



УНИВЕРЗИТЕТ У НОВОМ САДУ
ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА У
НОВОМ САДУ



Александар Бесермињи

**Реализација подсистема за претрагу и
анализу текста у персонализованом
електронском водичу кроз програм**

ДИПЛОМСКИ РАД
- Основне академске студије -

Нови Сад, 2014



УНИВЕРЗИТЕТ У НОВОМ САДУ ● ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА
21000 НОВИ САД, Трг Доситеја Обрадовића 6

КЉУЧНА ДОКУМЕНТАЦИЈСКА ИНФОРМАЦИЈА

| | | |
|---|--|------------------------|
| Редни број, РБР: | | |
| Идентификациони број, ИБР: | | |
| Тип документације, ТД: | Монографска документација | |
| Тип записа, ТЗ: | Текстуални штампани материјал | |
| Врста рада, ВР: | Завршни (Bachelor) рад | |
| Аутор, АУ: | Александар Бесермињи | |
| Ментор, МН: | проф. Др. Иштван Пап | |
| Наслов рада, НР: | Реализација подсистема за претрагу и анализу текста у персонализованом електронском водичу кроз програм | |
| Језик публикације, ЈП: | Српски / латиница | |
| Језик извода, ЈИ: | Српски | |
| Земља публикавања, ЗП: | Република Србија | |
| Уже географско подручје, УГП: | Војводина | |
| Година, ГО: | 2014 | |
| Издавач, ИЗ: | Ауторски репринт | |
| Место и адреса, МА: | Нови Сад; трг Доситеја Обрадовића 6 | |
| Физички опис рада, ФО: (поглавља/страна/ цитата/табела/слика/графика/прилога) | | |
| Научна област, НО: | Електротехника и рачунарство | |
| Научна дисциплина, НД: | Рачунарска техника | |
| Предметна одредница/Кључне речи, ПО: | Претрага, емисија, анализа | |
| УДК | | |
| Чува се, ЧУ: | У библиотеци Факултета техничких наука, Нови Сад | |
| Важна напомена, ВН: | | |
| Извод, ИЗ: | Анализа и претрага текста у персонализованом електронском водичу се заснива на коришћењу Lucene библиотеке. Програм који је наведен у Електронском водичу садржи информације о томе када програм почиње, колико траје, којем жанру припада, али и текстуални опис програма. Анализом описа, прави се индекс речи које се везују са програмима који се емитују. Такав индекс се касније претражује, а као резултат се враћају програми који су у вези са пронађеним речима. Реализација је урађена у оквиру Hibernate окружења у Java програмском језику. | |
| Датум прихватања теме, ДП: | | |
| Датум одбране, ДО: | | |
| Чланови комисије, КО: | Председник: | |
| | Члан: | Потпис ментора |
| | Члан, ментор: | Др. Иштван Пап, доцент |



UNIVERSITY OF NOVI SAD • FACULTY OF TECHNICAL SCIENCES
21000 NOVI SAD, Trg Dositeja Obradovića 6

KEY WORDS DOCUMENTATION

| | | |
|--|--|--------------------|
| Accession number, ANO : | | |
| Identification number, INO : | | |
| Document type, DT : | Monographic publication | |
| Type of record, TR : | Textual printed material | |
| Contents code, CC : | Bachelor Thesis | |
| Author, AU : | Aleksandar Beserminji | |
| Mentor, MN : | Ištvan Pap Ph. D. | |
| Title, TI : | One solution of subsystem for search and text analysis in personalized electronic program guide | |
| Language of text, LT : | Serbian | |
| Language of abstract, LA : | Serbian | |
| Country of publication, CP : | Republic of Serbia | |
| Locality of publication, LP : | Vojvodina | |
| Publication year, PY : | 2014 | |
| Publisher, PB : | Author's reprint | |
| Publication place, PP : | Novi Sad, Dositeja Obradovica sq. 6 | |
| Physical description, PD : (chapters/pages/ref./tables/pictures/graphs/appendixes) | | |
| Scientific field, SF : | Electrical Engineering | |
| Scientific discipline, SD : | Computer Engineering, Engineering of Computer Based Systems | |
| Subject/Key words, S/KW : | search, EPG, analysis | |
| UC | | |
| Holding data, HD : | The Library of Faculty of Technical Sciences, Novi Sad, Serbia | |
| Note, N : | | |
| Abstract, AB : | The text search and analysis in personalized electronic program guide is based on Lucene library. Event in electronic program guide (EPG) contain information about it's start time, duration, genre, but it also contain textual description. With analysis of that description, keywords index is made, where those keywords are in relation to the events from which they are extracted. When such index is searched, returned results are events which are connected to the found keywords. Realisation is done within Hibernate framework in Java programming language. | |
| Accepted by the Scientific Board on, ASB : | | |
| Defended on, DE : | | |
| Defended Board, DB : | President: | |
| | Member: | Menthor's sign |
| | Member, Mentor: | Ištvan Papp Ph. D. |

SADRŽAJ

| | |
|---|----|
| 1. Uvod..... | 1 |
| 2. Teorijske osnove | 3 |
| 2.1 Opis okruženja..... | 3 |
| 2.2 Pretraga teksta | 3 |
| 2.3 Važne osobine | 4 |
| 2.3.1 Relevantnost rezultata..... | 4 |
| 2.3.2 Ocena rezultata | 5 |
| 2.3.3 Brzina pretrage..... | 5 |
| 2.3.4 Horizontalna skalabilnost | 5 |
| 2.4 Pregled alata za pretragu teksta..... | 5 |
| 2.4.1 Pretraga pomoću SQL upita..... | 5 |
| 2.4.2 Sphinx | 6 |
| 2.4.3 Lucene, Solr, Hibernate Search | 6 |
| 2.4.4 Upporedni prikaz osobina alata za analizu i pretragu teksta | 7 |
| 3. Koncept rešenja..... | 8 |
| 3.1 Pravila za generisanje termina..... | 10 |
| 3.1.1 Razdvajanje reči..... | 10 |
| 3.1.2 Uklanjanje velikih slova | 10 |
| 3.1.3 Nepotrebne reči..... | 10 |
| 3.1.4 Stemovanje | 11 |
| 3.1.5 Primer primene opisanih pravila..... | 11 |
| 3.2 Pretraga..... | 12 |
| 3.2.1 Definisavanje HS-upita | 13 |

| | | |
|---------|--|----|
| 3.2.1.1 | TermQuery | 13 |
| 3.2.1.2 | WildcardQuery | 13 |
| 3.2.1.3 | BooleanQuery | 13 |
| 3.2.2 | Ocenjivanje rezultata | 13 |
| 3.3 | Tehnika bržeg pristupa rezultatima | 14 |
| 4. | Programsko rešenje | 15 |
| 4.1 | Indeksiranje informacija o emisijama | 16 |
| 4.2 | Pretraga indeksa | 16 |
| 4.3 | Opis programskih modula | 17 |
| 4.3.1 | DirectoryProvider | 17 |
| 4.3.2 | ShardingStrategy | 18 |
| 4.3.3 | LanguageDiscriminator | 19 |
| 4.3.4 | Pravila indeksiranja | 19 |
| 4.3.5 | Indeksiranje | 20 |
| 4.3.6 | Pretraga | 20 |
| 5. | Rezultati | 22 |
| 5.1 | Poređenje vremena izvršavanja | 22 |
| 5.2 | Poređenje kvaliteta rezultata | 23 |
| 6. | Zaključak | 24 |
| 6.1 | Dodatna proširenja | 25 |
| 7. | Literatura | 26 |

SPISAK SLIKA

| | |
|---|----|
| Slika 2.1 - Pretraga dokumenata korišćenjem indeksa termina | 4 |
| Slika 3.1 - Arhitektura sistema | 8 |
| Slika 3.2 - Primer čuvanja višejezičkih termina u istom indeksu | 9 |
| Slika 3.3 - Primer čuvanja višejezičkih termina u različitim indeksima..... | 10 |
| Slika 4.1 - Slučaj korišćenja Hibernate Search okruženja | 15 |

SPISAK TABELA

| | |
|--|----|
| Tabela 2.1 - Usporedni prikaz osobina alata za analizu i pretragu teksta..... | 7 |
| Tabela 3.1 - Primer primene opisanih pravila..... | 12 |
| Tabela 3.2 - Primer ocenjenih emisija..... | 14 |
| Tabela 4.1 - Primer rezultata za delimično poklapanje poslednjeg termina | 17 |
| Tabela 4.2 - Generisanje HS-upita od postojećeg K-Upita..... | 17 |
| Tabela 5.1 - Poređenje vremena izvršavanja za različite implementacije | 22 |

SKRAĆENICE

| | |
|------------|---|
| DTV | - <i>Digital television</i> , Digitalna Televizija |
| EPG | - <i>Electronic Programm Guide</i> , Programski vodič |
| SQL | - <i>Structured Query Language</i> |

1. Uvod

U digitalnoj televiziji (eng. Digital television - DTV) digitalni podaci (npr. audio i video) se modulišu u analogni signal i tako se prenose do krajnjeg korisnika. Na taj način programi su boljeg kvaliteta, moguće je prenositi programe visoke rezolucije, korisnik ima mogućnost odabira prevoda na željenom jeziku, interaktivnu televiziju i još mnogo toga. Takođe digitalni podaci se mogu komprimovati, a uticaj interferencije u maloj meri sa ostalim signalima u etru ne utiče na kvalitet podataka. Signali se multipleksiraju, frekvencijski opseg je bolje iskorišćen pa je moguće prenositi više programa na jednoj frekvenciji.

To dovodi do toga da korisnici imaju izbor od par stotina (nekada i preko hiljadu) programa na raspolaganju. Pregledanje svih programa u svrhu pronalaženja interesantnog sadržaja postaje zamorna procedura, a kao mogućnost koja može da olakša tu proceduru predstavlja Elektronski programski vodič (eng. Electronic program guide – EPG)[1]. On pruža informacije o TV emisijama, kao što su ime, opis, produžen opis, žanr, jezik, vreme početka, trajanje itd.

Personalizovani elektronski programski vodič je unapređena verzija standardnog elektronskog programskog vodiča koja na osnovu korisničkog profila izdvaja relevantne TV emisije u prvi plan. Korisnički profil se generiše na osnovu direktnog odabira korisnika šta mu se dopada ili posmatranjem korisnikovih akcija i navika.

U ovom radu će biti opisano jedno rešenje analize i pretrage teksta u personalizovanom elektronskom programskom vodiču, koji je deo sistema za preporuku sadržaja[2]. Pretraga je potrebna kada korisnik traži specifičnu TV emisiju ili onda kada sistem za preporuku ne preporučuje ono što korisnik očekuje. Od korisnika se očekuje da unese tekst koji na neki način opisuje traženu TV emisiju - program. To može da bude deo naslova ili opis TV emisije koju korisnik traži.

Podaci iz personalizovanog elektronskog programskog vodiča se analiziraju i generiše se indeks (eng. Index) reči, koji se kasnije pretražuje. Kada se značenja unetog opisa koji korisnik traži i opisa TV emisije poklapaju, te emisije se korisniku prikazuju kao rezultati pretrage.

Usluge pretrage su predviđene za TV i mobilne uređaje, gde je u nekim slučajevima unos teksta otežan (na primer kod daljinskog upravljača). Kako bi korisnicima sa takvim uređajima olakšali korišćenje pretrage, posebna pažnja je posvećena tome kako je pretraga implementirana.

Rešenje je bazirano na Java Enterprise Edition (JEE) [3]. Za potrebe objektno/relacionog mapiranja entiteta (eng. Object/Relational Mapping - ORM) korišćen je Hibernate [4]. To je okruženje za Java programski jezik, koja pruža mogućnosti mapiranja objektno-orijentisanog domenskog modela na tradicionalne relacije baze podataka.

Rad se sastoji od sedam poglavlja:

- U prvom poglavlju je ukratko opisan sadržaj samog rada.
- Drugo poglavlje opisuje teorijske osnove pretrage teksta, daje spisak osobina koje opisani sistem za pretragu treba da ima i na kraju izlaže neka od postojećih rešenja za pretragu teksta.
- Treće poglavlje je orijentisano na koncept rešenja, opis pojedinačnih delova analize teksta, generisanje indeksa i kasnije tehnike pretrage i dobavljanja rezultata.
- Četvrto poglavlje opisuje implementaciju programskog rešenja za pretragu teksta.
- U petom poglavlju su dati rezultati provere validnosti realizovanog rešenja kao i poređenje performansi sa još dva moguća rešenja.
- Šesto poglavlje predstavlja kratak osvrt na ceo rad, a dati su i dalji mogući pravci razvoja i mogućnosti unapređenja izloženog programskog rešenja.
- Na kraju, sedmo poglavlje pruža uvid u literaturu koja je korišćena tokom izrade ovog rada.

2. Teorijske osnove

2.1 Opis okruženja

Podsistem za pretragu teksta pripada sistemu za preporuku sadržaja. Sadržaj predstavljaju TV emisije, ali i sadržaj sa interneta koji može biti različitih tipova: vesti, biografije, najave, vremenska prognoza itd.

U trenutnoj realizaciji, pretraga je ograničena samo na pretraživanje TV emisija koje se na programu prikazuju u narednih sedam dana. Podaci o takvim TV emisijama se prikupljaju iz EPG podataka za svaki program – iz etra ili preko interneta za one programe koji ne emituju EPG podatke. Akvizicioni uređaji su set-top-box-ovi koji pretraživanjem određenih frekvencija prikupljaju EPG podatke. Ukoliko podaci nisu nađeni, pretraga se vrši na internetu na unapred definisanom portalu za određeni kanal. Prikupljeni podaci o TV emisijama se šalju poslužiocu koji ih skladišti u bazu podataka. Prilikom skladištenja, podaci se obrađuju radi kasnije lakše pretrage.

2.2 Pretraga teksta

Najosnovniji vid pretrage teksta je postupak pronalaženja određenog niza karaktera- upit (eng. Query) - unutar dokumenta. Dokument može sadržati samo tekst ili tekst podeljen u strukture kao što su naslov, autori, opis i slično. Kada se pretražuju dokumenti, traže se oni koji sadrže dati upit unutar svojih struktura.

Kada se pretražuje mali broj dokumenata, moguće je pretraživati dokumente direktno, tražeći upit u svakom dokumentu ponaosob (serijska pretraga). Ukoliko je broj dokumenata i/ili upita velik, takav pristup nije prihvatljiv zbog loših performansi.

Umesto toga, koristi se tehnika indeksiranja dokumenata, gde se tekst svih dokumenata analizira i pravi se indeks svih reči – termina. Termini su reči obrađene nekim skupom pravila, koji reči svodi na pogodniji oblik – u ovom slučaju za pretragu. Svaki termin je povezan sa dokumentom koji sadrži dati termin. Prilikom pretrage, pretražuje se indeks termina umesto teksta dokumenata, što ubrzava pretragu. Kao rezultati se vraćaju dokumenti sa kojima su pronađeni termini povezani – Slika 2.1.



Slika 2.1 - Pretraga dokumenata korišćenjem indeksa termina

2.3 Važne osobine

Za realizaciju podsistema pretrage u sistemu za preporuku, nije dovoljno vršiti samo osnovno poređenje upita sa opisom TV emisija. Potrebno je praviti naprednije poređenje – poređenje po značenju.

Pored toga, važno je u obzir uzeti i nekoliko stvari:

- Rezultati moraju biti relevantni
- Rezultati moraju biti ocenjeni
- Brzina pretrage
- Horizontalna skalabilnost

2.3.1 Relevantnost rezultata

Kada se vrši pretraga dokumenata, relevantni rezultati su oni dokumenti koji su slični po značenju, a ne oni dokumenti koji direktno sadrže tekst kojim je definisan upit. Primer nerelevantnog je dokument koji sadrži tekst „...flaširana voda...“, a koji je kao rezultat vraćen nakon pretrage definisane upitom „rana“. Relevantan rezultat bi bio dokument sa tekстом „...površinska rana...“, „...ispiranje rane...“ i sl.

2.3.2 Ocena rezultata

Dokumenti pronađeni prilikom pretrage mogu u različitim merama da se poklapaju sa upitom koji unosi korisnik. Meru koliko neki dokument odgovara upitu određuje algoritam koji definiše sistem za pretragu. Drugim rečima, svaki dokument je ocenjen nekom brojčanom vrednošću, a što je taj broj veći, prema definisanom algoritmu za ocenjivanje, taj dokument se više poklapa sa upitom.

2.3.3 Brzina pretrage

Brzina se definiše kao broj izvršenih zadataka u jedinici vremena. Pošto je pretraga deo sistema za preporuku koji je namenjen velikom broju korisnika, brzina je od presudnog značaja.

2.3.4 Horizontalna skalabilnost

Horizontalna skalabilnost označava postupak dodavanja resursa (poslužilaca) na kojima se izvršava data distribuirana aplikacija. Povećanjem broja poslužilaca na kojima se izvršava aplikacija, doprinosi propusnoj moći sistema odnosno brzini izvršavanja i omogućuje obradu većeg broja upita u jedinici vremena.

2.4 Pregled alata za pretragu teksta

U ovom delu su predstavljene tehnike i biblioteke koje se koriste za analizu i pretragu teksta. Opisana je pretraga pomoću SQL upita, Sphinx aplikacija za pretragu teksta i na kraju okruženje Hibernate Search, koje implementira Lucene biblioteku za indeksiranje i pretragu teksta.

2.4.1 Pretraga pomoću SQL upita

SQL (Structured Query Language) je jezik namenjen upravljanju podacima u relacionim bazama podataka. Podržava razne operacije, a kada je potrebno bliže odrediti podatke nad kojima operacija treba da se izvrši – recimo prilikom operacija brisanja podataka – koristi se klauzula WHERE.

Da bi se prilikom čitanja podataka iz baze dobili samo redovi koji sadrže određen tekst potrebno je napisati sledeći upit

```
SELECT * FROM <ime_tabele> WHERE <polje> = "<tekst>";
```

Izvršavanje ove komande će kao rezultat vratiti sve redove iz zadate tabele, čiji je sadržaj polja jednak zadatom tekstu. Kako bi u rezultatima bili i oni redovi čija polja sadrže zadati tekst, upit možemo prepraviti na sledeći način

```
SELECT * FROM <ime_tabele> WHERE <polje> LIKE "%<tekst>%";
```

Karakteristi ‘%’ označavaju zamenu za nula ili više karaktera, tj. ako je sadržaj polja “Minioni vole banane”, a traženi tekst “ana”, ovaj red će biti vraćen kao rezultat, dok u prethodnoj realizaciji upita to ne bi bio slučaj.

Postoje razne implementacije menadžera baza podataka koja podržavaju SQL kao što su MySQL, PostgreSQL, Oracle database manager i sl. U zavisnosti od implementacije, podržani su indeksiranje teksta i/ili ocenjivanje rezultata nije.

2.4.2 Sphinx

Sphinx je namenska aplikacija koja pruža usluge pretrage teksta. Može da komunicira sa drugim bazama podataka kao što su MySQL, MariaDB, PostgreSQL i sl. Takođe može se koristiti i kao samostalni poslužilac za skladištenje podataka.

Podržava grupno indeksiranje ili prilikom upisa, rad sa Non-SQL bazama podataka, spregu prema različitim programskim jezicima kao što su Java, PHP, Python, C itd. ocenjivanje rezultata i još mnoge druge mogućnosti koje su relevantne za pretragu teksta.

2.4.3 Lucene, Solr, Hibernate Search

Lucene je u osnovi biblioteka koja služi za indeksiranje i pretragu teksta[5]. Podržava razne upite kao što su džoker upiti, fuzzy upiti, kombinacija upita itd.

- Džoker upiti – korišćenje specijalnih „džoker“ karaktera kao što su „*“ ili „?“ radi zamene određenog broja karaktera. Npr. upit „lop*“ pronalazi termine „lopta“, „lopata“, „lopov“ i sl.
- Fuzzy upiti – omogućavaju definisanje koeficijenta sličnosti dva termina na osnovu dodavanja, uklanjanja ili zamene nekog karaktera. Npr. upit „mašina“ sa nekom vrednošću koeficijenta pronalazi sledeće termine: „mašine“, „mašna“, „masna“, „šina“ itd.
- Upit sa izrazom – pronalazi određeni izraz – niz reči unutar dokumenata. Kada su reči jedna pored druge, ocena takvog rezultata je veća u odnosu na one rezultate gde su reči na različitim mestima. Npr. upit „Ivan pije vodu“ vraća dokument „Naš pas Ivan pije vodu nakon šetnje.“ kao rezultat sa najvišom ocenom, dok dokument „Kupi mi vodu, baš mi se pije. A Ivan će doći kasnije.“ ima znatno manju ocenu.
- Kombinacija upita – omogućava kombinaciju više upita radi formiranja konačnog upita za pretragu.

Ocenjivanje rezultata se vrši prema količini preklapanja upita i rezultata. Kod biblioteke je objavljen pod Apache 2.0 licencom.

Solr je enterprajz (eng. Enterprise) platforma otvorenog koda za pretragu teksta, zasnovana na Lucene biblioteci[6]. Omogućava distribuirano indeksiranje, dinamičku klasterizaciju,

replikacije i još mnogo toga. Razvoj ove platforme je započet još 2004. godine, a od 2010. godine projekat Lucene i Solr su spojeni u jedan. Mnogi poznati servisi kao što su Instagram, NFL Sports, Netflix, eBay, NASA PDS i mnogi drugi, koriste Solr za realizaciju neke pretrage, što govori o ozbiljnosti ovog projekta.

Hibernate Search, kao poslednji u lancu, integriše Solr[7]. Podržava sve mogućnosti koje realizuju Solr i Lucene, a dodatno realizuje podršku za rad u Hibernate okruženju. Npr. prilikom rukovanja sa bazom podataka (operacije kao što su upis, izmene, brisanje i sl.), indeks se može automatski ažurirati da prati nastale izmene.

Prilikom indeksiranja, određivanje skupa pravila za obradu i određivanje lokacije indeksa se može vršiti dinamički na osnovu nekog parametra. Isto tako prilikom pretrage, Hibernate Search automatski određuje indeks koji se pretražuje, te nije potrebno eksplicitno zadati putanju do indeksa. Nakon pretrage indeksa Hibernate Search omogućava automatsko dobavljanje entiteta iz baze podataka koji su vezani za pronađene dokumente.

Opisanim mogućnostima, postignuto je da implementacija Solr odnosno Lucene biblioteka postanu transparentne.

Hibernate Search se razvija pod LGPL licencom što ga, uzimajući u obzir i sve ostale mogućnosti, u ovom slučaju, čini najboljim izborom za realizaciju ovog projekta.

2.4.4 Uporedni prikaz osobina alata za analizu i pretragu teksta

| | SQL | Sphinx | Hibernate Search |
|-------------------------------------|----------------------------------|----------------------|------------------|
| Podržava indeksiranje | ✓ | ✓ | ✓ |
| Uklanjanje nepotrebnih reči (3.1.3) | ✗ | ✓ | ✓ |
| Stemovanje (3.1.4) | ✗ | ✓ | ✓ |
| Džoker upiti | ✓ | ✓ | ✓ |
| Fuzzy upiti | ✗ | ✓ | ✓ |
| Kombinacija upita | ✓ | ✓ | ✓ |
| Ocenjivanje rezultata | ✗ | ✓ | ✓ |
| Licenca | Postoje različite implementacije | GPLv2 i komercijalna | LGPL |

Tabela 2.1 - Uporedni prikaz osobina alata za analizu i pretragu teksta

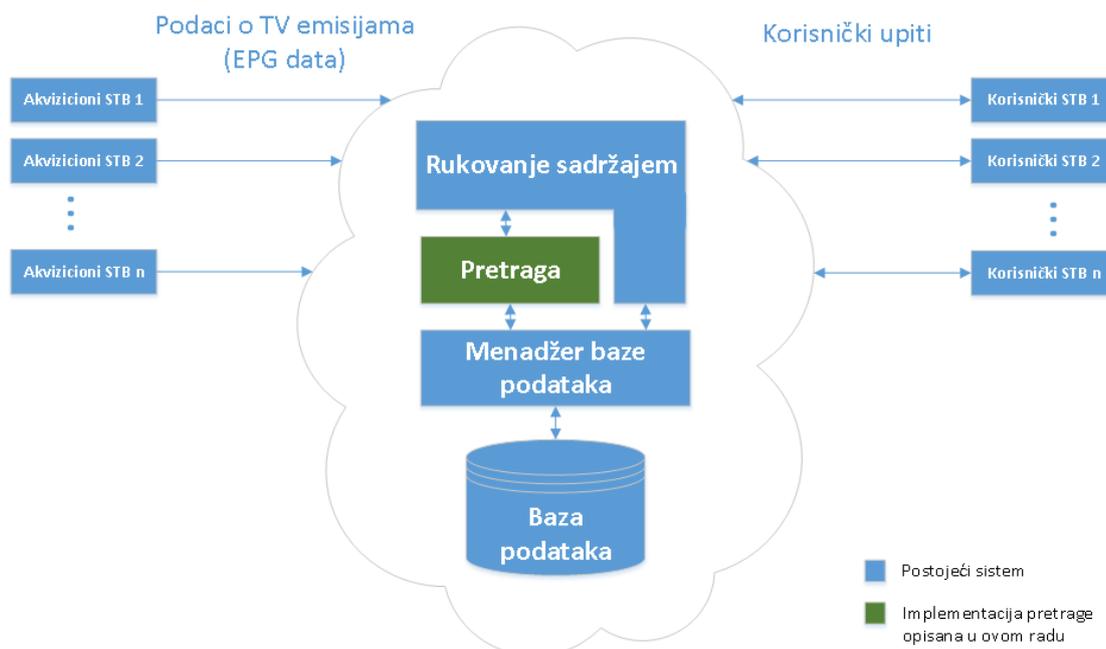
U *Tabela 2.1* - Uporedni prikaz osobina alata za analizu i pretragu teksta je dat uporedni prikaz osobina alata za analizu i pretragu teksta.

3. Koncept rešenja

U prethodnom poglavlju su predstavljene biblioteke koje pružaju mogućnost pretrage teksta, kao i osobine koje odabrana biblioteka mora da zadovoljava. Zbog toga, za potrebe realizacije podsistema pretrage, odabran je Hibernate Search i to iz dva razloga:

- Zadovoljava sve nametnute uslove i
- Oslanja se na već postojeću arhitekturu, tj. na Hibernate okruženje koje se koristi za realizaciju sistema komunikacije sa bazom podataka u sistemu za preporuku sadržaja gde je realizovan podsistem pretrage

U ovom poglavlju je dat pogled na arhitekturu sistema u kojem je realizovan podsistem za pretragu, kao i koncept rešenja za realizaciju pretrage, korišćenjem Hibernate Search okruženja.



Slika 3.1 - Arhitektura sistema

Na *Slika 3.1* je predstavljena arhitektura sistema za preporuku sadržaja (blokovi obojeni plavom bojom) u okviru koje je realizovan podsistem za pretragu i analizu elektronskog programskog vodiča (blok obojen zelenom bojom). Akvizicioni set-top-box-ovi prikupljaju informacije o programima i šalju ih poslužiocu u oblaku na skladištenje i dalju obradu. Korisnički set-top-box-ovi poslužiocu šalju podatke o korisnikovim akcijama, tačnije akcije promene kanala. Na osnovu toga poslužilac generiše korisnički profil. Kada korisnik zatraži od poslužioca neki sadržaj, taj sadržaj se odabira na osnovu generisanog korisničkog profila. Takav sadržaj se zove personalizovani sadržaj.

Prilikom pretrage, korisnik šalje izvorni upit poslužiocu, a on se u sistemu prosleđuje do modula „Pretraga“ koji vrši pretragu i vraća rezultate. Radi realizacije brze pretrage, ključna stvar je generisanje indeksa termina koje sadrže opisi TV emisija u elektronskom programskom vodiču (u daljem tekstu samo emisije). Indeksirani entitet (u ovom slučaju emisija) je u indeksu predstavljen dokumentom koji oslikava taj entitet. Tako dokumenti u indeksu imaju polja ime, opis i produženi opis, kao što to imaju i emisije.

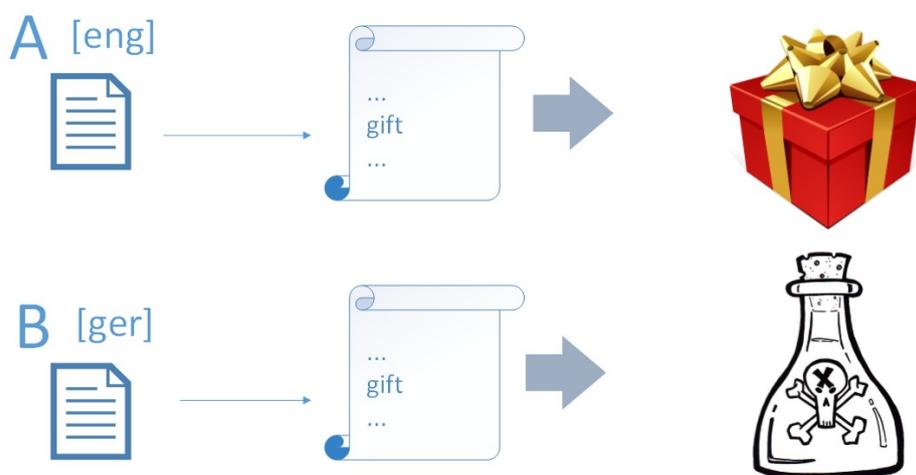
Ova strategija zahteva dodatni prostor u RAM memoriji ili na hard disku, za smeštanje indeksa termina. Indeks se može generisati od jednom u celosti od svih emisija trenutno prisutnih u bazi podataka ili dinamički dopunjavati svaki put kada se u bazu upiše nova emisija. Kada se indeks dopunjava dinamički, vreme upisa emisije u bazu je produženo vremenom potrebnim za indeksiranje informacija o toj emisiji.

Važno je napomenuti da se indeks generiše za svaki jezik posebno, kako bi se izbeglo preklapanje određenih termina koji se pišu isto, ali na različitim jezicima imaju različito značenje[8]. Tako se izbegava i problem pogrešnih relacija između emisija koje sadrže takve reči. Na primer, Engleska reč „gift“ (poklon) i Nemačka reč “gift” (otrov) su različitog značenja. U slučaju kada postoji jedan indeks za sve reči, emisije na Nemačkom jeziku bi bile povezane sa terminom “gift” iako nemaju veze sa značenjem te reči na engleskom jeziku - *Slika 3.2*.



Slika 3.2 - Primer čuvanja višejezičkih termina u istom indeksu

Hibernate Search podržava mehanizam koji omogućava podelu indeksa prema nekom kriterijumu, konkretno u ovom slučaju prema jeziku na kom su napisane informacije o emisijama - *Slika 3.3*.



Slika 3.3 - Primer čuvanja višejezičkih termina u različitim indeksima

3.1 Pravila za generisanje termina

3.1.1 Razdvajanje reči

Radi lakše obrade teksta i generisanja indeksa, izvorni tekst je potrebno razdvojiti na niz, gde je svaki element niza jedna reč tog teksta (eng. tokenization). Pri tom se izbacuju i znakovi interpunkcije.

3.1.2 Uklanjanje velikih slova

Niz reči dobijen u prethodnom postupku se zatim transformiše u niz reči čija su sva slova mala (eng. decapitalization). Ovo kasnije olakšava poređenje prilikom pretrage.

3.1.3 Nepotrebne reči

Prilikom pretrage, obraća se pažnja na imenice, prideve, glagole i sl. dok se zamenice predlozi, veznici, rečice i sl. zanemaruju (eng. common words discarding). Npr. u pokušaju da se pronađe serija „Two and a half men”, kao ključne reči za pretragu se smatraju reči „two”, „half“ i „men”, dok se reči „and” i „a” smatraju nevažnim. Uzimajući ovo kao pravilo, definisani su skupovi reči koje nisu od važnosti za pretragu na određenom jeziku tzv. liste nepotrebni reči (eng. stopwords lists).

Skupovi ovakvih reči su specifični za svaki jezik, te je potrebno definisati po jedan skup za svaki jezik koji se obrađuje. Na primer glagol “die” na Engleskom jeziku se smatra potrebnim dok se određeni član za ženski rod “die” na nemačkom smatra kao nepotrebna reč. Reči iz

ovakvih skupova ne ulaze u indeks, jer ne igraju nikakvu ulogu prilikom pretrage, a pri tom se smanjuje i veličina indeksa i potrebno je manje prostora za njegovo smeštanje u memoriju.

3.1.4 Stemovanje

Stemovanje (eng. stemming) je proces izdvajanja korena reči. Tako se od reči “run”, “running” ili “runs” uvek dobije koren reči “run”. Ovim se postiže fleksibilnost prilikom pretrage, jer korisnik prilikom pretrage ne mora da unosi upit tačno onako kako je napisano u informacijama o emisiji, već je važno da pogodi značenje onoga što traži.

3.1.5 Primer primene opisanih pravila

U *Tabela 3.1* - Primer primene opisanih pravila je grafički predstavljena lančana primena opisanih pravila na date informacije iz emisija i izgled indeksa nakon obrade tih informacija. Masna slova u primerima označavaju mesta gde nastaju izmene primenom datog pravila u levoj koloni. U poslednjem redu je dat spisak termina u indeksu gde se termin nalazi između velikih zagrada [i], a u vitičastim zagradama { i } su dati brojevi emisija odakle su ti termini dobijeni.

| Pravilo | # | Tekst | | | |
|---------------------------|---|---|--------------|-----------|--------------|
| Razdvajanje reči | 1 | The quick brown fox, JumPs over THE lazy Dog. | | | |
| | 2 | Can you jump over the brown rope Jon? | | | |
| | 3 | I am so lazy these days. | | | |
| | 4 | He really loves Ben's dogs. | | | |
| Uklanjanje kapitalizacije | 1 | [The] [quick] [brown] [fox] [JumPs] [over] [THE] [lazy] [Dog] | | | |
| | 2 | [Can] [you] [jump] [over] [the] [brown] [rope] [Jon] | | | |
| | 3 | [I] [am] [so] [lazy] [these] [days] | | | |
| | 4 | [He] [really] [loves] [Ben] [dogs] | | | |
| Nepotrebne reči | 1 | [the] [quick] [brown] [fox] [jumps] [over] [the] [lazy] [dog] | | | |
| | 2 | [can] [you] [jump] [over] [the] [brown] [rope] [jon] | | | |
| | 3 | [i] [am] [so] [lazy] [these] [days] | | | |
| | 4 | [he] [really] [loves] [ben] [dogs] | | | |
| Stemovanje | 1 | [quick] [brown] [fox] [jumps] [lazy] [dog] | | | |
| | 2 | [you] [jump] [brown] [rope] [jon] | | | |
| | 3 | [i] [lazy] [days] | | | |
| | 4 | [he] [really] [loves] [ben] [dogs] | | | |
| Sadržaj indeksa | | [quick] {1,2} | [lazy] {1,3} | [jon] {2} | [really] {3} |
| | | [brown] {1,2} | [dog] {1,4} | [i] {3} | [love] {3} |
| | | [fox] {1} | [you] {2} | [day] {3} | [ben] {3} |
| | | [jump] {1} | [rope] {2} | [he] {4} | |

Tabela 3.1 - Primer primene opisanih pravila

3.2 Pretraga

Pretraga se obavlja pronalaženjem poklapanja između korisnikovog upita (u daljem tekstu – K-upit) i termina sačuvanih u indeksu. K-upit je niz izvornih reči, dok se u indeksu nalaze obrađeni termini. U slučaju poređenja K-upita sa terminima u indeksu, rezultati ne bi bili odgovarajući, odnosno nije moguće njihovo direktno poređenje. Zbog toga se K-upit takođe obrađuje opisanim skupom pravila, a zatim se od dobijenih termina generiše Hibernate Search upit (u daljem tekstu – HS-upit). Sa takvim upitom, pretražuje se indeks termina. Pronađeni termini u indeksu su povezani sa dokumentima koji sadrže te termine, pa se onda ti dokumenti vraćaju kao rezultati.

3.2.1 Definisane HS-upita

HS-upit opisuje šta se traži u indeksu i kako se indeks pretražuje. Za opisivanje načina pretrage koriste se Query klase definisane u Lucene biblioteci. U podsistemu za pretragu korišćene su sledeće klase: TermQuery, WildcardQuery i BooleanQuery.

3.2.1.1 TermQuery

TermQuery obavlja proveru da li definisani termin postoji u indeksu. Ako postoji, dokumenti koji sadrže taj termin se smatraju rezultatima.

3.2.1.2 WildcardQuery

WildcardQuery klasa je već opisana u 2.4.3 - *Lucene, Solr, Hibernate Search* pod stavkom „Džoker upiti“.

3.2.1.3 BooleanQuery

BooleanQuery je pomenut u poglavlju 2.4.3 - *Lucene, Solr, Hibernate Search* pod stavkom „Kombinacija upita“. Koristi se za kombinaciju više upita i može biti definisan kao: MUST, MUST NOT i SHOULD. Kada je postavljen na MUST, onda dokumenti (rezultati) moraju da sadrže taj deo upita, dok za MUST NOT dokumenti ne smeju sadržati taj deo upita. BooleanQuery definisan kao SHOULD u rezultate ubraja i dokumente koji ne sadrže dati termin, ali su takvi rezultati rangirani sa slabijom ocenom (3.2.2 - *Ocenjivanje rezultata*).

3.2.2 Ocenjivanje rezultata

Nakon obavljene pretrage, rezultate je potrebno prikazati korisniku. Zbog strukture HS-upita koji omogućava da rezultati ne sadrže sve njegove termine, postoje „bolji“ i „lošiji“ rezultati. Rezultati koji imaju veće poklapanje (sadrže više termina koji su definisani u HS-upitu) ocenjuju se kao bolji rezultati u odnosu na one rezultate koji imaju manje poklapanje.

Takođe, rezultati gde se HS-upit poklapa sa imenom emisija, smatraju se kao bolji rezultati u odnosu na one rezultate koji imaju poklapanje u opisu. Hibernate Search omogućava definisanje koeficijenata koji određuju koliko puta su rezultati sa poklapanjem na određenim poljima (u ovom slučaju se radi o polju naslov) bolji u odnosu na rezultate koji imaju poklapanje na ostalim poljima.

U *Tabela 3.2* su date ocene za tri emisije za sledeći K-upit „Ana voli Milovana“

| | Emisija 1 | Emisija2 | Emisija3 |
|-------|---|------------------------------|--|
| Ime | Ana za Vas | Film - Bitka na Košarama | Serijsa – Milovan u ljubavi |
| Opis | Emisija koju vodi naša poznata voditeljica Ana. | Domaći film Milovana Drecuna | Domaća serija snimljena na osnovu poznate palindromne rečenice Ana voli Milovana |
| Ocena | 1.4 | 0.6 | 1.6 |

Tabela 3.2 - Primer ocenjenih emisija

Emisija 2 je ocenjen sa najmanjom ocenom, zato što je pronađen jedan termin iz upita i to u opisu emisije. Ocena emisije 1 je prilično visoka, a to je uzrokovano pronalaženjem termina „Ana“ u naslovu emisije. Poslednja emisija 3 ima tek malo veću ocenu od emisije 1. Iako su pronađeni svi termini definisani u K-upitu, oni se nalaze u opisu emisije, koji se boduje slabijom ocenom po pronađenom terminu.

3.3 Tehnika bržeg pristupa rezultatima

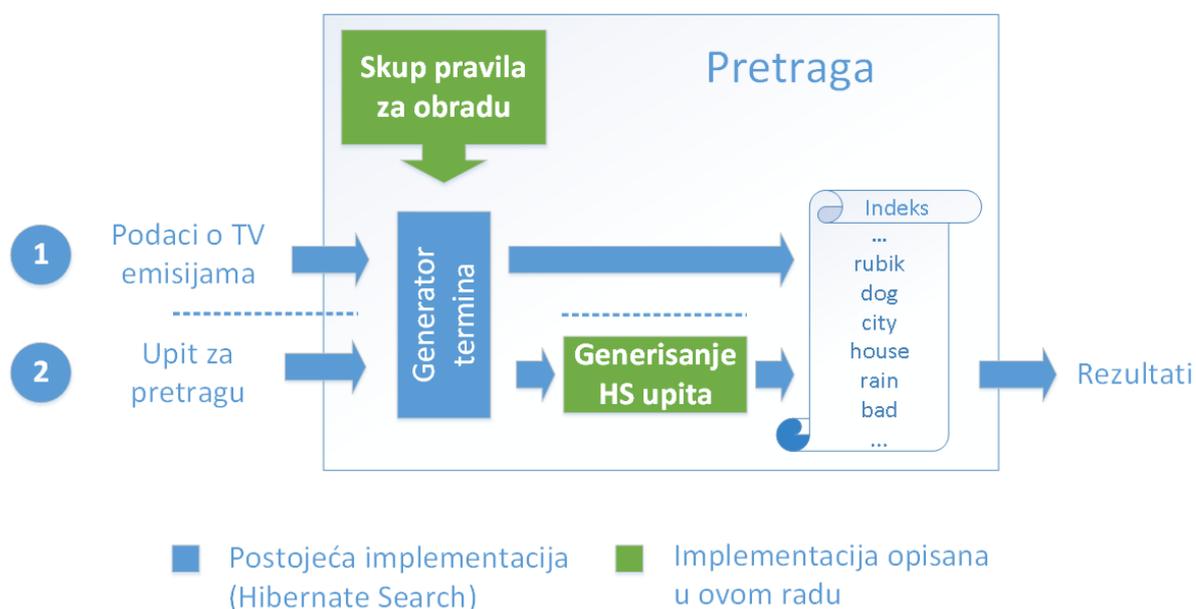
Kada se pretraži indeks, Hibernate Search, automatski dobavlja odgovarajuću emisiju iz baze podataka. Taj mehanizam je spor pošto se svaka emisija ponaosob dobavlja iz baze podataka[9][10].

Da bi se ubrzao proces dobavljanja emisija, koristi se drugačiji pristup. Prilikom indeksiranja emisije u indeksu se čuvaju i oni podaci koji se ne pretražuju opisanom tehnikom. Pošto indeks može da sadrži samo tekstualne podatke, tipovi podataka koji to nisu (npr. tip Set) moraju biti pretvoreni u tekst. Hibernate Search pruža mehanizam koji nam omogućava da opišemo na koji način se neki podaci pretvaraju u tekst. Nakon toga, kada se izvrši pretraga indeksa, kao rezultati se preuzimaju dokumenti koji sadrže sve podatke kao i emisija prilikom indeksiranja. To omogućava da se ta emisija ponovo formira, što je brže nego njeno dobavljanje iz baze podataka.

4. Programsko rešenje

U prethodnim poglavljima je dat uvod i opis problematike pretrage teksta. Izneti su zahtevi koje pretraga treba da zadovolji, kao principi po kojim pretraga treba da funkcioniše. Opisani su i mehanizmi Hibernate Search-a koji su korišćeni u ovom rešenju.

U ovom poglavlju je opisan postupak obrade informacija o TV emisijama (Slika 4.1, tok podataka označen brojem 1) i postupak pretraživanja (Slika 4.1, tok broj 2).



Slika 4.1 - Slučaj korišćenja Hibernate Search okruženja

Blokovi plave boje kao i indeks i pristup indeksu su deo implementacije Hibernate Search-a. Zelenom bojom su označeni blokovi koji predstavljaju implementaciju, a koja je opisana dalje u nastavku ovog poglavlja.

4.1 Indeksiranje informacija o emisijama

Kada akvizicioni uređaji prikupljene podatke pošalju na poslužilac (podaci o emisijama) i kada se ti podaci upišu u bazu podataka, iste je potrebno obraditi.

Zbog prirode problema, odnosno zbog čestog upisivanja novih informacija o emisijama u bazu, odabrano je dinamičko dopunjavanje indeksa.

Na osnovu podatka o tome na kojem su jeziku napisane informacije o emisijama, koje je definisano u polju *language* svake emisije, odabira se odgovarajući skup pravila, kojima će se data emisija obraditi i indeksirati. Potrebno je da jedan skup pravila za obradu ima definisana pravila opisana u poglavlju 3.1 - *Pravila za generisanje termina*.

Podaci kojima je emisija opisana kao recimo žanr ili vreme trajanja se ne pretražuju, ali se ipak skladište u indeks, bez obrade, radi kasnijeg bržeg pristupa rezultatima opisanog u poglavlju 3.3. To se postiže korišćenjem Hibernate Search klase *FieldBridge*, gde se definiše kako se neki podaci koji nisu tekst, prevode u tekstualni oblik radi smeštanja u indeks.

Nakon što je završena obrada informacija emisije, u zavisnosti od jezika na kom su napisane informacije o emisiji, bira se odgovarajući indeks i obrađeni podaci – dokumenti – se smeštaju u njega. Indeks se čuva na trajnoj memoriji (na hard disku) kako bi ostao dostupan i nakon ponovnog uključivanja sistema.

4.2 Pretraga indeksa

Prilikom pretrage, umesto da se pretražuju emisije smeštene u bazi, pretražuju se termini u indeksu. Pored brzine koja se dobija ovom tehnikom, veća je i fleksibilnost prilikom kreiranja upita.

Korisnički upit je niz reči koji opisuju sadržaj koji korisnik traži. Radi definisanja Hibernate Search upita za pretragu termina u indeksu, potrebno je reči iz korisničkog upita prevesti u termine kako bi poređenje bilo validno.

Svaki korisnički upit se obrađuje svim definisanim skupovima pravila i pretražuje u indeksima svih jezika, kako bi se korisnik što manje opterećivao odabirom jezika.

U slučaju da korisnik koristi daljinski uređaj za pretragu, algoritam pretrage omogućava korišćenje nedovršenih upita za pronalaženje željene TV emisije. Koristi se tehnika predviđanja, tako da K-upit ne mora biti napisan do kraja. Da bi se to postiglo, zadnja reč u upitu se posmatra na dva načina – kao da je napisana do kraja i kao reč koja još nije završena. To se postiže tako što se u HS-upitu definiše da dokumenti sadrže poslednji termin iz K-upita ili termin koji počinje kao poslednji termin u K-upitu - *Tabela 4.1*.

| Upit | Rezultat |
|-------------|--|
| This is spa | This is Sparta This is space team This is Sparky the dog |

Tabela 4.1 - Primer rezultata za delimično poklapanje poslednjeg termina

Kombinacijom `TermQuery`, `WildcardQuery` i `BooleanQuery` klasa, generiše se HS-upit. Za sve reči u K-upitu se definiše `TermQuery`, dok se za poslednju reč dodatno definiše `WildcardQuery`. Zatim se ti upiti kombinuju korišćenjem `BooleanQuery` klase, tako što se za sve upite postavlja vrednost `MUST` osim za upit definisan za poslednju reč u K-upitu, koji se definiše kao `SHOULD`.

Taj postupak je grafički prikazan u *Tabela 4.2*.

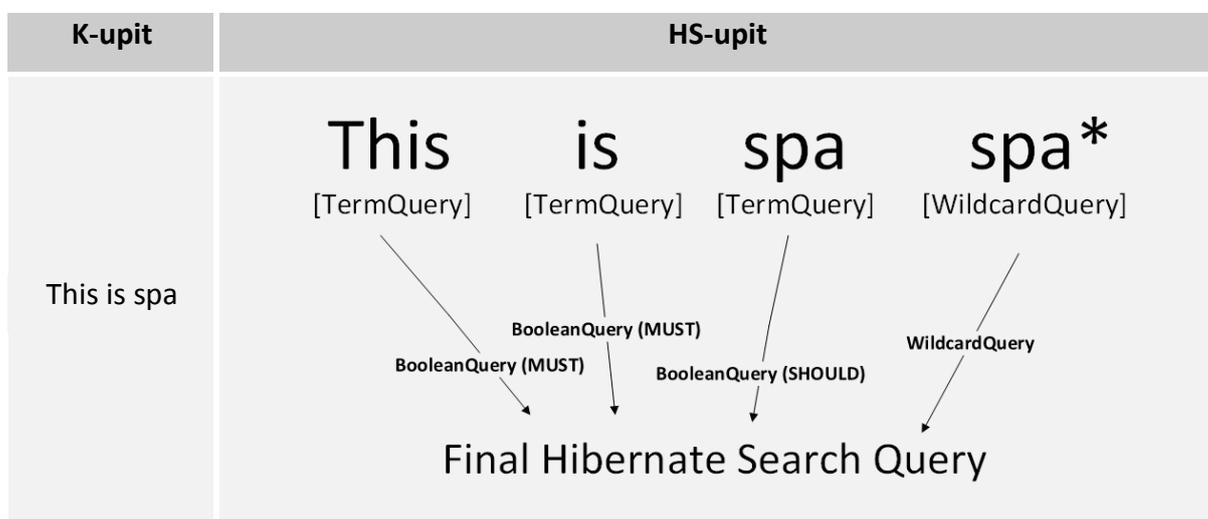


Tabela 4.2 - Generisanje HS-upita od postojećeg K-Upita

Izvršavanje ovako definisanog upita će vratiti niz ocenjenih dokumenata iz indeksa. Dokumenti se sortiraju od dokumenata sa najvišom ocenom do dokumenata sa najnižom ocenom. Nakon toga, formira se emisija na osnovu podataka iz dokumenata. Na kraju, korisniku se vraća lista emisija koje se najbolje poklapaju sa korisnikovim upitom.

4.3 Opis programskih modula

U ovom poglavlju je opisana implementacija pojedinih programskih modula, koji su neophodni za realizaciju opisanog sistema pretrage.

4.3.1 DirectoryProvider

Ovaj modul određuje direktorijum gde će biti sačuvan indeks u kojem se čuvaju generisani termini. Nasleđuje klasu `FSDirectoryProvider` i prepisuje funkciju `initialize`. Prilikom

inicijalizacije, podešava se parametar „*indexBase*“, na vrednost putanje gde treba da bude smešten indeks.

```
@Override
public void initialize(String directoryProviderName, Properties properties, BuildContext context) {
    Properties p = new Properties(properties);
    p.setProperty("indexBase", PUTANJA_DO_FOLDERA);
    super.initialize(directoryProviderName, p, context);
}
```

Opisani *DirectoryProvider* modul se uključuje u sistem, dodavanjem sledećih linija u konfiguracioni *hibernate.cfg.xml* fajl

```
<property name="hibernate.search.default.directory_provider">
    com.rtrk.cmore.lucene.DirectoryProvider
</property>
```

4.3.2 ShardingStrategy

Modul za deljenje indeksa na osnovu jezika na kojem je opis emisije napisan. Koristi se parametar *language* koji je prisutan u opisu emisije, kako bi se na osnovu njega kreirao određeni indeks. Nasleđuje klasu *ShardIdentifierProviderTemplate* i prepisuje funkcije *getShardIdentifier* i *loadInitialShardNames*.

```
@Override
public String getShardIdentifier(Class<?> entityType, Serializable id, String idAsString, Document document) {
    if (entityType.equals(EPGEvent.class)) {
        Fieldable f = document.getFieldable("language");
        String type = null;
        if (f != null) type = f.stringValue();
        if (type == null) type = "undefined";
        addShard(type);
        return type;
    }
    throw new RuntimeException("EPGEvent expected but found " + entityType);
}

@Override
protected Set<String> loadInitialShardNames(Properties properties,
    BuildContext buildContext) {
    ServiceManager serviceManager = buildContext.getServiceManager();
    SessionFactory sessionFactory = serviceManager.requestService(
        HibernateSessionFactoryServiceProvider.class, buildContext);
    Session session = sessionFactory.openSession();
    try {
        Criteria initialShardsCriteria = session.createCriteria(EPGEvent.class);
        initialShardsCriteria.setProjection(Projections.distinct(Property.forName("language")));

        @SuppressWarnings("unchecked")
        List<String> initialTypes = initialShardsCriteria.list();
        return new HashSet<String>(initialTypes);
    } finally {
        session.close();
    }
}
```

Opisani *ShardingStrategy* modul se uključuje u sistem, dodavanjem sledećih linija u konfiguracioni *hibernate.cfg.xml* fajl

```
<property name="hibernate.search.default.sharding_strategy">
  com.rtrk.cmore.lucene.ShardingStrategy
</property>
```

4.3.3 LanguageDiscriminator

LanguageDiscriminator modul služi za odlučivanje koji skup pravila će biti primenjen na obradu opisa emisije. Takođe se oslanja na parametar *language* koji je prisutan u opisu emisije. Implementira interfejs *Discriminator* i funkciju *getAnalyzerDefinitionName*.

```
@Override
public String getAnalyzerDefinitionName(Object value, Object entity, String field) {
  String defName = ((EPGEvent)entity).getLanguage();
  if(defName.equals("eng")) {
    return "eng";
  } else if(defName.equals("ger")) {
    return "ger";
  } else if(...) {
    .
    .
  } else {
    return "general";
  }
}
```

Prilikom definicije skupova pravila, *LanguageDiscriminator* se uključuje korišćenjem anotacije

```
@AnalyzerDiscriminator(impl = LanguageDiscriminator.class)
```

4.3.4 Pravila indeksiranja

Za svaki jezik je definisan skup pravila kojim se emisije čije su informacije napisane na datom jeziku indeksiraju. Svaki skup ima ime, a korišćenjem *LanguageDiscriminator*-a opisanog u prethodnom poglavlju, odabira se određeni skup koji se primenjuje u obradi. Dat je primer jednog skupa pravila za engleski jezik.

```
@AnalyzerDef(name = "eng", tokenizer = @TokenizerDef(factory = StandardTokenizerFactory.class), filters = {
  @TokenFilterDef(factory = StandardFilterFactory.class),
  @TokenFilterDef(factory = LowerCaseFilterFactory.class),
  @TokenFilterDef(factory = StopFilterFactory.class),
  @TokenFilterDef(factory = SnowballPorterFilterFactory.class, params = @Parameter(name = "language",
value = "English")),
  @TokenFilterDef(factory = SintagmaDetectorFactory.class) })
```

4.3.5 Indeksiranje

Indeksiranje se vrši automatski, prilikom upisa/izmene/brisanja informacija o emisiji u/iz baze. To se postiže dodavanjem sledećih konfiguracionih linija u *hibernate.cfg.xml* fajl

```
<event type="post-insert">
  <listener class="org.hibernate.search.event.FullTextIndexEventListener" />
</event>
<event type="post-update">
  <listener class="org.hibernate.search.event.FullTextIndexEventListener" />
</event>
<event type="post-delete">
  <listener class="org.hibernate.search.event.FullTextIndexEventListener" />
</event>
```

4.3.6 Pretraga

Sama pretraga indeksa i vraćanje rezultata predstavlja najkompleksniji deo ovog sistema za pretragu. K-upit se obrađuje i izdvajaju se termini:

```
Reader stringReader = new StringReader(new String(K_UPIT));
TokenStream tokenStream = analyzer.tokenStream("nan", stringReader);
CharTermAttribute charTermAttribute = tokenStream.addAttribute(CharTermAttribute.class);
tokenStream.incrementToken();
String term = charTermAttribute.toString();
```

Izdvojeni termini se zatim koriste za generisanje HS-upita. Generiše se upit opisan u 4.2 - *Pretraga indeksa*. Radi pojednostavljenja koda, prikazano je samo generisanje HS-upita koje vrši pretragu po imenima emisije. Analogno tome se generiše i upit za pretragu po opisu i produženom opisu emisije.

```
String lastToken = null;
try {
  while (tokenStream.incrementToken()) {
    if (lastToken != null) {
      partialQuery = new TermQuery(new Term("name",lastToken), Occur.MUST);
      partialQuery.setBoost(nameBoost);
      combinedTmpQuery.add(partialQuery, Occur.MUST);
      combinedQuery.add(combinedTmpQuery, Occur.MUST);
    }
    lastToken = charTermAttribute.toString();
  }
} catch (IOException e) {
  e.printStackTrace();
  return new ArrayList<EPGEvent>();
}

combinedTmpQuery = new BooleanQuery();

partialQuery = new TermQuery(new Term("name", lastToken),Occur.SHOULD);
combinedTmpQuery.add(partialQuery, Occur.SHOULD);
partialQuery = new WildcardQuery(new Term("name", lastToken + "*"));
combinedTmpQuery.add(partialQuery, Occur.SHOULD);

combinedQuery.add(combinedTmpQuery, Occur.SHOULD);
```

```
FullTextQuery query = fullTextEntityManager.createFullTextQuery(combinedQuery, EPGEvent.class);
```

Nakon generisanja HS-upita pretražuje se indeks, a nađeni dokumenti se koriste kako bi se informacije o emisijama ponovo formirale. Ponovo, radi pojednostavljenja koda i manjeg prikaza, neka polja su izostavljena.

```
List<EPGEvent> events = new ArrayList<EPGEvent>();

query.setProjection(FullTextQuery.DOCUMENT, FullTextQuery.SCORE);

List<Object[]> documents = (ArrayList<Object[]>) query.getResultList();

for (int j = 0; j < documents.size(); j++) {
    EPGEvent event = new EPGEvent();
    Document document = (Document) (documents.get(j)[0]);

    event.setId(Integer.valueOf(document.getFieldable("id").stringValue()));
    event.setName(document.getFieldable("name").stringValue());
    event.setDuration(Integer.parseInt(document.getFieldable("duration").stringValue()));
    event.setLanguage(document.getFieldable("language").stringValue());

    events.add(event);
}

return events;
```

5. Rezultati

Prilikom testiranja, realizovani podsistem za pretragu je pokazao duplo veći utrošak vremena za obavljanje pretrage od pretrage realizovane samo pomoću SQL upita. Kako bi utrošak tog vremena bio opravdan, realizovana su još dva sistema pretrage bazirana na SQL upitima, kako bi se poređenjem vremena izvršavanja pretrage kao i kvaliteta pronađenih rezultata, proverilo da li je Hibernate Search dobar izbor.

Prvi (SQL) od dva dodatna sistema pretrage za poređenje se zasniva na SQL pretrazi, gde se podaci u bazi podataka pretražuju sa K-upitom koji se ne obrađuje – opisano detaljnije u 2.4.1 - *Pretraga pomoću SQL upita*.

Drugi sistem pretrage (SQL + L) je hibrid, gde se K-upit obrađuje pomoću Lucene biblioteke, a SQL upit je generisan na osnovu dobijenih termina, slično generisanju HS-upita.

Implementacija pretrage opisana u ovom radu je u tabelama označena skraćenicom HS.

5.1 Poređenje vremena izvršavanja

Poređenje vremena izvršavanja upita je sprovedeno izvršavanjem 20 upita za svaku implementaciju i merenjem potrebnog vremena za izvršavanje svakog upita. Baza podataka je u trenutku testiranja sadržala oko 10.000 opisa emisija. Srednje vrednosti izvršavanja za svaku implementaciju su date u *Tabela 5.1*.

| Implementacija | Vreme izvršavanja [ms] |
|----------------|------------------------|
| SQL | 38 |
| SQL+L | 139 |
| HS | 94 |

Tabela 5.1 - Poređenje vremena izvršavanja za različite implementacije

Iz *Tabela 5.1* se vidi da se SQL upiti generisani samo na osnovu K-upita, izvršavaju najbrže. Nešto sporije se izvršava pretraga opisana u ovom radu, dok je implementacija, gde se

SQL upit generiše korišćenjem termina dobijenih obradom pomoću Lucene biblioteke, pokazala najlošije rezultate.

5.2 Poređenje kvaliteta rezultata

Kvalitet pronađenog rezultata je definisan kao odnos koliko se smisao rezultata poklapa sa smislom upita. Na primer, kada gledalac želi da pronađe film “Air Bud” (film o psu lualici koji igra košarku), pokušaće da ga pronađe uz pomoć nekoliko upita. Upiti su obrađeni sa svakim od opisanih implementacija pretraga, a rezultati su prikazani u *Tabela 5.2*.

| Upit | Implementacija | Broj rezultata | Poklapanje značenja | Air Bud u prvih 10? |
|------------------------|----------------|---------------------------|---------------------|---------------------|
| dog | SQL | >500 | < 10% | Ne |
| | SQL+L | | | |
| | HS | 34 | 85% | Da |
| dog playing | SQL | Nema pronađenih rezultata | | |
| | SQL+L | 67 | 46% | Ne |
| | HS | 14 | 93% | Da |
| dog playing basketball | SQL | Nema pronađenih rešenja | | |
| | SQL+L | 43 | 42% | Ne |
| | HS | 7 | 100% | Da |

Tabela 5.2 - Poređenje rezultata različitih implementacija

Kada se korisnički upit sastoji od samo jedne reči, SQL i SQL+L implementacije pronalaze jako veliki broj rezultata. Pri tom se jako mali broj rezultata poklapa po značenju sa K-upitom. U slučaju implementacije opisane u ovom radu, pronađen je znatno manji broj rezultata, ali procentualno broj rezultata koji se poklapaju po značenju je veći.

Kada se korisnički upit sastoji od dve reči, SQL implementacija ne vraća nikakve rezultate. SQL+L implementacija vraća 67 rezultata, gde se manje od polovine rezultata poklapa po značenju sa K-upitom. HS implementacija i u ovom slučaju pokazuje najbolje rezultate. Pronađeno je 14 rezultata i skoro svi rezultati se poklapaju po značenju sa K-upitom.

U slučaju gde je korisnički upit definisan sa tri reči, SQL implementacija takođe ne nalazi nikakve rezultate. SQL+L implementacija vraća 43 rezultata, a slično kao i za prethodni K-upit od dve reči, manje od polovine rezultata se poklapa sa značenjem K-upita. Na kraju, HS implementacija vraća 7 rezultata, gde se svi rezultati poklapaju sa značenjem K-upita.

Pored toga što HS implementacija pokazuje bolje rezultate u odnosu na SQL i SQL+L implementaciju, rezultati koji se dobiju su ocenjeni i korisniku je moguće servirati prvo najbolje ocenjene rezultate.

6. Zaključak

U ovom radu je predstavljeno jedno rešenje za realizaciju podsistema pretrage i analize teksta u personalizovanom elektronskom programskom vodiču. Posebna pažnja je posvećena realizaciji algoritma pretrage, kako bi i korisnici sa ograničenim unosom teksta mogli lako da koriste pretragu.

Rezultati pretrage moraju biti slični po značenju onome što gledalac unese kao upit za pretragu[11][12]. Da bi to postigli koristili smo tehnike indeksiranja i obrade teksta, kao i posebne upite za pretragu koje omogućava Hibernate Search biblioteka. Ona po svemu zadovoljava kriterijume za realizaciju ove analize i pretrage teksta, a omogućava i još dosta prostora za dalja proširenja.

Da bi opravdali izbor Hibernate Search-a, napravljeno je i poređenje sa pretragom pomoću SQL upita. Prilikom pretrage, tražen je film Air Bud, gde korisnik definiše upit na osnovu onoga što je predstavljeno u filmu, a to je pas koji igra košarku. Na osnovu dobijenih rezultata, vidi se da je odabir Hibernate Search biblioteke opravdan i pored toga što je vreme izvršavanja pretrage veće u odnosu na klasični SQL upit. Broj rezultata dobijenih pretragom je mali, a poklapanje po značenju sa upitom je veliko. Traženi film „Air Bud“ je pronađen za svaki izvršen upit i nalazi se među prvih 10 rezultata koji se prikazuju korisniku.

Među pronađenim rezultatima za date upite u poglavlju 5.2 - *Poređenje kvaliteta rezultata* pojavljuju se i filmovi kao što su: The Game Plan, Snow Dogs i Hotel for Dogs, koji u određenoj meri imaju veze sa datim upitom. Ostatak rezultata su uglavnom emisije čije su teme životinje.

6.1 Dodatna proširenja

Radi daljeg unapređenja pretrage, moguće je koristiti neke od upita koje omogućava Hibernate Search, a koji nisu korišćeni u ovoj implementaciji.

FuzzyQuery klasa omogućava poređenje termina sa definisanjem koeficijenta sličnosti. Opisana je u *2.4.3 - Lucene, Solr, Hibernate Search* pod stavkom Fuzzy upit. Pogodno za pretraživanje indeksa kada korisnik napravi štamparsku grešku prilikom definisanja upita.

PhraseQuery (takođe opisan u *2.4.3 - Lucene, Solr, Hibernate Search* kao Upit sa izrazom) se može koristiti radi boljeg ocenjivanja onih emisija čiji opis sadrži termine u tačno onakvom redosledu kako su oni definisani u upitu.

7. Literatura

- [1] Code of practice for an Electronic Programme Guide (EPG)
http://www.etsi.org/deliver/etsi_etr/200_299/288/01_60/etr_288e01p.pdf , posećeno 22.02.2014.
- [2] S. Pijetlovic, N. Jovanovic, N. Jovanov, S. Ocovaj, “One solution of a system for data acquisition and storage from the digital television transport stream and its exposure to the clients”, 21st Telecommunications Forum TELFOR, Belgrade, November 2013.
- [3] Java Enterprise Edition (JEE) hocs.oracle.com/javaee/, posećeno 30.07.2014.
- [4] Hibernate, hibernate.org, posećeno 17.05.2014.
- [5] Lucene, lucene.apache.org, posećeno 21.05.2014.
- [6] Solr, lucene.apache.org/solr, posećeno 18.05.2014.
- [7] Hibernate Search, hibernate.org/search, posećeno 17.05.2014.
- [8] S. D. Samantaray "An Intelligent Concept based Search Engine with Cross Linguility support", Industrial Electronics and Applications (ICIEA), 2012
- [9] Yinan Jing, Chunwang Zhang, Xueping Wang, “An empirical study on performance comparison of Lucene and relational database”, International Conference on Communication Software and Networks 2009
- [10] Shengdong Li, Xueqiang Lv, Feng Ling, Shuicai Shi “Study on Efficiency of Full-Text Retrieval Based on Lucene”, Information Engineering and Computer Science, 2009
- [11] Fei Shi, Mefford, C. "A New Indexing Method for Approximate Search in Text Databases", Computer and Information Technology, 2005
- [12] Charu C. Aggarwal, Philip S. Yu "On effective conceptual indexing and similarity search in text data", Data Mining, ICDM 2001