Zbornik radova Fakulteta tehničkih nauka, Novi Sad

**UDK: 004.738.1:766**
**DOI: https://doi.org/10.24867/29BE09Ivanic**

# POREĐENJE POKRIVENOSTI AKADEMSKIH BAZA

# COMPARISON OF COVERAGE OF ACADEMIC DATABASES

Aleksa Ivanić, *Fakultet tehničkih nauka, Novi Sad*

**Oblast – PRIMENJENE RAČUNARSKE NAUKE I INFORMATIKA**

**Kratak sadržaj –** *Opis implementacije sistema za poređenje pokrivenosti akademskih baza „OpenAlex“ i „Scopus“ korišćenjem Java programskog jezika, Spring Boot radnog okvira, Elasticsearch pretraživača i React JavaScript bilioteke.*

**Ključne reči:** *Akademske baze, poređenje, Elasticsearch*.

**Abstract** – *Description of the implementation of the system for comparing the coverage of the academic databases "OpenAlex" and "Scopus" using Java programming language, Spring Boot framework, Elasticsearch search engine and the React JavaScript library.*

**Keywords:** *Academic databases, comparing, Elasticsearch..*

## 1. UVOD

U ovom radu opisana je implementacija sistema za poređenje pokrivenosti akademskih baza koristeći *Java* programski jezik, *Spring Boot* radni okvir, *Elasticsearch* za skladištenje i pretragu naučnih radova i *ReactJS* biblioteke za implementaciju klijentske aplikacije.
Detaljno je opisan proces implementcije sistema za poređenje pokrivenosti akademskih baza. Opisana je konfiguracija *Elasticsearch*-a zatim korišćenje *Elasticsearch*-a zajedno sa *Java* programskim jezikom i *Spring Boot* radnim okvirom i na kraju sama implementacija indeksiranja naučnih radova i poređenje akademskih baza.

### 1.1. *Elasticsearch*

*Elasticsearch* pretraživač otvorenog koda (označava da je izvorni kod datog softvera dostupan i moguće ga je redistribuirati i modifikovati engl. *open-source*) i distribuiran je (softverski sistem u kojem komponente postavljene na povezane računare komuniciraju i koordiniraju svoje akcije slanjem poruka engl. *distributed*), a izgrađen je na vrhu *Apache Lucene*, biblioteke otvorenog koda, koja vam omogućava da implementirate funkciju pretraživanja u *Java* aplikaciji. *Elasticsearch* uzima ovu *Lucene* funkciju i proširuje je kako bi skladištenje, indeksiranje i pretraživanje učinili bržim, lakšim i, kao što ime kaže, elastičnim. Takođe, aplikacija ne mora da bude napisana u *Javi* da bi radila sa *Elasticsearch*-om;

**NAPOMENA:**
**Ovaj rad proistekao je iz master rada čiji mentor je bio dr Dragan Ivanović, red. prof.**

moguće je slati podatke preko *HTTP*-a u *JSON* formatu kako bi indeksirali, pretraživali i upravljali *Elasticsearch* klasterom.

U slučajevima kada je potrebno vraćanje relevantnih rezultata pretrage, dobavljanje statistike i sve to treba biti obaljveno brzo pretraživači poput *Elasticsearch*-a dolaze do izražaja jer su napravljeni da odgovore upravo na te izazove. Moguće je upariti pretraživač sa relacionom bazom podataka kako bi se kreirali indeksi i ubrzali *SQL* upiti. Ili, moguće je indeksirati podatke iz svog *NoSQL* skladišta podataka kako bi se omogućilo pretraživanje. Sve to moguće je postići sa *Elasticsearch*-om, a on dobro funkcioniše i sa dokument-orijentisanim bazama kao što je *MongoDB* jer su podaci u *Elasticsearch*-u takođe predstavljeni kao dokumenti. Moderni pretraživači kao što je *Elasticsearch* takođe su dobri u čuvanju podataka tako da ih je moguće koristiti kao *NoSQL* skladište podataka sa moćnim sposobnostima pretraživanja.

Elasticticsearch je samo pretraživač i nikada neće biti korišćen samostalno. Kao i svako drugo skladište podataka, potreban je način kako da se u njega unesu podaci i verovatno je neophodno obezbediti interfejs za korisnike koji pretražuju te podatke.
Da bismo stekli predstavu o tome kako bi se Elasticsearch mogao uklopiti u veći sistem, razmotrimo tri tipična scenarija:

- *Elasticsearch kao primarni bekend* (delovi aplikacije ili koda programa koji joj omogućavaju rad i kojima korisnik ne može pristupiti engl. *backend*) *za vebsajt* — ukoliko na primer postoji vebsajt koji omogućava ljudima da pišu objave na blogu, ali takođe želite mogućnost pretraživanja objava. Moguće je koristit *Elasticsearch* za skladištenje svih podataka u vezi sa ovim objavama kao i postavljanje upita.
- *Dodavanje Elasticsearch-a postojećem sistemu*—U slučajevima kada već postoji sistem koji obrađuje podatke a potrebno je dodati pretragu, ovo je najčešći scenario korišćenja *Elasticsearch*-a.
- *Elasticsearch kao bekend gotovog rešenja izgrađenog oko njega* — Budući da je Elasticsearch otvorenog koda i nudi jednostavan *HTTP* interfejs, veliki ekosistem ga podržava. Na primer, *Elasticsearch* je popularan za centralizovanje zapisa (zapis događaja ili radnji koje se dešavaju tokom izvršavanja programa engl. *log*); s obzirom na već dostupne alate koji mogu pisati i čitati sa *Elasticsearch*-a, osim konfigurisanja tih alata da rade onako kako želite, ne morate ništa dodatno da razvijate [1].

### 1.2. *API*

*API*-ji su mehanizmi koji omogućavaju da dve softverske komponente međusobno komuniciraju koristeći skup definicija i protokola. Na primer, softverski sistem meteorološkog zavoda sadrži dnevne podatke o vremenu. Aplikacija za vremensku prognozu na telefonu „razgovara“ sa ovim sistemom preko *API*-ja i prikazuje dnevne novosti o vremenu na telefonu.

*API* je skraćenica od engl. *Application Programming Interface*. U kontekstu *API*-ja, reč Aplikacija se odnosi na bilo koji softver sa jasnom funkcijom.
Interfejs se može posmatrati kao ugovor o usluzi između dve aplikacije. Ovaj ugovor definiše kako te dve aplikacije međusobno komuniciraju koristeći zahteve i odgovore. Njihova *API* dokumentacija sadrži informacije o tome kako programeri treba da strukturiraju te zahteve i odgovore.

*API* arhitektura se obično objašnjava terminima klijenta i servera. Aplikacija koja šalje zahtev naziva se klijent, a aplikacija koja šalje odgovor naziva se server. Dakle, u primeru vremenske prognoze, baza podataka meteorološkog zavoda je server, a mobilna aplikacija je klijent [2].

Postoje četiri različita načina na koje API-ji mogu da rade u zavisnosti od toga kada i zašto su kreirani:

- *SOAP API-ji* - koriste engl . *Simple Object Access Protocol*. Klijent i server razmenjuju poruke koristeći XML. Ovo je manje fleksibilan API koji je bio popularniji u prošlosti.
- *RPC API-ji* - engl. *Remote Procedure Calls*. Klijent kompletira funkciju (ili proceduru) na serveru, a server šalje izlaz nazad klijentu.
- *Vebsoket API-ji* - još jedan moderan razvoj veb API-ja koji koristi JSON objekte za prosleđivanje podataka. Vebsoket API podržava dvosmernu komunikaciju između klijentskih aplikacija i servera. Server može da šalje povratne poruke povezanim klijentima, što ga čini efikasnijim od REST API-ja.
- *REST API-ji* - ovo su najpopularniji i najfleksibilniji API-ji koji se danas nalaze na vebu. Klijent šalje zahteve serveru kao podatke. Server to koristi za pokretanje internih funkcija i vraća izlazne podatke nazad klijentu.

### 1.3. *REST*

***REST*** je akronim za ***RE**presentational **S**tate **T**ransfer* i arhitektonski stil za distribuirane hipermedijske sisteme. Roj Filding ga je prvi put predstavio 2000. godine u svojoj čuvenoj disertaciji. Od tada je postao jedan od najčešće korišćenih pristupa za izgradnju veb-baziranih *API*-ja (Aplikacijski programski interfejsi).
*REST* nije protokol ili standard, to je arhitektonski stil. Tokom faze razvoja, *API* programeri mogu implementirati *REST* na različite načine. Kao i drugi arhitektonski stilovi, *REST* takođe ima svoje vodeći principe i ograničenja. Ovi principi moraju biti zadovoljeni kako bi servisni interfejs bio *REST*ful [3].

### 1.4. Pregled relevantne literature

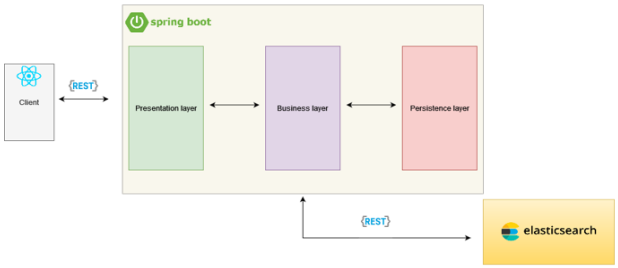
U radu [4] opisana je tehnika za merenje relativne veličine i preklapanja javnih veb pretraživača. Pretraživači su među najkorisnijim i najpopularnijim uslugama na Vebu. Korisnici su željni da saznaju kako se upoređuju. Koji ima najveću pokrivenost? Da li su indeksirali isti deo veba? Koliko stranica se tamo nalazi? Iako se o ovim pitanjima raspravljalo u popularnoj i u tehničkoj štampi, nije predložena objektivna metodologija evaluacije i nije se pojavilo dovoljno jasnih odgovora.

U ovom radu opisan je standardizovan, statistički način merenja pokrivenosti pretraživača i preklapanja putem nasumičnih upita. Ova tehnika ne zahteva privilegovan pristup nijednoj bazi podataka. Predstavljeni su rezultati obavljenih eksperimenata koji pokazuju procene veličine i preklapanja za *„HotBot“*, *„AltaVista“*, *„Excite“* i *„Infoseek“* kao i procente njihove ukupne pokrivenosti sredinom 1997. i novembra 1997.

Ova metoda ne daje apsolutne vrednosti. Međutim, koristeći podatke iz drugih izvora, procenjeno je da od novembra 1997. godine broj stranica koje su indeksirali *„HotBot“*, *„AltaVista“*, *„Excite“* i *„Infoseek“* bio otprilike 77 miliona, 100 miliona, 32 miliona i 17 miliona, a zajednička ukupna pokrivenost iznosila je 160 miliona stranica. Dalje se pretpostavlja da je veličina statičnog, javng *Web*-a od novembra bila preko 200 miliona stranica. Ono što je najviše iznenadilo jeste da je preklapanje veoma malo: manje od 1,4% od ukupne pokrivenosti, ili oko 2,2 miliona stranica indeksirano je od strane sva četiri pretraživača.

## 2. SPECIFIKACIJA SISTEMA ZA POREĐENJE POKRIVENOSTI AKADEMSKIH BAZA

U ovom poglavlju biće opisana arhitektura sistema za poređenje pokrivenosti akademski baza. Klijentska strana aplikacije implementirana je korišćenjem *„React“ JavaScript* biblioteke, dok je serverska aplikacija implementirana pomoću *Java* programskog jezika kao i *Spring Boot* radnog okvira. *Spring Boot* aplikacije zansivaju se na slojevitoj arhitekturi (engl. *Layered Architecture*) gde hijerarhiski susedni slojevi međusobno komuniciraju, takva arhitektura zastupljena je i u bekend-u ove aplikacije. Interakcija između *Elasticsearch*-a i bekend-a aplikacije odvijaće se pomoću *Spring Data Elasticsearch*-a, koji u mnogome olakšava indeksiranje i pretragu prilikom korišćenja *Elasticsearch*-a. Komunikacija između klijentske i serverske aplikacije odvija se posredstvom *API* ulazne tačke uz poštovanje ograničenja *REST*-a.




Slika 1. *Athitektura sistema*

## 3. POREĐENJE *„OpenAlex“* I *„Scopus“* BAZA NAUČNIH RADOVA

U ovom poglavlju biće opisan algoritam poređenja baza naučnih radova, koji se sastoji iz ciklusa. U svakom ciklusu će se nasumično birati određen broj dokumenata iz liste koja je dobijena pretraživanjem predthodno indeksiranih radova. Zatim je odabrane dokumente potrebno pronaći u bazi naučnih radova sa kojom poredimo trenutnu. Algoritam ciklusa će detaljnije biti opisan u nastavku:

- Odabir predefinisanog leksikona: potrebno je prvo pretražiti prethodno indeksirane radove kako bi se iz te inicijalne liste nasumično odabranali dokumenti za koje će se kasnije proveravati postojanje u drugoj bazi. Upit za pretragu sastavlja se od reči izabranih iz predefinisanog leksikona. Potrebno je izabrati jedan od tri postojeća leksikona: prvi je sastavljen od reči preuzetih iz ključnih reči naučnih radova, drugi sastavljen od reči iz apstrakta a treći od reči iz naslova.
- Pretraga naučnih radova: Nakon što je odabran leksikon sastavlja se upit nasumično birajući reči iz leksikona pa se zatim vrši pretraga radova odakle će se odabrati oni radovi čije će se postajanje proveravati u drugoj bazi. Vrši se *boolean* pretraga gde je upit sastavljen disjunkciom prethodno odabranih reči. Za pretragu je korišćen `ElasticsearchRestTemplate` i `NativeQuery` koji nam omogućavaju maksimalnu fleksibilnost prilikom sastavljanja upita.
- Pretraga odabranog naučnog rada u drugoj bazi – Za svaki rad potrebno je proveriti da li postoji u drugoj bazi, pa je pre svega neophodno sastaviti upit koji jedinstveno oslikava dati naučni rad. U zavisnosti od odabira leksikona, u svrhu prezentaciju uzećemo da je odabran leksikon sa rečima iz apstrakta naučnih radova, sastavlja se upit od reči iz apstrakta datog rada tako što se proverava frekventnost svake reči prolazeći kroz leksikon. Na kraju se za upit uzimaju najmanje frekventne reči. Na taj način postižemo jedinstvenost upita za taj naučni rad.
- Algoritam za izračunavanje poklapanja radova – Nakon izvršene pretraga u drugoj bazi *Elasticsearch* može vratiti veliki broj rezultata pa će se stoga proveravati prvih 40 vraćenih radova. Kako bi rad zadovoljio kriterijume i smatrao se odgovarajućim neophodno je da po *default*-u zadovoljava sledeće kriterijume:
  - Potpuno poklapanje identifikatora digitalnog objekta
  - Poklapanje naslova 80%
  - Poklapanje autora 40%

Korisniku je omogućeno da eksperimentiše i da menja ove kriterijume kao i da bira koji leksikon će se koristiti.

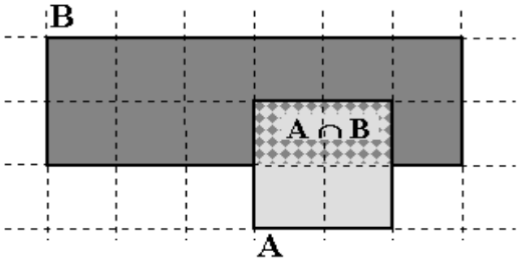
- Slanje podataka na klijentski deo aplikacije (engl. *frontend*) – nakon izvršenog kompleksnog izračunavanja poklapanja radova, liste sa uzorkovanim naučnim radovima gde je svaki rad uparen sa istim takvim u suprotnoj bazi, bazi sa kojom se poredi, šalju se na frontend gde će se dalje vršiti detaljni prikaz i izračunavanje mera poklapanja dveju baza.

## 4. REZULTATI I DISKUSIJA

Sistem za izračunavanje poklapanja baza naučnih radova omogućava merenje relativne veličine i preklapanja baza. Za svaki par naučnih baza DB1 i DB2 moguće je izračunati:

- njihove relativne veličine: Veličina(DB1):Veličina(DB2)
- Deo DB1 baze koji je indeksiran u DB2, izražen kao procenat veličine baze DB1

Ideja koja stoji iza ovog pristupa je jednostavna: razmotrimo skupove A i B veličine 4 i 12 jedinica respektivno. Neka veličina njihovog preseka A i B bude 2 (slika 2). Pretpostavimo da ne znamo nijednu od ovih veličina, ali možemo jednolično uzorkovati iz bilo kog skupa i proveriti postojanje u bilo kom skupu. Ako uzorkujemo iz A i ustanovimo da je oko 1/2 uzoraka takođe u B. Otuda je veličina A otprilike 2 puta veća od preseka, A i B. Slično tome ćemo otkriti da je veličina B otprilike 6 puta veća od veličine A i B. Ovo će nas dovesti do zaključka da je B oko 6 /2 = 3 puta veći od A.




Slika 2. *Izračunavanje odnosa veličine na osnovu procena preklapanja*

U nastavku će biti prikazani estimacija preklapanja i relativnih veličina računato na sledeći način:

- Procena preklapanja: deo baze podataka DB1 indeksiran i u bazi DB2 procenjuje se na sledeći način:
  `Deo naučnih radova uzorkovanih iz DB1 pronađenih u DB2`
- Poređenje veličine: Za baze podataka DB1 i DB2, odnos Veličina(DB1)/Veličina(DB2) se procenjuje na sledeći način:

```
Deo naučnih radova uzorkovanih iz
DB2 pronađenih u DB1
-------------------------------------------------
Deo naučnih radova uzorkovanih iz
DB1 pronađenih u DB2
```

U nastavku će biti prikazani rezultati merenja na po 1000 uzorkovanih radova iz svake baze.

| DB1 (izvor radova) | DB2 (baza sa kojom poredimo) | Procena preklapanja (radovi iz DB1 pronađeni u DB2) | | | | Poređenje veličine (veličina DB1/veličina DB2) | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 |
| OpenAlex | Scopus | 42% | 48% | 43% | 40% | 1.21 | 1.17 | 1.26 | 1.27 |
| Scopus | OpenAlex | 51% | 56% | 54% | 51% | 0.82 | 0.86 | 0.80 | 0.78 |

Tabela 1. *Rezultati merenja poređenja baza naučnih radova, korišćenjem leksikona sastavljenim od naslova radova*

| DB1 (izvor radova) | DB2 (baza sa kojom poredimo) | Procena preklapanja (radovi iz DB1 pronađeni u DB2) | | | | Poređenje veličine (veličina DB1/veličina DB2) | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 |
| OpenAlex | Scopus | 38% | 37% | 38% | 39% | 1.18 | 1.13 | 1.13 | 1.15 |
| Scopus | OpenAlex | 45% | 42% | 43% | 45% | 0.84 | 0.88 | 0.88 | 0.87 |

Tabela 2. *Rezultati merenja poređenja baza naučnih radova, korišćenjem leksikona sastavljenim od ključnih reči radova*

| DB1 (izvor radova) | DB2 (baza sa kojom poredimo) | Procena preklapanja (radovi iz DB1 pronađeni u DB2) | | | | Poređenje veličine (veličina DB1/veličina DB2) | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 |
| OpenAlex | Scopus | 41% | 39% | 41% | 42% | 1.17 | 1.15 | 1.12 | 1.14 |
| Scopus | OpenAlex | 48% | 45% | 46% | 48% | 0.85 | 0.87 | 0.89 | 0.87 |

Tabela 3. *Rezultati merenja poređenja baza naučnih radova, korišćenjem leksikona sastavljenim od apstrakta radova*

## 5. ZAKLJUČAK

U ovom radu opisan je sistem za poređenje pokrivenosti akademskih baza *„OpenAlex“* i *„Scopus“* korišćenjem *Java* programskog jezika, *Spring Boot* radnog okvira, *Elasticsearch* pretraživača *i React JavaScript* bilioteke.
Na osnovu rezultata dobijenih korišćenjem opisanog sistema može se zaključiti da baza *„OpenAlex“* indeksira veći broj radova. Korisniku je data mogućnost da konfiguriše proces poređenja ali kao što je prethodno prikazano u svim slučajevima poređenja dobija se približno ista razlika u pokrivenosti u korist *„OpenAlex“* baze.
Pored poređenja pokrivenosti baza sistem pruža korisniku složen uvid u uzorkovane dokumente koji su korišćeni u procesu poređenja tako da je moguće detaljnije analizirati rezultate. Takođe je omogućena pretraga indeksiranih naučnih radova iz *„OpenAlex“* i *„Scopus“* baza.

### 5.1. Dalji razvoj sistema

Ovaj sistem sigurno ima veliki potencijal za dalji razvoj, jedan od načina kako se može unaprediti jeste da se omogući jednostavno dodavanje novih akademskih baza za poređenje što bi zahtevalo izmene u samoj arhitekturi, modelu i implementaciji sistema ali bi uloženo vreme u razvoj ove funkcionalnosti sigurno rezultiralo u poboljšanju sistema. Takođe, pretpostavka je i da bi veštačka inteligencija mogla poboljšati algoritam za sastavljanje upita koji najbolje opisuju uzorkovane radove čije postojanje je potrebno proveriti u drugoj bazi.

## 6. LITERATURA


[1] Radu Gheorghe, Matthew Lee Hinman, and Roy Russo, „Elasticsearch in Action“, Novembar 2015.
[2] What is an API (Application Programming Interface)? https://aws.amazon.com/what-is/api/ (pristupljeno u aprilu 2024.)
[3] What is REST https://restfulapi.net/ (pristupljeno u aprilu 2024.)
[4] Krishna Bharat and Andrei Broder, „A technique for measuring the relative size and overlap of public Web search engines“, 1998.


**Kratka biografija:**


Aleksa Ivanić rođen 7.8.1998. godine u Zrenjaninu. Smer računarstvo i automatika na Fakultetu tehničkih nauka u Novom Sadu upisao je 2017. godine. Osnovne studije završio je 2021. godine i iste godine upisao master akademske studije. kontakt: ivanicaleksa98@gmail.com