Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Viktor Harej

Katalogizacijsko orodje za urejanje in bogatenje zapisov iz sistema COBISS

diplomsko delo

visokošolski strokovni študijski program prve stopnje računalništvo in informatika

Mentor: doc. dr. Dejan Lavbič

Ljubljana, 2017

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

Področje integracije podatkov je aktualno raziskovalno področje, kjer je v zadnjem času vedno večji poudarek na semantični integraciji. To velja tudi za področje urejanja bibliografskih zapisov, ki jih v Sloveniji obvladujemo v okviru sistema COBISS. V okviru diplomskega dela je potrebno najprej raziskati stanje bibliografskih podatkov v Sloveniji in nato predlagati pristop bogatenja podatkov v sistemu COBISS s pomočjo obstoječih spletnih virov. Rezultate raziskave predstavite v obliki delujočega prototipa orodja, ki katalogizatorju omogoča urejanje in bogatenje zapisov v sistemu COBISS. Na koncu svoj pristop tudi kritično ovrednotite.

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
COBISS	Co-operative Bibliographic System & Services	Kooperativni online bibliografski sistem in servisi
FRBR	Functional Requirements for bibliographic records	Funkcionalne zahteve za bibliografske zapise (FZBZ)
RDA	Resource Description and Access	Opis virov in dostop
MARC	MAchine-Readable Cataloging	Strojno berljiv bibliografski opis
VIAF	Virtual International Authority File	Virtualna mednarodna normativna datoteka
OCLC	Online Computer Library Center	
RDF	Resource Description Framework	Ogrodje za opis virov
SPARQL	SPARQL Protocol and RDF Query Language	protokol SPARQL in poizvedovalni jezik za RDF
NER	Named Entity Recognition	Razporaznavanje imenskih entitet
POS	Part of speech	Skladenjski deli jezika
ISBD	International Standard Bibliographic Description	Mednarodni standardni bibliografski opis
NLP	Natural Language Processing	Obdelava naravnega jezika
API	Application Programming Interface	Programski vmesnik

OCR	Optical Character Recognition	Optično razpoznavanje znakov
XML	Extensible Markup Language	Razširljivi označevalni jezik
JSON	JavaScript Object Notation	Opis JavaScript objekta
DOM	Document Object Model	Objektni model dokumenta
SWOT	strengths, weaknesses, opportunities, and threats	prednosti, slabosti, priložnosti in grožnje
URI	Universal Resource Identifier	Enolični identifikator vira
URL	Universal Resource Locator	Enolični krajevnik vira
HTML	Hyper Text Markup Language	Jezik za označevanje hiperteksta
OSGi	Open Services Gateway initiative	

Povzetek

Naslov: Katalogizacijsko orodje za urejanje in bogatenje zapisov iz sistema COBISS

Diplomsko delo predstavi katalogizacijsko orodje, ki omogoča urejanje in bogatenje bibliografskih zapisov iz sistema COBISS. V prvem koraku orodje deluje tako, da polušči zapise iz delno stukturirane oblike na spletu ter jih pripravi v skladu s poenostavljenim konceptualnim modelom FZBZ. Nadalje je katalogizatorju omogočeno, da s pomočjo bibliografskih podatkov, pridobljenih iz različnih spletnih storitev, preveri in obogati podatke iz sistema COBISS. Delo vsebuje predstavitev problemskega področja, analizo podatkovnih virov, analizo tehničnih rešitev pri manipulaciji teksta in združevanju podatkov. Končni prispevek dela je prototip spletne aplikacije, ki katalogizatorju omogoča, da lahko, na kar se da intuitiven in uporabniško prijazen način, preverja in povezuje podatke. Možen pristop k FRBRizaciji z našim prototipom skušamo še ovrednotiti, prikažemo pa tudi nekaj iskalnih uporabniških scenarijev, ki jih lahko bolje naslovimo.

Ključne besede: COBISS, katalogizacija, bogatenje zapisov.

Abstract

Title: Cataloguing tool for manipulation and enrichment of records from COBISS

Thesis presents cataloguing tool, which enables manipulation and enrichment of bibliographic records retrieved from COBISS. In the first step tool scrapes records from COBISS and prepares them in accordance with simplified FRBR conceptual model. In the next step, system enables cataloguer to check and enrich data from COBISS, using data retrieved from different data sources on the web. Our thesis includes presentation of problem domain, analysis of data sources, analyses of technical solutions for text manipulation and data integration. Final contribution is a web application prototype, which supports cataloguers work flow and enables him to check and integrate data in intuitive and user-friendly way. Thesis includes the evaluation of possible approach to FRBRization using our tool, and it also discusses the possible added value regarding better search options for the user.

Keywords: COBISS, cataloguing, enrichment of records.

Poglavje 1

Uvod

1.1 Motivacija

Namen diplomskega dela je izdelati orodje, ki katalogizatorju asistira pri urejanju in bogatitvi kataložnih zapisov iz COBISS-a. Poudarek je na t. i. FRBRizaciji – pripravi zapisov v skladu z modelom FZBZ (funkcionalne zahteve za bibliografske zapise, angl. FRBR). Gre za konceptualni model uporabnika bibliografskega univerzuma, predstavljen v entitetno relacijski tehniki. Pri postopku FRBRizacije orodje smiselno povezuje podatke, izluščene iz bibliografskih zapisov na COBISS-u, nadalje pa omogoča tudi povezovanje, preverjanje in bogatenje s podatki iz virov, ki so prosto dostopni na spletu. Namen je zajeti koristne atribute entitet in relacije med entitetami, ki nastopajo v bibliografskem univerzumu. Konkretneje, gre za relacije med posameznimi deli, pojavnimi oblikami in osebami ter korporacijami, ki so prispevale k nastanku dela in pojavne oblike.

Aplikacija predpostavlja nekoliko drugačno vlogo katalogizatorja, ki ni več zgolj prepisovalec podatkov iz fizičnih enot (knjig, revij, filmov itd.), temveč skuša s pomočjo podatkov s spleta, lastne razgledanosti in poznavanju uporabniških scenarijev zajeti čim več povezav med deli, pojavnimi oblikami in entitetami odgovornosti. Aplikacija bo torej katalogizatorju skušala omogočiti, da prevzame vlogo povezovalca bibliografskih podatkov s pomočjo

podatkov na spletu.

1.2 Metodologija

Razvoj prototipa najprej zahteva, da se podrobno seznanimo z izgledom kataložnih zapisov v sistemu COBISS. Predpriprava je potekala na tak način, da smo najprej podrobneje analizirali izgled kataložnih zapisov v sistemu COBIS (poglavje 4). Skušali smo identificirati polja, ki nosijo največjo semantično vrednost, v prvi vrsti za proces FRBRizacije, pa tudi za druge cilje, ki smo jih definirali vnaprej (poglavje 3).

V poglavju 5 sledi pregled prosto dostopnih podatkovnih virov na spletu in analiza, kateri podatkovni viri bi nam lahko najbolje koristili pri naših ciljih. Kvalitetnih virov podatkov na spletu je seveda več, vendar smo se za potrebe našega prototipa zadovoljili s tremi (LibraryThing, VIAF in DBpedia).

Poglavji 6 in 7 opisujeta funkcionalnosti, delovanje ter uporabo našega prototipa. Arhitekturne in tehnične odločitve in pojasnila zanje pa se nahajajo v poglavju 8.

Naše orodje oz. naš pristop k postopku FRBRizacije in bogatenja zapisov bomo skušali tudi evalvirati (poglavje 9). Prvi korak evalvacije bo, da bomo navedli nekaj uporabniških scenarijev v bibliografskem okolju in skušali prikazati, da lahko s pripravo podatkov z našim orodjem odgovorimo nanje. Drugi korak bo analiza SWOT, pri čemer ne bomo vrednotili orodja, temveč primerjali naš pristop k FRBRizaciji ter bogatenju in povezovanju podatkov, z alternativnim t.i. hkratnim pristopom.

Ker gre za diplomsko nalogo s specifičnega domenskega področja, je v diplomski nalogi kar nekaj knjižnične terminologije. Poglavje 2, ki sledi, služi kot kratek in poenostavljen uvod v področje.

Poglavje 2

Razlaga konceptov in pojmov

2.1 COBISS

COBISS [2] je platforma oz. sistem za vzajemno katalogizacijo, ki predstavlja platformo nacionalnih knjižničnih informacijskih sistemov v Sloveniji in tudi po drugih državah Balkana. Vsi navedeni sistemi so povezani v regionalno mrežo COBISS.Net [3]. Katalogizatorji iz knjižnic po Sloveniji, ki so vključene v sistem COBISS, prispevajo in prevzemajo kataložne zapise o gradivu, ter osebah in korporacijah v sistem COBISS. Sistem v celoti sicer zajema več orodij in baz, pri čemer sta za nas najzanimivejši vzajemna bibliografsko-kataložna baza podatkov COBIB ter baza CONOR – baza normativnih zapisov imen avtorjev (obe sta dostopni preko sistema COBISS).

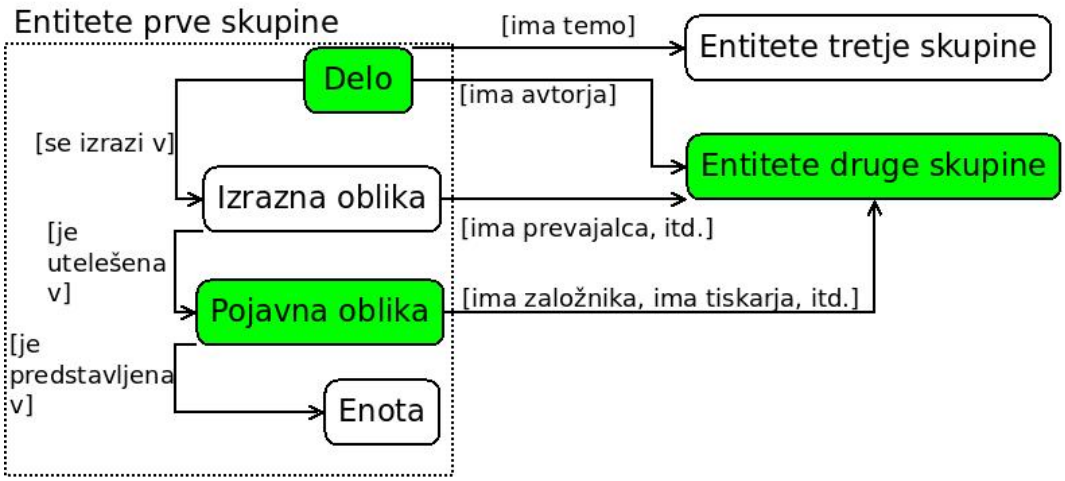
2.2 FZBZ oz. FRBR

FRBR oz. FZBZ je konceptualni model, predstavljen v entitetno relacijski tehniki. Eden od prispevkov modela, in hkrati glavni poudarek postopka FRBRizacije, je predpostavka, da vsaka bibliografska enota pripada delu, izrazni obliki, pojavni obliki in enoti [1]. Delo in izrazna oblika sta abstraktni entiteti, pri čemer delo predstavlja osnovno idejo, zajeto v gradivu, izrazna oblika pa je prvi poskus opisa ideje v obliki alfanumeričnih znakov. Pojavna oblika

in enota sta stvarna entitetna tipa, pojavno obliko pa si lahko predstavljamo kot celotno naklado določenega gradiva. Bibliografski zapis ponavadi opisuje prav pojavno obliko. Enota je posamezen predstavnik določene pojavne oblike, ki se od ostalih predstavnikov pojavne oblike razlikuje v individualnih lastnostih (avtogram na začetku knjige, strgane strani ipd.). Delo, izrazna oblika, pojavna oblika in enota so t.i. entitetni tipi prve skupine. Entitetnemu tipu druge skupine pripadajo tiste entitete, ki nosijo intelektualni ali umetniški prispevek za nastanek entitet prve skupine. V naši diplomski nalogi jih bomo poenostavljeno naslavljali tudi z izrazom "osebe in korporacije", in pri tem bomo imeli vselej v mislih bibliografski kontekst. Z entitetnimi tipi tretje skupine pa lahko opišemo vsebino dela.

Od entitet prve skupine bomo skušali implementirati le delo in pojavno obliko. Implementirali bomo tudi en splošni entitetni tip druge skupine (angl. contributor), ne pa tudi entitnih tipov tretje skupine (entitetni tipi povezani z vsebinskim opisom).

Delo in pojavno obliko najlažje pojasnimo s poenostavljenim primerom. Delo je torej abstraktna ideja oz. zaključen intelektualni in/ali umetniški prispevek, ki nosi neko poimenovanje. Npr.: zgodba o danskem princu, ki spozna, da sta njegov stric in mati skupaj ubila njegovega očeta in se polastila oblasti. To idejo tipično prepoznamo pod imenom (Shakespearjev) Hamlet. Pojavna oblika je manifestacija ideje v obliki alfanumeričnih znakov, lahko si jo predstavljamo kot naklado celotne izdaje. Slovenska izdaja v prevodu Janka Modra iz leta 2002 je primer pojavne oblike. Pozor: film Hamlet (1996) je manifestacija druge ideje oz. drugega dela, saj se ideja filma razlikuje od ideje drame oziroma je intelektualni in/ali umetniški vložek avtorjev filma znaten. Sicer pa obstajajo redki robni primeri, ko je težje presoditi, ali je ta vložek dovolj velik, da gre za novo delo. Diagram 2.1 prikazuje poenostavljen konceptualni model FZBZ. Z zeleno barvo so označeni entitetni tipi, ki jih bomo uporabljali v naši diplomski nalogi. Poenostavitev nekoliko poruši konsistentnost modela, npr. relacija ima "prevajalca" je relacija med izrazno obliko in entiteto druge skupine, vendar jo bomo v našem primeru primorani



Slika 2.1: Poenostavljeni model FZBZ – entitetni tipi, ki jih bomo uporabljali v naši nalogi, so obarvani zeleno

preseliti na nivo pojavne oblike – torej postane relacija med pojavno obliko in entiteto druge skupine. Prav tako bomo v našem modelu predpostavili relacijo med delom in pojavno obliko, čeprav med njima sicer ni neposredne relacije, temveč je vmes prisoten še entitetni tip izrazna oblika.

2.3 Postopek FRBRizacije

Postopek FRBRizacije je ekstrakcija strukture FZBZ iz obstoječih bibliografskih podatkov [4, str. 64]. Tipično gre za ekstrakcijo podatkov iz polj bibliografskih zapisov v kakšnem izmed formatov MARC. Natančno dokumentiran poskus FRBRizacije in opis omejitev trenutnih bibliografskih podatkov je opisan v [8].

2.4 Formati za zapis bibliografskih (meta)podatkov

COMARC (COBISS MARC) je binarni format za bibliografske zapise v sistemu COBISS. Gre za derivat iz družine MARC. Format MARC je nastal že leta 1965 in je namenjen shranjevanju in izmenjavi bibliografskih zapisov in sorodnih informacij v strojno berljivi obliki. Njegov glavni namen je viden tudi iz imena, MARC namreč pomeni MAchine-Readable Cataloging. Format za imena polj uporablja številke, danes pa se še vedno uporablja kot format za izmenjavo, manj pa za hranjenje zapisov [5].

Kot že omenjeno je MARC družina formatov – pravzaprav standard. Primer formata, ki temelji na standardu MARC je npr. MARC 21 [10], ki je danes najpopularnejši MARC derivat.

Format COMARC še natančneje temelji na formatu UNIMARC [6], katerega glavni namen je bil omogočiti mednarodno izmenjavo zapisov med nacionalnimi bibliografskimi ustanovami, hkrati pa lahko služi tudi kot osnovni model za nove formate za bibliografske podatke. UNIMARC ne določa interne oblike, vsebine ali strukture zapisov, temveč le priporoča obliko, vsebino in strukturo podatkov, kadar jih želimo izmenjevati. Ob nastanku sistema vzajemne katalogizacije v Sloveniji se je pri delu najprej uporabljalo prevod prve izdaje UNIMARC Manual. Z razvojem novih funkcij v sistemu COBISS je bilo treba format UNIMARC dopolnjevati z novimi polji in podpolji, kar je privedlo do razvoja formata COMARC [7].

2.5 Slovnice za zapis bibliografskih podatkov

Na polja formata MARC lahko gledamo kot na elemente slovnice, poleg tega pa obstajajo tudi modernejše oblike tega formata, ki nam olajšajo manipulacijo podatkov zakodiranih v tem formatu. Npr., format MARCXML, omogoča predstavitev podatkov MARC v notaciji XML [9].

Za potrebe diplomske naloge pa vendarle potrebujemo slovnico, ki nam

omogoča oblikovanje in zakodiranje v skladu z modelom FZBZ. Obstaja več takih slovnic RDF za opis bibliografskih podatkov, npr. FaBiO [12], FRBRoo [13], BIBFRAME [31] ter zastareli vocab [15].

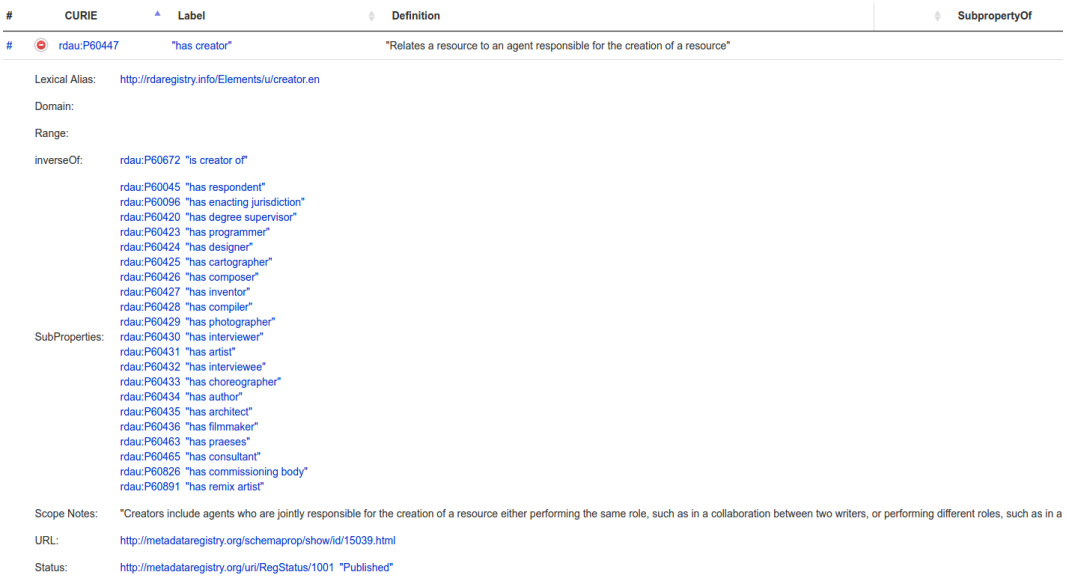
Tudi z drugimi bolj razširjenimi slovnicami za opis ontologij bi lahko zajeli nekatere koncepte iz modela FRBR. Slovnica za strukturiran opis podatkov na spletu schema.org [33] npr. vsebuje podoben element CreativeWork. Prav tako tudi ontologija, uporabljena na DBpediji [17] vsebuje nekatere elemente, ki omogočajo zapis konceptov, sorodnim tistim v modelu FRBR.

Mi bomo za del opisa delčka naših podatkov (diplomsko delo se osredotoča na relacije) uporabili slovnico RDA [18], ki velja za najbolj temeljito ontologijo za opis bibliografskih podatkov, po vzoru konceptualnega modela FRBR.

RDA (Resource Description and Access) poleg podatkovnih elementov (angl. data elements) izdaja tudi smernice in navodila za kreiranje metapodatkov za knjižnično gradivo in kulturno dediščino. Namen je izboljšati pripravo metapodatkov, da bodo ti čimbolj uporabni za kreiranje uporabniško prijaznih aplikacij, temelječih na povezanih podatkih (angl. Linked data) [34].

Poleg semantične izraznosti in prilagojenosti za platformo spleta in semantičnega spleta je prednost RDA tudi široko soglasje v knjižničarski skupnosti in širše. RDA je razvit kot skupni proces različnih inštitucij in posameznikov, ki ga nadzira in vodi RDA Steering Committee [19].

Uporabili bomo “unconstrained” verzijo te ontologije. Za razliko od “constrained” verzije RDA slovnice, elementi iz “uncostrained” verzije nimajo določenega ranga in domene. Ker bomo naš projekt naslonili le ne nekatere entitete iz konceptualnega modela FRBR (npr. izpuščamo izrazno obliko in enoto), je zato uporaba “unconstrained” verzije edina smiselna. Zaslonski posnetek 2.2 prikazuje izsek iz slovnice “unconstrained” RDA, natačneje relacijo rdau:P60447, z oznako angl. “has creator” oz. “ima avtorja”. Kot kaže slika, relacija nima navedenega niti ranga, niti domene, ima pa zato več podlastnosti.



#	CURIE	Label	Definition	SubpropertyOf
#	rdau:P60447	"has creator"	"Relates a resource to an agent responsible for the creation of a resource"	

Lexical Alias: http://rdaregistry.info/Elements/u/creator.en

Domain:

Range:

inverseOf: rdau:P60672 "is creator of"

SubProperties:
rdau:P60045 "has respondent"
rdau:P60096 "has enacting jurisdiction"
rdau:P60420 "has degree supervisor"
rdau:P60423 "has programmer"
rdau:P60424 "has designer"
rdau:P60425 "has cartographer"
rdau:P60426 "has composer"
rdau:P60427 "has inventor"
rdau:P60428 "has compiler"
rdau:P60429 "has photographer"
rdau:P60430 "has interviewer"
rdau:P60431 "has artist"
rdau:P60432 "has interviewee"
rdau:P60433 "has choreographer"
rdau:P60434 "has author"
rdau:P60435 "has architect"
rdau:P60436 "has filmmaker"
rdau:P60463 "has praeses"
rdau:P60465 "has consultant"
rdau:P60826 "has commissioning body"
rdau:P60891 "has remix artist"

Scope Notes: "Creators include agents who are jointly responsible for the creation of a resource either performing the same role, such as in a collaboration between two writers, or performing different roles, such as in a

URL: http://metadataregistry.org/schemaprop/show/id/15039.html

Status: http://metadataregistry.org/uri/RegStatus/1001 "Published"

Slika 2.2: Relacija “has creator” iz slovnice RDA “unconstrained”

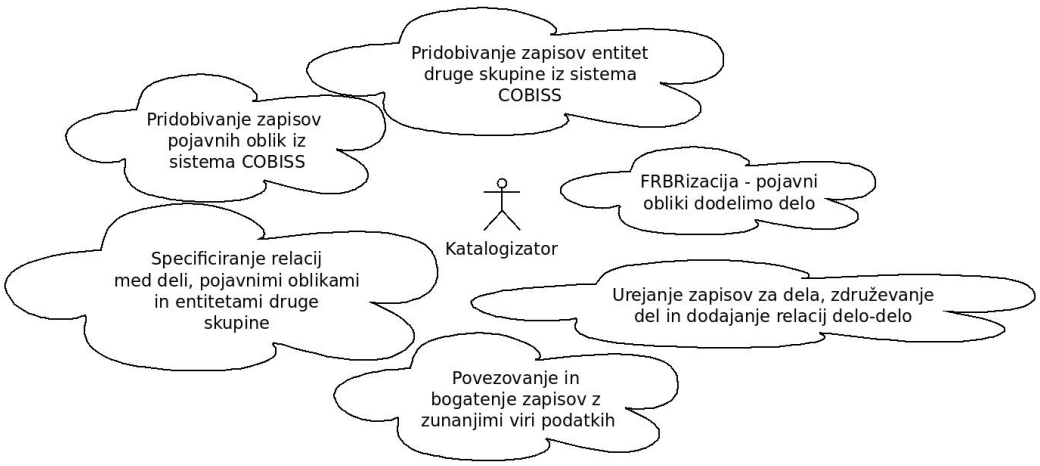
Poglavje 3

Cilji in željeni način delovanja orodja

Končni izdelek našega diplomskega dela je orodje, ki katalogizatorju asistira pri FRBRizaciji in obogatitvi kataložnih zapisov iz COBISS-a. Spodaj je natančneje razloženo, kaj vsebuje FRBRizacija in kaj obogatitev podatkov.

FRBRizacija:

- Oblikovanje zapisov in katalogiziranje v skladu s poenostavljenimi koncepti modela FRBR. Pojavne oblike, ki izvirajo iz iste ideje se združijo pod istim delom.
- Iskanje odgovornih oseb in korporacij oz. entitet odgovornosti (avtorjev, ilustratorjev in drugih, ki so dali umetniški ali intelektualni prispevek k nastanku dela, ter drugih entitet, ki so prespevali k nastanku pojavne oblike – npr. založnik, tiskar) in identifikacija relacij, ki jih ima ta entiteta z delom ali s pojavno obliko. Glavna relacija med delom kot abstraktno idejo in entiteto druge skupine je relacija, ki izraža avtorstvo.



Slika 3.1: Željene funkcionalnosti orodja

Obogatitev:

- V prvi vrsti lahko sam postopek FRBRizacija in kreiranje del ter "zbiranje" pojavnih oblik pod delom razumemo kot bogatitev.
- Viri na spletu opisujejo nekatere atribute in relacije med entitetami, ki jih ni v zapisih v sistemu COBISS.
- Ko našemu zapisu dodelimo identifikator zunanje storitve, smo ga bolje umestili v bibliografski univerzum.

Naše orodje bo torej omogočalo delno avtomatsko FRBRizacijo in bogatitev podatkov. Poleg tega bo lahko katalogizator vsak zapis naknadno še ročno popravil ter mu ročno dodajal atribute ali relacije. Orodje mora hkrati katalogizatorju ponuditi možnost odpravljanja napak v zapisih in razreševanje konfliktov – npr. združitev dveh del, če se izkaže, da gre za isto delo. Pričakovan obseg funkcionalnosti je grafično prikazan na diagramu 3.1

Namen rabe orodja pa je produkcija čimbolj kvalitetnih, torej semantično izraznih bibliografskih zapisov, s katerimi bomo lahko uporabniku ponudili čimveč možnosti brskanja in iskanja.

Poglavje 4

Analiza in obdelava teksta v kataložnih zapisih

V tem poglavju bomo najprej analizirali lastnosti teksta v kataložnih zapisih. Na podlagi tega bomo skušali identificirati tehnike in pristope, ki bi bili primerni za obdelavo teksta, konkretneje za strukturiranje teksta in izluščenje entitet ter relacij med njimi.

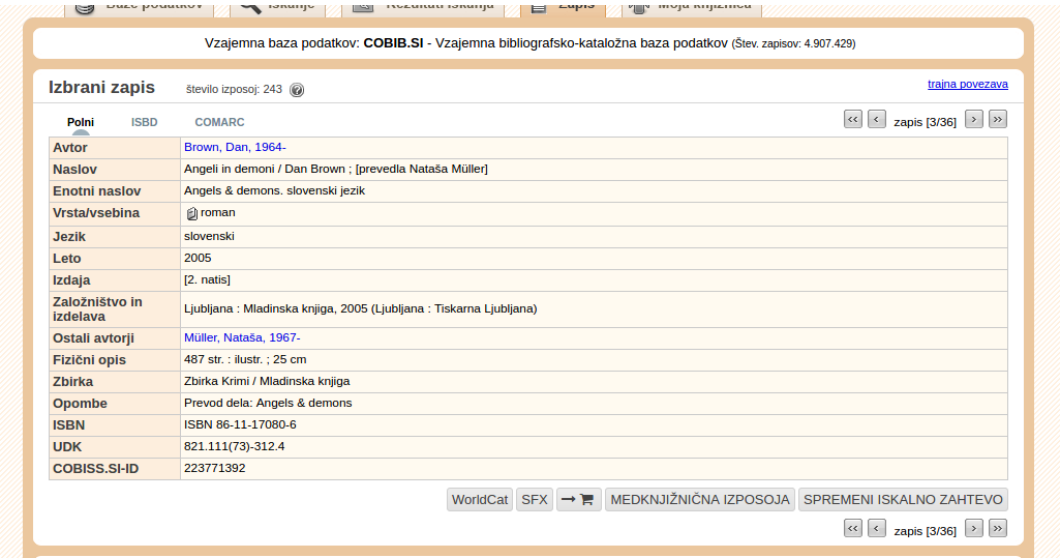
4.1 Lastnosti teksta v kataložnih zapisih

Kataložni zapis tipično opisuje pojavno obliko. Preden začnemo luščiti in strukturirati podatke iz kataložnih zapisov, moramo identificirati polja, ki nosijo za nas najpomembnejše informacije.

Nekoliko stiliziran kataložni zapis v formatu COMARC prikazuje zaslonski posnetek na sliki 4.1.

Prikaz v formatu COMARC je samo eden od možnih načinov prikaza zapisa v sistemu COBISS, na voljo imamo še formatiran in uporabniško prijaznejši format ("Polni"), ter zapis, stiliziran v skladu s standardom ISBD. V našem diplomskem delu bomo uporabljali luščenje podatkov iz zapisov v polnem formatu – primer takega zapisa je prikazan na zaslonskem posnetku 4.2.

Slika 4.1: Zapis iz COBISS-a, v formatu COMARC



Slika 4.2: Zapis iz COBISS-a, v polnem formatu

Relacije med deli lahko iščemo v določenih poljih kataložnega zapisa:

- V naslovu kataložnega zapisa. Naslov je v COBISS-u razumljen širše: prvi del je dejanski naslov, sledi znak "/", za tem znakom pa pride drugi del, kjer katalogizatorji zapišejo odgovorne osebe oz. korporacije. Primera naslovov, prepisanih iz COBISS-a, sta npr.
  "*Gospodar prstanov. [2], Stolpa / J. R. R. Tolkien ; [prevedel Branko Gradišnik]*" in
  "*Gospodar prstanov. [1], Bratovščina prstana / J. R. R. Tolkien ; [prevedel Branko Gradišnik]*",
  kjer gre za pojavni obliki dveh različnih del, med katerimi je relacija "nadaljevanje", se pravi delo Stolpa, ki ima poleg naslova navedeno št. 2, je nadaljevanje dela Bratovščine prstana. Drugi primer je npr. delo "Blade runner" oz. "Iztrebljevalec". V sistemu COBISS obstaja zapis za pojavno obliko tega dela, ki ima v drugem delu naslova navedeno tole: "based on the novel "Do androids dream of electric sheep?" by Philip K. Dick".

- V opombah, kjer so včasih zajete podobne informacije, kot jih lahko najdemo v naslovih, npr. da neko delo sloni oz. črpa inspiracijo iz nekega drugega dela ter tudi informacije o predhodnih delih oziroma nadaljevanjih. V sistemu COBISS najdemo zapis, ki ima v opombah zapisano tole: "Knjiga Rdeče zlata zgodovina je četrta v nizu Blato in druga v podnizu Blagor blata ...".

Relacije med deli, pojavnimi oblikami in entitetami odgovornosti najdemo:

- V polju naslov: npr. "Izbrani eseji in razprave Ivana Prijatelja / [uredil, uvod in opombe napisal A. Slodnjak]".

- V opombah.

- Morebitna dodatna opcija: s pomočjo optičnega razpoznavanja znakov bi lahko obdelali sliko kataložnega zapisa v formatu COMARC, ki je

tudi eden od ponujenih prikazov pojavne oblike v sistemu COBISS. V bloku polj 7XX najdemo imena entitet in njihovo vlogo oz. prispevek k nastanku pojavne oblike (primer polja 702: aVidmar bJosip f1895-1992 4340 – urednik). OCR je seveda zahteven proces in morali bi ga poganjati kot asinhroni proces v ozadju, uporabili pa bi ga lahko tudi za preverbo podatkov.

Pregled kataložnih zapisov pokaže, da ima tekst v kataložnih zapisih naslednje lastnosti:

- Prva lastnost je, da pri obdelavi zapisov iz COBISS-a srečujemo zgolj kratke skupke teksta – del naslova, ki označuje odgovornost in opombe.

- Druga lastnost teksta je, da sestoji iz besednjaka, ki je v veliki meri specifičen za opis bibliografskih podatkov.

- Tretja lastnost teksta je, da zaradi pravil katalogizacije in katalogizacijske prakse lahko približno predvidevamo, katera vrsta informacije se bo v določenem tekstu nahajala. Npr. zaradi pomanjkanja polja v formatu COMARC, kamor bi lahko katalogizator zabeležil naklado določene izdaje (npr. glasbene zgoščenke ali knjige), so katalogizatorji v Sloveniji začeli ta podatek zapisovati v polje opombe. Zaradi istega razloga je katalogizacijska praksa taka, da se v polje opombe zapiše tudi prevod dela, ki je za nas koristen podatek.

- Četrta lastnost teksta pa je, da je v večji meri v slovenskem jeziku.

V naši aplikaciji skušamo na več mestih čimbolj avtomatizirati iskanje entitet in relacij iz delno strukturiranega teksta. V naslednjem poglavju bomo naredili kratek pregled področja ter na podlagi identificiranih lastnosti teksta izbrali tehnike identifikacije entitet in relacij v tekstu.

4.2 Odkrivanje entitet in relacij iz teksta

4.2.1 Odkrivanje entitet

Odkrivanje entitet (in tudi relacij) iz teksta je eno od področij obdelave naravnega jezika. Odkrivanje entitet iz besedila (angl. Named Entity Recognition oz. NER) vključuje ekstrakcijo krajevnih, osebnih in stvarnih imen iz teksta. NER najpogosteje uporablja kot del procesiranja iskalne zahteve [27, str. 99] oz. kot eden izmed delov sistemov za poizvedovanje. Ponavadi jih najdemo znotraj tako imenovanih *parserjev* in *chunkerjev*, kjer skušamo prepoznati določeno entiteto in ugotoviti, ali ta ista entiteta nastopa tudi v drugih dokumentih, pa tudi ali je ta entiteta enaka entiteti iz drugega sistema, baze podatkov, storitve na spletu itd. V naši aplikaciji je izhodišče podobno – odkrivanje entitet izvajamo zato, da jo nato poskušamo povezati z njenim COBISS identifikatorjem in nadalje z drugimi identifikatorji iz različnih storitev: VIAF, DBpedia in LibraryThing. Kar tudi v našem primeru najbolj otežuje proces iskanja entitet iz teksta je dvoumnost jezika, saj je, npr. na enak način zapisano ime lahko osebno ali krajevno (npr. Dakota).

Grobo gledano, obstajata dva načina odkrivanja entitet iz teksta [26, str. 118]:

- s pomočjo pravil oz. vzorcev in
- s pomočjo statističnih klasifikatorjev.

Drugi način je uporaba statističnih klasifikatorjev – gre za pristop, ki ne zahteva dolgega seznama pravil, kot prvi pristop. Statistični klasifikator tipično pogleda vsako besedo v stavku in se odloči, ali gre za začetek imena entitete, nadaljevanje imena entitete, ali pa beseda sploh ni del imena. S kombinacijo teh predvidevanj lahko naučimo klasifikator, da identificira zaporedje besed, ki sestavljajo ime entitete [26, str. 119]. Potreben pogoj tega pristopa je označevanje delov jezika oz. besed (angl. POS Tagging oz. Part of speech tagging). Vsako besedo v stavku označimo glede na njeno skladenjsko kategorijo – npr. ali gre za samostalnik, glagol ipd. Obstaja več

popularnih in prosto dostopnih načinov označevanja besed v stavku v odprtokodni skupnosti. Nekateri označevalniki poleg skladenjske kategorije besedi določijo tudi glagolski čas, pa tudi število (ednina oz. množina). Različni označevalniki so različno uporabni za označevanje določenih jezikov. Npr. tekstov, ki uporabljajo zlogovne jezike, kot je kitajščina, zagotovo ne moremo označevati na enak način kot npr. tekstov v angleščini ali slovenščini. Pri tem angleščina velja za jezik, kjer se da, zaradi njene relativne enostavnosti, dele jezika avtomatizirano zelo dobro označiti.

Najzmogljivejši sistemi odkrivanja entitet v tekstu uporabljajo predvsem postopke strojnega učenja. Kot pišejo [41], je eden izmed najučinkovitejših pristopov izdelava modela na probablističnih grafih, npr. s pogojnimi naključnimi polji (angl. Conditional Random Fields) [41] ali pa npr. s pomočjo skritih markovskih modelov (angl. Hidden Markov Models) [40]. V praksi so ti sistemi implementirani z nadzorovanim učenjem na besedilu, pri katerem so entitete že označene. V procesu učenja se za vsako besedo generirajo posamezne lastnosti, kot npr. oblikoskladenjske oznake, velike začetnice, prisotnost pomišljaja in podobno, v procesu označevanja pa sistem uporabi model, zgrajen na osnovi teh lastnosti [28, str. 60].

Ne glede na pristop je vedno, ko so v uporabi metode strojnega učenja, pogoj, da na osnovi označenega teksta naučimo klasifikator prepoznati imena. To zahteva, da pripravimo učno in testno množico, kar pa ni preprosto. Ima pa zato ta pristop seveda svoje prednosti:

- še vedno lahko algoritem obogatimo ali dodatno preverjamo s seznami,
- premik algoritma na drugi jezik zahteva minimalne spremembe v kodi,
- lažje je modelirati kontekst znotraj stavka in tudi dokumenta,
- klasifikator lahko na novo učimo na novih tekstih in mu dodajamo funkcionalnosti [26, str. 118].

Nekateri sistemi uporabljajo tudi eksplicitno predznanje, njihova slabost pa je ta, da ne zaznajo neznanih entitet, če jih nimajo v obstoječem leksi-

konu. Zato se jih pogosto kombinira s sistemom, osnovanim na strojnem učenju, tako da oba skupaj tvorita hibridni sistem [28, str. 60].

Odkrivanje entitet v slovenščini

Pregled tehnik odkrivanja entitet in relacij iz teksta razkrije, da bi, če bi želeli v aplikaciji uporabiti kaj drugega kot pravila oz. vzorce, potrebovali testne podatke. Kot navedeno, imamo opravka z zelo kratkimi skupki teksta in za uporabo metod strojnega učenja bi imeli veliko težav pri izdelavi učne in testne množice. Ročno zapisana pravila morda res veljajo za najbolj primitiven način, vendar jih nikakor ne gre zavreči, če imamo opravka s specifično, dobro razumljeno domeno [26, str. 116].

4.2.2 Odkrivanje relacij

Skupaj z odkrivanjem entitet se v naši aplikaciji ukvarjamo tudi z odkrivanjem relacij. Tehnike odkrivanja relacij in entitet se nemalokrat, kot tudi v našem primeru, prekrivajo. Tudi v tem koraku bi lahko postopali na enega od naslednjih načinov [27, str. 187]:

1. z ročno zapisanimi pravili (angl. Patterns),
2. metode nadzorovanega učenja,
3. metode delno nadzorovanega učenja ali nenadzorovanega učenja:
    - Bootstrapping,
    - metoda oddaljenega nadzora (angl. Distant supervision),
    - nenadzorovane metode.

Tako kot pri iskanju entitet tudi pri iskanju relacij v naši aplikaciji izkoriščamo to, da gre za podobne vzorce pri strukturi teksta. Tekst iz zapisa,

v katerem iščemo relacije, na primer izgleda takole (navedeni so trije primeri):
"/ Philip K. Dick ; [prevedla Jan Jona Javoršek in Urša Vogrinc Javoršek]", "/ Philip K. Dick ; [prevedla Andrej Dolenc in Aleš Holz]" in "/ Edo Rodošek ; [spremna beseda Drago Bajt]"

Pomembna je tudi vsebina polja opombe, saj iz nje izluščimo originalni naslov ali naslov, iz katerega je delo prevedeno. Npr., "Prevod dela: Do androids dream of electric sheep?"

Pri iskanju relacij se torej zanašamo na iskanje vzorcev z ročno zapisanimi pravili, relacijo predvidimo glede na pozicijo velike začetnice in zakodiramo v obliki regularnih izrazov. Shranimo celotni naslov, skupaj z delom o odgovornosti in seveda tudi izluščene relacije. Smislena nadgradnja našega prototipa bi bila, da bi naš sistem, potem ko bi obdelal in shranil večje število zapisov, lahko besede (strojno ali avtomatsko) označili in na podlagi tega razvili statistični klasifikator.

V naši nalogi se osredotočamo na slovenski tekst, čeprav je pri tujih pojavnih oblikah polje "Naslov" v jeziku te pojavne oblike. Te so v sistemu COBISS v manjšini. Npr., "/ Philip K. Dick ; retold by Andy Hopkins and Joc Potter ; [illustrated by Steven Player]" ali pa "/ Rossella Cattaneo ; prefazione di Franco Cavalli ; con un contributo di Jacques Bernier"

Pri primerih iskanja relacij in entitet v katerem izmed razširjenih svetovnih jezikov bi posredno lahko uporabljali katero izmed storitev na spletu, ki v ozadju uporabljajo tehnike strojnega učenja. Storitve na spletu, naštete tukaj [35], povečini nudijo storitve zgolj za bolj razširjene svetovne jezike, še posebej za angleščino.

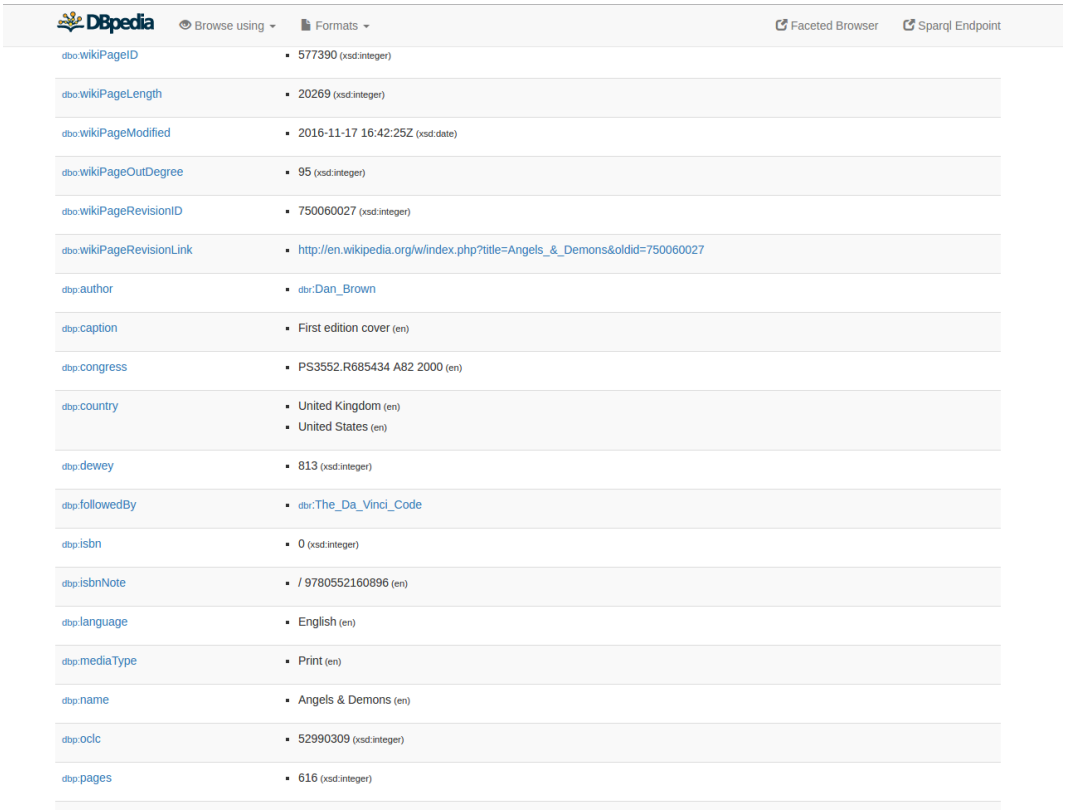
Poglavje 5

Pregled podatkovnih virov na spletu in povezovanje

5.1 Identifikacija virov

Na spletu obstaja kar nekaj virov, ki bi jih lahko uporabili kot podatkovni vir za preverjanje in bogatitev naših zapisov. Pri virih na spletu je ključno razlikovanje, ali gre za podatke, ki so bili narejeni s strani katalogizatorja ali pa so jih ustvarili uporabniki.

Vir informacij, ki združuje zapise, kreirane s strani katalogizatorjev, je storitev VIAF (Virtual International Authority File). Storitev VIAF je nastala pod okriljem OCLC-ja – kooperative ameriških knjižnic in širše [20]. VIAF združuje različne normativne zapise z različnimi nacionalnimi identifikatorji v enotno storitev z globalnim identifikatorjem. Namen tega je v prvi vrsti racionalizacija stroškov in izboljšanje uporabnosti normativnih zapisov, s povezovanjem različnih, širše uporabnih normativnih zapisov, v obliki primerni za diseminacijo na spletu [21]. Tudi v naši aplikaciji bomo omogočili katalogizatorju, da entiteto poveže z identifikatorjem VIAF ter nato obogati zapise iz COBISS-a z morebitnimi dodatnimi podatki iz zapisa VIAF. S pomočjo storitve VIAF bi želeli predvsem preveriti ali na novo pridobiti relacijo odgovornosti. Z istim namenom bomo skušali zapis, ki predstavlja



Slika 5.1: Izsek seznama razredov slovarja DBpedia

entiteto, povezati z zapisom na DBpediji.

DBpedia je t. i. *crowdsourcing* projekt, katerega namen je ponuditi podatke iz Wikipedije v strukturirani obliki na spletu [16]. Za nas je koristna predvsem, ker vsebuje tako zapise o npr. knjigah, filmih, glasbenih albumih in podatke o entitetah druge skupine. Slika 5.1 prikazuje formatiran zapis iz DBpedije v brskalniku, sicer pa bomo mi podatke pridobivali v strukturirani obliki, opisani s pomočjo DBpediji lastne ontologije – zaslonska slika 5.2. Natančen opis razredov in lastnosti razredov je moč najti na [17]

Iz socialnega omrežja LibraryThing [11] bomo s pomočjo programskega vmesnika in nadaljnega luščenja podatkov skušali najti informacije o povezavah delo – delo (vključivši relacijo, kateri filmi bazirajo na določeni knjigi). Prikaz entitet na omrežju LibraryThing pravzaprav delno sledi mo-

```
{ "head": { "link": [], "vars": ["creation", "tit"] },
  "results": { "distinct": false, "ordered": true, "bindings": [
    { "creation": { "type": "uri", "value": "http://dbpedia.org/resource/Do_Androids_Dream_of_Electric_Sheep%3F" }  , "tit": { "type": "uri", "value": "http://dbpedia.org/resource/Philip_K._Dick" }},
    { "creation": { "type": "uri", "value": "http://dbpedia.org/resource/The_Three_Stigmata_of_Palmer_Eldritch" }  , "tit": { "type": "uri", "value": "http://dbpedia.org/resource/Philip_K._Dick" }},
    { "creation": { "type": "uri", "value": "http://dbpedia.org/resource/Time_Out_of_Joint" }  , "tit": { "type": "uri", "value": "http://dbpedia.org/resource/Philip_K._Dick" }},
    { "creation": { "type": "uri", "value": "http://dbpedia.org/resource/Ubik" }  , "tit": { "type": "uri", "value": "http://dbpedia.org/resource/Philip_K._Dick" }},
    { "creation": { "type": "uri", "value": "http://dbpedia.org/resource/The_Collected_Stories_of_Philip_K._Dick" } , "tit": { "type": "uri", "value": "http://dbpedia.org/resource/Philip_K._Dick" }},
    { "creation": { "type": "uri", "value": "http://dbpedia.org/resource/The_Man_Who_Japed" }  , "tit": { "type": "uri", "value": "http://dbpedia.org/resource/Philip_K._Dick" }},
    { "creation": { "type": "uri", "value": "http://dbpedia.org/resource/We_Can_Remember_It_for_You_Wholesale_(collection)" }  , "tit": { "type": "uri", "value": "http://dbpedia.org/resource/Philip_K._Dick"
}},
    { "creation": { "type": "uri", "value": "http://dbpedia.org/resource/The_Golden_Man_(collection)" } , "tit": { "type": "uri", "value": "http://dbpedia.org/resource/Philip_K._Dick" }},
    { "creation": { "type": "uri", "value": "http://dbpedia.org/resource/The_Crack_in_Space" }  , "tit": { "type": "uri", "value": "http://dbpedia.org/resource/Philip_K._Dick" }},
    { "creation": { "type": "uri", "value": "http://dbpedia.org/resource/Gather_Yourselves_Together" }  , "tit": { "type": "uri", "value": "http://dbpedia.org/resource/Philip_K._Dick" }},
    { "creation": { "type": "uri", "value": "http://dbpedia.org/resource/Nicholas_and_the_Higgs" }  , "tit": { "type": "uri", "value": "http://dbpedia.org/resource/Philip_K._Dick" }},
```


Slika 5.2: Izsek strukturiranih rezultatov poizvedbe v formatu JSON – vsa dela Philipa K. Dicka

delu FZBZ. Navaja zapise o delih – delo ima svojo podstran in identifikator, znotraj dela pa so naštete vse edicije (tako rekoč pojavne oblike) tega dela, v različnih jezikih (prevodih) in formatih.

Podatki na našem solr strežniku so opisani s pomočjo ontologije RDA. Apache Solr [24] je primer iskalnega strežnika, v katerega vsak primerek dela, pojavne oblike ali entitete druge skupine shranimo kot dokument, v polja dokumenta pa shranimo vrednosti atributov primerka. Osredotočili se bomo na bogatenje zapisov o delih, in sicer iz VIAF-a in omrežja LibraryThing. V naslednjem poglavju naredimo pregled vidikov in tehnik povezovanja podatkov, ki so relevantni za nas.

5.2 Načini povezovanja podatkov

Povezovanje podatkov je predmet preučevanja različnih vej računalniške znanosti. V osnovi se ukvarja s problemom integracije podatkov. Področje integracija podatkov (angl. data integration) obsega povezovanje oz. kombiniranje podatkov iz različnih podatkovnih virov. Sistemi za integracijo podaktov so formalno definirani kot trojica

$$< G, S, M >, \tag{5.1}$$

kjer G predstavlja globalno shemo, S predstavlja heterogeno shemo, M pa pretvarja zahteve med obema shemama [36].

Ko govorimo o integraciji podatkov, imamo ponavadi v mislih integracijo zelo velikih podatkovnih virov. Kot taka je integracijo zato ključna [37]:

- v velikih korporacijah, ki imajo v lasti mnogo virov podatkov,
- v znanstvenih sferah, kjer se prav tako ustvari ogromno podatkov, ki bi jih bilo vredno povezati,
- za sodelovanje med vladnimi agencijami, od katerih ima vsaka v lasti veliko virov podatkov,
- pri zagotavljanju dobre kvalitete iskanja strukturiranih virov podatkov na spletu in semantičnem spletu.

Tehnike povezovanje podatkov (angl. data mapping) so tipično prvi korak pri integraciji podatkov. Cilj je identifikacija povezav med elementi dveh različnih podatkovnih modelov. Primeri povezovanja podatkov z namenom kasnejše združitve podatkovnih virov so:

- transformacija podatkov med našim in zunanjim podatkovnim virom,
- identifikacija relacij med podatki za potrebe iskanja izvora in spreminjanja podatkov skozi čas (angl. data lineage),
- odkrivanje skritih občutljivih podatkov, ki so zakodirani kot del nekega polja,
- konsolidacija več podatkovnih baz in združitev v eno bazo z eno shemo, brez odvečnih polj.

Naša aplikacija, ki je predstavljena v naslednjem poglavju, je zgolj prototip in njen namen ni razvoj tehnike avtomatskega ali delno avtomatskega povezovanja in ujemanja velikih virov podatkov. S področja povezovanja podatkov bomo zato posegali le na področje transformacije podatkov z namenom bogatenja (dopolnjevanja) in preverjanja ter za potrebe prikaza podatkov iz različnih virov na enem mestu. Pravila ujemanja med našimi podatki, ki so opisani s slovnico RDA, in zunanjimi viri in shemami bomo določili ročno. Slovnica RDA je semantično zelo izrazna in obsežna, podatki na spletu pa

so opisani z manj izraznimi shemami. Zato smo lahko v slovnici RDA vedno našli vsaj en element, ki bi ustrezal manj natančni shemi, v kateri so zapisani podatki pridobljeni iz spleta, npr. DBpedijina ontologija ali format MARCXML.

Pri bibliografskem univerzumu gre za dobro definirano in konsolidirano domeno, tudi zato so načini povezovanja dobro razviti in stara praksa v okolju knjižničnih metapodatkov. Zelo zgodaj se je namreč pojavila potreba po izmenjavi in posledično tudi transformaciji podatkov. Nastali so preverjeni "nasveti oz. recepti", kako polja ene sheme preslikati v polja druge sheme. Ekvivaletna polja v dveh različnih shemah prikažemo s tabelo ujemanj (angl. schema crosswalk). Izčrpen seznam takih ujemanj med različnimi shemami oz. formati za bibliografske podatke je na voljo [38]. Rast števila virov na spletu in digitalnih zbirk ter nastanek semantičnega spleta sta še poudarila potrebo, da bi bili metapodatki, sicer tipično zakodirani v katerem od derivatov formata MARC, v spletnem okolju dostopni v modernejših formatih in shemah [29].

Če je struktura polj ene sheme zelo različna od strukture druge sheme, taka tabela seveda ne zadošča. Postopek dejanske pretvorbe podatkov iz ene sheme v drugo imenujeno mapiranje metapodatkov (angl. metadata mapping).

V svetu velikih shramb podatkov, z zapletenimi shemami opisanimi z ontologijami, govorimo o ujemanju ontologij (angl. Ontology alignment oz. Ontology matching). Gre za postopek, ko elemente iz enega imenskega prostora oz. nabora elementov pretvarjamo v drug imenski prostor oz. nabor elementov. Preden skušamo najti ujemanje med takimi podatki ali pa zgolj ujemanje med različnimi ontologijami, ponavadi za predstavitev teh podatkov uporabimo graf. Kvaliteta semantičnega spleta v tem kontekstu je, da že imamo na voljo ogrodje za predstavitev podatkov v obliki trojic, ki jih lahko povezujemo v poljubno velike grafe. Pri tem postopku izkoriščamo semantično izraznost elementov ontologije, in ker gre za ogromne množice podatkov, so bolj kot ne uporabni zgolj avtomatski ali pa vsaj polavtomatski

pristopi.

V naši aplikaciji bi lahko ujemanje ontologij prakticirali zgolj s podatki iz DBpedije, ki so edini izmed naših virov podatkov predstavljeni v obliki trojic [30].

Poglavje 6

Funkcionalnosti in delovanje orodja

6.1 Prvi sklop – luščenje in FRBRizacija

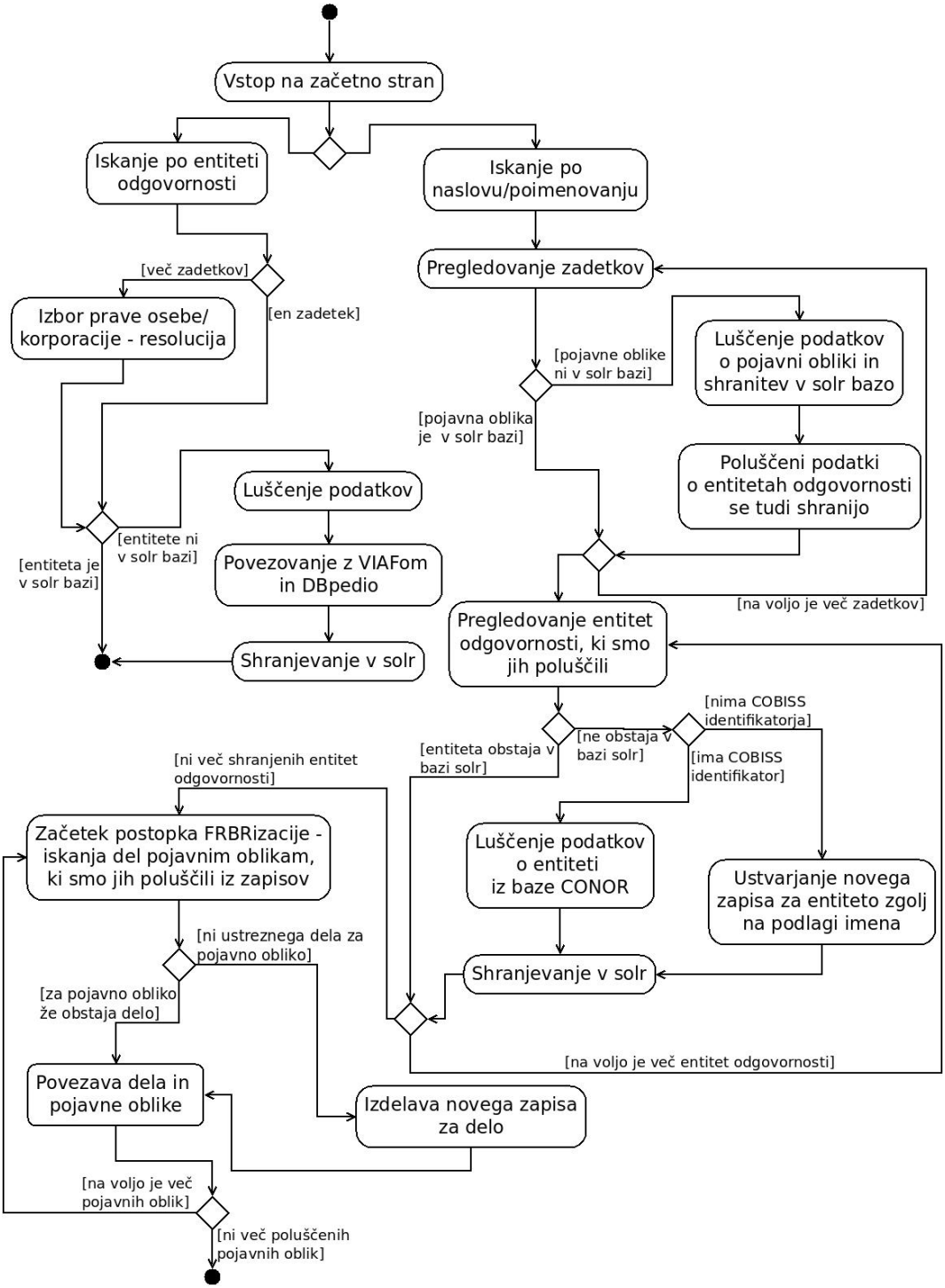
Osnovni funkcionalnosti našega orodja sta pridobivanje in obdelava podatkov (luščenje) in FRBRizacija. Diagram 6.1 predstavlja visokonivojski prikaz dveh načinov delovanja našega orodja. Prvič, katalogizator vpiše iskalno zahtevo v polje "Iskanje po naslovu/poimenovanju" (glej sliko 7.1) in iskalna zahteva se pošlje na sistem COBISS. Zapisi o pojavnih oblikah, ki ustrezajo tej zahtevi v sistemu COBISS, se poluščijo. Pri iskanju pojavnih oblik poskrbimo, da sistem COBISS ne išče samo po glavnem naslovu, ampak po večih poljih – ključnih besedah. Tako zajamemo tudi poimenovanja, ki so zajeta v drugih poljih, npr. druge vrste naslovov, opombe itd. Pri tem lahko poluščimo več zapisov naenkrat. V sklopu luščenja podatkov o pojavnih oblikah sistem polušči še podatke o osebah in korporacijah, ki so sodelovale pri nastanku pojavnih oblik, ter skuša identificirati in shraniti relacije teh entitet s pojavnimi oblikami. Sledi še korak FRBRizacije, ko skušamo poluščene pojavne oblike povezati z delom. V drugem primeru, na podlagi iskalne zahteve, sistem polušči le podatke o entiteti/entitetah druge skupine iz baze CONOR, ki ustrezajo iskalni zahtevi. Prav tako kot pri pojavnih oblikah se

proces luščenja ne izvede, če imamo entiteto že poluščeno v naši bazi.

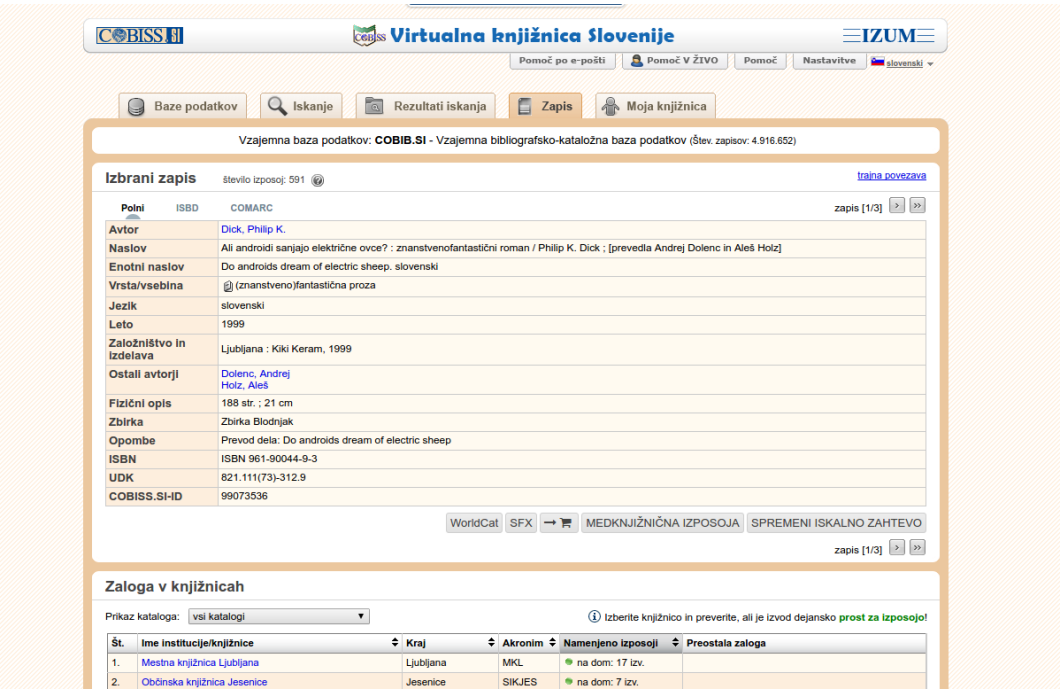
Kot rečeno, med luščenjem podatkov iz zapisa COBISS te podatke tudi obdelujemo in shranimo, in sicer s pomočjo slovnice RDA.

Luščenje entitet odgovornosti kot del luščenja pojavnih oblik vključuje tudi identifikacijo relacij, ki jih imajo entitete druge skupine s pojavno obliko. Podatke za ta namen najprej najdemo v poljih "Avtor" in "Ostali avtorji", kjer so imena entitet, ki so prispevale k nastanku pojavne oblike, navedena po pravilih za oblikovanje značnic (angl. heading, enotne oblike imena entitete). Oblika zapisa značnice na primeru enostavnih osebnih imen je taka, da je na prvem mestu priimek, nato sledi vejica, za vejico pa še ime. V tekstih, ki jih obdelujemo z namenom iskanja relacij (polje "Naslov") pa so ta imena zapisana v običajnem vrstnem redu. Primer prikazuje slika zaslona 6.2.

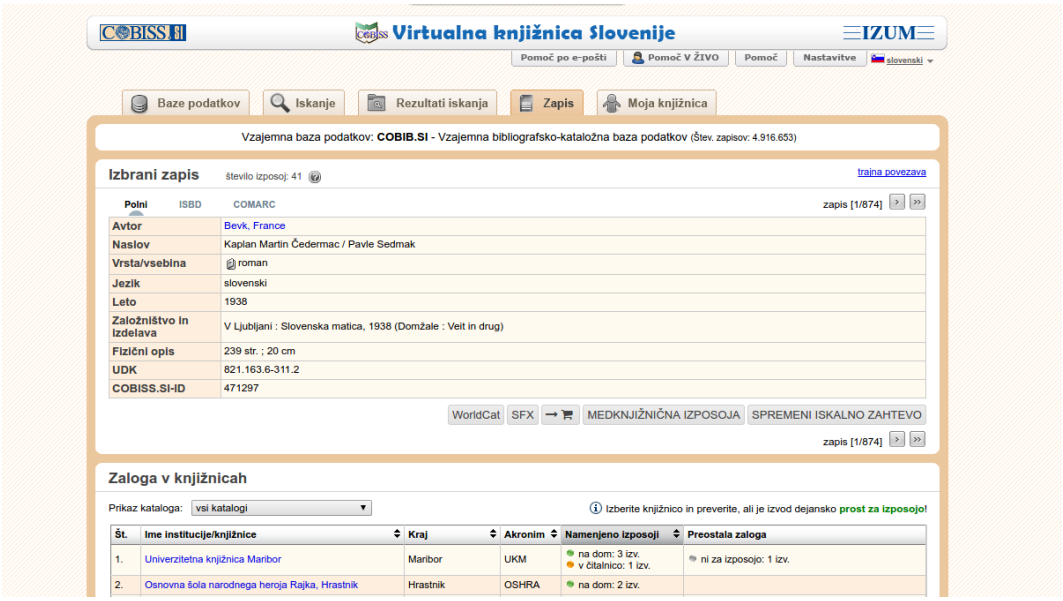
Ko poluščimo značnice entitet druge skupine iz polj "Avtor" in "Ostali avtorji" in ko želimo te značnice povezati z imeni oseb iz polja "Naslov" ali "Opombe", je postopek za večino primerov dokaj preprost. V polju "Naslov" iščemo vnaprej opredeljene nize, kot npr. "prev", s čimer iščemo besede tipa prevod, prevedla, prev. itd. Relaciji sledi ime osebe ali korporacije in s pomočjo ločevanja po velikih začetnicah ločeno shranimo elemente RDA in pripadajoča imena oseb. Nato skušamo med temi izluščenimi imeni oseb in korporacij ter med značnicami imen najti ujemanje. Idealno je, če ujemanje dobimo že, ko obrnemo vrstni red imena in priimka v značnici. Če ne, preverimo tudi psevdonime osebe ali korporacije (dobljene iz baze CONOR), v primeru, da bi bilo ime osebe v delu naslova navedeno s psevdonimom. Tak primer kaže zaslonska slika 6.3, kjer je bil na pojavni obliki naveden psevdonim Franceta Bevka – Pavle Sedmak. Tako vemo, da gre v resnici za dve različni poimenovanji iste osebe in ne za dva avtorja. Če zadetka še vedno ni, preverimo še ali gre za morebitno slovnično napako, ali malenkost drugače zapisano ime avtorja. V tem koraku si pomagamo s primerjanjem nizov s Levensteinovo razdaljo. Če sta si niza zelo podobna, predpostavimo, da gre za poimenovanji iste entitete. V zadnjem koraku z najdenimi relacijami povežemo pojavno obliko z ustreznimi entitetami druge skupine.



Slika 6.1: Diagram aktivnosti, ki prikazuje proces luščenja in FRBRizacije

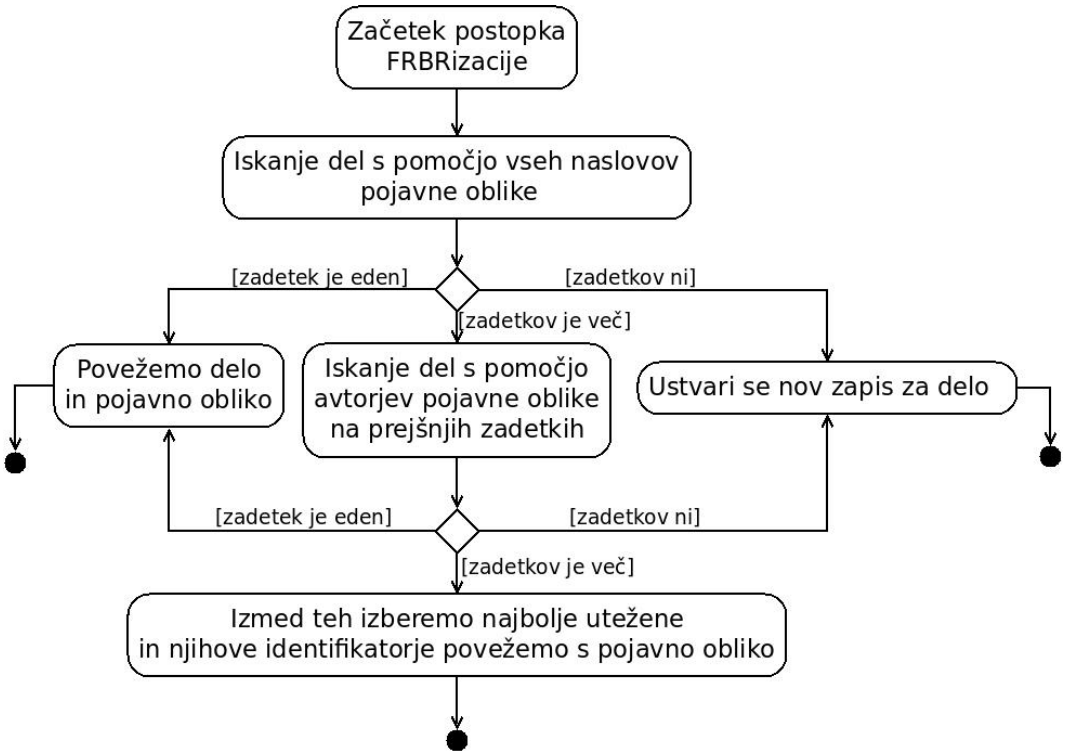


Slika 6.2: Primer izgleda značnic, ki se nahajajo v poljih “Avtor” in “Ostali avtorji”



Slika 6.3: Primer, ko je avtor v drugem delu polja “Naslov” naveden s psevdonimom, v polju “Avtor” pa z značnico

Pri FRBRizaciji skušamo na novo poluščeni pojavni obliki dodeliti delo. Vzamemo vse naslove pojavne oblike (glavni naslov iz sistema COBISS, morebitne alternativne naslove, enotni naslov ali naslov originalnega dela, če gre za prevod) in preverimo, ali obstaja delo, ki bi imelo tak ali podoben naslov oz. poimenovanje. Naslednji korak vključuje preverjanje avtorjev. Vzamemo značnico imena avtorja in primerjamo, ali ima kakšno izmed prej najdenih del enakega avtorja. Če tako ostanemo z enim zadetkom, id pojavne oblike povežemo z identifikatorjem dela in relacijo “has work manifested”. Če zadetkov ni, ustvarimo novo delo na osnovi podatkov o pojavni obliki. Če je zadetkov več, jih razvrstimo od najbolj do najmanj uteženega. Najbolj uteženi so tisti, pri katerih se je enotni naslov (ali enotni naslovi) ujemal z naslovom dela. Dopuščati moramo, da je v pojavni obliki zajetih več del, zato zelo permisivno pripišemo pojavni obliki kar vsa preostala dela (ponavadi sta največ dve). Je pa res, da se možnost, da bomo ostali z več deli, zmanjšuje s tem, kako pozoren je katalogizator s preverjanjem morebitnih podvojenih



Slika 6.4: Diagram postopka FRBRizacije

del. Če jih bo v bazi veliko, bodo tudi nekatere pojavne oblike imele več del samo zato, ker bodo ta dela dvojniki. Diagram procesa FRBRizacije je prikazan na sliki 6.4.

6.2 Drugi sklop – povezovanje

Drugi sklop funkcionalnosti je povezovanje in bogatenje podatkov z drugimi viri podatkov in poteka ločeno od glavne funkcionalnosti luščenja in FRBRizacije. Ker je že korak luščenja zelo potraten z vidika vzhodno/izhodnih operacij, bi poskus avtomatskega povezovanja in bogatenja, pri katerem gre prav tako za veliko število vhodno/izhodnih operacij, še dodatno upočasnil delovanje in uporabnost našega sistema. Poleg tega bi težko zagotovili dovolj visoko kakovost rezultatov avtomatskih opravil. Pozitiven vidik takega

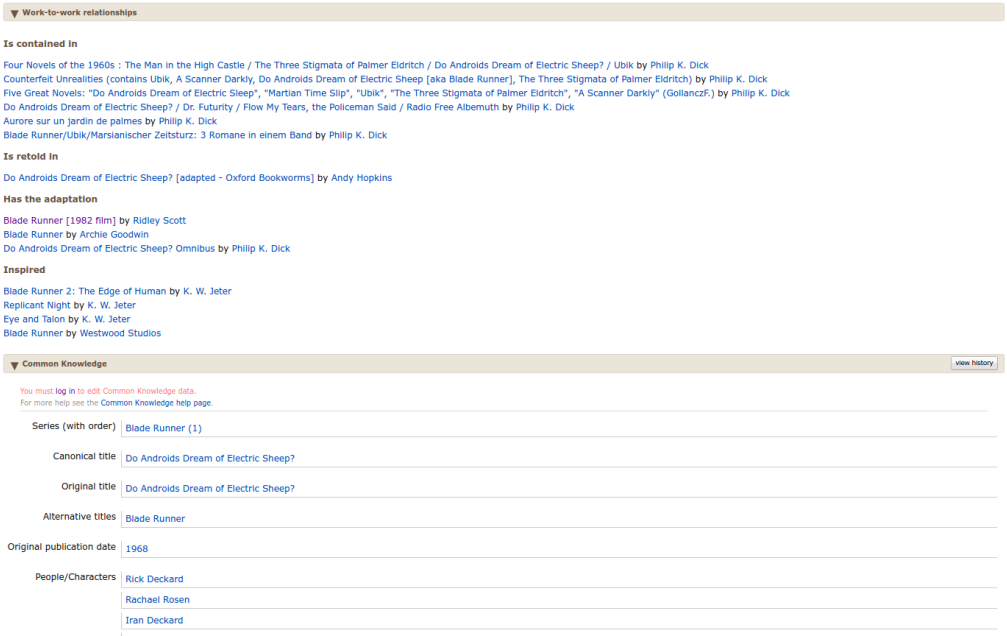
pristopa je, da katalogizator bolje razume delovanje sistema in od kod je določena informacija prišla, zato zna bolje utežiti njeno vrednost. Slabost pa je seveda čas, ki ga katalogizator porabi za izvedbo dodatnih korakov.

Povezovanje in bogatenje podatkov lahko izvedemo šele, ko poznamo identifikator zunanje entitete. Pred povezovanjem nam sistem omogoča, da skušamo kar iz naše aplikacije najti zapise o ustreznih entitetah na zunanji storitvi. Pri tem sistem pošlje zahtevo na zunanjo storitev z uporabo različnih vmesnikov. Programski vmesnik omrežja LibraryThing se je izkazal za najbolj uporabnega, precej manj uporabna pa sta programski vmesnik storitve VIAF in SPARQL vstopna točka (angl. endpoint) DBpedije. Tudi zato so načini povezovanja in pridobivanja podatkov pri naših treh zunanjih virih zelo različni. Tehnični vidik tega je opisan v poglavju 8 .

Naš prototip se osredotoča na pridobivanje podatkov o delih. Iz storitve VIAF s pomočjo programskega vmesnika pridobimo podatke o naslovih in avtorjih, nekoliko kompleksnejše pa je pridobivanje podatkov iz storitve LibraryThing. Poleg splošnih informacij o delu (različne vrste naslovov, originalni jezik) skuša naš sistem s pomočjo storitve LibraryThing najti tudi povezave med deli (relacije naše delo – drugo naše delo). Sistem iz profila dela na omrežju LibraryThing najprej polušči tip relacije, skupaj s LibraryThing identifikatorji pripadajočih del.

Slika 6.5 prikazuje profil dela “Do androids dream of electric sheep?” na storitvi LibraryThing, kjer so navedeni podatki o relacijah z drugimi deli. Pod njimi je območje, iz katerega luščimo originalni naslov in morebitne alternativne naslove.

Poluščeni relaciji sistem najde ustrezen element iz slovarja RDA, npr. relacija “Has the adaptation” iz LibraryThinga postane “rdau:P60260”. Sistem pa mora nato najti še ustrezno delo iz storitve LibraryThing v naši bazi del. Da pojasnimo; v prvem koraku sistem ponudi relacijo: *delo iz baze solr (naše delo) z LibrayThing identifikatorjem – drugo delo iz storitve LibraryThing*. Želeli pa bi relacijo: *naše delo – drugo naše delo iz baze solr*. Drugemu delu iz storitve LibraryThing moramo zato najti ustreznik v naši bazi. V primeru,



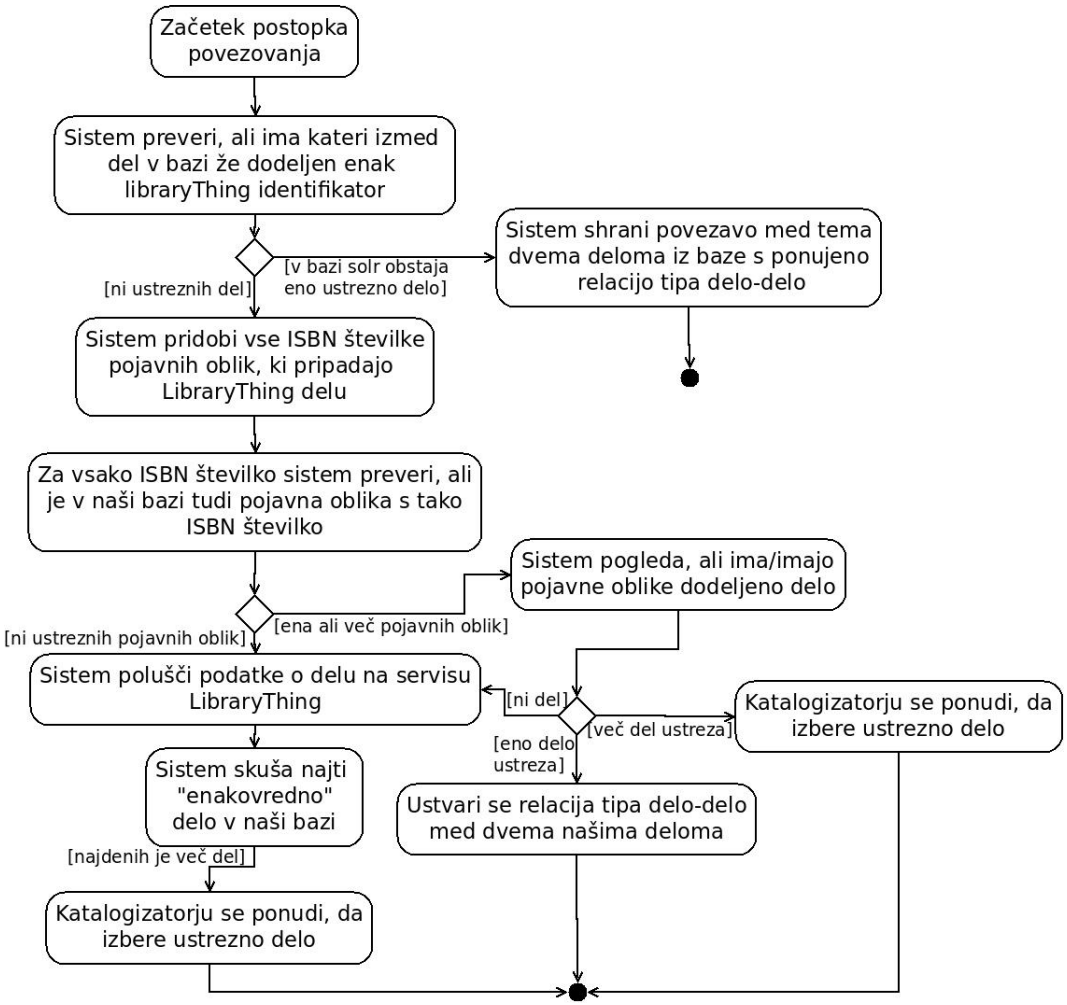
Slika 6.5: Del profila dela

da v sistemu ni kandidata, se relacija ne more dodati. Sistem kandidata išče v večih korakih (opisani postopek je prikazan na diagramu 6.6.):

- v prvem koraku skuša najti, ali v bazi solr že obstaja delo s takim LibraryThing identikatorjem,
- če ni kandidata, v drugem koraku sistem pridobi iz storitve LibraryThing ISBN številke pojavnih oblik, ki sodijo pod LibraryThing delo. Sistem skuša nato najti "naše" pojavne obike, ki bi nosile kakšno izmed vrnjenih številk ISBN. Če kaka taka "naša" pojavna oblika obstaja, pogledamo, ali ima dodeljeno delo. To delo je potem morebiten ustreznik dela iz storitve LibraryThing in posledično kandidat za subjekt relacije naše delo – drugo naše delo. Če je kandidatov več, katalogizator odloči, kateri kandidat je pravi, če pa je kandidat samo eden, se relacija ustvari avtomatsko,
- če še vedno ni kandidatov, poluščimo podatke o drugem delu iz storitve

LibraryThing, natančneje naslove in avtorje, ter skušamo s pomočjo iskanja najti morebitnega kadidata za delo v naši bazi.

Med osnovne funkcionalnosti orodja spada tudi urejanje podatkov o pojavnih oblikah, entitetah druge skupine in predvsem delih. Z vidika delovanja gre pri tem sklopu funkcionalnosti za dodajanje, popravljanje ali brisanje polj iz profila entitete ali brisanje, dodajanje in združevanje entitet.



Slika 6.6: Diagram aktivnosti, ki prikazuje poskus povezovanja

Poglavje 7

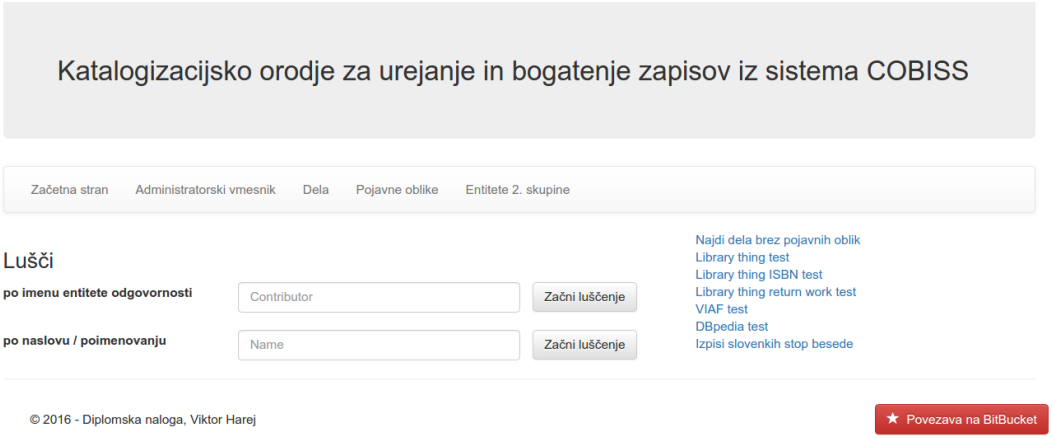
Uporaba orodja in uporabniški vmesnik

7.1 Scenariji uporabe

Orodje skuša katalogizatorju, na čimbolj intuitiven in preprost način, ponuditi možnost, da entiteto umesti v kontekst bibliografskega univerzuma, s pomočjo povezovanja. V osnovi gre za delno avtomatsko orodje, pri čemer katalogizator pregleduje in ureja zapise ter izbira in potrjuje odločitve, ki jih predlaga sistem.

Osnovni način uporabe orodja je tak, da katalogizator vstopi na osnovno stran, kjer ima dve možnosti (glej sliko 7.1). Pri prvi opciji "Iskanje po naslovu/poimenovanju" se iskalni pojmi pošljejo na COBISS, kjer s pomočjo luščenja podatkov pridobimo podatke o pojavnih oblikah in entitetah druge skupine, vezane na te pojavne oblike, ter iz pojavnih oblik posledično tudi podatke o delih. Scenarij dela katalogizatorja, ko ta z iskalno zahtevo sproži luščenje in FRBRizacijo zapisov pojavnih oblik iz sistema COBISS, prikazuje diagram 7.2.

Druga možnost je iskanje po imenu entitete odgovornosti, ki služi preverjanju, ali v sistemu COBISS (baza normativnih zapisov za osebe in korporacije se imenuje CONOR) obstaja zapis za določeno osebo/korporacijo. Cilj



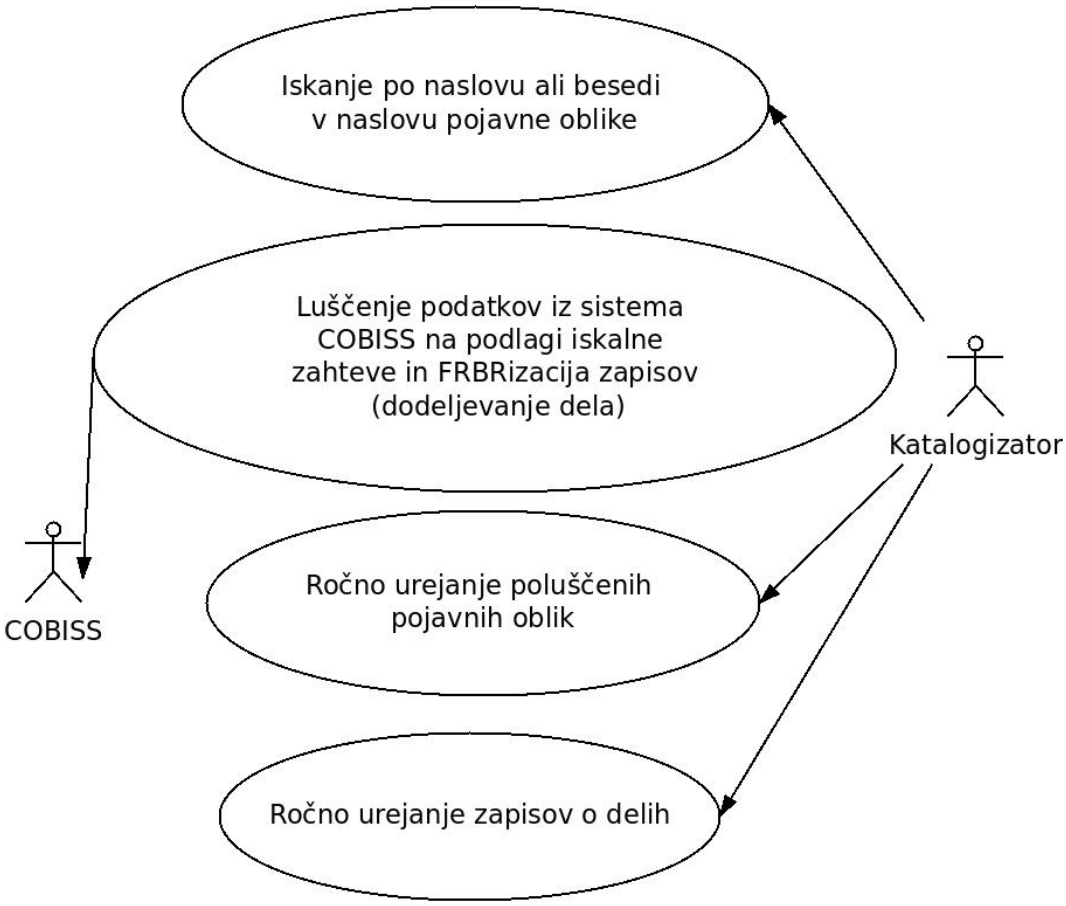
Slika 7.1: Vstopna stran našega orodja in obe glavni opciji

je, da tudi na našem lokalnem strežniku zgradimo bazo imen oseb in korporacij. Scenarij, ko katalogizator s pomočjo iskalne zahteve polušči zapise o entitetah druge skupine, prikazuje diagram 7.3. Scenarij je preprostejši, saj tu ni koraka FRBRizacije.

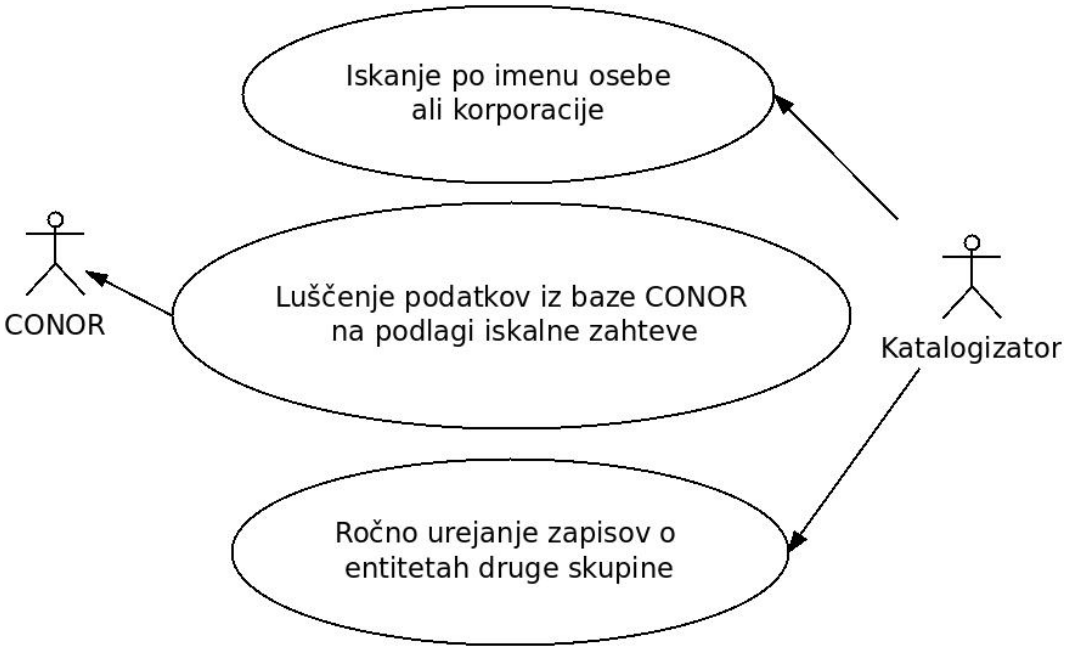
Če so poluščeni podatki za pojavne oblike in entitete druge skupine relativno visoke kvalitete, drugače velja za zapise o delih. Katalogizatorjev prispevek pri urejanju teh zapisov je precej velik, pri tem pa je ključnega pomena uporabnost orodja. Tipičen scenarij urejanja dela po luščenju pojavnih oblik in FRBRizaciji prikazuje diagram primerov uporabe 7.4

Po koncu luščenja sledi povzetek rezultatov. Prikazane so pojavne oblike iz baze, ki ustrezajo iskalnemu nizu. Te pojavne oblike so lahko že obstajale v bazi ali pa so bile na novo poluščene in FRBRizirane. Slika 7.5 predstavlja povzetek rezultatov in je posledica iskalnega niza “androidi sanjajo”. V sistemu COBISS obstaja 7 zapisov na to zahtevo in vse smo poluščili. Hkrati smo te pojavne oblike FRBRizirali in sistem je ustvaril 4 zapise za delo. Pravilno je združil tri pojavne oblike pod eno delo, dve pojavni obliki pod drugo delo, dve pojavni obliki pa sta dobili vsaka svoje delo.

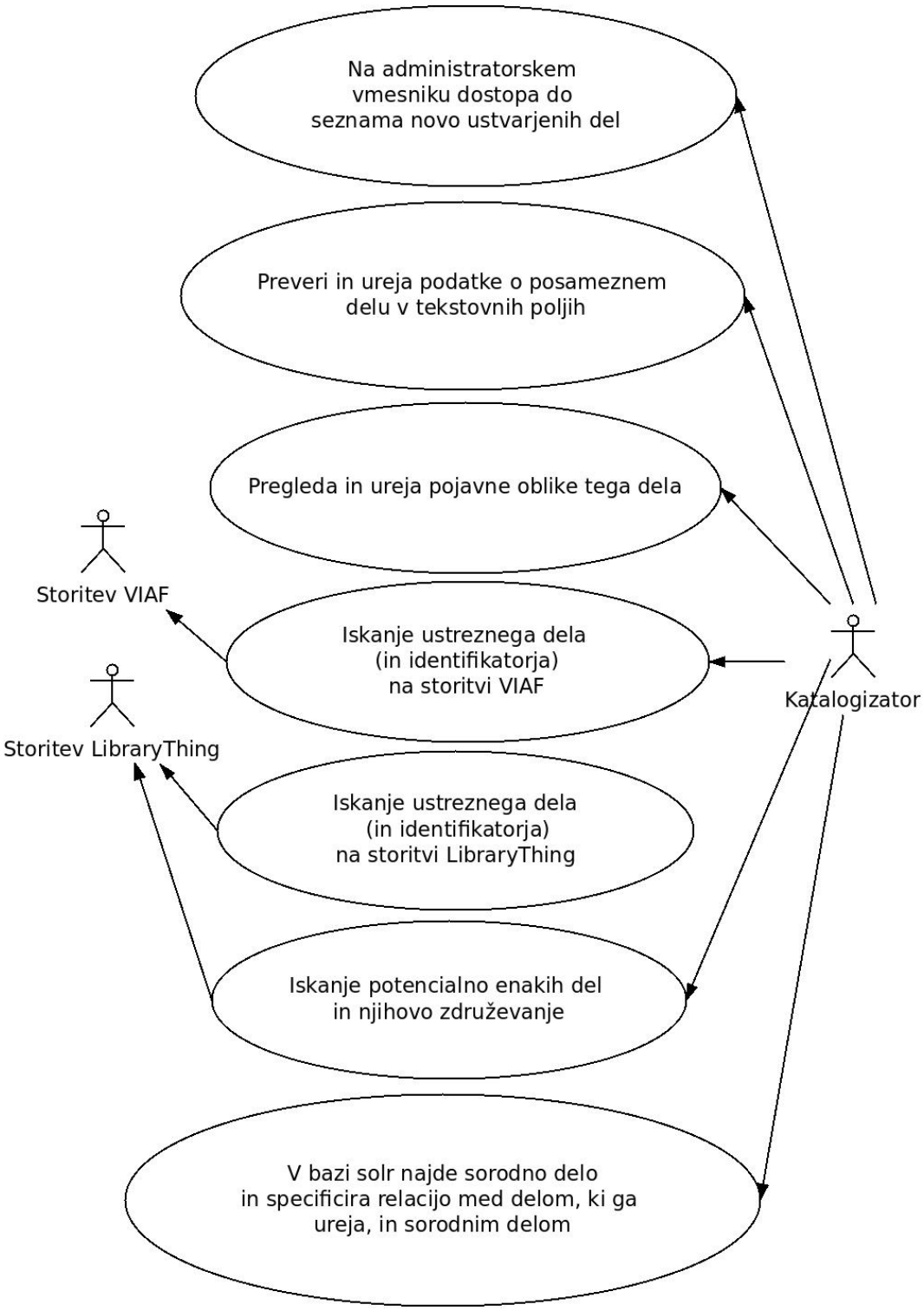
Katalogizator lahko klikne pojavno obliko ali delo kar na strani, ki prikazuje povzetek rezultatov, in tako dostopa do profila pojavne oblike ali dela.



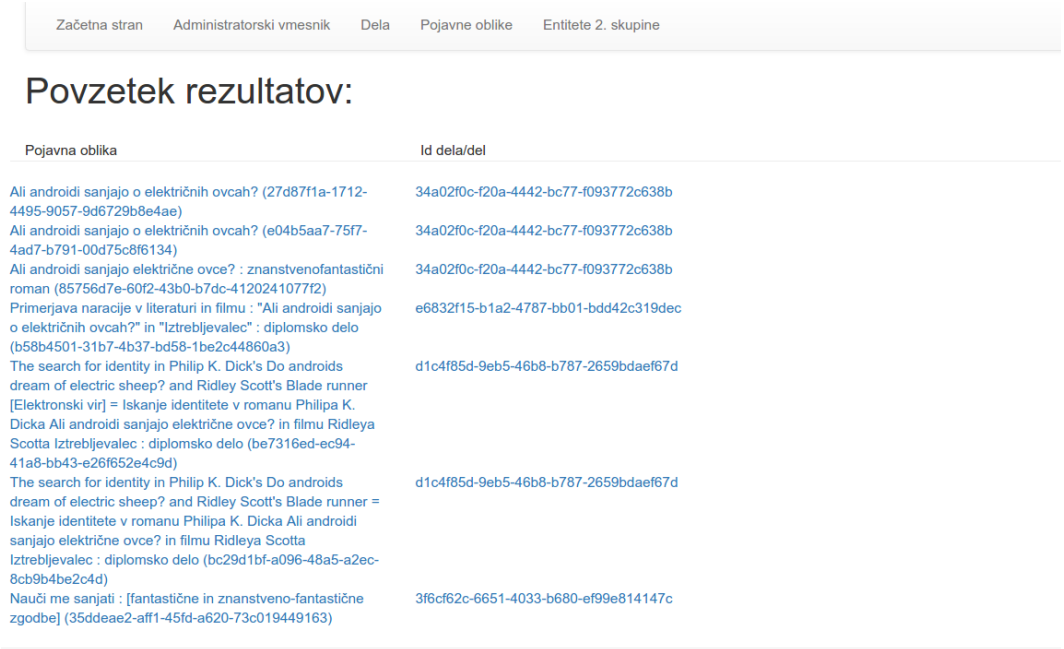
Slika 7.2: Primer uporabe, ki prikazuje luščenje podatkov in FRBRizacjio zapisov



Slika 7.3: Primer uporabe, ki prikazuje luščenje podatkov o entitetah druge skupine



Slika 7.4: Uporabniški diagram, ki prikazuje urejanje dela



Slika 7.5: Stran s povzetkom rezultatov.

Sicer pa je vstopna točka za urejanje nedavno poluščenih pojavnih oblik, del in entitet druge skupine, administratorska stran, ki jo prikazuje slika 7.6.

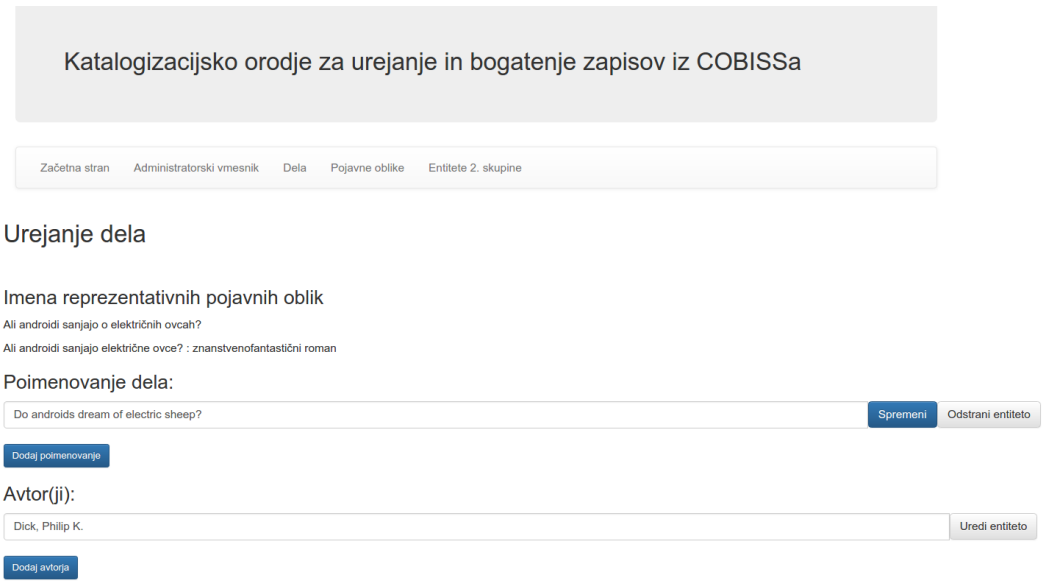
Drugi način dostopanja do zapisov vseh treh entitet je glavni meni, ki je vselej prisoten na vrhu aplikacije.

Na sliki 7.7 je prikazan prvi del profila dela. Podatki so bili predhodno obdelani in kot taki nadalje ponujeni katalogizatorju. Katalogizator lahko s pomočjo spustnega menuja zamenja relacijo (jo npr. dodatno specificira) ali popravi morebitno slovnično napako v imenu.

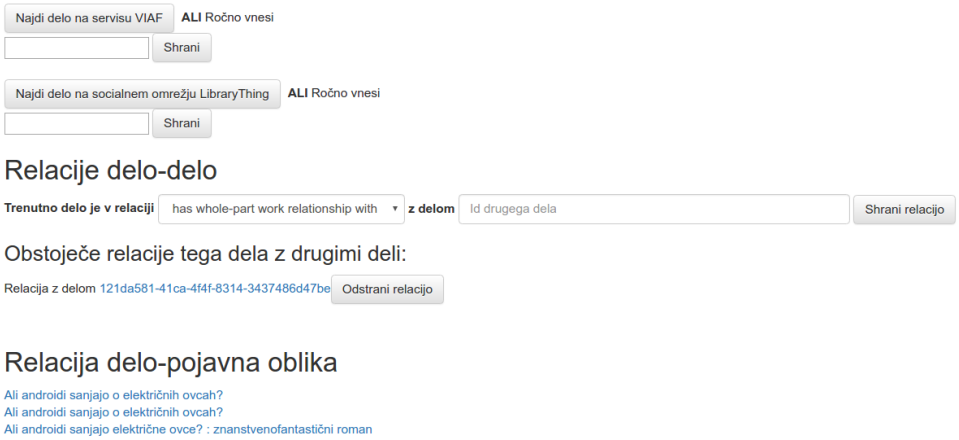
Na koncu profila dela je moč najti del profila, ki se dotika povezovanja naše pojavne oblike s storitvama VIAF in LibraryThing. Na sliki 7.8 je prikazano tudi območje, kjer urejamo in dodajamo relacije delo – delo.

Kot že omenjeno je ciljni uporabnik našega orodja katalogizator, ki bi želel izboljšati obstoječe bibliografske zapise iz sistema COBISS oz. podatke o bibliografskih entitetah, ki jih zapisi opisujejo. Pri delu smo operirali z identifikatorji VIAF in LibraryThing. Prvo, s čimer bi želeli izboljšati naše

Slika 7.6: Administratorska stran, kjer dostopamo do zapisov



Slika 7.7: Prvi del profilne strani enega od del

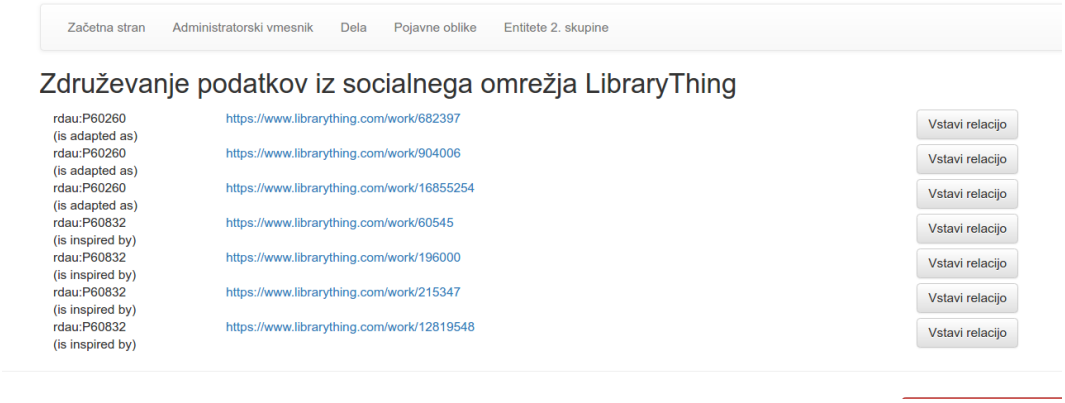


Slika 7.8: Drugi del profilne strani enega od del

zapise, so relacije, pri čemer je ključna predhodna identifikacija z enim od identifikatorjev. Po tem, ko npr. katalogizator vnese identifikator ustreznega dela na storitvi LibraryThing, orodje izvede mapiranje polj, in prikaže rezultate. Ker že imamo zabeleženo ime za delo "Do androids dream of electric sheep?", ga sistem ne ponudi, kljub temu da je bil pridobljen iz profila dela na storitvi LibraryThing. Kot kaže slika 7.9, pa ponudi kar nekaj relacij delo – delo.

V naslednjem koraku katalogizator išče ujemanje med delom, ki je na spletni strani prikazan z LibraryThing identifikatorjem URL. Po opisanem postopku v poglavju 6 sistem bodisi sam izbere delo ali delo iz seznama izbere katalogizator.

V primeru, da katalogizator opazi, da dva zapisa za delo opisujeta isto delo, lahko ti dve deli združi. Sistem za izbrano delo najprej sam predlaga potencialno enaka dela. Pri tem na zelo permisiven način išče ujemanje med besedami v naslovih del in naslovi pojavnih oblik teh del. Katalogizator pa lahko drugo delo, s katerim želi združiti prvo delo, poišče tudi ročno. Kot je vidno na sliki 7.10, mu je pri tem v pomoč iskalnik s funkcijo samodejnega dopolnjevanja iskalnega termina (angl. autocomplete).



Slika 7.9: Možnosti mapiranja, ki je katalogizator sprejme ali zavrne

Slika 7.10: Stran za združevanje del

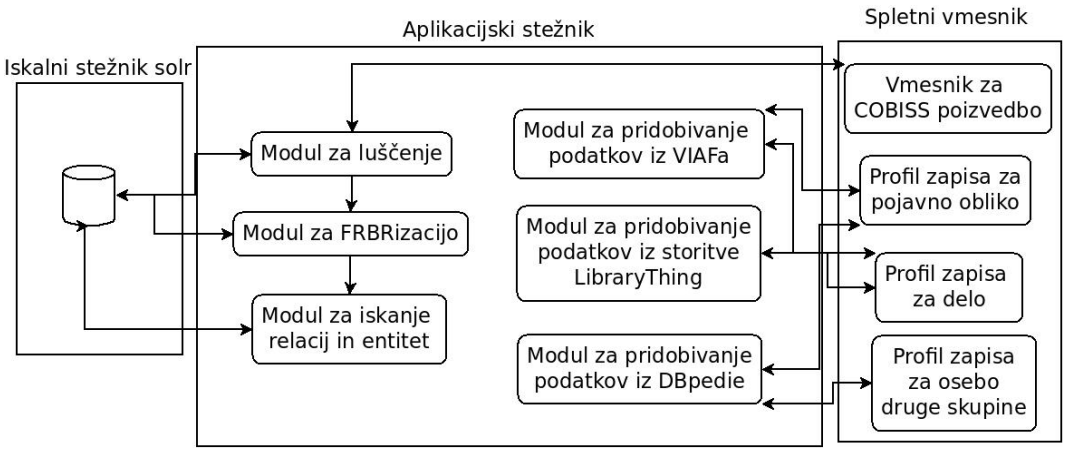
Poglavje 8

Implementacija z arhitekturnega in s tehničnega vidika

8.1 Uporabljene tehnologije

Naše orodje je spletna aplikacija, narejena v programskem jeziku java. Uporabili smo spletno ogrodje Spring boot [23] in strežniško programsko opremo Tomcat. Gre za aplikacijski strežnik za izvedljivo javansko spletno kodo, narejeno v skladu s specifikacijo Servlet [22].

Iskalni strežnik solr je prav tako produkt iniciative Apache. Kot navaja spletna stran, gre za zanesljiv in skalabilen iskalni strežnik, ki lahko poganja tudi velike distribuirane sisteme [24]. Solr je razvit na osnovi preverjene in zelo popularne javanske knjižnice za iskanje in manipulacijo teksta lucene [25]. Za potrebe naše diplomske naloge smo ga izbrali predvsem zaradi prilagodljivosti, prijaznega vmesnika, možnosti izvajanja poizvedb s pomočjo programskega vmesnika ter zaradi naprednih možnosti indeksiranja. Sicer pa je koda našega prototipa dostopna v Git repozitoriju Bitbucket, in sicer na naslovu [39].



Slika 8.1: Oris arhitekture našega prototipa

8.2 Opis arhitekture

Orodje smo poizkušali razviti na čimbolj modularen način, kar nam pride prav pri preglednosti kode in pri skalabilnosti pri morebitnem dodajanju novih funkcionalnosti. S pojmom modul, v kontektu naše diplomske naloge, razumemo java razred, v katerem je zajeta specifična funkcionalnost (in ne pravi java modul ali modul OSGI). Tudi pri opisu delovanja naše aplikacije na tehničnem nivoju bomo najprej opisali arhitekturo aplikacije na višjem nivoju, potem pa bomo nadaljevali z opisom posameznih modulov. Grobi oris arhitekture našega prototipa prikazuje diagram 8.1.

Pri pisanju in strukturiranju kode smo sledili objektno orientiranemu načinu razvoja. Entitete iz konceptualnega modela FZBZ, s katerimi operiramo v aplikacijo, smo zakodirali v javanske razrede:

- Work.java
- Manifestation.java
- Contributor.java

Pri tem smo sledili principom enkapsulacije: polja razredov imajo zgolj

privaten dostop, objekt spreminjamo s pomočjo metod, angl. getters and setters.

Tudi način operiranja s temi objekti (kreiranje, popravljanje in brisanje), taki imenovani CRUD del aplikacije, sledi objektni paradigmi. Pri shranjevanju v podatkovni stežnik solr polja objekta "ovijemo" v polja dokumenta – za ta namen smo pri vsakem razredu napisali dve ključni funkciji prepareForSolr in createFromSolr. Lahko bi rekli, da uporabljamo princip, ki je zelo podoben prevajanju objektov v strukture relacijske baze (angl. Object Relational Mapping) [32].

8.3 Moduli aplikacije

Aplikacijo smo strukturirali na module, naštete v nadaljevanju poglavja. Nekoliko podrobneje bomo opisali zgolj tri module za pridobivanje in bogatenje podatkov.

8.3.1 Modul za luščenje podatkov

Znotraj tega modula s pomočjo javanske knjižnice za luščenje podatkov pridobimo podatke o zapisih, ki ustrezajo uporabniški zahtevi. Za pomikanje po naslovih URL uporabljamo razred HttpClient iz javanske knjižnice apache commons. Podatke pa luščimo kar s pomočjo metod jezika java za odkrivanje pozicije podnizov iz vsebine na spletni strani. Zaradi velikega števila manipulacij teksta in bojazni, da bomo upočasnili delovanje sistema, se nismo odločili za uporabo preglednejšega jezika xPath in v javo vgrajene knjižnice za xPath.

8.3.2 Modul za iskanje entitet in relacij med entitetami druge skupine

V tem modulu s pomočjo definiranih vzorcev iščemo relacije in entitete v delu naslova, ki se tiče odgovornosti. V modul ContributorRelationship.java

pošljemo objekt pojavne oblike, modul pa skuša najprej iz dela naslova o odgovornosti izluščiti relacije in pripadajoče ime entitete. Sistem se pri tem zanaša na pozicijo velikih začetnic in predvideva, da bo relacija nastopila pred imenom. Nato shrani relacijo in pripadajoče ime. Relaciji zapisani z nizom, s pomočjo predefiniranih vzorcev dodeli ustrezen element iz slovarja RDA. Sistem skuša nato najti ujemanje med značnicami entitete druge skupine, ki smo jih shranili s pojavno obliko, in z imeni, ki smo jih izluščili iz naslova. Značnico obrnemo in primerjamo podobnost z imeni. Nato imena primerjamo še s psevdonimi oseb in korporacij, ki so identificirane z našimi značnicami. V tretjem koraku pa implementiramo še primerjavo imen in značnic z Levensteinovo razdaljo. Uporabljamo paket StringUtils iz paketa apache.commons. Če normirana podobnost (od 0 od 100) presega določeno vrednost, predpostavimo, da gre kljub vsemu, za poimenovanji iste entitete, in sicer zaradi morebitnih napak katalogizatorjev ali tiska ali preprosto malenkost različnih zapisov imen, kot je npr. Phillip Dick in Phillip K. Dick.

8.3.3 Modul za FRBRizacijo podatkov

Gre za modul, s katerim iz poluščene pojavne oblike ustvarimo delo ali pa pojavni obliki določimo delo, ki že obstaja. Pri postopku iskanja najprej iščemo z uporabo vseh naslovov pojavne oblike. V bazi del skušamo najti ujemanje s temi naslovi, pri tem pa iščemo ujemanje v poljih “representativeManifestationTitle” (naslovi pojavnih oblik, ki sodijo pod to delo) in “rdau:P60367” (relacija “ima naslov”, angl. “has title”). V to polje zapišemo poimenovanja iz polj “enotni naslov” pojavne oblike, iz morebitne opombe “Prevod dela:”. Sem spada tudi ime, ki smo ga delu dodelili s pomočjo storitev VIAF ali LibraryThing. Če dobimo kandidate, število zadetkov zmanjšujemo tako, da iščemo še ujemanje med avtorji pojavne oblike in avtorji dela. Če se je delo ujemalo po vrednosti iz polja “ima naslov”, to delo rangira višje kot drugo delo, pri katerem tega ujemanja ni bilo. Glede na to rangiranje sistem pojavni obliki naivno pripiše vsaj eno delo. Na katalogizatorju je, da preveri, ali je rezultat FRBRizacije smiseln. Če noben naslov pojavne oblike ne vrne

kandidata za delo, ustvarimo nov vnos za delo – ustvarimo novo instanco razreda Work in jo shranimo kot dokument v bazo solr, skupaj z id-jem pojavne oblike, ki pripada temu delu. Dokument ima morebiti zapolnjena polja "representativeManifestationTitle", polje "ima naslov" ("P60367") in polje z značnicami avtorjev.

8.3.4 Moduli za pridobivanje in bogatenje podatkov

Aplikacija pridobiva podatke iz različnih virov in na različne načine ter jih tudi na različne načine povezuje.

1. Modul za pridobivanje podatkov iz storitve VIAF

   Ta modul lahko išče po storitvi VIAF z uporabo programskega vmesnika in skuša najti zapise o delih (v formatu MARCXML), ki ustrezajo našemu delu. Rezultate nato obdela in ponudi možne kandidate za našemu delu enakovredno delo iz storitve VIAF. Iskalnik, ki ga uporablja API, ni preveč uporaben, in število rezultatov je prevečkrat neobvladljivo. Poleg tega pri API-ju delujejo le nekateri parametri iz dokumentacije. Iz seznama rezultatov moramo zato sami izluščiti zapise za delo, in sicer s pomočjo Googlove knjižnice gson za manipulacijo teksta v notaciji JSON. Ko imamo VIAF identifikator za naše delo, lahko poluščimo podatke o delu iz storitve VIAF. Tokrat imamo opravka z XML-jem, zato za identifikacijo elementov v DOM drevesu dokumenta znova uporabljamo knjižnico java.xpath. Poluščimo naslove in podatke o avtorju in tem vrednostim pripišemo ustrezno relacijo s pomočja elementa iz slovarja RDA. Poluščene rezultate vrnemo v javanski podatkovni strukturi tipa

   $$HashMap < String, ArrayList < String >> \tag{8.1}$$

   Pri tem je ključ strukture HashMap relacija iz slovarja RDA, vrednost pa seznam nizov, ki so subjekti te relacije (delo ima lahko več naslovov ali avtorjev pod isto oznako RDA). Te podatke nato prikažemo katalogizatorju.

2. Modul za pridobivanje podatkov iz DBpedije

   Na DBpediji najdemo zapise za t. i. "Work", vendar ne gre za delo v smislu modela FZBZ, temveč je, sodeč po atributih, nekje med delom, izrazno in pojavno obliko. Zelo poenostavljeno bi lahko rekli, da gre v večini primerov za opis najprej izdane pojavne oblike (v kronološkem smilsu). Zato v naši nalogi poenostavljeno pripišemo DBpedia WrittenWork identikator kar naši pojavni obliki. Ujemanje iščemo po besedah iz naslova, in sicer s pomočjo jezika SPARQL in s pomočjo javanske knjižnice Apache Jena. Taka iskanja tipično vrnejo preveč rezultatov, zato je zaželjeno, da DBpedia identifikator katalogizator vnese ročno. Način preiskovanja modela grafa s pomočjo knjižnice Apache Jena smo zgolj nastavili, v pridobivanje in povezovanje podatkov iz DBpedije pa se v okviru diplomske naloge nismo spuščali. Pozorni bi morali biti, da bi nekatere atribute pripisali našemu delu, nekatere pa določeni pojavni obliki tega dela.

3. Modul za pridobivanje podatkov iz socialnega omrežja LibraryThing

   Ta modul poskrbi za pridobivanje in strukturiranje podatkov o delu, dobljenih iz njegovega profila na storitvi LibraryThing. Za omrežno povezovanje in pridobivanje vsebine na oddaljenem strežniku vselej uporabljamo knjižnico apache.commons. Po postopku, razloženem v poglavju 6, pridobimo podatke o delu in jih na enak način, kot pri modulu za storitev VIAF, shranimo v podatkovno strukturo tipa hashMap (prikaz 8.1) ter jih nato prikažemo katalogizatorju, ki jih lahko sprejme ali zavrne.

Poglavje 9

Evalvacija

Evalvacije našega orodja smo se lotili na dva načina.

V diplomskem delu smo opisovali različne načine povezovanja in bogatenja podatkov in tudi vrednost takšnega povezovanja. Zato bomo v prvem delu navedli nekatere uporabniške iskalne scenarije in skušali oceniti, kako lahko s pripravo podatkov z našim orodjem odgovorimo na te potrebe.

V drugem delu pa ocenjujemo vrednost našega orodja s katalogizatorjevega vidika. S pomočjo analize SWOT bomo primerjali dva scenarija:

- postopni prehod na FRBRizirane in obogatene zapise z našim orodjem,
- hkratni FRBRizacija in bogatenje zapisov.

9.1 Evalvacija z vidika uporabniških scenarijev

Našteli bomo nekaj enostavnih uporabniških scenarijev in skušali ugotoviti, kako lahko naše pilotno orodje obogati podatke na ustrezen način:

1. Iskanje filma, ki je adaptacija določene knjige

   Gre za relacijo delo – delo. Naše orodje skuša, s pomočjo storitve LibrayThing, predlagati povezave med različnimi deli. Sicer pa lahko

povezave specificira tudi katalogizator ročno. Z našim orodjem lahko zajamemo različne relacije med deli, npr. da je neko delo smotrno uporabljati skupaj z nekimi drugim delom, da je prvo delo adaptacija drugega, da drugo delo nadaljuje prvo itd.

2. Iskanje novejšega izvoda knjige in iskanje pojavne oblike v drugem jeziku

   Naše orodje naslovi vse pojavne oblike (popravljena verzija ali drugi prevod) in združi pod eno delo. Tako lahko uporabnik lažje najde izvod, ki ga potrebuje. Večjezični uporabnik bo cenil tudi dejstvo, da se mu lahko na pregleden način prikaže, v katerih jezikih je delo na voljo. Včasih uporabnik niti ne ve, da ima na voljo tudi druge opcije in katere so te. Uporabniku lahko vrnemo zgolj (neurejen) seznam zadetkov ali pa mu s smiselnim prikazom in grupiranjem vseh pojavnih oblik enega dela omogočimo, da se lažje orientira v bibliografskem univerzumu.

3. Podatek, da je v določeni pojavni obliki zajetih več del

   Pojavna oblika lahko vsebuje dve deli ali več del – služi zgolj kot vsebnik za dve ali več samostojnih del. Npr. knjiga z večimi Cankarjevimi črticami. Ali pa npr. literalno delo skupaj z obsežno študijo. Prav tako obstajajo glasbeni nosilci, ki vsebujejo več samostojnih del, npr. CD z izbranimi Mozartovimi sonatami za klavir. Uporabnik se lahko tako odloči za verzijo, s katero bo dobil več. Naše orodje omogoča, da določeni pojavni obliki določimo dve ali več del. Uporabniku lahko posledično damo informacijo, da je določeno delo na voljo tudi v drugi pojavni obliki, kjer bo imel na voljo tudi spremno študijo ali druga dela istega avtorja.

4. Iskanje del po psevdonimu/personi

   Večkrat se zgodi, da avtorji uporabljajo psevdonime, pri čemer za različne kategorije del, ki jih ustvarijo, uporabijo drug psevdonim. Včasih je pomembno, s katerim psevdonimom se avtor podpisuje na katero

delo, saj lahko psevdonim označuje persono. Npr., ni malo avtorjev in avtoric, ki otroška besedila objavljajo pod drugačnim imenom kot dela za odrasle. Informacija, katero ime je avtor uporabil na pojavni obliki, lahko tako razkrije, kateri ciljni publiki je namenjena.

5. Iskanje vloge osebe

   Včasih nas zanima vloga neke osebe ali korporacije pri delu in pojavnih oblikah tega dela. Želeli bi najti filme, kjer je nek zvezdnik nastopal kot režiser in ne kot igralec. Z našim orodjem lahko katalogizator na semantičen način definira relacije s pomočjo slovnice RDA. Tako lahko uporabnik natančno ve, kakšno vlogo je entiteta imela pri nastanku dela ali pojavne oblike in pri katerih delih in pojavnih oblikah je še nastopala v tej vlogi. Ena od možnih vlog, ki jo lahko zajamemo z našim orodjem, je tudi, da je bila določena korporacija sponzor nekega dela. V primeru, ko beremo kako knjigo ali prispevek o zdravstvenih proizvodih, je ta informacija zagotovo koristen podatek za presojo kredibilnosti vira.

9.2 Evalvacija postopnega prehoda z našim orodjem

Naše orodje temelji na luščenju podatkov iz sistema COBISS in omogoča postopno obdelavo zapisov, vsakič ko bodisi katalogizator bodisi uporabnik izvedeta iskalno zahtevo.

Alternativa našemu pristopu je opcija takojšnjega prehoda, to je hkratna FRBRizacija in obogatitev vseh zapisov iz COBISS-a naenkrat. Kljub temu, da ta opcija za nas ne pride v poštev, je z vidika vrednotenja vseeno smiselno da primerjamo način takojšnjega prehoda na nov sistem in postopni prehod z uporabo našega orodja. Pri obeh pristopih bomo izvedli (ohlapni) analizi SWOT, pri čemer se bomo osredotočili na vidike delovnega procesa katalogizatorja, ekonomski vidik in posredno tudi na vidik računalniške arhitekture.

POSTOPNI (NAŠ) PREHOD

PREDNOSTI

- obogatitev in preverba podatkov iz COBISS-a,
- relativno dober vpogled v postopke aplikacije, ki so razdelani na module,
- možnost sodelovanja uporabnika,
- relativno poceni način FRBRizacije,
- možnost postopnega nakupa strojne opreme, glede na hitrost večanja zapisov.

SLABOSTI

- vzporedno vzdrževanje dveh sistemov,
- katalogizatorji morajo poznati dva načine obdelave,
- še vedno bi morali vzporedno izdelovati zapise v zastarelem formatu MARC,
- luščenje zapisov na spletu je počasno (vhodno-izhodne omrežne operacije), povsem drugače bi bilo, če bi imeli na voljo zapise lokalno,
- prehod je lahko zelo dolgotrajen.

PRILOŽNOSTI

- prehod na nov sistem je postopen, kar nam daje več možnosti, da aplikacijo sproti dopolnjujemo in izboljšujemo,
- katalogizatorji se lahko sproti, na pravih podatkih, priučijo novega načina katalogizacije, hkrati pa podrobno spoznajo in bolje razumejo njegove prednosti,

- testiramo lahko razne načine prikaza podatkov uporabniku in po potrebi lahko (v majhni meri) celo prilagajamo način delovanja našega sistema.

GROŽNJE NAŠEGA PRISTOPA

- prehod se lahko zavleče v nedogled, vsem prej omenjenim priložnostim je treba določiti časovne roke,
- sistem COBISS, ter predvsem koda strani HTML se lahko spremeni, pri čemer moramo kodo za luščenje podatkov napisati na novo.

TAKOJŠNJI PREHOD

PREDNOSTI

- v trenutku, ko sistem deluje, lahko prenehamo z izdelovanjem zapisov MARC in preidemo na obdelavo v novejšem, semantično močnejšem, formatu,
- če je faza testiranja zelo temeljita in v celoti vnaprej pripravljena, je prehod zelo gladek.

SLABOSTI

- strojno opremo je treba nabaviti predhodno, kar je lahko (pre)velik strošek; lahko se celo izkaže, da je ne bomo potrebovali,
- katalogizatorji morajo opravljati nadure ali pa je treba zaposliti novo delovno silo za čas prehoda,
- katalogizatorji morajo biti šolani vnaprej.

PRILOŽNOSTI

- uporabnikom lahko takoj ponudimo sistem, ki temelji na obogatenih podatkih.

GROŽNJE

- katalogizatorji se po implemetaciji novega sistema, zaradi povsem drugačnega načina dela, kljub izobraževanju lahko ne znajdejo.

Poglavje 10

Zaključek

V diplomski nalogi smo poskušali izdelati orodje, ki bo katalogizatorju služilo pri urejanju in bogatenju podatkov iz sistema COBISS. Orodje je uporabno, ko bi želeli ustvariti (in kasneje tudi izboljšati) zapise za dela pojavne oblike iz sistema COBISS in te zapise umestiti v kontekst bibliografskega univerzuma.

S postopkom FRBRizacije ustvarimo zapise o delih. Prisotnost koncepta dela že sama po sebi združuje pojavne oblike na osnovi iste (podobne) ideje, naš sistem pa skuša pomagati najti tudi relacije delo – delo. Na ta način lahko uporabniku ponudimo bolje povezane podatke, ki mu bodo olajšali orientacijo in mu pomagali pri zadostitvi iskalne poizvedbe. Skušali smo pokazati nekaj možnosti, kako je mogoče s povezovanjem na zunanje storitve pridobiti koristne in kvalitetne podatke. Ti lahko neposredno koristijo uporabniku pri njegovih iskalnih scenarijih. Nekaj takih scenarijev smo povzeli v poglavju 9. Seveda je možnosti izboljšave ogromno, konec koncev smo se v naši nalogi osredotočali zgolj na knjižno gradivo, ki pa je le eden od tipov gradiva v bibliografskem univerzumu.

Ker pa je potreba po kakovosti podatkov velika, mora imeti katalogizator na voljo možnost, da te podatke ureja. Pri našem prototipu smo se osredotočili na urejanje in bogatenje podatkov o delih; zapisi o pojavnih oblikah in entitetah druge skupine so že v osnovi zelo kvalitetni. V poglavju 7 prikazujemo načine dela katalogizatorja, ki tipično preuči ponujen podatek,

presodi, ali naj se vključi v sistem COBISS, in ga s pomočjo sistema priredi ali popravi. Prispevek diplomske naloge je torej prototip [39], ki podpira določen način dela katalogizatorja. Uporabnost tega prototipa pri scenariju, ko bi orodje uporabili za postopno FRBRizacijo in urejanje, smo skušali oceniti v drugem delu poglavja 9. Tudi tu je možnosti za izboljšavo še ogromno, predvsem je tu ključno, da si ustvarimo pregled nad viri na spletu in znamo oceniti, kako lahko zunanje podatke integriramo v naše orodje, tako da bomo lahko čim bolje naslovili uporabniške scenarije iskanja in prikaza informacij.

Literatura

[1] IFLA Study Group on the Functional Requirements for Bibliographic Records. *Functional Requirements for Bibliographic Records*. K.G. Saur, 1998.

[2] COBISS.si: Kooperativni online bibliografski sistem in servisi. [Online]. Dosegljivo:
http://cobiss.si/. [Dostopano 14. 1. 2017].

[3] O mreži COBISS-Net. [Online]. Dosegljivo:
http://www.cobiss.net/o_mrezi_COBISS-Net.htm. [Dostopano 7. 1. 2017].

[4] J. Pisanski, M. Žumer. "Funkcionalne zahteve za bibliografske zapise (FZBZ): analiza uporabnosti konceptualnega modela bibliografskega sveta", *Knjižnica*, št. 53, zv. 1-2, str. 61–76, 2009. Dosegljivo:
http://revija-knjiznica.zbds-zveza.si/Izvodi/K0912/Pisanski-Zumer.pdf. [Dostopano 21. 12. 2016].

[5] K. Spicher. "The development of the MARC format", *Cataloging and classification quarterly* , št. 21, zv. 3-4, str. 75–90, 1996.

[6] IFLA. *UNIMARC formats and related documentation*. Dosegljivo:
http://www.ifla.org/publications/unimarc-formats-and-related-documentation. [Dostopano 21. 1. 2017].

[7] T. Brešar. "Primerjava formatov MARC 21 – UNIMARC – COMARC", *Organizacija znanja* , št. 9, zv. 3, 2004. Dosegljivo:

http://home.izum.si/cobiss/oz/2004_3/html/clanek_04.html. [Dostopano 9. 1. 2017].

[8] E. T. O'Neill. "FRBR: Functional Requirements for Bibliographic Records: Application of the Entity-Relationship Model to Humphry Clinker ", *Library Resources & Technical Services* , št. 46, zv. 4, str. 61–76, 2002. Dosegljivo: http://www.oclc.org/content/dam/research/publications/library/2002/oneill_frbr22.pdf. [Dostopano 9. 1. 2017].

[9] MARCXML: MARC 21 XML Schema. [Online]. Dosegljivo: https://www.loc.gov/standards/marcxml. [Dostopano 7. 1. 2017].

[10] Library of Congress Network Development and MARC Standards Office. *MARC 21 Format for Bibliographic Data*. Dosegljivo: https://www.loc.gov/marc/bibliographic/.[Dostopano 18. 1. 2017].

[11] LibraryThing. [Online]. Dosegljivo: https://www.librarything.com/. [Dostopano 11. 1. 2017].

[12] FaBiO, the FRBR-aligned Bibliographic Ontology. [Online]. Dosegljivo: http://www.sparontologies.net/ontologies/fabio/source.html. [Dostopano 7. 1. 2017].

[13] FRBRoo: object-oriented definition and mapping from FRBREer, FRAD and FRSAD (version 2.2). [Online]. Dosegljivo: http://www.ifla.org/files/assets/cataloguing/frbr/frbroo_v2.2.pdf. [Dostopano 7. 1. 2017].

[14] Metadata: Mapping between metadata formats. [Online]. Dosegljivo: http://www.ukoln.ac.uk/metadata/interoperability. [Dostopano 7. 1. 2017].

[15] Expression of Core FRBR Concepts in RDF. [Online]. Dosegljivo: http://vocab.org/frbr/core. [Dostopano 7. 1. 2017].

[16] About DBpedia. [Online]. Dosegljivo: http://wiki.dbpedia.org/about. [Dostopano 7. 1. 2017].

[17] DBpedia Ontology Classes. [Online]. Dosegljivo: http://mappings.dbpedia.org/server/ontology/classes/. [Dostopano 7. 1. 2017].

[18] RDA Registry. [Online]. Dosegljivo: http://www.rdaregistry.info. [Dostopano 9. 1. 2017].

[19] Abaut RDA Tookit. [Online]. Dosegljivo: http://www.rdatoolkit.org/about. [Dostopano 9. 1. 2017].

[20] Abaut OCLC. [Online]. Dosegljivo: https://www.oclc.org/about.en.html. [Dostopano 9. 1. 2017].

[21] VIAF: The Virtual International Authority File. [Online]. Dosegljivo: https://viaf.org/. [Dostopano 9. 1. 2017].

[22] Apache Tomcat. [Online]. Dosegljivo: http://tomcat.apache.org/. [Dostopano 9. 1. 2017].

[23] Spring Boot. [Online]. Dosegljivo: https://projects.spring.io/spring-boot/. [Dostopano 9. 1. 2017].

[24] Apache Solr. [Online]. Dosegljivo: http://lucene.apache.org/solr/. [Dostopano 9. 1. 2017].

[25] Apache Lucene. [Online]. Dosegljivo: http://lucene.apache.org/core/. [Dostopano 9. 1. 2017].

[26] G. S. Ingersoll, T. S. Morton, A. L. Farris. *Taming text: How to Find, Organize, and Manipulate It*. Manning, 2012.

[27] R. M. Reese. *Natural Language Processing with Java*. Packt, 2015.

[28] T. Štajner, T. Erjavec, S. Krek. "Razpoznavanje imenskih entitet v slovenskem besedilu", *Slovenščina 2.0* , št. 1, zv. 2, str. 58–81, 2013. Dosegljivo: http://www.trojina.org/slovenscina2.0/arhiv/2013/2/Slo2.0_2013_2_04.pdf. [Dostopano 9. 1. 2017].

[29] L. M. Chan, M. L. Zeng. "Metadata Interoperability and Standardization – A Study of Methodology Part I: Achieving Interoperability at the Schema Level", *D-Lib Magazine* , št. 12, zv. 6, 2006. Dosegljivo: http://www.dlib.org/dlib/june06/chan/06chan.html. [Dostopano 9. 1. 2017].

[30] The DBpedia Data Set. [Online]. Dosegljivo: http://wiki.dbpedia.org/services-resources/dbpedia-data-set-2014. [Dostopano 7. 1. 2017].

[31] BIBFRAME: Bibliographic Framework Initiative. [Online]. Dosegljivo: https://www.loc.gov/bibframe/. [Dostopano 7. 1. 2017].

[32] Mapping Objects to Relational Databases: O/R Mapping In Detail . [Online]. Dosegljivo: http://www.agiledata.org/essays/mappingObjects.html. [Dostopano 7. 1. 2017].

[33] Schema.org. [Online]. Dosegljivo: http://schema.org/. [Dostopano 7. 1. 2017].

[34] Linked Data. [Online]. Dosegljivo: https://www.w3.org/standards/semanticweb/data. [Dostopano 7. 1. 2017].

[35] Quora: What is the best entity extraction API + service? [Online]. Dosegljivo: https://www.quora.com/What-is-the-best-entity-extraction-API-+-service. [Dostopano 9. 1. 2017].

[36] M. Lanzerini.“ Data integration: A theoretical perspective”. [Online]. Dosegljivo: http://www.dis.uniroma1.it/ lenzerin/homepagine/talks/ TutorialPODS02.pdf. [Dostopano 19. 1. 2017].

[37] A. Halevy, A. Rajaraman, J. Ordille. “Data Integration: The Teenage Years”, v zborniku *Proceedings of the 32nd international conference on Very large data bases*, Seul, Korea, sept. 2006, str. 9-16. Dosegljivo: https://homes.cs.washington.edu/ alon/files/halevyVldb06.pdf. [Dostopano 19. 1. 2017].

[38] M. Day. “Metadata: Mapping between metadata formats”. [Online]. Dosegljivo: http://www.ukoln.ac.uk/metadata/interoperability/. [Dostopano 19. 1. 2017].

[39] Katalogizacijsko orodje za urejanje zapisov iz sistema COBISS. [Online]. Dosegljivo: https://bitbucket.org/dlavbic/cobiss-enrichment/wiki/Home. [Dostopano 19. 1. 2017].

[40] S. Morwal, N. Jahan, D. Chopra. “Named Entity Recognition using Hidden Markov Model (HMM) ”, *International Journal on Natural Language Computing (IJNLC)* , št. 4, zv. 1, str. 15–23, 2012.

[41] C. Sutton, A. McCallum. “An Introduction to Conditional Random Fields ”, *Foundations and Trends® in Machine Learning*, št. 4, zv. 4, str. 267–373, 2012.