

Slovenská technická univerzita v Bratislave
FAKULTA INFORMATIKY A INFORMAČNÝCH TECHNOLÓGIÍ
Študijný program: Informatika
Autor: Martin Černák
Bakalárska práca: Dynamické odporúčanie
Vedúci práce: prof. Ing. Pavol Návrat, PhD.
Máj 2015

Dynamické odporúčanie v kontexte hudobných dokumentov je vďaka svojmu úzkemu zamerania a vďaka menšej komunite značne nepreskúmané. Existuje niekoľko riešení, ktoré ale nevyužívajú plný potenciál dynamického odporúčania. Jednou z možností ako tieto systémy vylepšiť, je začať uvažovať starnutie ako používateľových tak globálnych preferencií. V hudobnom odvetví môžeme častejšie ako v ostatných vidieť príchod mimoriadne populárnych nových interprétov, piesni a štýlov, ktoré rýchlo vymiznú z povedomia verejnosti, prípadne zostane okolo nich úzka skupina fanúšikov. Kontrastom k nim sú piesne, autori a hudobné štýly, ktoré pretrvávajú dlhodobo v povedomí ľudí a vypadajú, že starnutie na nich nemá vplyv.

Annotation

Slovak University of Technology Bratislava
FACULTY OF INFORMATICS AND INFORMATION TECHNOLOGIES
Degree Course: Informatics
Author: Martin Černák
Bachelor thesis: Dynamic recommendation
Supervisor: prof. Ing. Pavol Návrat, PhD.
May 2015

Dynamic recommendation in the context of musical documents thanks to its narrow focus and with smaller community largely unexplored. There are several solutions but that don't using full potential of dynamic recommendation. One way to improve these systems is to start thinking of aging user and global preferences. In the music sector this can be more frequent than in other sectors, extremely popular new artists, songs and styles that quickly disapear from public awareness or remain around them a small group of fans. A contrast to them are songs, authors and musical styles that persist long time in the minds of people and looks like aging does not affect them.

POĎAKOVANIE

Chcel by som v prvom rade poďakovať pánu profesorovi Pavlovi Návratovi za jeho odbornú pomoc a motiváciu a za všetky konzultácie ktoré sme spoločne absolvovali. Zároveň by som rád poďakoval učástníkom jeho výskumného seminára za konštruktívnu kritiku, ktorá taktiež prispela k zdokonaleniu tejto práce.

ČESTNÉ PREHLÁSENIE

Čestne prehlasujem, že záverečnú prácu som vypracoval samostatne s použitím uvedenej literatúry a na základe svojich vedomostí a znalostí.

.....

Martin Černák

Obsah

Zoznam obrázkov

Zoznam ukážok

1 Úvod

Ako z jej názvu vyplýva, informatika je predmet zameraný na prácu s informáciami. To, čo kedysi bolo najväčším problémom, teda dostať nejaké informácie k používateľom už dávno nie je problém. Vďaka internetu sa dajú informácie dostať prakticky všade. No teraz čelíme väčšiemu problému. Naša spoločnosť dokáže za účelom zábavy, rozvoja, alebo produktivity vyprodukovať neuveriteľné množstvo informácií. Precíznosť archivácie údajov je asi najväčšia v histórii, problém nastáva, ak chceme nejaké údaje vyhľadať. Klasický prístup spravovania informácií už nie je dostačujúce a jednoduché vyhľadávanie, už nie je dostatočne efektívne na to, aby sme boli schopní nájsť požadované informácie.

Dokonca aj vyhľadávanie ako také prestáva byť dostatočne efektívne, namiesto neho sa dostáva do popredia odporúčanie, ktoré doslova používateľovi ponúkne informácie, ktoré by ho mohli zaujímať bez toho, aby musel vynaložiť akúkoľvek námahu na hľadanie. Aby mohol systém robiť takúto predikciu, potrebuje poznať používateľa a to mu umožňuje profilovanie používateľov. Profil používateľa je komplexná vec. Záujmy používateľa môžu byť ovplyvnené jeho demografickými parametrami (vek, vzdelanie, miesto pobytu), záujmami a všeobecnými novinkami ako vydanie nového albumu obľúbenej kapely, alebo uvedenie nového zariadenia na trh. Do úvahy musíme brať aj udalosti v živote používateľa, napríklad narodenie potomka tiež v určitom smere ovplyvní používateľove záujmy. Z toho vyplýva, že profil musí byť dynamický, a preto je potrebné nejakým spôsobom aj odoberať záujmy, o ktoré sa používateľ už viac nezaujíma.

Cieľom tohto projektu je vytvoriť aplikáciu, ktorá bude schopná dynamicky odporúčať. Na riešenie hore spomenutých problémov existuje množstvo prístupov. Každý z týchto prístupov má mierne lepšie výsledky v iných situáciách, čiže dosť závisí od domény, pre ktorú bude systém odporúčať. V tomto projekte sa budeme zaoberať doménou hudobných dokumentov (akordy, texty, taby, preklady). Táto oblasť ešte nie je prebádaná, čo nám prináša nové možnosti ako aj nové problémy.

1.1 Použité pojmy a skratky

sedenie - Sekvencia zobrazení dokumentov, ktorá je časovo ohraničená.

akcia - Jedna elementárna interakcia používateľa so systémom, kliknutie na odkaz, zadanie vyhľadávacieho reťazca.

dokument - Je jedno hudobné dielo reprezentované tabmi, textom, notami, prekladom, alebo iným spôsobom nápomocným k prevedeniu hudobného diela.

vlastnosť dokumentu - Špecifický črt dokumentu, ktorý môže ovplyvniť používateľové preferencie.

značka dokumentu - Označená vlastnosť dokumentu, ktorá je rozoznávana vyhľadávacím systémom.

preferencia - Je vlastnosť dokumentu, ktorú nejakým spôsobom používateľ preferuje.

užitočnosť - Vlastnosť je hodnota určujúca, či je daný dokument preferovaný používateľom.

2 Existujúce riešenia

Počas analýzy som našiel niekoľko riešení problému dynamického odporúčania. Každé z týchto riešení poskytuje rôzne výhody a nevýhody pri rôznych oblastiach nasadenia. To, ktorý model je najvhodnejší pre nás je ovplyvnené cieľovým artiklom a cieľovou skupinou. Pre potreby tejto práce bolo potrebné najskôr vybrať doménu, v ktorej chcem riešenie implementovať. Následne zanalyzovať prístupy v danej oblasti.

2.1 Odporúčanie hudobných dokumentov

Pre potreby tejto práce som zvolil oblasť hudby, avšak nie v úplne klasickom ponímaní, zameral som sa na dokumenty umožňujúce hudobníkom naučiť sa hrať určité hudobné dielo. Za základné vlastnosti hudby sa považuje rytmus, sila a farba tónu. Na zaznamenanie týchto vlastností vzniklo viacero zápisov podľa potrieb určitých skupín hudobníkov. Odporúčanie v tejto oblasti je pomerne nové, a preto sa budem skôr snažiť nájsť riešenia z iných oblastí a preskúmať ich aplikovateľnosť v tejto oblasti.

Na presné odporúčanie akéhokoľvek artiklu musíme najskôr nájsť nejaké jeho vlastnosti na základe, ktorých môžeme odporúčať. Keďže táto doména je úzko spojená s hudbou, budem sa snažiť vychádzať z nej.

Najznámejším spôsobom kategorizovania hudby je rozdelenie na žánre a podžánre. Problém pri žánroch a podžánroch je, že neexistuje jednotná definícia, ani spôsob ich kategorizácie. Určovanie žánrov má nasledujúce typy pravidiel:

- **formálne a technické pravidlá aplikované na obsah (sila, výška a farba tónu)**,
- **semiotické pravidlá** (abstraktný vopred dohodnutý koncept, napríklad politická situácia),
- **pravidlá správania sa** (črty správania sa fanúšikov, alebo interpretov daného žánru),

- **ekonomické a jurisdikčné pravidlá** (zákonné a právne aspekty, ktoré daný žáner podporujú, napríklad český protestsong[1]).

Tieto pravidlá definoval Franco Fabbri[4].

Pravidlá sú pomerne abstraktné a i napriek viacerým pokusom o vytvorenie kompletne ontológie žánrov či už z akademických alebo komerčných kruhov [2], neexistuje jednotná ontológia hudobných žánrov.

Pre potreby odporúčania by bolo najvhodnejšie automatické určovanie žánru, ako napríklad navrhli autori článku [6], kde analyzovali výšku a snažili sa odhaliť akcent nôt, čo im umožnilo odhaliť rytmus piesne. Následne sa zamerali na určenie jednotlivých časti hudobného diela ako predohra, hlavná časť, refrén a sloha.

Ďalším prístupom je nechať označovať vlastnosti dokumentu používateľom, tento prístup používa napríklad služba last.fm[3], ktorá sa následne snaží používať najpopulárnejšie značky ako žánre. Tento prístup je bližšie opísaný v článku Paual Lamera[5] a je známy pod názvom sociálne značenie (angl. social tagging).

Podobnosť odporúčania hudobných dokumentov a hudby

I keď sú tieto dve domény veľmi podobné, existujú rozdiely. Jeden z rozdielov sú ich vlastnosti dokumentov. Na presnú identifikáciu pesničky nám stačí názov piesne, autor piesne, interpret a prevedenie. Pri hudobných dokumentoch sa môžu okrem týchto vlastností líšiť aj v type dokumentov (taby, akordy, text, preklad), prípadne niektoré dokumenty môžu obsahovať iba časť daného hudobného diela (predohra, medzi-hra, refrén, sólo atď.).

Podobnosť s odporúčaním textu

Ďalšia doména, ktorú je možné využiť, je odporúčanie textových dokumentov. To je najčastejšie založené na anlýze výskytu slov v texte. Avšak jediné typy dokumentov, ktoré by sa dali takto analyzovať sú preklady a texty.

[1] www.wikipedia.sk
[2] https://www.apple.com/itunes/affiliates/resources/documentation/genre-mapping.html
[3] http://www.last.fm/home

2.2 Rôzne prístupy k odporúčaniu

Hlavným účelom odporúčacích systémov je odhadnúť užitočnosť dokumentu pre používateľa[8], pričom v mnohých prípadoch je potrebné užitočnosť dokumentu odhadovať. Užitočnosť ako taká môže závisieť od veľkého množstva parametrov, skupina týhto parametrov sa všeobence nazýva kontextové premenné. Na základe toho ako systém nakladá z danými údajmi, delím odporúčacie systémy na niekoľko kategórií. Hlavnou charakteristikou systému je **funkcia užitočnosti**.

Filtrovanie na základe obsahu

Pri tomto prístupe odporúčame používateľovi dokumenty podobné tým, čo sa mu páčili v minulosti. Funkciou užitočnosti dokumentu je teda podobnosť dokumentu s už zobrazenými dokumentmi používateľa. Podobnosť dokumentov sa zisťuje pomocou porovnávania značiek dokumentov.

Kolaboratívne filtrovanie

Toto je najpopulárnejší spôsob implementácie odporúčania. Najjednoduchšia a originálna implementácia odporúča aktívnemu používateľovi dokumenty, ktoré sa páčili ľuďom s podobným vkusom. Podobnosť používateľov je založená na histórii hodnotenia dokumentov používateľov. Takže úlohou funkcie užitočnosti je nájsť najpodobnejších používateľov a vrátiť dokument, ktorý mal najkladnejšie hodnotenia od týchto používateľov.

Najväčším problémom kolaboratívneho filtrovania, tzv. problém studeného štartu. Spočíva v tom, že ak pribudne do zbierky nový dokument, nemá ešte žiadne hodnotenia, takže sa nebude nikomu odporúčať.

Demograficke odporúčanie

Tieto odporúčacie systémy odporúčajú používateľovi dokumenty na základe jeho demografického profilu (vek, národnosť, jazyk atď.). Za istú formu tohoto odporúčania môžeme považovať multijazyčnosť dnešných stránok. Zaujímavým príkladom je aj domovská stránka google, ktorá sa vo významné dni zobrazuje v

rôznych krajinách rôzne[1].

Znalostné odporúčanie

Zanlostné odporúčacie systémy odporúčajú na základe znalostí o tom ako nejaká vlastnosť dokumentu ovplyvňuje užitočnosť dokumentu pre používateľa. V princípe ide o systém, ktorý dáva používateľovi otázky a na základe zistených faktov mu odporučí dokument. Takéto odporúčanie sa najčastejšie využíva pri zákazníckej podpore. Veľmi dobrým príkladom na tento prístup je zákaznícka podpora Microsoftu[2].

Komunitné odporúčanie

Toto odporúčanie je veľmi podobné s kolaboratívnym filtrovaním, avšak na rozdiel od neho uprednostňuje implicitne dané priateľstvá medzi používateľmi. Tento druh odporúčania zažíva rozkvet najmä v poslednej dobe spolu s rozkvetom používania sociálnych sietí. V dokumente[8] sa dokonca uvádza, že v špeciálnych prípadoch sú efektívnejšie ako kolaboratívne filtrovanie. Tento druh odporúčania sa často nazýva aj sociálne odporúčanie. Funkcia užitočnosti v tomto prípade najskôr zistí vzťahy medzi používateľmi a preferencie priateľov používateľa a na základe ich preferencií určí užitočnosť dokumentov pre používateľa.

Hybridné odporúčanie

Dané odporúčanie kombinuje vlastnosti predchádzajúcich prístupov na vyriešenie ich vzájomných problémov. Napríklad časté riešenie je kombinovanie colaboratívneho filtrovania s filtrovaním založeným na obsahu, kedy v podstate filtrovanie na základe obsahu rieši problém studeného štartu pre kolaboratívne filtrovanie. Tak isto sa v poslednej dobe zvykne kombinovať kolaboratívne filtrovanie s komunitným odporúčaním vďaka ich dobrým výsledkom.

[1] http://www.google.com/doodles/
[2] https://support.microsoft.com/sk-sk

Spätná väzba

Aby odporúčacie systémy mali ako odporúčať, potrebujú zistiť preferované vlastnosti dokumentov z korpusu odporúčaných dokumentov. Z tohto dôvodu je potrebné nejakým spôsobom zistiť, ktoré vlastnosti používateľ preferuje. Na toto zisťovanie slúži spätná väzba. Spätnú väzbu v zásade delíme na dve skupiny:

- **implicitná spätná väzba** (je získavanie spätnej väzby používateľa z jeho akcií, ktoré nesúvisia priamo s hodnotením, napríklad stiahnutie dokumentu, prípadne jeho vytlačenie, hlavnou výhodou je, že nevyžaduje vedomý zásah používateľa, avšak zvyšuje technické nároky na systém),
- **explicitná spätná väzba** (je vedome ponúknutie spätnej väzby od používateľa, napríklad ak používateľ označí, že sa mu dokument páči, [1] zvykne byť efektívnejšia, avšak vyžaduje vedomý zásah používateľa).

Explicitnú spätnu väzbu môžeme ďalej deliť na základe toho, akú hodnotu nám vracia napríklad na **binárnu**, „páči sa mi“ a „nepáči sa mi“, alebo **na hodnotu** (napríklad pridelovanie 1 až 5 hviezdičiek). S týmto priamo súvisí aj konštrukcia používateľských profilov.

2.3 Používateľské profily

Ďalšou dôležitou súčasťou odporúčacích systémov je spôsob akým konštruujú používateľské profily. Rôzne druhy profilov umožňujú použitie rôznych algoritmov avšak môžu mať rôzny dopad na pamäťovú či výkonovú stránku systému[1].

Binárny vektor

V tomto prípade sú preferencie reprezentované vektorom v dvojrozmernom priestore, kde jeden rozmer sú značky dokumentov a druhý sú používatelia. Vektor je tvorený binárnou hodnotou kde 1 na pozícií p_{ij}, pri predpoklade že j je identifikátor j.-teho používateľa a i je identifikátor i-tej značky, znamená, že i-ta značka je preferencia používateľa j. V tabuľke 1 môžeme vidieť príklad, v ktorom máme používateľa p_0 ktorý nepreferuje žiadné značky, následne Používateľa p_1 ktorý preferuje značky z_1, z_2 a Používateľa p_2 ktorý preferuje značku z_0.

z_p	p_0	p_1	p_2
z_0	0	0	1
z_1	0	1	0
z_2	0	1	0

Tabuľka 1: *Ukážka modelu profilu ako binárneho vektora*

Vahovaný vektor

Tento profil je veľmi podobný s predchádzajúcim profilom, avšak namiesto binarnych hodnôt sú hodnotami súradníc užitočnosti daných preferencií 2.

z_p	p_0	p_1	p_2
z_0	0	0	2
z_1	0	2	0
z_2	0	5	1

Tabuľka 2: *Ukážka modelu profilu ako váhovaného vektora.*

Trojrozmerný vektor

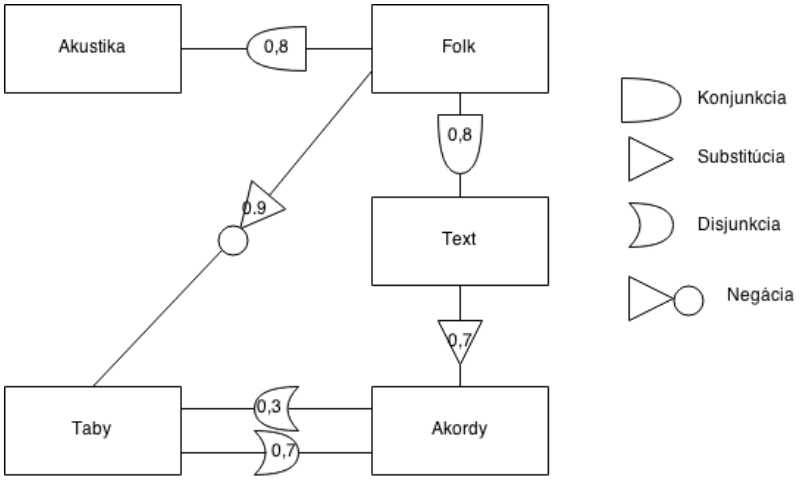
V prípade, že vieme aj rozdeliť značky do domén alebo kontextov, môžeme do vektorového profilu pridať ešte jednu súradnicu, ktorá reprezentuje doménu inej značky. Napríklad ak odporúčame text a značky sú reprezentované slovami, ktoré sa našli v texte, tak značke, ktorá pochádza z nadpisu, môžeme automaticky priradiť väčšiu hodnotu.

Profil sémantickej siete

Profil sémantickej siete (angl. Semantic network profile) je semantická sieť ktorá je vybudovaná pre konkrétneho používateľa a vyjadruje vzťahy medzi značkami, ktoré používateľ preferuje.

Sémantická sieť[11] je orientovaný graf, v ktorom sú vrcholmi značky, zatiaľ čo hrany sú ich vzťahy. napríklad v [1] sú použité vzťahy typov.

- konjunkcia,



Obr. 1: *Ukážka sémantickej siete*

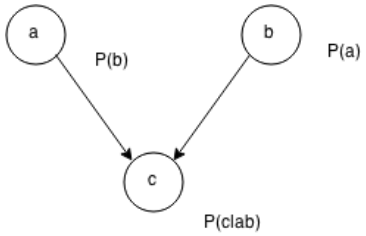
- disjunkcia,
- substitúcia,
- negácia.

Takéto semantické siete sa používajú najmä pri dopĺňaní slov do vyhľadávacích fráz, kde v prípade krátkej vyhľadávacej frázy môžeme zredukovať jej viacznačnosť doplnením slov, ktoré majú v sémantickej sieti najsilnejšiu pozitívnu väzbu. Na obrázku 1 môžeme vidieť príklad takejto siete.

Teda ak by mi prišiel od používateľa výraz *folk*, tak môžem výraz rozšíriť slovami *akustika* a *text*. Keď budem pokračovať, slovo text sa dá nahradiť slovom akordy (keďže v podstate akordy sú texty doplnené o skratky akordov). Následne však taby už nemôžem doplniť aj keď majú disjunktný vzťah s akordami, pretože majú silnú negatívnú väzbu s folkom. Teda výsledný výraz by bol *folk a akustika akordy*.

Bayesova sieť

Ďalšou možnosťou ako ukladať používateľské dáta je bayesová sieť. Bayesová sieť vychádza z bayesovej teoréma, o ktorej využití v kontexte odporúčania sa budeme zaoberať neskôr. Bayesová sieť slúži na výpočet pravdepodobnosti hypotézy



Obr. 2: *Ukážka bayesovej siete*

pri zmene jej evidencie. Jej vrcholmi sú latentné premenné (závisia od ostatných premenných) a jej hranami ich vzťahy. Takže v prípade, že sa mi zmení hodnota jednej premennej v grafe, po hranách viem upraviť hodnoty všetkých premenných, ktoré od nej závisia.

Napríklad na obrázku 2 môžeme vidieť bayesovú sieť zloženú z troch premenných a, b a c. Tieto sú vo vzájomnom vzťahu. V prípade zmeny hodnoty a alebo b sa hodnota c automaticky prepočíta.

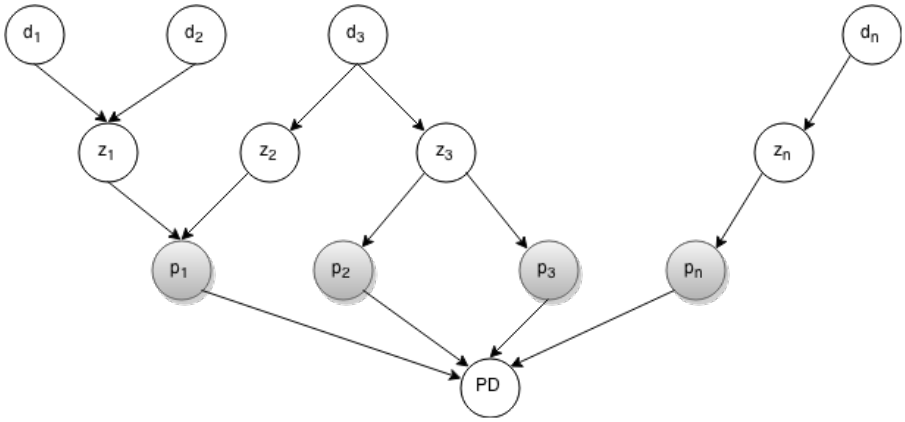
Ako príklad ako využiť takúto sieť môžem uviesť zjednodušený príklad z prezentácie [3], kde najhornejšie vrcholy (tie, ktoré nezáviseli od iných premenných) boli dokumenty $D = d_1, d_2...d_n$ a pod nimy boli ich značky $Z = z_1, z_2...d_n$, následne ak sme chceli odporúčať (príklad je aplikovaný na vyhľadávací dotaz, avšak dotaz môže byť nahradený používateľským profilom), na najspodnejšie body sme pripojili preferencie používateľa $P = p_1, p_2...p_n$. Náčrt siete môžeme vidieť na obrázku 3.

Pričom značky a dokumenty si môžeme držať predpočítané v pamäti a tak isto aj používateľské preferencie. V prípade odporúčania ich iba poprepájame.

Bayesov theorem

Bayesová teoréma sa zaoberá vplyvom nových poznatkov na existujúce domienky o určitej hypotéze. Vďaka nej vieme kombinovať nové dáta s existujúcimi poznatkami. Matematicky je táto teoréma vyjadrená v kontexte vyhľadávania pomocou rovnice 1 a rovnice 2.

$$P(P|d) = \frac{P(d|P) * P(P)}{P(d)} \tag{1}$$



Obr. 3: *Ukážka odporúčacej bayesovej siete*

$$P(NP|d) = \frac{P(d|NP) * P(NP)}{P(d)} \tag{2}$$

Rovnice obsahujú,

- pravdepodobnosť, že dokument je užitočný P,
- prevdepodobonsť, že dokuemnt nie je užitočný NP,
- pravdepodobnosť, že vrátený dokument d je užitočný $P(P|d)$,
- $P(d|P)$ prevdepodobnosť, že ak je vrátený užitočný dokument, je to dokument d,
- pravdepodobnosť vrátenia užitočného dokumentu $P(P)$,
- pravdepodobnosť výberu dokumentu $P(d)$,
- pravdepodobnosť neužitočnosti dokumentu $P(NP|d)$,
- pravdepodobnosť $P(d|NP)$ že ak je ak je vrátený neužitočný dokument, je to dokument d,
- $P(NP)$ je pravdepodobnosť že je vrátený neužitočný dokument,
- pravdepodobnosť $P(d)$ vrátenia dokumentu d.

To, či je dokument užitočný sa následne určuje tým, či je $P(P|d)$ väčšie ako $P(NP|d)$.

2.4 Váhovanie značiek

Ak mám dokument, alebo profil reprezentovaný značkami, je potrebné zistiť ako veľmi sú tieto značky preferované. Nie všetky značky majú rovnakú váhu, takže potrebujeme dosiahnuť, aby užitočnosť dokumentu závisela od unikátnosti značiek v ňom. Základy váhovania sú popísané v prezentácií Heinrich Schütze [9].

Frekvencia pojmov

Frekvencia pojmov (angl. Term Frequency ďalej TF), je jeden z najjednoduchších a najstarších prístupov k váhovaniu značiek, pri tomto prístupe sa jednoducho počíta počet výskytov značiek v dokumente. Existuje niekoľko druhov tohoto váhovania.

Binárne váhovanie znamená, že napríklad nepočítame počet výskytov slova v texte, ale berieme iba či sa v texte nachádza, alebo nie. Matematicky vyjadrené rovnicou 3.

$$w_{TF_BIN}(t_i) = \begin{cases} 1 & \text{ak } t_i \in d_j \\ 0 & \text{ak } t_i \notin d_j \end{cases} \tag{3}$$

Kde d_j je dokument a t_i je slovo.

Čistá frekvencia sa dá tiež použiť. V tomto prípade sa váha slova určuje podľa počtu jeho výskytov v texte. Matematicky vyjadrené rovnicou 4

$$w_{TF_RAW}(t_i) = t_{i_{d_j}} \tag{4}$$

Kde $t_{i_{d_j}}$ je počet výskytov slova t_i v dokumente d_j.

Logaritmická váha sa taktiež používa najmä kvôli tomu, že relevancia dokumentu nerastie proporcionálne s počtom výskytov slova v dokumente. Toto môžeme matematicky vyjadriť napríklad rovnicou 5.

$$w_{TF_LOG}(t_i) = \begin{cases} 1 + \log 10 t_{i_{d_j}} \text{ ak } t_{i_{d_j}} > 0 \\ 0 \qquad\qquad\qquad\quad \text{inak} \end{cases} \tag{5}$$

Existuje ešte viac spôsobov ako sa dá vyhodnotiť frekvencia pojmov, medzi ktoré patrí napríklad dvojitá normalizácia 0.5 (angl. double normalization 0.5) alebo k-dvojitá normalizácia (angl. double normalization K) [9].

Frequencie pojmov, inverzná frekvencia dokumentov

Frequencia pojmov, inverzná frekvencia pojmov (angl. Term Frequency, Inverse Document Frequency ďalej TF*IDF) je prístup, pri ktorom zahrňujeme do užitočnosti aj počet dokumentov, ktoré majú danú značku. Základom je zníženie užitočnosti často sa vyskytujúcim značkám. Toto znižovanie je reprezentované rovnicou 6 ďalej IDF.

$$idf_i = log10\frac{N}{dt_i} \tag{6}$$

Kde dt_i je počet dokumentov v ktorých sa pojem t_i nachádza.

Výsledná rovnica TF*IDF je rovnica 7 ktorá je vlastne súčin TF a IDF.

$$w_{TF*IDF} = w_{TF} * log10\frac{N}{dt_i} \tag{7}$$

Presonalizované BM25 váhovanie

Daný model je jeden zo štatistických modelov. Tu uvedená verzia je jeho modifikácia podľa S. Cronen a spol. [2], ktorá je matematicky reprezentovaná rovnicou 8.

$$w_{BM25}(t_i) = \log \frac{(r_{t_i} + 0.5)(N - n_{t_i} + 0.5)}{(n_{t_i} + 0.5)(R - r_{t_i} + 0.5)} \tag{8}$$

Kde N je počet všetkých dokumentov, n_{t_i} je počet dokumentov obsahujúcich pojem t_i, R je počet dokumentov, ktoré používateľ už navštívil a r_{t_i} je počet dokumentov, ktoré už používateľ navštívil obsahujúcich pojem t_i

2.5 Evolúcia používateľských preferencií

Preferencie používateľa sa časom menia, môžu vznikať nové a zanikať staré, prípadne sa vracať predošlé. Na základe toho môžeme používateľské preferencie

rozdeliť na:

- krátkodobé preferencie,
- dlhodobé preferencie,
- sezónne preferencie.

Odhalenie krátkodobých záujmov je pomerne triviálne, stačí agregovať používateľové záujmy za časové obdobie, ktoré považujeme za „krátku dobu“ a vrátiť značky, ktoré používateľ preferoval najčastejšie.

Dlhodobé záujmy

Problém dlhodobých záujmov je o dosť komplikovanejší. Väčšina riešení, ktoré sme preskúmali používala na tento problém kombináciu rôznych váhovacích algoritmov a štatistických metód. Napríklad v článku [7], kde použili už spomínané váhovacie algoritmy.

Používateľský profil je reprezentovaný trojrozmerný váhovaným vektorovým profilom a zoznamom zobrazených dokumentov (navštívené url). Následne sa najskôr vytvorí zoznam adekvátnych dokumentov zo značiek. Následne na porovnávanie značiek dokumentov a profilov sa použíjú tri rôzne algoritmy.

Jednoduché porovnávanie, kde sa vlastne spočítajú váhy značiek, ktoré majú spoločné používateľ a dokument podľa vzorca 9.

$$u_j(d_i) = \sum_{t=1}^{N} {}_{z_t} f(z_t) * u(z_t) \tag{9}$$

Vysledkom je funkcia užitočnosti pre dokument i $u_j(d_i)$, N_{z_i} je počet unikátnych značiek v dokumente d_i, $f(z_t)$ je počet výskytvo značky z_t v dokumente a $u(z_t)$ je vypočítaná užitočnosť značky z_t.

Porovnávanie unikátnych značiek, teda zanedbávame váhu určenú váhovacími algoritmami a iba spočítame unikátne značky podľa vzorca 10.

$$u_u(d_i) = \sum_{t=1}^{N} {}_{z_i} u(z_t) \tag{10}$$

Jazykový model (angl. Language model), ktorý generuje unigramový jazykový model (angl. unigram language model), kde užitočnosť značiek je použitá ako frekvencia značiek v rovnici 11.

$$u_{lm}(d_i) = \sum_{t=1}^{N} z_i \log u(z_t) + 1 \sum_{j=1}^{N} z_i \tag{11}$$

Ďalej sa ešte výsledné dokumenty poskytnuté jedným z týchto algoritmov filtruju algoritmom PClick, ktorý pracuje s históriou navštívených dokumentov. Tento algoritmus vracia iba dokumenty, ktoré používateľ v histórii často navštevoval. Matematicky je to vyjadrené rovnicou 12. Tento algoritmus berie do úvahy aj vyhľadávací dotaz.

$$u_{pclick}(d_i) = \frac{|Zobrazenia(d_i, u_j, q_n)|}{|Zobrazenia(q_n, u_j)| + \beta} \tag{12}$$

Kde $Zobrazenia(d_i, u_j, q_n)$ je počet zobrazení dokumentu d_i a $Zobrazenia(q_n, u_j)$ je celkový počet zobrazení dokumentu d_i pre vyhľadávací dotaz q_i.

3 Návrh aplikácie

V tejto kapitole sa budeme zaoberať zvolenými technológiami a abstraktným návrhom aplikácie.

3.1 Platforma

Prácu sme sa rozhodoli vypracovať vo forme webovej aplikácie. Daná forma bola zvolená najmä pre väčšiu dostupnosť pre koncového používateľa. Tak isto veľkým plus je, že celá aplikácie beží na strane tvorcu, takže prípadné zmeny v systéme nebude potrebne synchronizovať naprieč viacerými zariadeniami. Ďalšou výhodou je redukovanie problémov so synchronizáciou databáz medzi koncovými zariadeniami.

Taktež protabilita riešenia je pomerne dobrá, nepotrebujeme sa zaoberať rôznymi verziami operačných systémov. Proti tomuto by sa dalo argumentovať rozdielmi medzi internetovými prehliadačmi a potrebu responzívneho dizajnu pre mobilné zariadenia, ale toto všetko sa dá jednoducho vyriešiť voľbou správnach grafických knižníc.

Keďže sa jedná o webovú aplikáciu, bude bežať čiastočne vo forme klient server ako html, css a javascript kód bežiaci vo vyhľadávači používateľa a php kód bežiaci na strane servera.

Klient

Klientská časť je postavená na JavaScripte, HTML a CSS3. Aby bol zabezpečený responzívny dizajn a portabilitu naprieč prehliadačmiy, rozhodoli sme sa použiť knižnicu Bootstrap[3], ktorá ja postavená na CSS3 a jQuery. Vďaka nej nie je potrebné kontrolovať vizuálnu stránku v rôznych prehliadačoch a vytvorenie dizajnu, ktorý dobre spolupracuje s rôznymi rozmermi obrazoviek a rozlíšení je pomerne málo námahavé. Taktiež plánujeme použiť jQuery[4] v prípade potreby implementácie logiky na klientskej strane a CSS3[5] v prípade potreby hlbších

[3] http://getbootstrap.com/
[4] https://jquery.com/
[5] http://www.css3.info/

zásahov do grafiky.

Server

Na strane servera sme sa rozhodoli použiť jazyk PHP5[6] najmä z dôvodu doterajších skúseností, tak isto pre tento jazyk sa dá ľahko vyhľadať hosting a má veľmi rozsiahlu dokumentáciu. Avšak v tomto jazyku je nedostatok štruktúry, čo niekedy môže viesť k veľkým neudržateľným aplikáciam, preto sme sa rozhodoli použiť aplikačný rámec (angl. framework) Yii2[7], ktorý má sadú odporúčaných praktík vďaka ktorým je aplikácia dlhodobo udržateľná. Ďalšou výhodou je, že tento aplikačný rámec má v sebe zapracované všetky hore uvedené klientské knižnice, čím uľahčuje prácu s nimi. Tak isto nám tento aplikačný rámec umožňuje používať rôzne databázy vďaka svojmu modulu DAO[8] pomocou ktorého modeluje SQL dotazy. Avšak niektoré zložitejšie dotazy sa v ňom nedajú namodelovať. Ako databázu sme zvolili MySQL[9] vďaka jej rýchlosti, popularite, ale najmä dostupnosti.

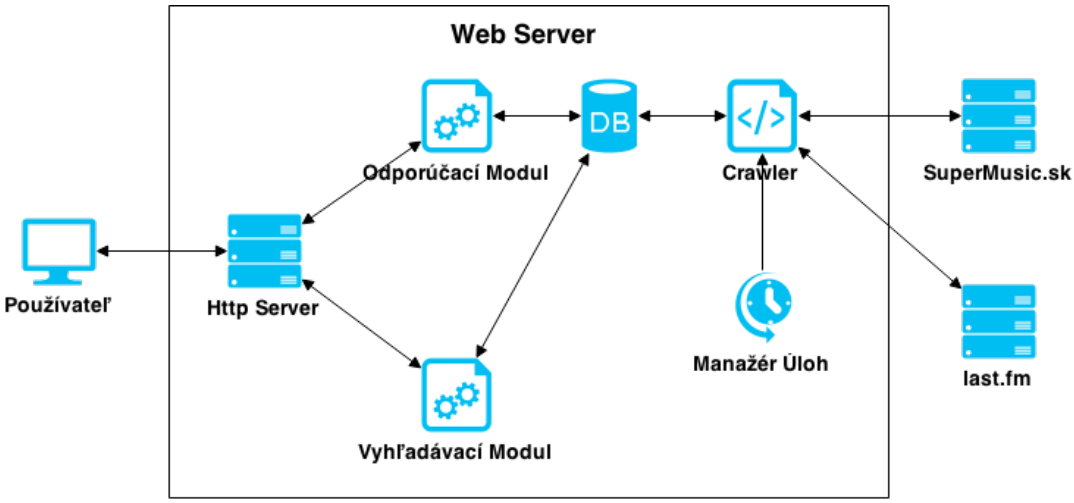
Vývojová platforma

Ako správcu verzií sme sa rozhodoli použiť git[10]. Tento nástroj je pomerne jednoduchý na ovládanie a spoľahlivý. Tak isto existuje množstvo internetových služieb, kde sa pomocou neho dá zálohovať. Verziovanie dopĺňam správou balíkov pomocou manažéra balíkov Composer[11]

Na testovanie sme použili na PHPUnit postavený Codeception, ktorý je integrovaný do Yii2.

Ako editor používame Vim s množstvom prídavných modulov.

[6] http://php.net/
[7] http://www.yiiframework.com/
[8] https://github.com/yiisoft/yii2/blob/master/docs/guide/db-dao.md
[9] https://www.mysql.com/
[10] http://git-scm.com/
[11] https://getcomposer.org/



Obr. 4: *Štruktúra aplikácie*

3.2 Základné rozvrhnutie aplikácie

Keďže ide o webovskú aplikáciu, na najvyššej úrovni bude postavená na vzore Klient-Server. Avšak z dôvodu bezpečnosti sa bude väčšina úloh vykonávať na serveri. Aplikácia bude získavať dáta z už existujúcej databázy hudobných dokumentov[12]. Tieto dáta budú doplnené o sociálne tagy zo služby last.fm[13]. Nad týmito dátamy sa bude vykonávať vyhľadávanie, ale najmä odporúčanie. Základný náčrt aplikácie môžeme vidieť na obrázku 4.

Aplikácia sa bude skladať z dvôch hlavných oddelených subsystémov

- odporúčanie (spúšťané používateľom cez prehliadač),
- indexovanie dokumentov (séria úloh spúšťaná manažérom úloh).

3.3 Špecifikácia služieb

Aplikácia by mala poskytovať niekoľko základných služieb:

- vyhľadávanie dokumentov,

[12] http://www.supermusic.sk/
[13] http://www.last.fm/home

- nájdenie podobných dokumentov aktuálne zobrazenému dokumentu,
- odporúčanie dokumentov,
- zostavenie spevníka.

Prvé dve funkcionality sú vytvorené najmä z dôvodu, že v danej doméne sa ťažko získavajú preferencie. Ich jedinou úlohou je získanie základných preferencií používateľa na základe ktorých budeme vytvárať odporúčania.

Pre zabezpečenie týchto funkcií bude potrebované udržiavať databázu dokumentov, ktorá sa bude pravidelne aktualizovať. Toto bude zabepečovať Kravler (angl. crawler), ktorý bude vykonávať činnosti:

- prehľadávanie supermusic po nových dokumentoch,
- prehľadávanie supermusic po nových interprétoch,
- sťahovanie dokumentov zo supermusic,
- sťahovanie značiek z last.fm,
- váhovanie značiek.

Kravler bude implementovaný ako aplikácia pre príkazový riadok aby sa dal použiť s manažérom úloh.

Odporúčanie

Na odporúčanie by sme chceli využiť filtrovanie na základe obsahu spolu s vlastným algoritmom na určenie dlhodobých, krátkodobých a sezónnych záujmov.

4 Implementácia

4.1 Datový model

Model dokumentu

Dokument bude reprezentovaný svojím názvom, typom a značkami, pričom značky budú mať doménu na základe toho, či vznikli z názvu dokumentu, jeho typu, názvu interpréta, alebo boli získane zo služby last.fm.

Čiže dokument bude uložený v trojrozmernom vektorovom priestore, kde prvá súradnica budú dokumenty $D = d_1, d_2, ...d_n$, ďalšia súradnica budú značky $Z = z_1, z_2, ..z_n$ a posledná súradnica budú domény $D = d_{názov}, d_{interpret}, d_{typ}, d_{tag}$.

Model používateľa

Používateľov profil bude reprezentovaný históriou navštívených značiek. Bude sa ukladať pre každú navštívenu značku a jeden riadok bude obsahovať informácie $row = d_i, u_j, z_l, čas$ kde d_i je zobrazený dokument, u_j je používateľ, ktorý dokument zobrazil, z_l je značka ktorá bola zobrazená.

Celú túto schému môžeme vidieť na obrázku 5.

4.2 Indexovanie dokumentov a váhovanie

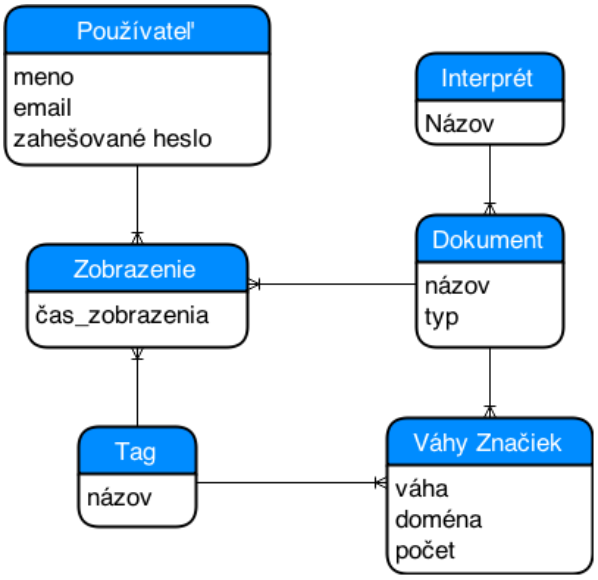
Kravler

Kravler (angl. crawler) je program slúžiaci na prehľadávanie webu za účelom indexovania[14]. Kravler vykonáva niekoľko operácií v rámci aplikácie, je implementovaný ako jednoduchý program príkazového riadku. Každá z jeho operácií sa v prípade potreby dá vykonať osobitne. Toto je hlavne potrebné ak by sa zmenili parametre indexovania. Na obrázku 6 môžeme vidieť diagram akcií kravleru.

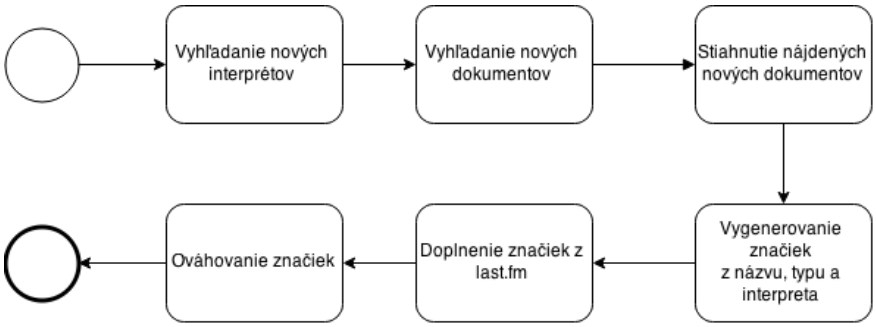
Vyhľadanie nových interpretov

Táto akcia využíva spôsob, akým sú na super music triedení interpreti. Interpreti sa zobrazujú na stránke „http://www.supermusic.sk/skupiny.php?od=“ pričom

[14] http://en.wikipedia.org/wiki/Web_crawler



Obr. 5: *Zjednodušený datový model*



Obr. 6: *BPMN Digram kravleru*

parameter *od* sú počiatočné písmena názvu kapely. Experimentálne sme overili, že na prejdenie celej databázy potrebujem vygenerovať všetky trojpísmenové začiatky názvov, pričom nezáleží na malých a veľkých písmenách. Pre generovanie používame anglickú abecedu až na prvé písmeno, kde musím brať do úvahy aj niektorú diakritiku.

Prvé písmeno sa vyberá z množiny „A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, R, S, T, U, V, X, Y, *, Ž, Ť, Č“. Tento príkaz sa dá spustiť aj paralelne, kedy vytvorí niekoľko procesov, avšak príkaz nie je spoľahlivý, vzhľadom na to, že v rámci zachovania portability sme použili iba základnu knižnicu pcntl. Táto knižnica neobsahuje zámky, ani iný spôsob ako ošetriť prítupy ku zdrojom.

Kravler pre interpretov využíva XPath výrazy [1, 2, 3] na získanie dát zo stiahnutej stránky.

Ukážka 1: *XPath na vyhľadanie interpretovho identifikátora*

```
//table[@bgcolor="#333333" and position() = 2]//a/@id
```

Ukážka 2: *XPath na vyhľadanie interpretovho mena*

```
//table[@bgcolor="#333333" and position() = 2]//a/@href
```

Ukážka 3: *XPath na vyhľadanie interpretovho aliasu*

```
//table[@bgcolor="#333333" and position() = 2]//a/text()
```

Tieto hodnoty sa ešte ďalej spracúvajú, napríklad vytiahnutie pri druhom XPath-te 2 získame url z ktorej musíme aktuálne meno získať pomocou regulárneho výrazu.

Ukážka 4: *Regulárny výraz na získanie mena interpreta z url interpreta*

```
/&name=.*\$/
```

Vyhľadanie nových dokumentov

Táto akcia funguje podobne ako akcia Vyhľadanie nových interpretov. Tiež generuje adresy z abecedy a adresy „http://www.supermusic.sk/piesne.php?od=“. pričom dopĺňa písmená na koniec adresy.

Prvý znak je anglická abeceda rozšírená o niekoľko slovenský a českých interpunkčných písmen plus hviezda. Teda v prvom rade sú písmená z množiny „A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, X, Y, Z, Č. Ď, Ľ, Ř, Š, Ť, Ž, *“, v druhej rade už je anglická abeceda.

Následne sa snaží získať zoznam piesní zo stiahnutej stránky, za týmto účelom používame XPath výrazy [6, 5, 8, 7].

Ukážka 5: *XPath na vyhľadanie názvov piesní*

```
//table[@width=740]//td/a/text()
```

Ukážka 6: *XPath na vyhľadanie názvov piesní*

```
//table[@width=740]//td/a/@href
```

Ukážka 7: *XPath na vyhľadanie názvov piesní*

```
//table[@width=740]//td/imt/@src
```

Ukážka 8: *XPath na vyhľadanie názvov piesní*

```
//table[@width=740]//td/text()
```

Stiahnutie nových dokumentov

Počas hľadania nových dokumentov sa ešte nesťahujú obsahy dokumentov. Takže všetky nové dokumenty ešte nemajú žiadny obsah, táto akcia si vyberie nové dokumenty a následne ich sťahuje, url dokumenty sa dájú získať tak, že zoberieme url „http://www.supermusic.sk/skupina.php?action=piesenidpiesne=“ a na jej koniec pridám identifikátor dokumentu.

Následne sa dokument spracuje a ak to je potrebné, prispôsobí sa aplikácii.

Vygenerovanie značiek

Aby bolo možné odporúčať, je potrebné vygenerovať značky pre dokumenty. Značky sa generujú z názvu dokumentu, názvu interpréta a typu dokumentu tak, že sa rozdelia na slová. Následne sa odstránia slová so slabou semantickou silou[15]

[15] http://en.wikipedia.org/wiki/Stop_words

(angl. stopwords). Na filtrovanie takýchto slov použije sa už hotový zoznam slóv vytvorený v rámci projektu stop-words[16] Z tohoto projektu využívame sady:

- stop-words_czech_3_cz,
- stop-words_english_3_en,
- stop-words_slovak_2_sk.

Text sa ešte predtým normalizuje a tak isto sa určí doména značky, pre jednoduchosť ukladania neukladáme značku pre každú doménu osobitne, ale máme vytvorený číselník, ktorý priraďuje presné domény značkám:

- 8 znamená, že značka je názov dokumentu aj názov interpreta,
- 7 znamená, že značka je názov dokumentu a značka z názvu interpreta,
- 6 znamená, že značka je názov interpreta a značkou z názvu dokumentu,
- 5 znamená, že značka je názov dokumentu,
- 4 znamená, že značka je názov interpreta,
- 3 znamená, že značka je značkou v názve dokumentu aj interpreta,
- 2 znamená, že značka je značkou v názve dokumentu,
- 1 znamená, že značka je značkou v názve interpreta,
- 0 znamená, že značka je buď značka získaná z typu dokumentu alebo služby last.fm.

Doplnenie značiek z last.fm

Na dopĺňanie značiek používam službu track.getTopTags z last.fm apy, ktorá vráti pre interpreta a názvu piesne najčastejšie prideľované značky. Tie následne pridáme k danej piesni. Keďže počet pridelení vrátených z last.fm vysoko prebíja počty vytvorené z názvu dokumentu a interpreta, rozhodoli sme sa túto hodnotu nevyužiť.

[16] https://code.google.com/p/stop-words/

Váhovanie značiek dokumentov

Tento model sa v MySQL nazýva model prirodzeného jazyka (angl. Natural Language Model), ktorý porovnáva vlastnosti dokumentov na základe abstrakcie priestoru, v ktorom sú jednou dimenziou vlastnosti jedného dokumentu a druhou vlastnosti druhého dokumentu, prípadne vyhľadávacieho reťazca, alebo používateľský profil. Následne sa vracajú dokumenty, ktoré majú najpodobnejší smer vektora k požadovanej fráze.

V MySQL[17] je tento prístup implementovaný pomocou nasledujúcej rovnice 13.

$$w_d = \frac{\log(dtf_d) + 1}{\sum_{i=1}^{t} \log(dtf_i) + 1} . \frac{U}{1 + 0.0115 * U} . \log \frac{N}{nf} \tag{13}$$

Rovnica obsahuje:

- dtf_d je množstvo koľkokrát sa nachádza značka v dokumente,
- dtf_i množstvo i-tej značky dokument,
- U počet unikátnych značiek dokumentu,
- N počet všetkých dokumentov,
- nf je počet dokumentov, ktoré obsahujú danú značku

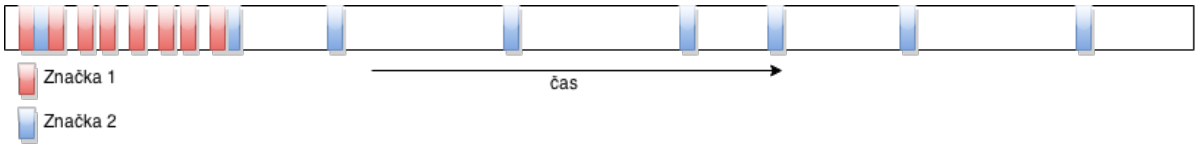
Rovnica sa dá rozdeliť na tri časti:

- **základná časť** je to primárna rovnica určujúca váhu pojmu,
- **normalizačný faktor** spôsobí, že ak je dokument kratší ako priemerná dĺžka dokumentu, jeho relevancia stúpa, [10]
- **inverzná frekvencia** zabezpečuje, že menej časté pojmy majú vyššiu váhu.

4.3 Odporúčanie

V rámci odporúčania je vytvorených niekoľko algoritmov.

[17] http://dev.mysql.com/doc/internals/en/full-text-search.html



Obr. 7: *Zobrazenia značiek na časovej osi.*

Agregované odporúčanie

Ako referenčné odporúčanie sme zvolili agregovanie všetkých zobrazených značiek, a vrátenie tých, ktoré používateľ navštevoval najčastejšie. Matematicky je tento algoritmus vyjadrený rovnicou 14,

$$u(d_i, u_k) = \sum_{j=0}^{j=N_{d_i u_k}} u(z_j) \tag{14}$$

kde $u(d_i, u_k)$ je užitočnosť dokumentu d_i pre používateľa u_k, ďalej $N_{d_i u_k}$ čo je spoločný počet značiek dokumentu d_i a používateľa u_k a $u(z_j, d_i)$, čo je užitočnosť značky z_j vzhľadom na dokument d_i.

Vlastné odporúčanie

Pri agregovanom odporúčaní však nastáva jeden zásadný problém, ak používateľ v istom čase zobrazí jeden dokument veľakrát, môže to tak ovplyvniť jeho záujmy na dlhší čas. Napríklad na obrázku 7 by značka 1 aj značka 2 mala rovnakú váhu.

Na vyriešenie tohoto problému sme sa rozhodoli implementovať algoritmus, pri ktorom si rozdelíme históriu zobrazení značiek na n rovnakých častí, značky v každej časti si zoradíme podľa počtu zobrazení a vrátim si i najzobrazovanejších značiek. Pri každej časti považujeme pozíciu, na ktorej sa značka umiestnila za jej váhu.

Následne pre každú značku spočítame jej váhy zo všetkých skupín, v ktorých sa vyskytla. Túto váhu nakoniec ešte zlogaritmuje z dôvodu, že značky, ktoré boli na prvých priečkach zvykli mať obrovský náskok oprotí nižším. Na ukážke 9 môžeme vidieť pseudo kód tohoto algoritmu.

Ukážka 9: *Získanie značiek profilu*

```
Znacky ziskajZnackyProfilu(zobrazeneZnacky):
    znackyProfilu = []
    velkostJednejSkupiny =
        pocet(zobrazeneZnacky) / configuracia('pocetSkupin')

    for i=0; i<pocet(zobrazeneZnacky); i+= velkostJednejSkupiny:
        zoskupeneZnacky = zoskup(
            zobrazeneZnacky, podla=znacky, pocet(*)
        )
        zoradeneZnacky = zorad(zoskupeneZnacky, podla=pocet(*))
        for j=0; j<configuracia('vrchSkupiny'); j++:
            znackyProfilu[zoradeneZnacky[j].nazov] += j;

    foreach znackyProfilu as znackaProfilu:
        znackaProfilu = log(znackaProfilu)
```

Skupinové odporúčanie

Aplikácia tak isto umožňuje vytváranie spevníkov pre viacerých používateľov. Pre zjednodušenie táto funkcionalita je implementovaná v dvoch krokoch. V prvom kroku sa získaju a ováhujú značky vybranej skupiny používateľov, v druhom sa vrátia najužitočnejšie dokumenty na základe značiek. Vyhodnotenie užitočnosti tagov zo zobrazení sa vypočítava podľa rovnice 15,

$$u(z_i) = \frac{N_{u_i}}{N_u} * \log(N_{z_i} + 1) \tag{15}$$

kde $u(z_i)$ je užitočnosť, N_{u_i} je počet používateľov, ktorí už zobrazili značku z_i, N_u je počet používateľov pre ktorých chcem získať značky a $_{z_i}$ je celkový počet zobrazení značky.

Podobnosť dokumentov

Pre vylepšenie odporúčaní počas prieskumného vyhľadávania, berieme do úvahy aj dokument, ktorý sa najviac podobá na aktuálny dokument. Podobnosť dokumentov je vyhodnotená pomocou rovnice 16.

$$podobnost(d_i, d_j) = \sum_{k=0}^{N_{ij}} u(d_i, z_k) * u(d_j, z_k) \tag{16}$$

kde N_{ij} je počet spoločných značiek dokumentu d_i a dokumentu d_j a d_i, z_k je užitočnosť značky z_k vzhľadom na dokument d_i.

Porovnávanie s dokumentom

Dokument môžem porovnať s akoukoľvek sadou ováhovaných značiek, či už vytvorenej z profilu používateľa, alebo z vyhľadávacieho reťazca. Toto porovnávanie je vyjadrené rovnicou 17.

$$vyhovuje(d_i, Z) = \sum_{k=0}^{N_z} u(d_i, z_k) * qu(z_k) \tag{17}$$

kde Z je sada značiek, pre ktorú sa snažím nájsť vyhovujúci dokument. N_z je počet značiek, ktoré má dokument a sada značiek na porovnanie rovnaké, $u(d_i, z_k)$ je užitočnosť značky k vzhľadom na dokument d_i a $qu(z_k)$ je užitočnosť značky z_k v sade značiek Z.

5 Testovanie

Cieľom overovania bolo zistiť či algoritmus na odhaľovanie dlhodobých preferencií postavený na obdobiach dokáže poskytnúť lepšie výsledky ako agregácia hodnôt.

Vlastné testovanie

Preto som sa rozhodol najskôr vytvoriť jedného testovacieho požívateľa a sledovať ako sa s časom bude menit účinnosť oboch algorimtov.

Pri tomto testovaní sa porovnával používateľov zámer s niekoľkými odporučeniami algoritmov. Zámer bol tvorený skupinou kapiel alebo typov dokumentov. Vzhľadom na unikátnosť piesní sme sa rozhodoli nepoužiť pri testovani tento parameter.

Podobnosť užitočnosti odporúčaní sme hodnotili podľa toho, koľko kapiel a typov dokumentov sa nachádza aj v zámere aj vo výsledku. Tento prístup je vyjadrený matematicky rovnicou 18.

$$užitočnosť(R) = \frac{\sum_{i=0}^{N} 1 + \sum_{i=0}^{M} 1 + \sum_{i=0}^{K} 1}{3} \tag{18}$$

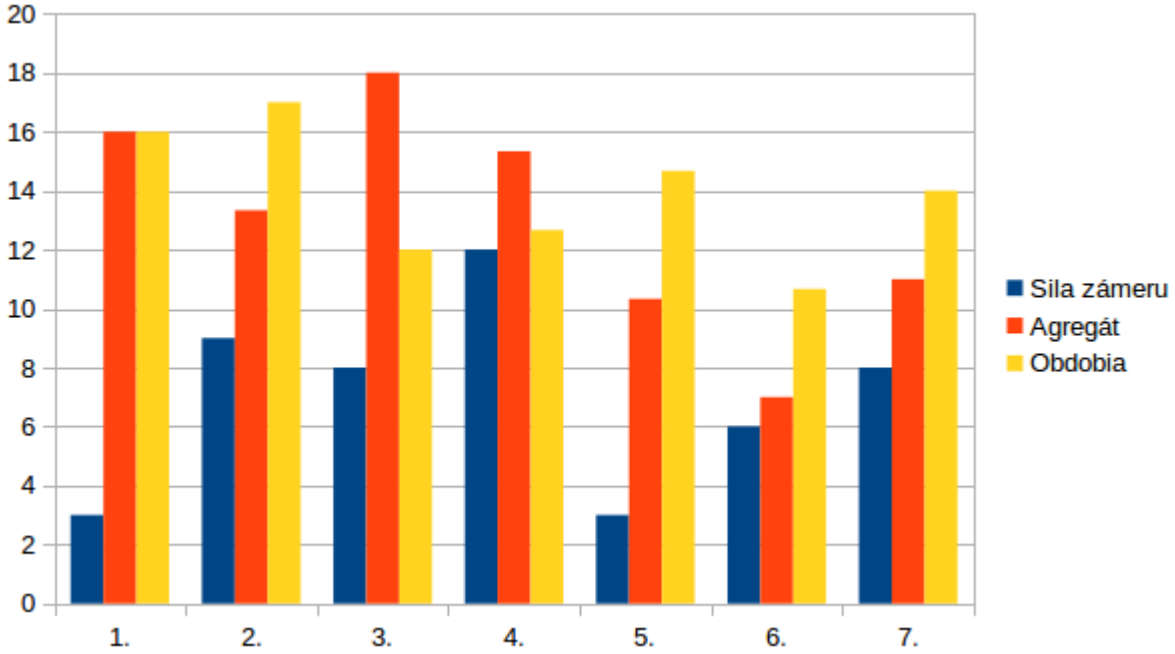
Rovnica obsahuje R čo je množina odporúčaní, N, čo je počet piesni ktoré sú aj v zámere aj v odporúčaní, M je počet interpretov, ktorí sú aj v zámere aj v odporučení a K je počet typov dokumentov, ktoré su tiež prienikom odporučení a zámeru.

Na obrázku 8 môžeme vidieť výsledok tohoto testu.

Pokračovať v testoch nemalo zmysel vzhľadom na to, že agregačný algoritmus v predposlednom aj v poslednom teste vrátil rovnaké výsledky. Čo vzhľadom na jeho povahu znamená, že menšie zámery už nemajú vplyv na jeho výsledok.

5.1 Dotazník

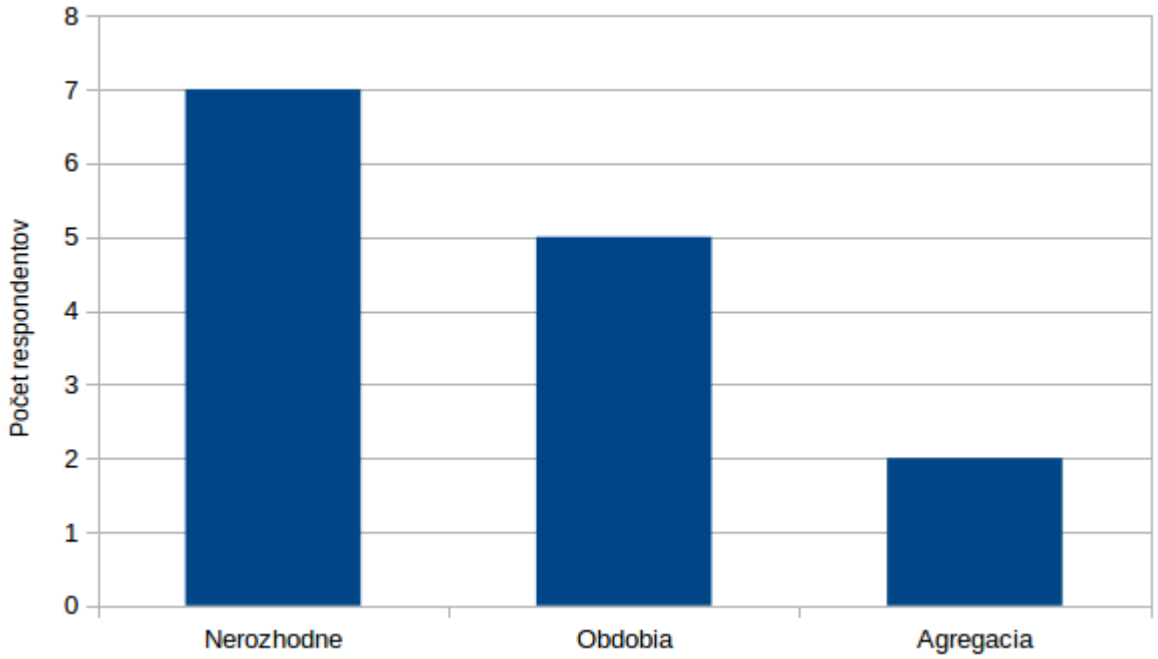
Ďalšie testovanie prebehlo formov testovania aplikácie na webe pomocov respondentov, ktorý používali aplikáciu rôzne časove obdobia, následne sa im boli



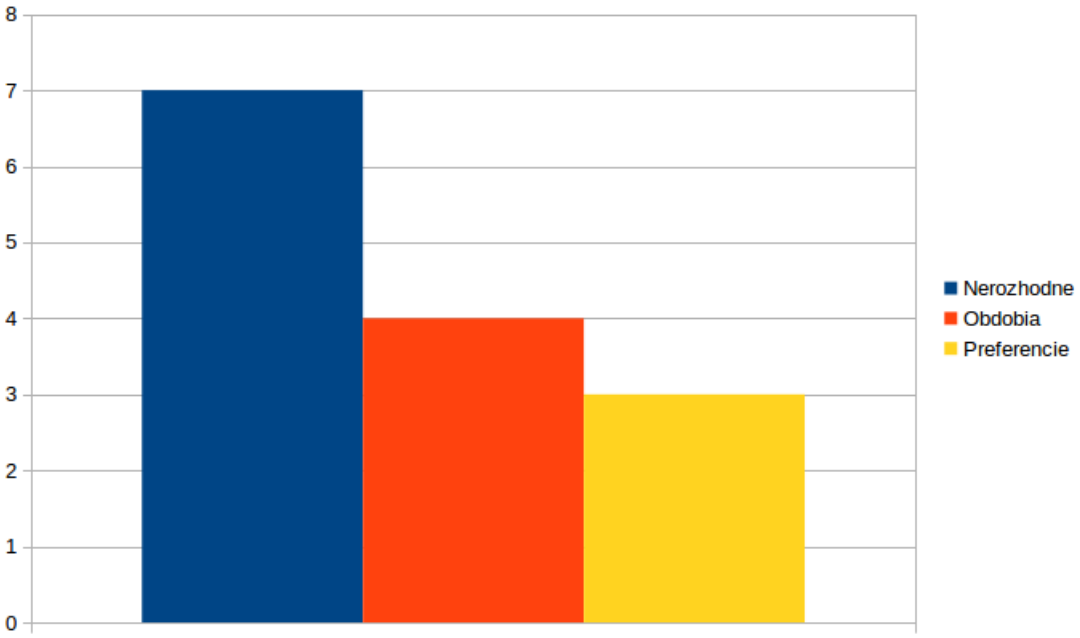
Obr. 8: *Graf výsledkov pre manualny test*

ponúknute odporúčania z obidvoch algoritmov a oni vyhodnotili ktorí im viacej vyhovuje. Na obrázku 9 môžeme vydieť výsledky dotazníku.

Okrem tohoto porovnávania som ešte zisťoval či na stránke dokumentu používatelia viac preferovali panel s podobnými dokumentamy alebo panel z odporúčaním na báze obdobý. Výsledky môžeme vidieť na obrázku 10



Obr. 9: *Graf výsledkov pre dotazník*



Obr. 10: *Graf výsledkov pre dotazník*

6 Zhodnotenie

Počas tejto práce sme porovnávali algoritmus, ktorý berie do úvahy starnutie používateľovho profilu s niektorými inými odporúčacími algoritmami a to najmä agregáciou zobrazení a odporúčaním najpodobnejšieho dokumentu.

V oboch prípadoch sme dosiahli riešenie iba s mierne lepšími výsledkami a je pravdepodobné, že pri dlhodobejšom používaní by tiež sa mohola stratiť citlivosť na zmeny v používateľových preferenciach.

6.1 Možné vylepšenia

Algoritmus by bolo možné vylepšiť rôznymi spôsobmi, napríklad využitím exponenciálneho rozpadu, kedy by sa ukladala iba jedna väzba používateľa a značky. Starnutie by sa zabezpečilo menením rýchlosti starnutia.

Ďalším možným vylepšením je uvažovanie sezónnych záujmov a kombinácia s krátkodobými záujmamy. Kedy by napríklad v tomto prípade bolo možné namapovanie zobrazených značiek podľa rokvo, nasledne by sme mohli pre každú značku robiť klastrovanie v tomto priestore.

Literatúra

[1] Peter Brusilovsky. User profiles for personalized information access. School of Information Sciences University of Pittsburgh, USA, 2009.

[2] S. Cronen-Townsend and W. B. Croft. Quantifying query ambiguity. 2002.

[3] Alexander Dekhtyar. Probabilistic information retrieval part 1: Survey. [navštívené 10 januara 2015], 2000.

[4] Franco Fabbri. A theory of musical genres: Two applications. 1980. [navštívené 27 apríla 2015].

[5] Paul Lamere. Social tagging and music information retrieval. 2009. [navštívené 24 apríla 2015].

[6] Namunu C. Maddage, Li Haithou, and Mohan S. Kankanhalli. Music structure analysis statistics for popular songs. November 2009.

[7] Nicolaas Matthijs and Filip Fadlinski. Presonalizing web search using long term borwsing history.

[8] Francesc Ricci, Lior Rokach, Bracha Shapira, and Paul B. Kantor. *RECOMMENDER SYSTEMS HANDBOOK*, chapter Recommendation Techniques, pages 11 – 14. Springer New York Dordrecht Heidelberg London. [navštívené 1 máj 2015].

[9] Heinrich Schütze. Introduction to information retrieval. [navštívené 6 januara 2015], 2011.

[10] Amit Singhal, Chris Buckley, and Mandar Mitra. Pivoted document length normalization. 1996.

[11] John F. Sowa. Semantic networks. [navštívené 4 februára 2015].

A Technická dokumentácia

A.1 Špecifikácia požiadaviek

Nefunkčné požiadavky

Na systém sú kladené nasledujúce nefunkčné požiadavky

itemize

Rýchlosť hlavným sledovaným parametrom pri webových aplikáciach je čas načítania stránky, podľa štatistík, stránka ktorá sa načíta nad 2 sekundy má veľkú pravdepodobnosť, že sa používateľ na ňu nevráti[18],

Bezpečnosť je dôležitá najmä skrz to, že stránka bude profilovať používateľov, čo by sa mohlo dať zneužiť. Je potrebné použiť vhodný spôsob prihlasovania a ukladania prihlasovacích údajov,

Stabilita aplikácia by mal vydržať fungovať dostačujúce časové obdobie bez výskytu väčšej chyby,

Modularita by mala byť zabezpečená pre prípad budúcich úprav aplikácie.

Aplikácia by mala ďalej byť schopná správne pracovať na platforme PHP verzie vyššej ako 5.4 pričom je predpokladaná prítomnosť GD a mcrypt rozšírenia. Po vizuálnej stránke by sa aplikácia mala dobre zobraziť vo všetkých moderných prehliadačoch a to responzívne. Ďalej aplikácia bude potrebovať prítomnosť databázy MySQL verzie aspoň 5.1, prípadne PostrgreSQL.

Funkčné požiadavky

Aplikácia bude poskytovať možnosť vyhľadania piesne, pričom bude následne poskytovať odporúčania, vďaka čomu umožní prieskumné vyhľadávanie. Ďalej bude používateľovi umožňovať nechať si odporučiť dokumenty na základe ním zobrazených dokumentov a vytvorenie spevníka pre kombináciu niekoľkých používateľov.

Aplikácia bude mať tak isto administračné rozhranie, kde bude možné zobrazovať a spravovať (mazať a upravovať) zaregistrovaných používateľov.

[18] http://www.mcrinc.com/Documents/Newsletters/201110_why_web_performance_matters.pdf

A.2 Návrh projektu

Projekt ja implementovaný v aplikačnom rámci Yii2. V rámci tohoto aplikačného rámca je použitých niekoľko návrhových vzorov. Základom je návrhový vzor MVC[19], ktorý následne na správu dát dopĺňa návrhovým vzorom ActiveRecord[20], ktorý slúži na prácu s dátami. Tak isto program umožňuje použiť buď LazyLoading[21], alebo EagerLoading[22].

Kód je napísaný tak, že každý objekt má vlastný súbor a zdrojové súbory sú rozdelené podľa toho, čo reprezentujú (modely, pohľady, kontrolery, widgety, asety, komponenty, kravlery, migracie a testy).

A.3 Implementácia

Funkcie v modeli dokumentu

Tieto funkcie sa nachádzajú v súbore models/Document.php

Ukážka 10: *Funkcia na porovnanie dokumentu so sadou ováhovaných značiek*

```
public static function match($tags, $exclude = false) {
    $case = Globals::sqlCase($tags,'tag.name');

    $subQuery = (new Query)->select('id')
        ->from('tag')
        ->where(['name' => array_keys($tags)]);

    $query = (new Query)->select('document_id')
        ->from('map_document_tag map')
        ->innerJoin('tag', new Expression('tag.id = map.tag_id')»
          » )
        ->where(['tag_id' => $subQuery])
        ->groupBy('document_id')
        ->orderBy(new Expression("SUM(weight* $case ) DESC"))
        ->limit("50");
```

[19] http://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller
[20] http://en.wikipedia.org/wiki/Active_record_pattern
[21] http://en.wikipedia.org/wiki/Lazy_loading
[22] Pravý opak LazyLoading kedy je snaha vybrať všetky data z databázy naraz

```
    if($exclude)
        $query->andWhere(['not in', 'document_id', $exclude]);

    return Document::find()->where(['id' => $query]);
}
```

Ukážka 11: *Porovnanie dokumentu so sadou značiek akurát*

```
public static function matchIds($tags) {

    $case = Globals::sqlCase(ArrayHelper::map($tags, 'tag_id', '»
      » weight'),'tag_id');

    return Document::find()
        ->innerJoin('map_document_tag map', 'map.document_id=»
          » document.id')
        ->where(['map.tag_id' => ArrayHelper::getColumn($tags, '»
          » tag_id')])
        ->groupBy('document.id, document.name')
        ->orderBy(new Expression("SUM(weight * $case) DESC"));
}
```

Ukážka 12: *Funkcia*

```
public static function search($query)
{
    $query_tags = array_count_values(Tag::escape($query));
    return self::match($query_tags);
}
```

Ukážka 13: *Funkcia*

```
public static function getTagType($IN, $DN, $IT, $DT) {
    if($DN && $IN) return 8;
    elseif($DN && $IT && !$IN) return 7;
    elseif($IN && $DT) return 6;
    elseif($DN) return 5;
    elseif($IN) return 4;
    elseif($DT && $IT) return 3;
    elseif($DT) return 2;
    elseif($IT) return 1;
```

```
    else return 0;
}
```

Ukážka 14: *Funkcia*

```
public static function similiarTags($document1, $document2) {
    return Tag::find()
        ->innerJoin('map_document_tag map1', 'map1.tag_id=tag.id»
          » ')
        ->where(['map1.document_id' => $document1->id])
        ->innerJoin('map_document_tag map2', 'map2.tag_id=tag.id»
          » ')
        ->andWhere(['map2.document_id' => $document2->id])
        ->andWhere(new Expression('map1.document_id = map2.»
          » document_id'));
}
```

Ukážka 15: *Funkcia vracajúca podobné dokumenty aktuálnemu dokumentu*

```
public function getSimiliar() {
    return static::find()
        ->innerJoin('map_document_tag map1', 'map1.document_id=»
          » document.id')
        ->innerJoin('map_document_tag map2', ['AND',
                'map2.tag_id=map1.tag_id',
                ['map2.document_id' => $this->id],
                ['<>', 'map1.document_id', $this->id]
            ])
        ->groupBy('document.id')
        ->orderBy(new Expression('SUM(map1.weight * map2.weight)»
          »  DESC'))
        ->having(new Expression('COUNT(*) > 1'));
}
```

Funkcie v modely používateľa

Tieto funkcie sa nachádzajú v súbore models/User.php

Ukážka 16: *Odporúčanie pre viacerých používateľov vráti ováhované značky*

```
public static function recommendFor($ids) {
    $expression = new Expression(
        '((COUNT(DISTINCT user_id) / '.
        count($ids).'.00) * '.
        'LOG(COUNT(*) + 1))'
    );
    return (new Query())
        ->select(['tag_id', 'weight' => $expression])
        ->from('view')
        ->where(['user_id' => $ids])
        ->orderBy($expression)
        ->groupBy('tag_id');
}
```

Ukážka 17: *Algoritmus*

```
public function getTimeAwareRecommendDocuments($exclude = false)»
  »  {
        $view_count = View::find()->where(['user_id' => $this->»
          » id])->count();
    $per_cluster_top = 50 / Yii::$app->params['long_term_groups»
      » '];
    $cluster_size = floor(
        $view_count / Yii::$app->params['long_term_groups']
    );

    $tags = [];
    Yii::info("Counting clusters\n");

    for($i=0; $i<$view_count; $i+=$cluster_size) {
        $querySlice = (new Query)->select('*')
            ->from('view')
            ->limit("$cluster_size")
            ->offset("$i")
            ->orderBy('id');

        $slice_tags = (new Query)->select('tag_id')
            ->from(['cluster' => $querySlice])
            ->groupBy('tag_id')
            ->limit($per_cluster_top)
```

```
            ->orderBy(new Expression('COUNT(*)'))
            ->all();

        for($j=0; $j<$per_cluster_top; $j++) {
            if(array_key_exists($slice_tags[$j]["tag_id"], $tags»
              » ))
                $tags[$slice_tags[$j]['tag_id']] += log10($j + »
                  » 1);
            else $tags[$slice_tags[$j]['tag_id']] = log10($j + »
              » 1);
        }
    }

    $case = "CASE map.tag_id \n";
    foreach($tags as $tag_id => $weight) {
        $case .= "WHEN $tag_id THEN $weight\n";
    }
    $case .= "END\n";

    $query = Document::find()
        ->innerJoin('map_document_tag map',
            new Expression('map.document_id = document.id'))
        ->where(['map.tag_id' => array_keys($tags)])
        ->limit(50)
        ->orderBy(new Expression('SUM(map.weight * '.$case.')'))
        ->groupBy('document.id');

    if($exclude)
        $query->andWhere(['<>', 'document.id', $exclude]);

    return $query;
}
```

Ukážka 18: *Vráti agregáciu počtov zobrazení značiek*

```
public function getRecommendDocuments() {
    $userTagWeights = (new Query)
        ->select(['tag_id', new Expression('LOG(COUNT(*)) AS »
          » weight')])
        ->from('view')
```

```
            ->where(['user_id' => $this->id])
            ->groupBy('tag_id')
            ->having(new Expression('LOG(COUNT(*)) > 0'));

        return Document::find()
            ->innerJoin('map_document_tag map','map.document_id=»
              » document.id')
            ->innerJoin(['user_tag' => $userTagWeights],'user_tag.id»
              » =map.tag_id')
            ->where(new Expression('(user_tag.weight * map.weight) >»
              »  0'))
            ->orderBy(new Expression('(user_tag.weight * map.weight)»
              »  DESC'));
    }
```

Model relačnej tabuľky dokumentov a tagov aj s váhami

Táto trieda sa nachádza v súbore models/MapDocumentTag.php

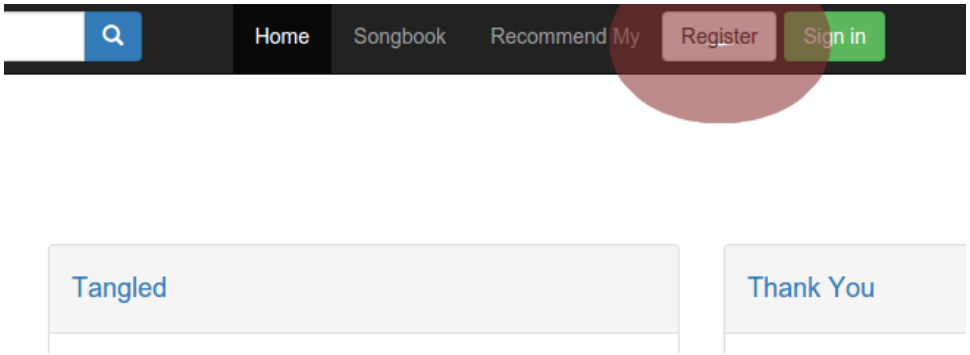
Ukážka 19: *Ohodnotí vytvorené referencie dokumentov a značiek*

```
public static function calculateWeights() {
    $transaction = Yii::$app->db->beginTransaction();
    extract(Yii::$app->params['tag_appereance_weights']);
    try {
        Yii::$app->db->createCommand(
            "UPDATE map_document_tag AS tg2 ".
            "SET weight = ".
            "   (LOG(tg2.count) + 1)/t2.sumdtf * ".
            "   t2.U / (1 + 0.0115*t2.U) * ".
            "   LOG((SELECT COUNT(*) FROM document) / nf) * ".
            "   CASE    WHEN tg2.type_id = 0 THEN $none".
            "           WHEN tg2.type_id = 1 THEN »
              » $document_name_tag".
            "           WHEN tg2.type_id = 2 THEN »
              » $interpret_name_tag".
            "           WHen tg2.type_id = 3 THEN $name_tag".
            "           WHEN tg2.type_id = 4 THEN »
              » $interpret_name".
```

```
            "            WHEN tg2.type_id = 5 THEN $document_name»
              » ".
            "            WHEN tg2.type_id = 6 THEN »
              » $interpret_name_document_tag".
            "            WHEN tg2.type_id = 7 THEN »
              » $document_name_interpret_tag".
            "            WHEN tg2.type_id = 8 THEN $name".
            "   END ".

            "FROM map_document_tag AS tg ".
            "INNER JOIN (".
            "   SELECT document_id, ".
            "       SUM(LOG(count) +1) AS sumdtf, ".
            "       COUNT(tag_id) AS U ".
            "   FROM map_document_tag ".
            "   GROUP BY document_id ".
            ") AS t2 ON t2.document_id = tg.document_id ".
            "INNER JOIN (".
            "   SELECT tag_id, COUNT(document_id) AS nf ".
            "   FROM map_document_tag ".
            "   GROUP BY tag_id ".
            ") AS t3 ON t3.tag_id = tg.tag_id ".
            "WHERE tg2.id = tg.id "
        )->execute();
    } catch (Exception $e) {
        $transaction->rollBack();
    }

    $transaction->commit();
}
```



Obr. 11: *Registrační tlačidlo*

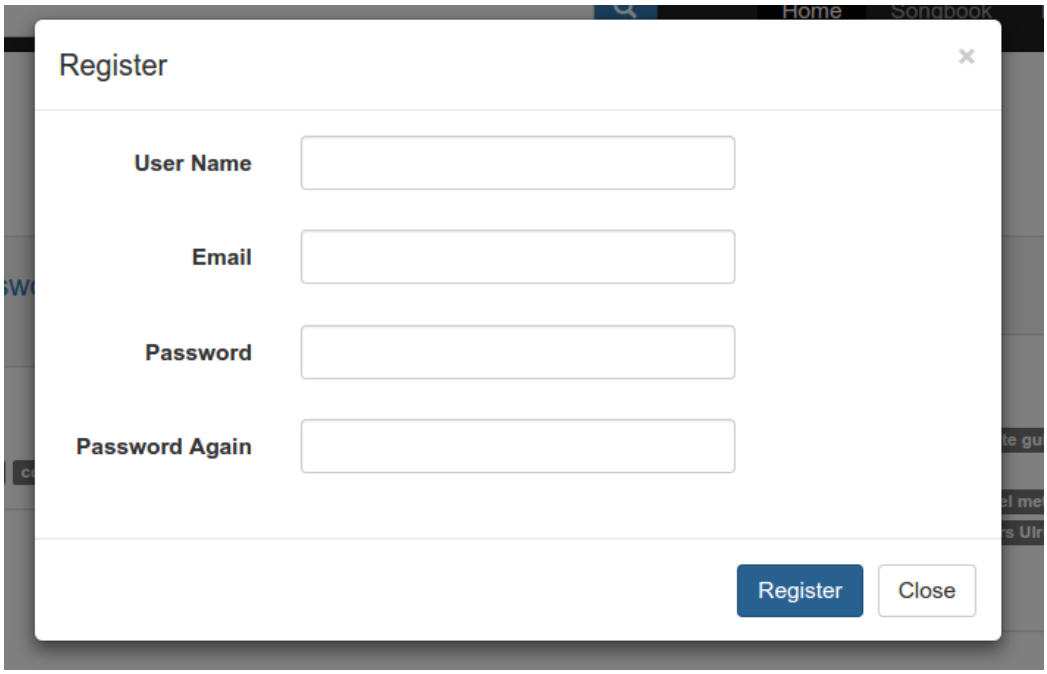
B Používateľská dokumentácia

B.1 Registrácia

1. Navigujeme sa na ktorúkoľvek stránku, napríklad http://bcmusic.yweb.sk/web/.
2. Klikneme na tlačítko Register v pravom hornom rohu11.
3. Následne sa nám zobrazí registračný formulár12
4. Vyplníme požadované údaje.
5. Zvolíme Register.
6. Ak nevypísalo žiadnu chybu, mali by ste byť zaregistrovaní a prihlásení.

B.2 Prihlásenie

1. Navigujeme sa na ktorúkoľvek podstránku aplikácie.
2. Kliknem na tlačitko Sign In vpravo hore.
3. Následne sa nám zobrazí registračný formulár 14.
4. Vyplníme svoje prihlasovacie meno a heslo.
5. Klikne na Login.



Obr. 12: *Registračný formulár*

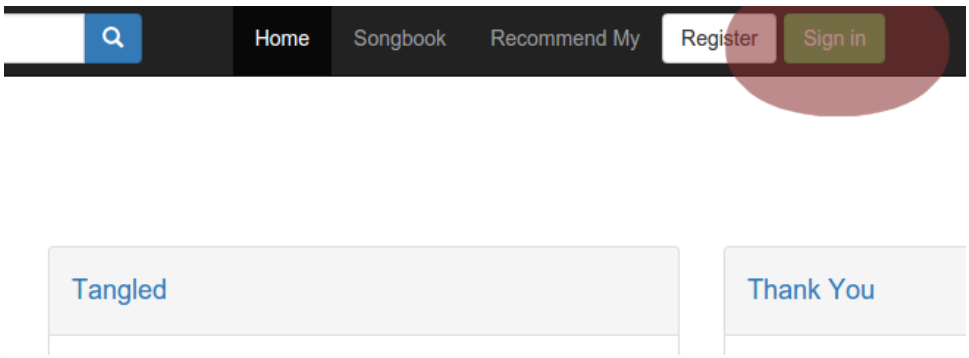
6. Ak nenastala žiadna chyba mali by sme byť prihlásení.

B.3 Vyhľadávanie

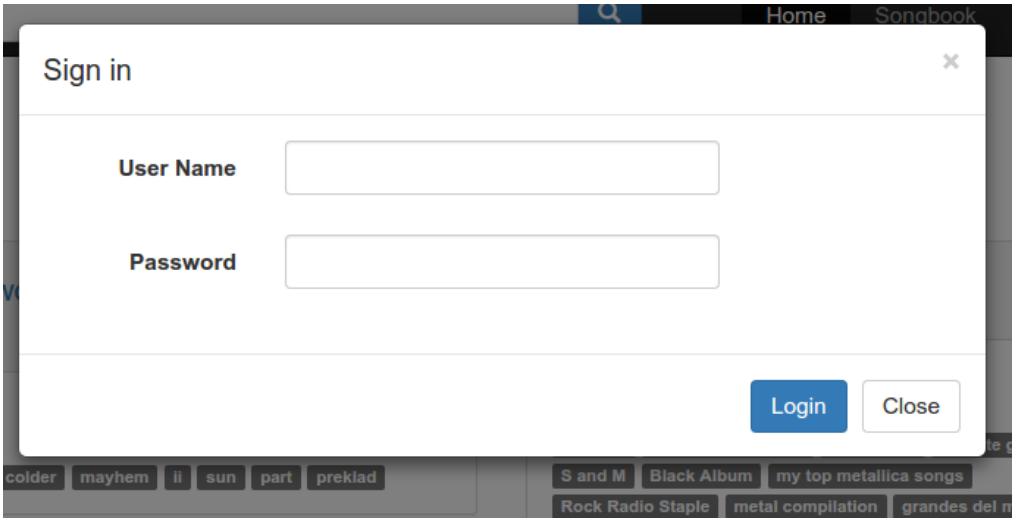
Vyhľadáva sa pomocov textového poľa v hlavičke stránky 15 (oblasť číslo 1), ktoré je dostupné na každej podstránke. V prípade, že sa do nedá zadať vyhľadávací reťazec a stlačí sa buď modrá lupa vpravo od poľa alebo enter, zobrazia sa výsledky ako na obrázku 15 (oblasť číslo 2).

B.4 Vytvorenie spevníku

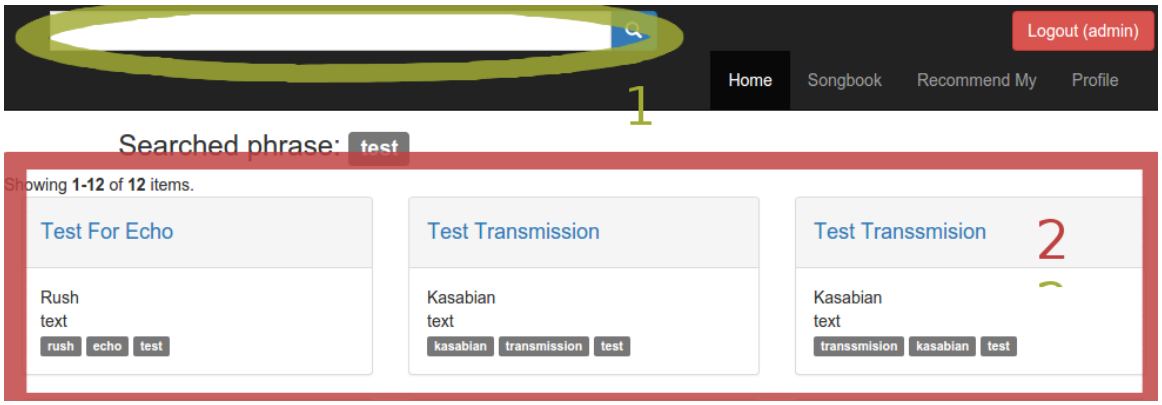
1. Klineme na tlačidlo Songbook v menu 16 (oblasť 1).
2. Zaškrtneme, pre ktorých používateľov chceme generovať spevník 16 (oblasť 2).
3. Kliknem na tlačidlo „Create“ 16 (oblasť 3).



Obr. 13: *Prihlasovacie tlačidlo*



Obr. 14: *Prihlasovací formulár*



Obr. 15: *Vyhľadávacia obrazovka*

4. Na obrázku 16 (oblasť 4) by mali byť zobrazené zoznamy odporúčaných piesni do spevníku.

B.5 Zobrazenie odporúčaní

Odporúčania sa zobrazujú na stránke v dvoch prípadoch. V prípade, že navštívite sekciu stránky „Recommend My“ alebo pri zobrazení dokumentu.

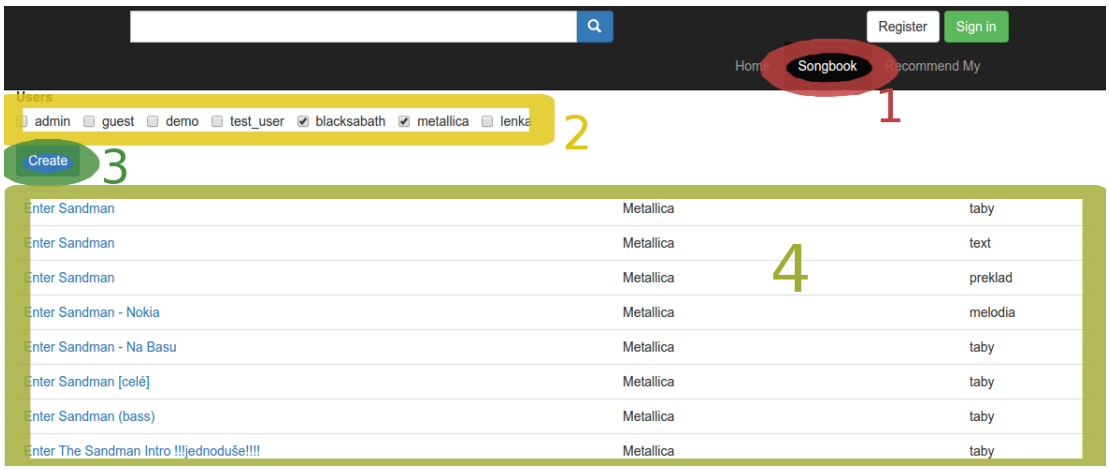
B.5.1 Zobrazenie odporúčaní „Recommend My“

Na túto obrazovku sa dostaneme po kliknutí na „Recommend My“ v menu 17 (oblasť 1). Na obrazovke sa následne nachádzajú dve tabuľky reprezentujúce dva rôzne algoritmy na odporúčanie.

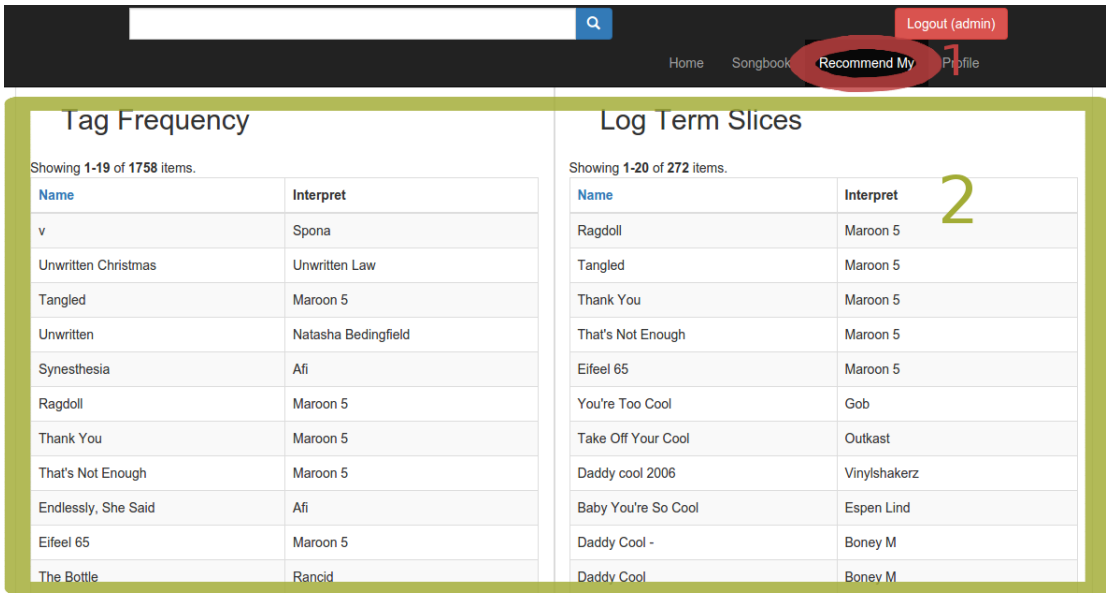
B.5.2 Zobrazenie odporúčaní pri zobrazení dokumentu

Na túto obrazovku sa môžeme dostať buď z vyhľadávania opísanom vo Vyhľadávanie B.3. Obrazovka obsahuje prvky:

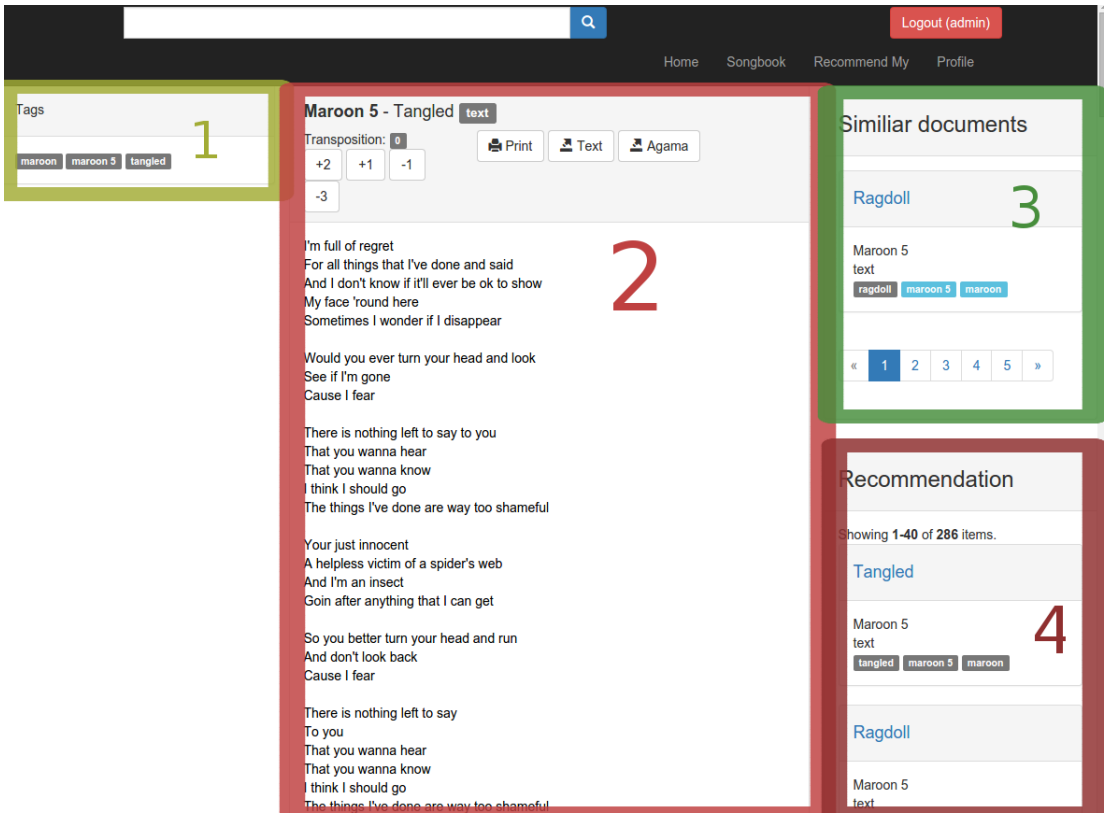
- Na obrázku 18 vľavo (oblasť 1) sa nachádza panel s tagmi dokumentu.
- V strede (oblasť 2) sa nachádza aktuálny obsah dokumentu.



Obr. 16: *Vyhľadávacia obrazovka*



Obr. 17: *Obrazovka odporúčaní*



Obr. 18: *Zobrazenie dokumentu*

- Vpravo hore (oblasť 3) je panel, ktorý zobrazuje dokumenty najviac podobné aktuálnemu dokumentu.
- Vpravo dole (oblast 4) je panel, ktorý obsahuje odporúčania pre aktuálneho používateľa.

B.6 Inštalačná príručka

Zdrojové súbory sú na digitálnom nosiči v podpriečinku src/. Aby aplikácia mohla fungovať, potrebuje z webu prístupný podpriečinok src/web a potrebuje aby PHP skripty sa mohli zapisovať do priečinka src/runtime.

Kontrola minimálnych požiadaviek

Pre presné stanovenie splnenia minimálnych požiadaviek je v priečinku so zdrojovými súbormi skript „requirements.php“. Tento skript je potrebné spustiť v konzole. Skript upozornení na prípadné možné problémy s kompatibilitou. Ak budú vypísané upozornenia znamená to, že aplikácia bude fungovať, avšak v niektorých situáciach môže nastať nedefinovaný stav.

Inštalácia balíkov Composer-a

Na nainštalovanie všetkých knižníc použitých pri vývoji a závislostiach aplikácie je potrebne z koreňového priečinka inštalácie (priečinok src/) spustiť príkaz20. V pripade že nemáte nainštalovaný Composer, môžete postupovať podľa oficiálneho návodu[23].

Ukážka 20: *Inštalácia balíkov Composer-a*

```
composer install
```

Inštalácia databázy

Prvý krok na pripojenie k databáze je nastavenie prístupových parametrov databázy. Tieto sa nastavujú v súbore „src/config/db.php“. Príklad na nastavenie MySQL databázy je v ukážke 21a príklad na nastavenie PostgreSQL je v ukážke 22.

Ukážka 21: *Príklad nastavenia databázy MySQL*

```
<?php

return [
    'class' => 'yii\db\Connection',
```

[23] https://getcomposer.org/doc/00-intro.md

```
    'dsn' => 'mysql:'.
            'host=localhost;'.
            'port=3303;'.
            'dbname=database',
    'username' => 'username',
    'password' => 'password',
    'charset' => 'utf8',
];
```

Ukážka 22: *Príklad nastavenia databázy PostgreSQL*

```
<?php

return [
    'class' => 'yii\db\Connection',
    'dsn' => 'pgsql:'.
            'host=localhost;'.
            'port=5432;'.
            'dbname=database',
    'username' => 'username',
    'password' => 'password',
    'charset' => 'utf8',
    'schemaMap' => [
        'pgsql' => [
            'class' => 'yii\db\pgsql\Schema',
            'defaultSchema' => 'public',
        ]
    ]
];
```

Po nastavení pripojenia k databáze je treba ešte vytvoriť v databáze potrebnú štruktúru tabuliek. Na spravovanie zmien v štruktúre databázy boli použité migrácie aplikačného rámca Yii2. Na vytvorenie tabuliek a cudzích kľučov stačí spustiť prikaz „./yii migrate“ v koreňovom priečinku aplikácie („src/“).

Otestovanie funkčnosti súčiastok

V prípade potreby je možné v tomto bode spustiť automatizované testovanie, čo však vyžaduje prítomnosť ďalšej databázy (testovanie prepisuje základné dáta),

prístupy k databáze pre tester sa nastavujú v súbore „src/tests/codeception/config/config.php“.

Ďalej je potrebne spustiť migrácie, tento krát ich ale púšťame z priečinka „src/tests“ a použijeme príkaz „./codeception/bin/yii migrate“. Následne môžeme spustiť testy pomocov „codecept run unit“. Aby testovanie fungovalo, vyžaduje sa aby v danom prostredí bol nainštalovaný testovací aplikačný rámec Codeception[24].

Príkazy pre manažéra úloh

Pokiaľ nechceme manuálne spúšťať indexovanie a prednačítanie externých databáz je dobré pridať tieto príkazy do manažéra úloh,

- **./yii interpret/explore** - získava názvy a identifikátory interpretov zo stránok ktoré obsahuju zoznamy interpretov,
- **./yii document/explore** - snaží sa získať identifikátory dokumentov, ich interpretov a typ,
- **./yii document/parallelnametags** - paralelne vygeneruje značky z názvu dokumentu, názvu interpreta a typu dokumentu,
- **./yii document/lunloaded** - snaží sa stiahnuť obsahy dokumentov a značky z last.fm, tento príkaz sa dá spustiť aj paralelne ako **./yii document/parallelunloaded**, tento prístup však nemusí fungovať všade a nie je spoľahlivý,
- **./yii mapdocumenttag/weight** - vypočítava užitočnosť značiek.

Aplikácia obsahuje aj ďalšie príkazy:

- **./yii document/paralleltaglfm** - paralélne sťahuje značky z last.fm,
- **./yii document/tagtype $id** - $id je parameter identifikátor dokumentu, vygeneruje značky pre dokument z jeho názvu, názvu jeho interpreta a typu dokumentu,
- **./yii document/tagtypes** - skontroluje, či všetky značky priradené dokumentom sú správneho typu, existuje aj paralelna verzia **./yii document/parallel-tagtypes**,

[24] http://codeception.com/install

Nastavenia

Aplikácia po nainštalovaní bude bežať vo vývojovom móde. Pre produkčné nastavenie je potrebné zmeniť nastavenia v „src/web/index.php“. V danom súbore je potrebné zmeniť zakomentovať riadky 4 a 5 a odkomentovať riadky 8 a 9.

Ďalej aplikácia obsahuje nastavenia v súbore „src/config/params.php“.

- **tag_appereance_weights** - váhy jednotlivých domén značiek,
- **min_tag_lenght** - minimálna dĺžka slova aby sa mohlo stať značkou,
- **time_aware_recomendation** - ak je nastavený na „true“, pri zobrazení piesne sa na paneli z odporúčaniami zobrazujú výsledky nášho algoritmu, ak je hodnota „false“, zobrazujú sa výsledky agregačného algoritmu.

C Elektronické médium

K dokumentu priložené elektronické médium má nasledovnú štruktúru:

/doc

- bakalárska práca spolu s anotáciami v slovenskom a anglickom jazyku

/src/doc

- súbor s referenciami vo formáte BibTeX a súbory dokumentácia vo formáte Latex

/src

- zdrojové kódy samotnej implementovanej aplikácie

readme.txt

- popis obsahu média v slovenskom a anglickom jazyku