

❖ Poizvedovanje po povezanih podatkih s porazdelitvijo dela med strežnikom in odjemalcem

Jan Robas, Dejan Lavbič

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko, Večna pot 113, 1000 Ljubljana
JanRobas@gmail.com ; Dejan.Lavbic@fri.uni-lj.si

Izveček

Na spletu je veliko podatkov, ki so objavljeni na različnih mestih v različnih, nezdružljivih oblikah. Zaradi tega nad njimi ne moremo izvajati poizvedb. Iskanje odgovora na specifično vprašanje po navadi terja iskanje po ključnih besedah, branje in tolmačenje vsebine. To lahko rešimo tako, da uporabimo povezane podatke. Težava pri tem je dostopnost in učinkovitost poizvedovanja. V članku predstavljamo obstoječo rešitev, ki omogoča poizvedovanje po povezanih podatkih, tako da je delo porazdeljeno med strežnikom in odjemalcem. Poleg tega predstavljamo obstoječo razširitev, ki poveča učinkovitost, in našo lastno razširitev, s katero poskušamo še boljšati učinkovitost. Našo razširitev primerjamo z že obstoječo razširitvijo in z osnovno različico omenjene rešitve.

Ključne besede: spletne tehnologije, povezani podatki, semantični splet, SPARQL, procesiranje na odjemalcu.

Abstract

Querying Linked Data by distributing work between the server and the client

On the web, there is plenty of data presented in various incompatible forms. Consequently, querying such data can be difficult. To find an answer to a specific question, we have to resort to searching by keywords and interpreting the results. This problem can be solved via the use of Linked Data. The problem with querying Linked Data is availability and performance. In this article, we present an existing solution that enables Linked Data querying by distributing the work between the server and the client. In addition, we present an existing extension that increases efficiency. Finally, we present our own extension, an attempt to improve further the efficiency. We compare our extension to the existing one and the basic version of the subject solution.

Keywords: web technology, Linked Data, semantic web, SPARQL, client-side processing.

1 UVOD

Uporabniki se vedno bolj zanašamo na pridobivanje podatkov prek svetovnega spleta. Na spletu se nahaja velika količina podatkov, ki je predstavljena v različnih, nezdružljivih oblikah. Enakovredni podatki so pogosto različno predstavljeni na več mestih, kar pripelje do nekonsistence. Tudi spletni vmesniki, ki ponujajo podatke, so različni. Če želimo implementirati aplikacijo, ki uporablja te spletne vmesnike, imamo lahko precej dela z usklajevanjem naših podatkov s podatki, ki jih dobimo preko njih. Kot omenjata v članku avtorjev Heath in Bizer (2011), moramo v svoji aplikaciji za vsak spletni vmesnik posebej napisati kodo za odjemalca, kar zahteva znaten napor. Podatki, ki jih ponujajo spletni vmesniki, niso na voljo iskalnikom oziroma splošnim spletnim agentom. Za začetek reševanja teh težav potrebujemo način objave podatkov, ki je prilagojen za internet. Prav tako je pomembno, da

so podatki predstavljeni v strojno berljivi obliki, saj jih le tako lahko računalniško obdelujemo in po njih poizvedujemo. Eden od ustreznih načinov so povezani podatki, ki temeljijo na standardnih spletnih tehnologijah. V splošnem gre za tipizirane povezave med stvarmi iz različnih virov, pri čemer so tako stvari kot tipi povezav predstavljeni z enotnimi označevalniki virov (angl. URI, uniform resource identifier).

Po povezanih podatkih lahko poizvedujemo s pomočjo semantičnega poizvedovalnega jezika SPARQL. Pri tem se pojavijo težave, povezane z dostopnostjo in učinkovitostjo poizvedovanja. V članku predstavljamo obstoječi način objave povezanih podatkov, ki nam omogoča poizvedovanje s porazdelitvijo dela med strežnikom in odjemalcem. To rešitev smo nadgradili s svojo razširitvijo in jo primerjali z že obstoječo razširitvijo AMF.

V drugem razdelku bomo predstavili idejo povezanih podatkov. V tretjem razdelku bomo predstavili delce povezanih podatkov in delce vzorcev trojčkov, ki omogočajo porazdelitev dela pri poizvedovanju. Nato bomo predstavili obstoječo razširitev AMF in našo razširitev. V četrtem razdelku bomo našo razširitev evaluirali in jo primerjali z osnovno različico ter z razširitvijo AMF, sledi sklep.

2 POVEZANI PODATKI

Pojem povezani podatki se nanaša na nabor dobrih praks objavljanja in povezovanja strukturiranih podatkov na spletu. V splošnem gre pri povezanih podatkih za tipizirane povezave med podatki iz različnih virov. Ti viri so lahko bodisi znotraj iste organizacije bodisi porazdeljeni med več različnih organizacij. Povezani podatki so objavljeni na spletu tako, da so strojno berljivi in jih je mogoče naknadno povezovati z drugimi viri. Gre za podobno povezovanje podatkov, kot so na spletu povezani dokumenti HTML. Medtem ko so dokumenti HTML povezani z netipiziranimi povezavami, povezani podatki temeljijo na formatu RDF, pri katerem so povezave med objekti tipizirane. Rezultat je splet podatkov oziroma bolj točno splet objektov in pojmov, ki so predstavljeni s podatki na spletu (Bizer, Heath in Berners-Lee, 2009).

Glavna načela povezanih podatkov se skladajo z načeli in standardi svetovnega spleta in so (Tim Berners-Lee, 2006):

1. Uporabi URI-je kot imena za stvari (tako za objekte kot pojme).
2. Uporabi HTTP URI-je tako, da lahko uporabniki dostopajo do teh imen (npr. prek spletnega brskalnika).
3. Ko uporabnik dostopa do URI-ja, ponudi koristne informacije z uporabo standardov (RDF, SPARQL).
4. Vključi povezave do drugih URI-jev, da lahko odkrijejo več stvari.

Povezani podatki temeljijo na temeljnih spletnih tehnologijah: URI in HTTP. Medtem ko z URL-ji naslavljamo posamezne dokumente in druge vire na spletu, URI-ji omogočajo bolj splošno naslavljanje poljubnih entitet na svetu. Če se URI-ji začnejo s `http://`, potem jih lahko kot URL-je zahtevamo prek protokola HTTP in o njih pridobimo dodatne informacije. Čeprav URI-ji lahko predstavljajo pojme ali fizične stvari, ki jih ne moremo prenesti prek sple-

ta, lahko o njih pridobimo opise, dodatne informacije in povezane podatke. Dve omenjeni tehnologiji sta dopolnjeni z RDF, ki definira podatkovni model v obliki grafa, v katerem povezujemo entitete s tipiziranimi povezavami. Podatki so zapisani v obliki trojčkov oblike subjekt, predikat, objekt. Tako subjekt kot objekt sta URI-ja, ki predstavljata vir oziroma v primeru objekta opcijski niz znakov (literal, ki lahko predstavlja bodisi niz znakov bodisi število). Predikat je tudi predstavljen z URI in pove, v kakšnem razmerju sta subjekt in objekt. Če imamo npr. predikat `http://xmlns.com/foaf/0.1/knows` in je prva oseba (subjekt) A s tem predikatom povezana z drugo osebo (objekt) B, si to razlagamo kot: oseba A pozna osebo B. V modelu RDF je to trojček (A `http://xmlns.com/foaf/0.1/knows` B), pri čemer sta A in B URI-ja, pri čemer vsak predstavlja svojo osebo. Dva vira (v našem primeru osebi A in B) sta lahko izvzeta iz dveh različnih podatkovnih virov na spletu (npr. iz dveh družbenih omrežij). Ti podatki so med sabo združljivi. Trojčke RDF si zato lahko predstavljamo na podoben način kot spletne povezave med stranmi HTML.

2.1 Osnovni pojmi

Vsak trojček vsebuje subjekt, predikat in objekt. Vsak od teh je predstavljen z URI-jem, pri čemer je objekt lahko, opcijsko, niz znakov.

Vzorec trojčka je podoben kot trojček, vendar za razliko od dejanskega trojčka lahko namesto URI-jev in nizov znakov vsebuje spremenljivke.

Rezultati vzorca trojčka so trojčki v bazi, ki jih lahko dobimo s preslikavo spremenljivk danega vzorca trojčka.

Vsebnostna poizvedba je poizvedba, ki preverja, ali je neki trojček vsebovan v bazi.

Osnovni vzorec grafa BGP je sestavljen iz več vzorcev trojčka.

Poizvedba SPARQL je sestavljena iz enega ali več BGP-jev, ki so povezani z različnimi operatorji. Zaradi poenostavitve se v tem delu osredinjamo na poizvedbe, ki vsebujejo en BGP. Rezultati poizvedbe SPARQL so preslikave spremenljivk, ki vodijo do rešitve.

3 DELCI POVEZANIH PODATKOV

Članek avtorjem C. B. Aranda, A. Hogan, J. Umbrich in P.-Y. Vandenbussche (2013) govori o problemu dostopnosti oddaljenih točk SPARQL, saj je samo okoli

tretjina teh dostopna 99 odstotkov časa. Ob koncu leta 2013 je bila povprečna dostopna točka SPARQL nedosegljiva več kot dan in pol vsak mesec. Poleg tega obstajajo velike razlike v učinkovitosti poizvedovanja (3–4 velikostnih redov). To pomeni, da ne moremo razviti zanesljive aplikacije, ki bi se zanašala na poizvedovanje prek tovrstnih dostopnih točk. Ena rešitev tega problema je, da uporabnikom ponudimo statične datoteke s podatki RDF. Težava pri ponujanju podatkov RDF v statičnih datotekah je, da so podatkovne zbirke lahko zelo velike, uporabnik pa morda potrebuje samo majhen del podatkov. Poleg tega mora imeti uporabnik nameščeno ustrezno programsko opremo. Omenjena načina dostopa do podatkov se da predstaviti z vidika delcev povezanih podatkov kot dve skrajnosti (glej sliko 1).

Osnovna ideja delcev povezanih podatkov je, da so na dostopni točki objavljeni delci celotnega nabora povezanih podatkov (v obliki trojčkov), ki so razdeljeni po straneh. Pri tem lahko trojčke filtriramo z izbirnikom, ki ga lahko poljubno definiramo (R. Verborgh, O. Hartig, B. De Meester, G. Haesendonck, L. De Vocht, M. Vander Sande, R. Cyganiak, P. Colpaert, E. Mannens, R. Van de Walle, 2014). V skladu z idejo povezanih podatkov so pri delcih povezanih podatkov prisotni metapodatki z dodatnimi informacijami o podatkih ter, kot jih imenuje omenjeni članek, kontrole, ki vsebujejo povezave do drugih strani rezultatov in do drugih delcev povezanih podatkov. Tako datoteko s podatki kot dostopno točko SPARQL lahko predstavimo z vidika delcev povezanih podatkov. Kot vidimo na sliki 1, sta to skrajna pristopa, pri čemer datoteka s podatki pomeni večjo obremenitev za odjemalca, dostopna točka SPARQL pa večjo obremenitev za strežnik. V članku bomo obravnavali eno od konkretnih implementacij delcev povezanih podatkov – delce vzorcev trojčkov, ki so opisani v članku avtorjev R. Verborgh, M. Vander Sande, O. Hartig, J. Van Herwegen, L. De Vocht, B. De Meester, G. Haesendonck, P. Colpaert

(2016). Namen delcev vzorcev trojčkov je izboljšati učinkovitost poizvedovanja po povezanih podatkih s porazdelitvijo obremenitve med strežnikom in odjemalcem.

3.1 Strežnik

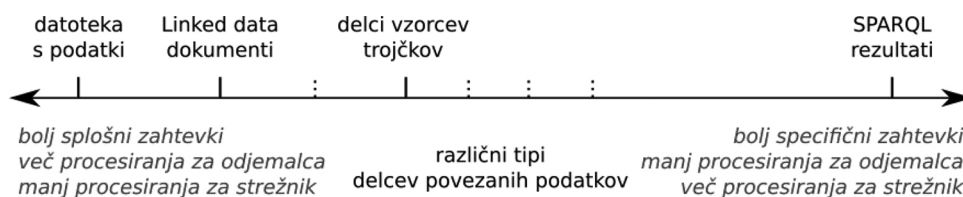
Pri delcih vzorcev trojčkov je izbirnik vzorec trojčka. Strežnik ponuja rezultate vzorca trojčka, razdeljene po straneh. Delec vzorca trojčka, ki ga vrača strežnik, vsebuje:

- podatke: rezultate vzorca trojčka (na trenutni strani),
- metapodatke: približno število vseh trojčkov v rezultatih (po vseh straneh),
- kontrole: povezave (URL) do drugih strani rezultatov podanega vzorca trojčka in do drugih delcev vzorcev trojčkov (korenski URL do delca z vsemi trojčki v bazi).

Ker je izbirnik namesto specifične poizvedbe SPARQL v vsakem primeru le vzorec trojčka, se pogosto zgodi, da se pri več različnih poizvedbah med izvajanjem pojavi isti izbirnik. Če je vsaj en vzorec trojčka v BGP-ju skupen več poizvedbam (pri čemer ime spremenljivke ni pomembno), pride do enakih zahtevkov. Zaradi tega je smiselno uporabiti posredniški predpomnilniški strežnik, ki posreduje podatke med strežnikom in odjemalcem. Ob tem si odgovore shranjuje v predpomnilnik, da jih potem ob enakih zahtevah lahko vrne odjemalcu brez obremenjevanja aplikacijskega strežnika.

3.2 Odjemalec

Odjemalec omogoča poizvedovanje po podatkih s pomočjo poizvedovalnega jezika SPARQL. Ker strežnik omogoča dostop do delcev vzorcev trojčkov, mora odjemalec pridobiti ustrezne podatke z njihovo pomočjo. Zaželeno je, da odjemalec zaradi izvedbe poizvedbe SPARQL naredi čim manj zahtevkov HTTP in prenese čim manj podatkov. Odjemalec uporablja metapodatke, konkretno število strani, za planiranje



Slika 1: Vse spletne vmesnike, ki ponujajo trojčke RDF, si lahko predstavljamo kot ponudnike delcev – vzorcev trojčkov.

poizvedbe. BGP-ji vsebujejo enega ali več vzorcev trojčkov in so glavni gradniki poizvedb SPARQL. Drugi operatorji, ki se nahajajo med BGP-ji, se izvajajo na strani odjemalca in niso specifični za predstavljeni način pridobivanja podatkov. Podpora teh je omejena in ni v obsegu članka. Odjemalec, razvit v okviru našega pristopa, je napisan v jeziku JavaScript in deluje v večini priljubljenih spletnih brskalnikov.

Odjemalec izvede BGP tako, da prenese prve strani delcev vseh nastopajočih vzorcev trojčkov. Tako v metapodatkih dobi število rezultatov za vsak vzorec trojčka. Nato izbere vzorec trojčka z najmanj rezultati in se začne sprehajati po njem. Za vsak trojček v rezultatih pripadajočo preslikavo (preslikavo spremenljivke v URI ali niz znakov) uporabi nad drugimi vzorci trojčka v BGP-ju in nad novim BGP-jem rekurzivno izvede isti algoritem. Če vmes pride do delca brez rezultatov, potem preslikave, ki so vodile do tega, ne vodijo do rešitve in jih zavržemo. Kadar po preslikavi vzorec trojčka ne vsebuje spremenljivk,

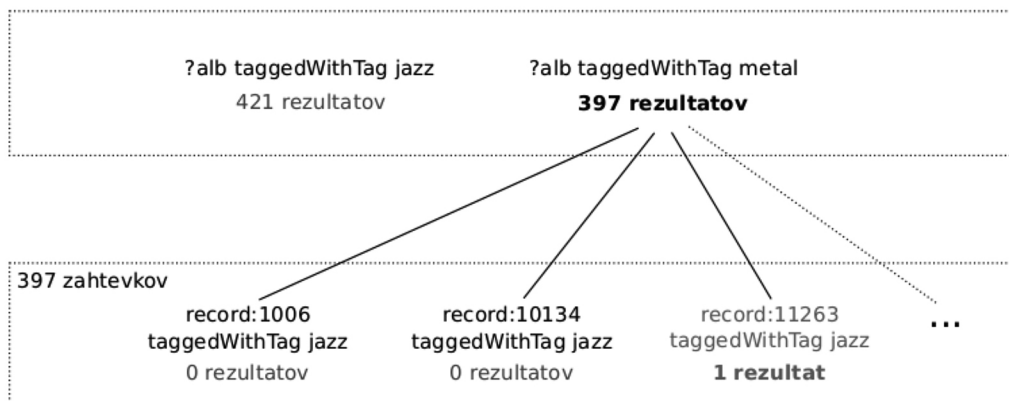
pride do t. i. vsebnostne poizvedbe. Gre za poizvedbe, s katerimi preverjamo vsebnost trojčka v bazi.

Za primer si pogledajmo preprosto poizvedbo (glej poizvedbo 1) iskanja ustrezne preslikave za spremenljivko `?alb`, ki nastopa kot subjekt v obeh vzorcih trojčkov. V tem primeru iščemo subjekte, ki so označeni tako z oznako `tag:jazz` kot z oznako `tag:metal`. V poizvedbi nastopa BGP, ki vsebuje vzorca trojčkov (`?alb tags:taggedWithTag tag:jazz`) in (`?alb tags:taggedWithTag tag:metal`). Iščemo torej subjekt, ki je prisoten v obeh delcih vzorcev trojčkov. Odjemalec izvede poizvedbo tako, da prenese prve strani obeh delcev in izbere vzorec trojčka, ki ima najmanj rezultatov. To je v tem primeru vzorec trojčka (`?alb tags:taggedWithTag tag:metal`) s 397 rezultati. Nato se začne po tem delcu sprehajati (po vseh straneh) in za vsak subjekt preveri, ali se nahaja tudi v drugem pripadajočem delcu. Če se, je pripadajoča preslikava prisotna v rezultatih. Tako pride do 397 vsebnostnih poizvedb, kot je prikazano na sliki 2.

```
PREFIX tags: <http://www.holygoat.co.uk/owl/redwood/0.1/tags/>
PREFIX tag: <http://dbtune.org/jamendo/tag/>
SELECT ?alb WHERE {
    ?alb tags:taggedWithTag tag:jazz. # 421
    ?alb tags:taggedWithTag tag:metal. # 397
}
```

Poizvedba 1: Poizvedba SPARQL, za katero odjemalec delcev vzorcev trojčkov naredi 397 vsebnostnih poizvedb.

Rezultat algoritma so vse preslikave, ki vodijo do rešitve.



Slika 2: Prikaz izvedbe poizvedbe 1. Namesto celotnih URI-jev so zaradi berljivosti prikazani njihovi zadnji deli.

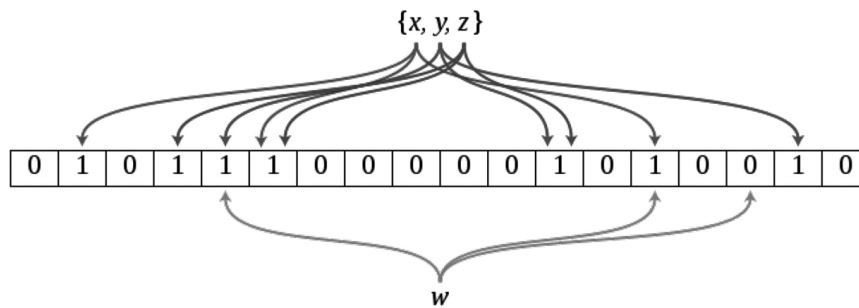
3.3 Obstoječa razširitev AMF

Razširitev AMF, kot jo opisuje članek avtorjev M. Vander Sande, R. Verborgh, J. Van Herwegen, E. Mannens, R. Van de Walle (2015), na metapodatke delca vzorca trojčka doda verjetnostno podatkovno strukturo za množico, ki vsebuje vse trojčke, nastopajoče kot rezultati vzorca trojčka (ni razdelitve po straneh). Pri tem uporablja Bloomov filter, s pomočjo katerega lahko preverimo, ali je možno, da se poljubni element nahaja v množici. Na vprašanje, ali je element v množici, lahko odgovorimo z *mogoče* ali *ne*. Prazen Bloomov filter je niz m bitov, nastavljenih na 0. Vsak element se s pomočjo k zgoščevalnih funkcij preslika v bite v tem nizu. Pri dodajanju elementa v Bloomov filter se pripadajoči biti nastavijo na 1, pri preverjanju vsebnosti elementa pa se preveri, ali so vsi pripadajoči biti nastavljeni na 1. Če so, potem je mogoče, da je element v množici, če niso, pa zagotovo vemo, da tega elementa ni v množici. Parametra m (velikost filtra) in k (število zgoščevalnih funkcij) sta lahko odvisna od velikosti množice, za katero gradimo Bloomov filter. S podano fiksno želeno verjetno-

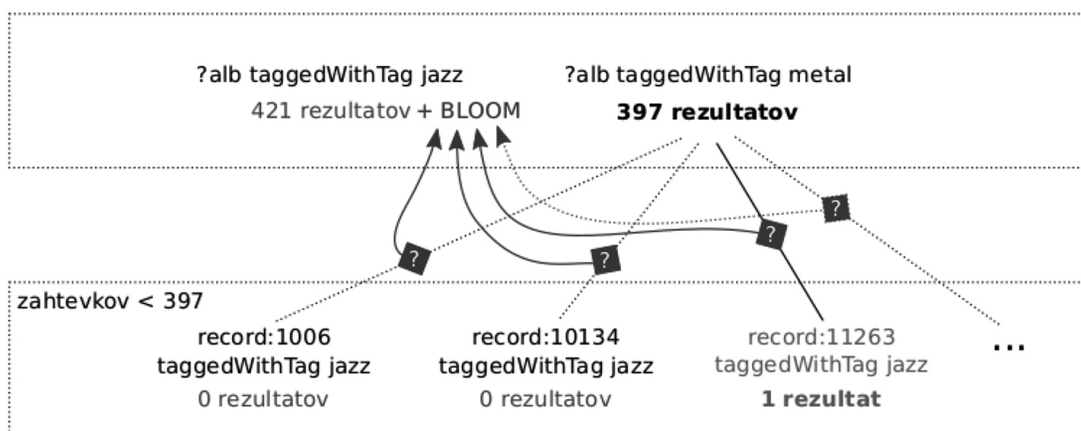
stjo lažnih pozitivnih odgovorov velikost Bloomovega filtra linearno narašča s številom elementov v množici (F. Bonomi, M. Mitzenmacher, R. Panigrahy, S. Singh, G. Varghese, 2006). Slika 3 prikazuje Bloomov filter, ki vsebuje elemente x , y in z , poizvedujemo pa po elementu w . Ker se w preslika v mesto, kjer je vrednost 0, vemo, da ga zagotovo ni v množici.

Razširitev poleg Bloomovih filtrov omogoča tudi uporabo podatkovne strukture GCS, ki je prostorsko malo učinkovitejša, vendar porabi več procesorske moči (F. Putze, P. Sanders, J. Singler, 2007). Medtem ko so rezultati razdeljeni po straneh, verjetnostna podatkovna struktura v metapodatkih obravnava celoten vzorec trojčka. Prednost te metode je, da zmanjša število vsebnostnih poizvedb (in posledično zahtevkov), saj nam ni treba preverjati vsebnosti trojčkov, za katere verjetnostna podatkovna struktura v metapodatkih ne izkazuje možnega vsebovanja v celotnem delcu vzorca trojčka.

Na sliki 4 je prikazana izvedba poizvedbe 1 preko delcev vzorcev trojčkov z opisano razširitvijo. Kvadratke z vprašaji si predstavljamo kot pogojne



Slika 3: Bloomov filter



Slika 4: Prikaz izvedbe poizvedbe 1 z razširitvijo AMF

stavke, pri katerih ne izvedemo vsebnostnih poi-zvedb, če glede na Bloomov filter v metapodatkih ni možnosti vsebovanja danega trojčka. Parametra Bloomovega filtra m (dolžina bitnega polja) in k (šte-vilo zgoščevalnih funkcij) se nastavitve glede na šte-vilo rezultatov v celotnem delcu vzorca trojčka tako, da imamo fiksno verjetnost napake oziroma lažno pozitivnega odgovora. Privzeta vrednost je 0,001 oziroma 0,1 %, s katero, glede na meritve, dobimo dobre rezultate.

3.4 Naš predlog izboljšave

Razširitev AMF zmanjša število vsebnostnih poi-zvedb, vendar lahko pri tem močno poveča velikost delcev. Naša ideja je, da v metapodatke dodamo trojčke, povezane z rezultati na *prikazani strani* delca vzorca trojčka, ki jih odjemalec prav tako lahko izkoristi za zmanjšanje števila vsebnostnih poi-zvedb. Pri tem se osredinimo na izboljšanje v smislu količine prenesenih podatkov.

Tako kot razširitev AMF tudi naša razširitev uporablja verjetnostno podatkovno strukturo, bolj specifično Bloomove filtre, vendar pri naši razširitvi obravnavamo druge podatke. Naša razširitev na metapodatke delca vzorca trojčka doda Bloomov filter, ki vsebuje trojčke, povezane s subjekti in objekti trojčkov na prikazani strani. Katere trojčke pri tem dejansko vsebuje Bloomov filter, je odvisno od različice naše razširitve.

Bloomov filter pri naši različici na strani i vsebuje:

- vse trojčke, povezane z objekti trojčkov na strani i , če je spremenljivka vzorca trojčka objekt,
- vse trojčke, povezane s subjekti trojčkov na strani i , če je spremenljivka vzorca trojčka subjekt,
- 4 različice, če sta spremenljivki vzorca trojčka tako subjekt kot objekt:
 - filtra ne vključimo (različica Jan 1),
 - v filtru so vsi trojčki, povezani s subjekti trojčkov na strani i (različica Jan 2),
 - v filtru so vsi trojčki, povezani z objekti trojčkov na strani i (različica Jan 3),
 - v filtru so vsi trojčki, povezani s subjekti ali objekti trojčkov na strani i (različica Jan 4).

Ker je lahko v odvisnosti od podatkov velikost

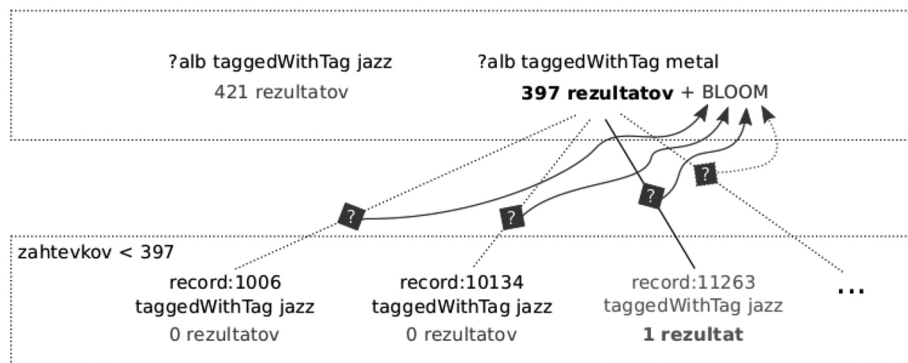
Bloomovega filtra zelo velika, se odločimo za imple-mentacijo omejitve števila trojčkov v Bloomovem filtru. Ko je določena omejitev števila trojčkov preko-račena, filtra ne pošljemo. Ta omejitev je nastavljiva.

Dodatna možnost naše razširitve so različice, v tem članku imenovane Jan N Filter, pri čemer je N številka ene od zgoraj navedenih različic. Pri teh različicah v filter dodajamo le trojčke z določenimi predikati. Gre za omejitev trojčkov v filtru glede na predikat. V filtru so namreč uporabni le trojčki s predikati, ki nastopajo v vsebnostnih poi-zvedbah. S tem lahko dodatno zmanjšamo količino prene-senih podatkov, če vemo, kateri predikati se bodo pojavljali v vsebnostnih poi-zvedbah. Protokol za uskladitev seznama v filtru uporabljenih predika-tov med strežnikom in odjemalcem ni določen; pri meritvah smo ta seznam ročno vnesli v programsko kodo strežnika in odjemalca, kot da gre za posebno različico, ki obravnava le določene predikate. Za splošno rabo bi bila potrebna sinhronizacija sezna-ma predikatov pred vsakim izvajanjem poi-zvedbe, vendar bi bil zagotovo del teh podatkov proti podatkom, potrebnim za izvedbo celotne poi-zvedbe, zelo majhen. Če ta seznam ni popoln, še vedno lahko izvajamo poi-zvedbe, ki terjajo vsebnostne poi-zvedbe z drugimi predikati, vendar njihovo iz-vajanje ni optimalno. Te seznam bi lahko strežnik sestavil samodejno glede na vsebnostne poi-zvedbe, ki bi jih dobival.

Kot je razvidno iz slike 5, se uporaben Bloomov filter pri naši razširitvi nahaja na drugem delcu vzorca trojčka kot pri razširitvi AMF. Razlika je, da pri naši razširitvi Bloomov filter vsebuje trojčke, poveza-ne z rezultati na prikazani strani, pri razširitvi AMF pa so v Bloomovem filtru vsi dejanski rezultati vzorca trojčka ne glede na prikazano stran.

Z algoritmom 1 pridobimo Bloomov filter, ki ga priložimo delcu vzorca trojčka. Prikazani algoritem je uporabljen v različici Jan 2, saj v primeru, da sta tako objekt kot subjekt spremenljivki, v filter dodamo trojčke, povezane s subjekti rezultatov.

Naša razširitev je bolj podrobno opisana v magi-strskem delu J. Robas (2016).



Slika 5: Prikaz izvedbe poizvedbe 1 z našo razširitvijo

```

tp          ... vzorec trojčka
stran      ... prikazana stran rezultatov vzorca trojčka tp
p_napaka   ... verjetnost napake za Bloomov filter

VrniFilter(stran, tp) {
  if(tp["predikat"] je spremenljivka) {
    vrni prazen filter
  } else if(tp["subjekt"] je spremenljivka || tp["objekt"] je spremenljivka) {
    del = "subjekt"
  } else if(tp["objekt"] je spremenljivka) {
    del = "objekt"
  }

  rezultati = prazna množica
  za vsak trojček na strani {
    q1 = poizvedba: trojček[del] ?p ?o
    q2 = poizvedba: ?s ?p trojček[del]
    rezultati = rezultati ∪ rezultati q1 ∪ rezultati q2
  }

  if(|rezultati| > omejitev) {
    vrni prazen filter
  }

  // nastavi optimalna parametra m in k
  m = (⌈-|ln(rezultati)|*ln(p_napaka)) / (ln(2)*ln(2))
  k = m / |ln(rezultati)|
  bloom = nov Bloomov filter s parametroma m in k

  za vsak trojček v rezultatih {
    bloom.dodaj(trojček["subjekt"] + "|" + trojček["predikat"] + "|" + trojček["subjekt"])
  }

  // bloom tukaj predstavlja vrednost Bloomovega filtra v zapisu base64
  vrni m + "," + k + "," + bloom
}

```

Algoritem 1: Pseudokoda za izgradnjo našega filtra

4 EVALVACIJA PREDLAGANE IZBOLJŠAVE

4.1 Metodologija

Za avtomatizirano izvajanje meritev smo uporabili BrowserMob Proxy¹ in Selenium WebDriver² z brskalnikom Chromium.

Pri ocenjevanju uspešnosti razširitve smo obravnavali naslednji metriki:

- število zahtevkov do strežnika: z našo razširitvijo želimo zmanjšati število vsebnostnih poizvedb; vsaka vsebnostna poizvedba predstavlja zahtevek HTTP;
- količina prenesenih podatkov od strežnika proti odjemalcu: z zmanjšanjem števila vsebnostnih poizvedb se zmanjša količina prenesenih podatkov, vendar pa filter v metapodatkih poveča velikost delcev vzorcev trojčkov. Ker dandanes za dostop do informacij pogosto uporabljamo mobilne naprave, je ta metrika pomembna. Pri naši razširitvi smo se osredinili na optimizacijo z vidika te metrike.

Računalnik, na katerem je potekalo testiranje, ima CPE Intel Core i7 6700, ki teče s frekvenco 3,4 GHz, in 16 GiB delovnega pomnilnika. Testiranje je potekalo v operacijskem sistemu Ubuntu 16.04 LTS. Pri obeh metriki velja, da je manjša vrednost boljša, saj obe predstavljata vir, ki ga hočemo pri izvajanju poizvedb čim manj izrabit.

Za testiranje smo uporabili Waterloo SPARQL Diversity Test Suite (G. Aluç, O. Hartig, M. T. Özsu, K. Daudjee, 2014) z zbirko, ki vsebuje deset milijonov trojčkov. Različne razširitve in osnovno različico brez razširitev smo testirali s 340 poizvedbami SPARQL. Te poizvedbe so generirane z WatDiv, tako da je za vsako od predlog C1, F1, F2, F3, F4, F5, L1, L2, L3, L4, L5, S1, S2, S3, S4, S5 in S6 generiranih 20 poizvedb. Vsaka od teh predlog predstavlja določeno obliko poizvedb, naključno pa se spreminjajo določeni URI-ji. Enkrat npr. iščemo igralce, ki se nahajajo v mestu A, drugič pa igralce, ki se nahajajo v mestu B, sama oblika poizvedbe, če bi jo predstavili z grafom, pa ostaja enaka. Izvedba poizvedb, ustvarjenih iz predlog F1, L2, L5, S2, S3, S4, S5 in S6, terja vsebnostne poizvedbe. Pri izvajanju teh poizvedb z našo razširitvijo pričakujemo manjše število zahtevkov, manjšo količino prenesenih podatkov in (posledično) krajši čas izvajanja v primerjavi z osnovno različico. Pri drugih poizvedbah

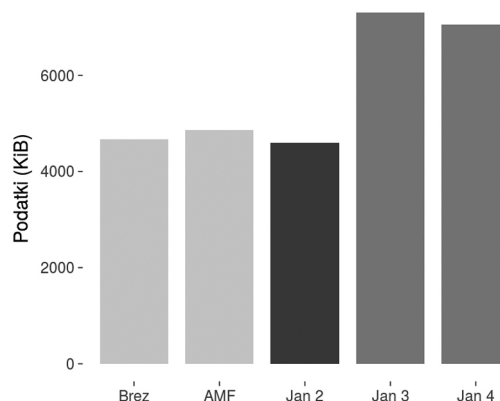
pričakujemo večjo količino prenesenih podatkov zaradi dodanega Bloomovega filtra. Verjetnost napake za Bloomove filtre je bila pri vseh razširitvah nastavljena na 0,1 odstotka. Na vseh stolpcičnih grafih je prikazano povprečje obravnavane metrike pri izvedbi vseh poizvedb oziroma njihove podmnožice. Pri intervalih zaupanja smo uporabili petodstotno stopnjo tveganja.

Pri razširitvah Jan N Filter smo v filter dodali trojčke, katerih predikati nastopajo v vsebnostnih poizvedbah katere koli od vseh 340 poizvedb, ki nastopajo v testu. Te predikate smo dobili tako, da smo vse poizvedbe pognali z osnovno različico in si ob vsaki vsebnostni poizvedbi shranili njen predikat. Dobili smo sedem različnih predikatov, ki smo jih pred testom sinhronizirali med strežnikom in odjemalcem. S protokolom, pri katerem bi potrebovali uskladitev enkrat na izvedbo poizvedbe, bi z obravnavanimi podatki in poizvedbami potrebovali okoli 1 KiB dodatnih podatkov na poizvedbo. Pri naših testnih podatkih in poizvedbah je povprečna količina prenesenih podatkov pri izvajanju poizvedbe z razširitvijo Jan 2 Filter okoli 3,5 MiB, s tem da je mediana okoli 1 MiB, torej bi bila razlika sorazmerno majhna. Pri vseh različicah Jan N Filter je omejitev števila trojčkov v filtru nastavljena na 20 tisoč.

4.2 Rezultati meritev

4.2.1 Količina prenesenih podatkov

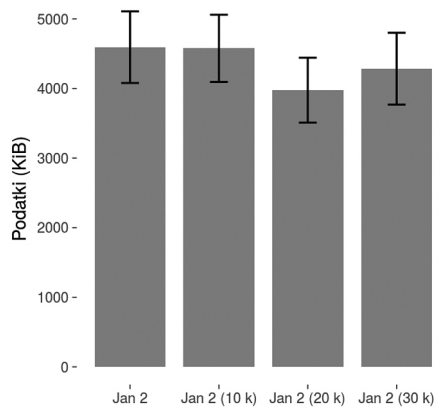
Na sliki 6 je prikazan graf primerjave različic glede na povprečno količino prenesenih podatkov pri izvedbi vseh poizvedb. Najmanj podatkov prenese različica z razširitvijo Jan 2. V nadaljevanju podrobnejše analize našega pristopa bomo obravnavali razširitvi Jan 2 in Jan 4.



Slika 6: Primerjava povprečne količine prenesenih podatkov pri izvedbi vseh 340 poizvedb SPARQL z različnimi razširitvami

¹ Dostopno na spletnem naslovu <https://github.com/lightbody/browsermobproxy>.

² Dostopno na spletnem naslovu <http://www.seleniumhq.org/>.

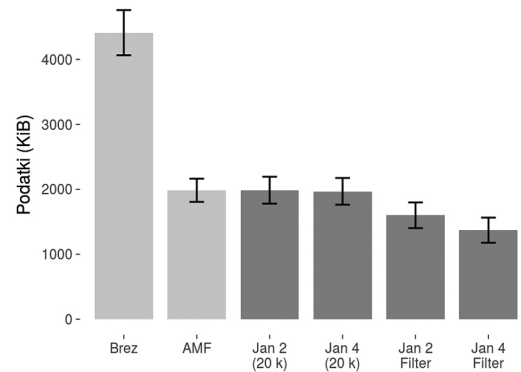


Slika 7: Primerjava povprečne količine prenesenih podatkov pri izvedbi vseh 340 poizvedb SPARQL z razširitvijo Jan 2 in uporabo različnih omejitev števila trojčkov v filtru

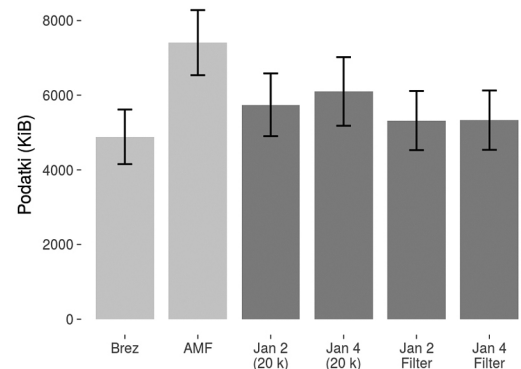
Količino prenesenih podatkov smo pri naši razširitvi poskusili zmanjšati z omejitvijo števila trojčkov v filtru. Ker je pri nas filter odvisen od trojčkov na prikazani strani, za poljuben vzorec trojčka prekoračitev na eni strani ne implicira prekoračitve na drugih straneh. Številka v oklepaju poleg oznake predstavlja omenjeno omejitev. Črka k je oznaka za kilo (10 k npr. predstavlja 10 tisoč). Izmerili smo količino prenesenih podatkov ob uporabi razširitve Jan 2 pri omejitvah 10 tisoč, 20 tisoč in 30 tisoč. Kot je razvidno iz grafa na sliki 7, se izkaže, da je količina prenesenih podatkov najmanjša pri omejitvi 20 tisoč, zato smo se odločili za nastavitev omejitve na to vrednost.

Primerjava povprečne količine prenesenih podatkov pri izvedbi poizvedb, ki terjajo vsebnostne poizvedbe, je prikazana na sliki 8. Iz grafa je razvidno, da, izvzemši rezultate različic Jan N Filter, od naših različic najmanj podatkov prenese Jan 2. Kar se tiče prenosa podatkov, je najbolj učinkovita različica Jan 4 Filter, vendar moramo za učinkovito uporabo te različice predhodno vedeti, kateri predikati bodo nastopali v vsebnostnih poizvedbah.

Na sliki 9 vidimo primerjavo povprečne količine prenesenih podatkov pri izvedbi poizvedb, ki ne terjajo vsebnostnih poizvedb. Učinkovitost je pri teh poizvedbah ob uporabi vseh opisanih razširitev slabša kot pri osnovni različici, saj vse razširitve z dodajanjem metapodatkov poskušajo zmanjšati število pri izvedbi teh poizvedb neobstojećih vsebnostnih poizvedb. Iz grafa je razvidno, da naše različice pri teh poizvedbah dodajo manj podatkov kor različica z razširitvijo AMF.

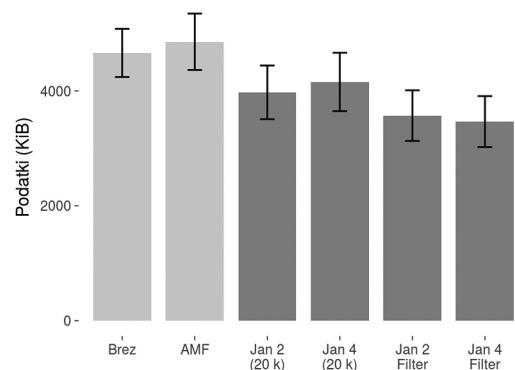


Slika 8: Primerjava povprečne količine prenesenih podatkov pri izvedbi sklopov poizvedb, ki terjajo vsebnostne poizvedbe (F1, L2, L5, S2, S3, S4, S5, S6), z različnimi razširitvami



Slika 9: Primerjava povprečne količine prenesenih podatkov pri izvedbi sklopov poizvedb, ki ne terjajo vsebnostnih poizvedb (C1, F2, F3, F4, F5, L1, L3, L4, S1), z različnimi razširitvami

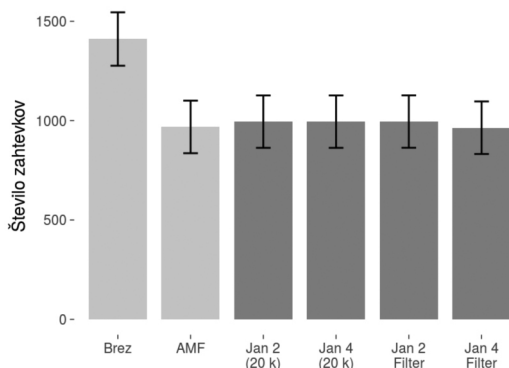
Povprečna količina prenesenih podatkov pri izvedbi vseh poizvedb je prikazana na sliki 10. Na grafu lahko vidimo, da najmanj podatkov prenese pri uporabi različice Jan 4 Filter, vendar moramo za take rezultate predhodno vedeti, kateri predikati bodo nastopali v vsebnostnih poizvedbah.



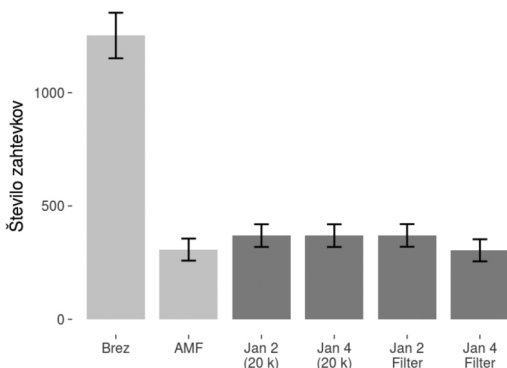
Slika 10: Primerjava povprečne količine prenesenih podatkov pri izvedbi vseh 340 poizvedb z različnimi razširitvami, s tem da pri naših razširitvah uporabljamo omejitve števila trojčkov v filtru

4.2.2 Število zahtevkov

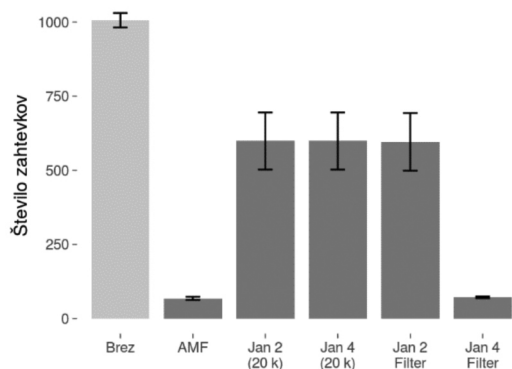
Na sliki 11 je prikazana primerjava razširitev glede na povprečno število zahtevkov pri izvedbi vseh 340 poizvedb. Vidimo lahko, da sta razširitvi Jan 2 in Jan 4 malo manj učinkoviti kot razširitvi AMF in Jan 4 Filter. Ker z vsemi razširitvami, navedenimi v tem članku, zmanjšujemo število vsebnostnih poizvedb, je število zahtevkov pri poizvedbah, ki ne terjajo vsebnostnih poizvedb, enako kot pri osnovni različici. Zaradi tega je razlika v številu zahtevkov bolj opazna na sliki 12. Pri razširitvi Jan 4, ki je najbolj splošna, je pričakovano podobno zmanjšanje števila zahtevkov kot pri razširitvi AMF. Izkaže se, da ob uporabi razširitve Jan 4 pri poizvedbah iz sklopa S4 prihaja do prekoračitve števila trojčkov v filtru in posledično filter ni poslan. Na sliki 13 vidimo primerjavo razširitev pri izvajanju poizvedb iz sklopa S4. Ob omejitvi trojčkov v filtru glede na predikat (Jan 4 Filter) se filter spleta poslati in je učinkovitost podobna kot pri razširitvi AMF. Razširitvi Jan 2 in Jan 4 sta brez uporabe filtra predikatov pri tem sklopu poizvedb manj učinkoviti kot razširitev AMF.



Slika 11: Primerjava povprečnega števila zahtevkov pri izvedbi vseh 340 poizvedb



Slika 12: Primerjava povprečnega števila zahtevkov pri izvedbi sklopov poizvedb, ki terjajo vsebnostne poizvedbe (F1, L2, L5, S2, S3, S4, S5, S6), z različnimi razširitvami



Slika 13: Primerjava povprečnega števila zahtevkov pri izvedbi poizvedb v sklopu S4

5 SKLEP

Čeprav je nevtralnost interneta zaradi različnih interesov pogosto pod vprašanjem (P. Ganley, B. Allgrove, 2006), nam spletne tehnologije omogočajo prosto prejemanje in objavljanje vsebine. Velikokrat pa je težava, kako ponuditi to vsebino, da jo bo kdo lahko našel. Rešitev za to so lahko povezani podatki, vendar je pri tem treba razviti način, kako iskati po veliki količini povezanih podatkov. Delci vzorcev trojčkov so splošen način iskanja po povezanih podatkih s porazdelitvijo bremena med strežnikom in odjemalcem. Poleg tega so zasnovani tako, da so razširljivi. V tem članku smo razvili svojo razširitev in jo primerjali z že obstoječo razširitvijo AMF. Ugotovili smo, da naša razširitev glede na test WatDiv z ustrežno nastavitvijo parametrov dosega dobre rezultate. Pri našem testu različica z našo razširitvijo prenese manj podatkov kot različica z razširitvijo AMF in različica brez razširitve.

V nadaljevanju bi bilo treba za metode Jan N Filter določiti protokol za usklajevanje seznama uporabljenih predikatov med strežnikom in odjemalcem. Lahko bi implementirali tudi algoritem, ki bi ta seznam na strani strežnika sproti prilagajal in bi tako omogočil bolj optimalno izvajanje poizvedb brez predhodne konfiguracije. Dobre rezultate bi lahko dobili tudi z nepopolnim seznamom predikatov.

Glede na pridobljene rezultate menimo, da je naša razširitev praktično uporabna. Še posebno dobro se obnese v primerih, ko nad podatki delamo poizvedbe, pri katerih se pogosto pojavljajo enaki predikati. Če uporabljamo razširitev Jan 4 Filter, lahko iskanje optimiziramo za najpogostejše poizvedbe in tako dosežemo še boljšo učinkovitost. Učinkovitost vsake razširitve pa je seveda odvisna tako od podatkov kot od poizvedb, ki jih izvajamo.

6 LITERATURA

- [1] Aluç, G., Hartig, O., Özsu, M. T., Daudjee, K. (2014). Diversified Stress Testing of RDF Data Management Systems, v *International Semantic Web Conference*, str. 197–212.
- [2] Aranda, C. B., Hogan, A., Umbrich, J., Vandenbussche, P.-Y. (2013). SPARQL Web-Querying Infrastructure: Ready for Action?, v *The 12th International Semantic Web Conference (ISWC2013)*, str. 273–288.
- [3] Berners-Lee, T. (2006). *Linked Data*. <https://www.w3.org/DesignIssues/LinkedData.html>. Dostopano 13. 11. 2017.
- [4] Bizer, C., Heath, T. (2011). Linked Data: Evolving the Web into a Global Data Space, *Synthesis Lectures on the Semantic Web: Theory and Technology*, zv. 1, št. 1, str. 1–136.
- [5] Bizer, C., Heath, T., Berners-Lee, T. (2009). *Linked Data - The Story So Far*, zv. 5, št. 3, str. 1–22.
- [6] Bonomi, F., Mitzenmacher, M., Panigrahy, R., Singh, S., Varghese, G. (2006). An Improved Construction for Counting Bloom Filters, v *European Symposium on Algorithms*, str. 684–695.
- [7] Ganley, P., Allgrove, B. (2006). Net neutrality: A user's guide, *Computer Law & Security Review*, zv. 22, št. 6, str. 454–463.
- [8] Putze, F., Sanders, P., Singler, J. (2007). Cache-, Hash- and Space-Efficient Bloom Filters, v *International Workshop on Experimental and Efficient Algorithms*. Springer, str. 108–121.
- [9] Robas, J. (2016). *Učinkovito poizvedovanje po povezanih podatkih s porazdelitvijo obremenitve*. Magistrsko delo, Fakulteta za računalništvo in informatiko, Ljubljana.
- [10] Vander Sande, M., Verborgh, R., Van Herwegen, J., Mannens, E., Van de Walle, R. (2015). Opportunistic Linked Data Querying through Approximate Membership Metadata, v *The Semantic Web – ISWC 2015*.
- [11] Verborgh, R., Hartig, O., De Meester, B., Haesendonck, G., De Vocht, L., Vander Sande, M., Cyganiak, R., Colpaert, P., Mannens, E., Van de Walle, R. (2014). Querying Datasets on the Web with High Availability, v *Proceedings of the 13th International Semantic Web Conference, ser. Lecture Notes in Computer Science*, zv. 8796, str. 180–196.
- [12] Verborgh, R., Vander Sande, M., Hartig, O., Van Herwegen, J., De Vocht, L., De Meester, B., Haesendonck, G., Colpaert, P. (2016). Triple Pattern Fragments: a Low-cost Knowledge Graph Interface for the Web, *Journal of Web Semantics*, zv. 37–38, str. 184–206.

■

Jan Robas je leta 2016 magistriral na področju računalništva in informatike na Fakulteti za računalništvo in informatiko Univerze v Ljubljani. Trenutno je zaposlen kot razvijalec programske opreme v podjetju Datastat, d. o. o. Ima večletne izkušnje z razvojem študijske informatike in pedagoškim delom na področju programiranja. Glavna področja, ki ga zanimajo in s katerimi se ukvarja, so razvoj informacijskih sistemov, programske paradigme, podatkovne zbirke, spletne tehnologije, semantični splet in poučevanje na področju računalništva. Zavzema se za ohranjanje odprtosti in nevtralnosti interneta.

■

Dejan Lavbič je leta 2010 doktoriral na področju računalništva in informatike na Fakulteti za računalništvo in informatiko Univerze v Ljubljani, kjer je zaposlen kot docent. Na raziskovalnem področju se ukvarja z inteligentnimi agenti, večagentnimi sistemi, ontologijami, poslovnimi pravili, semantičnim spletom, odkrivanjem plagiatorstva s pomočjo socialnih omrežij, kakovostjo informacij in naprednimi pristopi za porazdelitev podatkov. Sodeloval je pri številnih gospodarskih in raziskovalnih projektih s področja strateškega planiranja, metodologij razvoja informacijskih sistemov, uporabe inteligentnih sistemov, avtomatizacije poslovnih procesov in obvladovanja ter porazdelitve velike količine podatkov.