Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.738.1:766
DOI: https://doi.org/10.24867/10BE13Bokic

SISTEM ZA DETEKCIJU PLAGIJARIZAMA U DOKUMENTIMA PISANIM NA SRPSKOM JEZIKU

SYSTEM FOR DETECTION OF PLAGIARISM IN DOCUMENTS WRITTEN IN SERBIAN

Stefan Bokić, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *Rad opisuje arhitekturu i implementaciju sistema za proveru naučnih radova (diplomski, master, doktorat). U sistemu je moguće dodavati naučne radove i proveravati da li postoje slični radovi u kolekciji. Ako postoje slični radovi, koliko su slični. Takođe nastavnici imaju mogućnost da provere da li ima previše poklapanja uvidom u naučne radove.*

Ključne reči: *Elasticsearch, plagijarizam, naučni radovi*

Abstract – *This paper describes the architecture and implementation of the system for checking scientific papers (graduate, master, doctoral). It is possible to add scientific papers to the system and check if there are similar papers in the collection. If there are similar papers, how similar are they. Professors also have the opportunity to check if there are too many matches by looking at scientific papers.*

Keywords: *Elasticsearch, plagiarism, scientific papers*

1. UVOD

Razvoj tehnologija uslovio je njihovu sve veću integraciju u ljudske živote. Takođe sa razvojem interneta dolazi do transformisanja načina na koji funkcionišu komunikacioni sistemi. S obzirom da je internet dostupan svima, a na internetu se nalazi veliki broj informacija, dolazi se do situacije da je jednim klikom moguće kopirati informacije sa interneta. Zbog toga je od izuzetne važnosti razlikovati verodostojne naučne radove i plagijate. Na taj način se doprinosi kredibilitetu naučnih radova

2. TEORIJSKE OSNOVE

Plagijarizam predstavlja skup algoritama koje omogućavaju proveru radova, tj. pokazuje da li jedan rad predstavlja kopiju drugog. Zašto je važno u današnjem svetu razlikovati prave radove od njihovih kopija, plagijata?! Danas plagijarizam predstavlja veliki problem u profesionalnom i edukativnom svetu. S obzirom da je internet dostupan svima, i da se veliki izvor informacija nalazi na internetu, vrlo je lako kopirati te informacije. Plagijarizam se takođe odnosi na upotrebu tuđih informacija, jezika ili pisanja, upotrebljenih bez odgovarajuće potvrde originalnog izvora [1, 2].

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

Plagijarizam u okviru tekstualnih dokumenata može se pojaviti u nekoliko oblika [1, 2]:

- plagirani tekst se može kopirati jedan na jedan
- odlomci se mogu kopirati u većem ili manjem obimu
- odlomci se mogu prevesti sa stranih jezika

Uspeh postojećih pristupa u otkrivanju plagijata teksta zavisi od načina plagijarizma, tj. koji od tri navedena načina je upotrebljen. Postoje tri glavne ideje koje se mogu iskoristi za traženje plagijata [3]:

- *substring matching* – podudaranje delova teksta
- *keyword similarity* – sličnost ključnih reči
- *fingerprint analysis* – generisanje digitalnog otiska

Podudaranje delova teksta pokušava da identifikuje maksimalno poklapanja delova teksta plagijata i originalnog teksta [3].

Kod pristupa sličnosti ključnih reči, ideja je da se identifikuju značajne reči iz dokumenta i da se uporede sa značajnim rečima drugih dokumenata. Ako sličnost premaši zadati prag, dokument predstavlja plagijat.

Dokument je podeljen u delove koji se rekurzivno porede. Ovaj algoritam pretpostavlja da se plagijarizam dešava u dokumentima sa sličnim temama [3].

Kod pristupa generisanje digitalnog otiska, koristi se kriptografska heš funkcija. Dokumenti su podeljeni u delove teksta. Heš funkcija služi za generisanje digitalnih otisaka dela teksta (niza termova), koji se zatim upoređuje sa bazom originalnog teksta. S obzirom da heš funkcija identifikuje niz reči jedinstveno, to znači da kvalitet ovog pristupa zavisi od veličine delova teksta i odstupanja od originalnog teksta [3].

3. KORIŠĆENE TEHNOLOGIJE

Java je programski jezik opšte namene koji je zasnovan na klasama i objektno orijentisanom programiranju. Pored toga Java je osmišljena da ima što je moguće manje zavisnosti od implementacije. Dakle, glavna ideja tvoraca Jave je da kod jednom napisan može da se pokrene sa bilo koje mašine.
To znači da kod iskompajliran u Javi može da se pokrene na bilo kojoj platfomi koja podržava Javu [5].

Spring radni okvir je platforma koja pruža sveobuhvatnu infrastrukturnu podršku za razvoj Java aplikacija. Spring upravlja sa infrastrukturom aplikacije, tako da se programeri mogu skoncentrisati na poslovnu logiku svoje aplikacije. Ovo je omogućeno kroz korišćenje objekata (engl. *Plain old Java objects* - POJO), anotacija i konfiguracijonih fajlova. Spring je zamišljen kao lagan radni okvir. Dva glavna koncepta na kojima se bazira Spring su [6]:

- injektovanje zavisnosti (engl. *dependency injection – DI*) [7]
- aspekti orijentisanog programiranja (engl. *aspect-oriented programming* - AOP) [8]

Elasticsearch je pretraživač otvorenog koda koji nudi pretragu u gotovo realnom vremenu, mogućnost pretraživanja celokupnog teksta, kao i RESTful API. Dakle, *Elasticsearch* je pretraživač celokupnog teksta napisan u Javi, koji je osmišljen da bude distributivan, skalabilan i da vraća rezultate pretrage u realnom vremenu. Takođe, *Elasticsearch* server je jednostavan za instalaciju, može se podići bez ikakvog podešavanja, mada većina korisnika želi da podešava funkcionalnosti što je isto moguće kroz podešavanje raznih parametara. Pokrenuta instanca *Elasticsearch*-a se naziva čvor, dva ili više čvorova formiraju klaster. Da bi se postavio klaster potrebno mu je samo zadati ime. *Elasticsearch* će se sam pobrinuti za otkrivanje čvorova na mreži i povezati ih u klaster [9].

Angular predstavlja radni okvir za dizajniranje aplikacija i razvojnu platformu za kreiranje efikasnih i sofisticiranih jednostraničnih aplikacija (engl. *single page aplication* [12]). Angular je poput mnogih drugih veb radnih okvira baziran na komponentama. To znači da su komponente glavni gradivni blokovi u Angular-u. Blokovi mogu da prikazuju informacije, renderuju šablone i izvršavaju akcije nad podacima. Najbolje prakse sugerišu da bi komponente trebale da se sastoje iz tri dela [10, 11]:

- HTML fajl za šablone
- CSS fajl za stilove
- *TypeScript* fajl za kontrolu

4. SPECIFIKACIJA

Sistem za detekciju plagijata je veb aplikacija namenjena za više korisnika koji uz pomoć aplikacije mogu da unose seminarske i završne radove različitih nivoa studija (diplomski, master ili doktorski rad) u sistem i proveravaju poklapanja sa drugim radovima. Sistem za detekciju plagijata olakšava i ubrzava provere prepisivanja sa drugim radovima. Kako radi aplikacija:

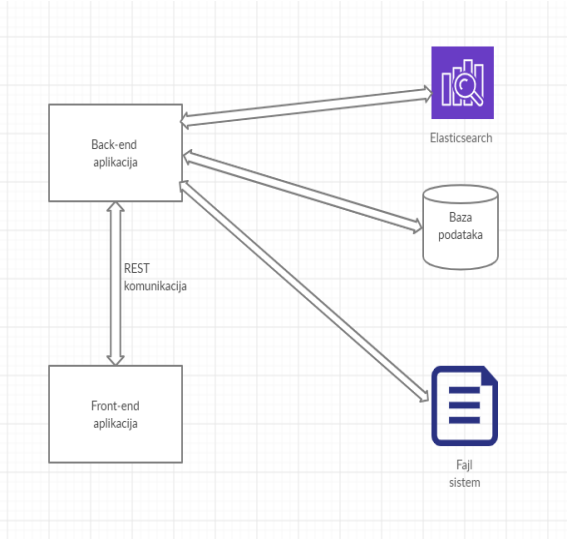
- Potrebno je da korisnik napravi svoj nalog
- Nakon toga je moguće logovanje na aplikaciju
- Moguće je otrpemanje fajlova koji ne prolaze kroz proveru plagijarizma, tj. oni su već objavljeni
- Nakon napunjene baze sa naučnim radovima, je moguće vršiti proveru na plagijate
- Po otpremanju novog fajla, dobija se rezultat provere na kom su prikazani najsličniji rezultati
- Ukoliko korisnik želi može da pristupi detaljima određenog rada gde su prikazane sličnosti između dva dokumenta po delovima
- Postoji mogućnost preuzimanja oba rada kako bi korisnik ručno proverio poklapanja i napisao komentar za ta dva rada

4.1. Arhitektura sistema

Sistem se sastoji iz sledećih delova:

- *Backend* aplikacija
- *Frontend* aplikacija
- Baza podataka
- Fajl sistem
- *Elasticsearch*

Na slici 1. je prikazana arhitektura sistema koja je prethodno opisana.



Slika 1. *Arhitektura sistema*

Backend aplikacija predstavlja glavni deo ovog sistema u kojoj se nalazi sva poslovna logika. Takođe ona obezbeđuje servise svim prethodno navedenim delovima aplikacije. Realizuje se kao servisno orijentisana veb aplikacija. *Backend* aplikacija je implementirana pomoću Spring radnog okvira.

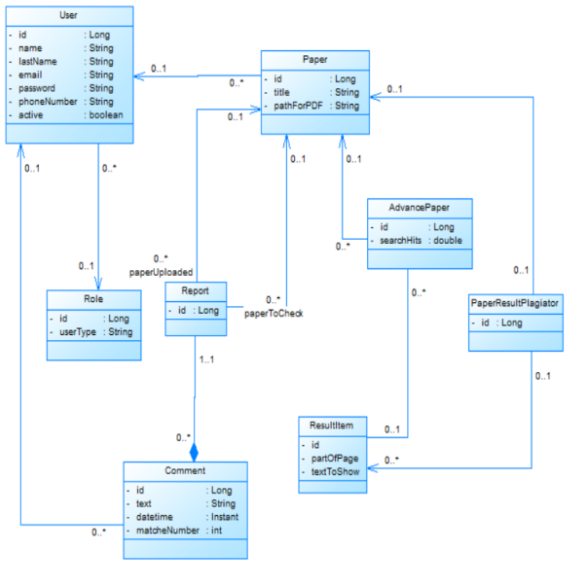
Frontend aplikacija obezbeđuje interfejs i funkcionalnosti koje su neophodne korisnicima. *Frontend* aplikacija je implementirana pomoću Angular radnog okvira.

Baza podataka se koristi za skladištenje podataka vezanih za korisnike, metapodatke o naučnim radovima, kao i putanje na fajl sistemu gde su uskladišteni otpremljeni naučni radovi.

Fajl sistem se koristi za skladištenje naučnih radova.

Elasticsearch se koristi za skladištenje metapodataka o naučnim radovima, pretprocesiranih tekstova izvučenih iz naučnih radova, njihovu obradu i odgovor na upite koji su vezani za sličnost između naučnih radova.

Na slici 2. je prikazan model klasa sistema koji je zadužen za fizičku reprezentaciju tabela u bazi podataka.



Slika 2. *Dijagram klasa sistema*

5. IMPLEMENTACIJA

Ono što je od najvećeg interesa je pitanje na koji način se podaci čuvaju u *Elasticsearch*-u. U okviru podešavanja *Elasticsearch*-a je podešen analizator (engl. *analyzer*) koji je zadužen za analiziranje teksta koji stiže na *Elasticsearch* server. U okviru njega je ubačen *Shingle token filter* [13]. Koji je podešen da od teksta pravi niz reči od veličine četiri pa do veličine šest reči. Parametri koji su zaduženi za to su *min_shingle_size* i *max_shingle_size*. Takođe pored toga je podešeno da ne želimo da se dodatno prave reči od veličine jedan. Šta to znači? Pogledajmo na sledećem primeru. Ako imamo rečenicu *Danas je lep i sunčan dan*. Ova rečenica se sastoji od šest reči. Ako bi se primenio *Shingle token filter* koji je podešen izlaz bi bili sledeći nizovi reči:

- Danas je lep i
- Danas je lep i sunčan
- Danas je lep i sunčan dan
- je lep i sunčan
- je lep i sunčan dan
- lep i sunčan dan

Iz ovoga zaključujemo da rezultat predstavljaju svi podstringovi veličine četiri, pet i šest. Podešavanja su prikazana u listingu 1.

Naravno nije jedini cilj da izdelimo tekst na jednake delove i da se nadamo da će se baš ti identični isečci naći u drugim radovima. Naš cilj je da postignemo da svi radovi koji su pisani na sprskom jeziku, bilo da su napisani na ćirilici, latinici, bez dijakritika, u drugim padežima, licima, rodovima, budu prepoznati sa tim isečćima ako predstavljaju isti tekst.

To se postiže dodavanjem *plugin-a* za srpski jezik koji se bavi rešavanjem prethodno navedenih problema.

```
settingsBuilder =
    XContentFactory.jsonBuilder()

    .startObject()
      .startObject("analysis")
        .startObject("filter")
          .startObject(
                 "filter_shingle")
          .field("type","shingle")
          .field(
            "min_shingle_size", 4)
          .field(
            "max_shingle_size", 6)
          .field(
         "output_unigrams", false)
          .endObject()
        .endObject()
        .startObject("analyzer")
          .startObject(
          "ShingleAnalyzer")
          .field("type","custom")
          .field("tokenizer",
                  "standard")
          .array("filter",
                "lowercase",
                "filter_shingle")
          .endObject()
        .endObject()
      .endObject()
    .endObject();
```

Listing 1. *Podešavanja analizatora*

Definisan je parametar koji govori na koliko termova će se deliti rad prilikom kreiranja segmenata naučnog rada. Na ovaj način je postignuto otkrivanje plagijata koji su posledica dva ili više naučnih radova. Na primer naučni rad radX je nastao tako što je autor kopirao prvu trećinu iz rad1, drugu iz rad2 i treću iz rad3. Nakon što je rad podeljen na segmente, taj segment prolazi kroz definisani analizator i šalje se upit ka *Elasticsearch* serveru. Kao rezultat tog upita se dobija *searchHits* za svaki naučni rad. *SearchHits* je jedan realan broj koji govori o tome koliko je taj upit uspeo da pogodi taj rad.
U prevodu provere se svi radovi i za svaki rad se dobije jedan realan broj. Rezultati su vraćeni sortirano. Međutim taj broj ne znači ništa sam po sebi, pošto *searchHits* zavisi od broja termova u radu i od sadržaja samog rada.
Međutim mi možemo izračunati koliki bi mogao da bude maksimalan *searchHits*. To se uradi tako što se proveri koliki bi bio *searchHits* da se rad poredi sa samim sobom. Kada podelimo ta dva broja dobijamo procenat sličnosti ta dva naučna rada. Jako je važno napomenuti da u zavisnosti od pogođenog terma u radu se *searchHits* povećava.
Konfiguracije *shingle filter tokena* utiče na to da povećavanje *searchHits* broja bude veće ukoliko se u radu nađu jedni pored drugih nizovi termova kreiranih prilikom otpremanja rada, nego kad se pogodi pojedinačan term.

6. ZAKLJUČAK

U radu je predstavljen sistem za proveru plagijarizama. Nakon toga je objašnjeno šta to predstavlja plagijarizam i na koje načine se može detektovati. Zatim su opisane tehnologije koje su korišćene za implementaciju sistema. Nakon toga, predstavljen je kompletan opis sistema, kao i njegove komponente i moduli. Detaljno su opisani moduli, funkcionalnosti modula i komunikacija između istih. Pored toga je prikazan model klasa sistema koji je zadužen za fizičku reprezentaciju tabela u bazi podataka.

U nastavku je prikazana implementacija sistema. Obrazloženi su zanimljivi delovi koda u kojima je implementirana provera plagijarizma, odnosno oni delovi koda koji su zaduženi za pretprocesiranje delova teksta, pripremu naučnih radova za proveru na plagijarizam.

Cilj ovog rada je bio da pronađe radove koji dosta liče, imaju dosta napisanog sličnog teksta i da ih prikaže korisniku kako bi on mogao detaljnije da to pregleda i utvrdi da li su stvarno plagijati. Međutim i dalje postoje neki nedostaci. Na primer ukoliko bi dva naučna rada bila napisana na istu temu i poseduju isti fond reči, Sistem za detekciju plagijata bi rekao da su dosta slični čak iako ne predstavljaju plagijate. Tako da je to jedna od stvari koja bi mogla da bude unapređena. Druga stvar, kada Sistem za detekciju plagijata da rezultate on može samo da nabroji najsličnije naučne radove i da eventualno po delovima kaže koji su delovi najsličniji. To znači da bi jedan od pravaca nastavka razvoja ovog rada bio da se nakon liste najsličnijih radova koje vrati Sistem za detekciju plagijata, na svakom od njih označi koji je to deo tačno kopiran.

7. LITERATURA

[1] Zu Eissen, Sven Meyer, and Benno Stein. "Intrinsic plagiarism detection." In *European conference on information retrieval*, 2006.

[2] Zu Eissen, Sven Meyer, Benno Stein, and Marion Kulig. "Plagiarism detection without reference collections." In *Advances in data analysis*, 2007.

[3] Benno Stein and Sven Meyer zu Eißen. Near Similarity Search and Plagiarism Analysis. In Proc. 29th Annual Conference of the GfKl Springer, 2006.

[4] https://ieee-dataport.org/sites/default/files/analysis/27/IEEE%20Citation%20Guidelines.pdf (pristupljeno u avgustu 2020.)

[5] Java 8, *Java dokumentacija*, https://docs.oracle.com/javase/8/docs/ (pristupljeno u avgustu 2020.)

[6] Webb, Phillip, Dave Syer, Josh Long, Stéphane Nicoll, Rob Winch, Andy Wilkinson, Marcel Overdijk, Christian Dupuis, and Sébastien Deleuze. "*Spring boot reference guide*." *Part IV. Spring Boot features* 24 (2013).

[7] Prasanna, Dhanji R. "Dependency injection." (2009).

[8] Irwin, John, Gregor Kickzales, John Lamping, Anurag Mendhekar, Chris Maeda, Cristina Videira Lopes, and J. Loingtier. "Aspect-oriented programming." *Proceedings of ECOOP, IEEE, Finland* (1997)

[9] Kononenko, Oleksii, Olga Baysal, Reid Holmes, and Michael W. Godfrey. "Mining modern repositories with elasticsearch." In *Proceedings of the 11th working conference on mining software repositories*, pp. 328-331. 2014.

[10] Angular dokumentacija, https://angular.io/docs (pristupljeno u avgustu 2020.)

[11] Wohlgethan, Eric. "Supporting Web Development Decisions by Comparing Three Major JavaScript Frameworks: Angular, React and Vue. js." PhD diss., Hochschule für Angewandte Wissenschaften Hamburg, 2018.

[12] Mikowski, Michael, and Josh Powell. *Single page web applications: JavaScript end-to-end*. Manning Publications Co., 2013.

[13] Shingle token filter, https://www.elastic.co/guide/en/elasticsearch/reference/current/analysis-shingle-tokenfilter.html (pristupljeno u avgustu 2020.)

Kratka biografija:

Stefan Bokić rođen je 02.10.1996. godine u Novom Sadu. Osnovnu školu „Žarko Zrenjanin“ završio je 2011. godine. Gimnaziju „Jovan Jovanović Zmaj“ u Novom Sadu završio je 2015. godine. Iste godine upisao se na Fakultet tehničkih nauka, odsek Računarstvo i automatika. Školske 2019. godine je diplomirao sa prosekom 9.97. Iste godine upisao je master studije, smer Računarstvo i automatika, modul Elektronsko poslovanje. Položio je sve ispite predviđene planom i programom.