SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 2232

Detekcija fragmenata koda raspoloživih na portalu Stack Overflow u aplikacijama na GitHubu

Lea Rački

Zagreb, srpanj 2020.

Umjesto ove stranice umetnite izvornik Vašeg rada.
Da bi ste uklonili ovu stranicu obrišite naredbu `\izvornik`.

SADRŽAJ

1. Uvod **1**

2. Kloniranje kodova **3**

2.1. Klonovi tipa 1 - T1 4

2.2. Klonovi tipa 2 - T2 5

2.3. Klonovi tipa 3 - T3 6

2.4. Klonovi tipa 4 - T4 8

2.5. Klonovi kodova u Internetu 8

3. Prikupljanje i priprema podataka **10**

3.1. *Stack Overflow* 10

3.2. *GitHub* 12

4. Alat za detekciju klonova - *SourcererCC* **14**

4.1. Algoritma za otkrivanje klonova 15

4.1.1. Faza kreiranja parcijalnog indeksa 15

4.1.2. Detekcija klonova 16

5. Obrada podataka **17**

5.1. Raščlanjivanje kodova na skup tokena 18

5.2. Pronalazak klonova 21

5.3. Konfiguracijski parametri 22

6. Rezultati **23**

7. Zaključak **24**

Literatura **25**

Popis slika **27**

Popis tablica **28**

Popis ispisa **29**

Sažetak **30**

A. Posts.xml **32**

B. ArcSSLSocketFactory.java - *GitHub* **34**

C. fragment_395433.java - *Stack Overflow* **36**

1. Uvod

Programiranje, odnosno razvoj programske podrške (enlg. *software development*) i programsko inženjerstvo (enlg. *software engineering*) aktivnosti su koje se temelje na suradnji pojedinaca, timova ili čitavih zajednica. Programeri koriste različite portale i alate kao pomoć u svom radu, kako bi se koordinirali jedni s drugima, stekli nova znanja i informirali se o novim tehnologijama. Takvi su portali javno dostupni i osmišljeni na način da integriraju mnoge značajke koje podržavaju i olakšavaju suradnju pojedinaca, njihovu komunikaciju i koordinaciju [11].

Među internetskim portalima koje programeri svakodnevno posjećuju svakako se ističe *Stack Overflow* [6]. *Stack Overflow* je forum koji se temelji na konceptu pitanja i odgovora. Postavljajući pitanja na *Stack Overflow*-u, programeri mogu potražiti pomoć i dobiti savjete svojih kolega primjerice o vlastitom isječku koda, problemu na koji su naišli ili o nekim nedovoljno dokumentiranim tehnologijama. Odgovarajući na pitanja drugih, programeri mogu dijeliti svoje znanje i stručnost, pomagati i educirati druge ili postizati veći ugled na forumu [12]. *Stack Overflow* se nekad čak rangira više od službenih dokumentacija i knjiga kao resursa za traženje programskog rješenja za neki problem. Programeri se oslanjaju na odgovore sa *Stack Overflow*-a iz razloga jer ih je lako i intuitivno za pronaći na webu. Uz sve to, svako pitanje i odgovor na *Stack Overflow*-u daju opis problema odnosno njegov kontekst što programeri smatraju korisnim prilikom rješavanja svoga problema, a glasanje i prihvaćanje odgovora pružaju pouzdanost u točnost rješenja ponuđenih na *Stack Overflow*-u [7]. Tako se može tvrditi da sudjelovanje na *Stack Overflow*-u ubrzava razvojne aktivnosti jer se pružaju brza rješenja tehničkih izazova, čime se programerima štedi dragocjeno vrijeme [12].

Još jedan internetski portal kojeg vrijedi spomenuti je svakako *GitHub* [3]. *GitHub* je jedan od najvećih internetskih portala za pohranu programskog koda u kojemu se nalazi, prema podacima iz siječnja 2020. godine, više od sto milijuna softverskih projekata koje održava preko četrdeset milijuna registriranih korisnika odnosno programera. *GitHub* pruža podršku i korisničko sučelje za kontrolu verzija i upravljanje izvornim programskim kodom te kontrolu pristupa projektnim repozitorijima [14].

Dvije se platforme preklapaju u ekosustavu za razmjenu znanja: *GitHub* programeri mogu zatražiti pomoć u StackOverflow-u za rješavanje vlastitih tehničkih izazova. Na sličan način, oni se mogu uključiti u StackOverflow da zadovolje potražnju za znanjem drugih, možda manje iskusnih od sebe [12].

Developeri za vrijeme razvoja programa i njihovog održavanja recikliraju, to jest ponovno koriste već napisane dijelove programskih komponenata, isječke kodova i programske knjižice [12]. Navedena tvrdnja dovodi do pretpostavke da programeri za vrijeme svoga rada kopiraju tuđe, odnosno već napisane fragmente programskog koda u svoje aplikacije i projekte te ih potom u određenoj mjeri izmjenjuju i prilagođavaju svojim potrebama. Proces korištenja gotovog koda koji uključuje ponovnu upotrebu odnosno kopiranje izvornog koda naziva se kloniranje koda (engl. *code cloning*). Kloniranje koda je aktivnost ponovne uporabe izvornog koda kopiranjem s nekog javno dostupnog izvora u vlastiti projekt [7]. Navedena pretpostavka o kloniranju kodova, ako je točna, može biti korisna u nizu situacija. Primjerice, prilikom dekompajliranja koda moguće je detektirati klonirane fragmente i na taj način poboljšati kvalitetu dekompajliranja. Isto tako, ako su fragmenti koje pojedinci u svome radu kopiraju sa StackOverflow ranjivi, detekcijom tih fragmenata moguće je detektirati ranjivosti u aplikacijama [7].

Svrha ovog rada je istražiti pretpostavku o kloniranju kodova, koliko je ono često i na koji se način mogu detektirati klonirani fragmenti u izvornim kodovima različitih aplikacija i programa. U sklopu ovog diplomskog rada istražuje se povezanost kodova objavljenih na *Stack Overflow*-u i onih na *GitHub*-u, odnosno u kojoj se mjeri isječci kodova dostupni na *Stack Overflow* portalu kopiraju u odabranom skupu aplikacija raspoloživih u javno dostupnim repozitoriju koda *GitHub*, a pritom će se uspoređivati samo kodovi pisani programskim jezikom *Java*.

Rad je strukturiran na sljedeći način. Na početku rada detaljnije je opisan pojam kloniranja kodova i vrste kloniranja. U idućem poglavlju opisuje se na koji su način prikupljeni podaci s obje platforme, *StackOverflow*-a i *GitHub*-a, te proces pripreme prikupljenih podataka za daljnju analizu. U 4. poglavlju opisan je alat za otkrivanje klonova - *SourcererCC*. Sljedeće, 5. poglavlje sadrži detaljni opis postupka obrade podataka uz korištenje *SourcererCC* alata. Rezultati rada nalaze se u 6 poglavlju, a u posljenjem, 7. poglavlju nalazi se zaključak rada.

2. Kloniranje kodova

Duplicirani ili klonirani programski kod (engl. *code clone*) je računarski izraz za dio izvornog koda koji se u istom ili sličnom obliku pojavljuje više puta, bilo u okviru jednog ili kroz različite programe. Kloniranje programskih kodova je uobičajena pojava u razvoju softvera i očekivani postotak klonova u tipičnim softverskim sustavima iznosi od 7 do 23 posto [7].

Automatizirani postupak pronalaska klonova u izvornom kodu naziva se otkrivanje ili detekcija kloniranja koda (engl. *code clone detection*) [13]. Detekcija kloniranja koda tema je mnogih istraživačkih radova u programskom inženjerstvu, a pitanje jesu li klonovi pozitivni ili ne, odnosni jesi li poželjni u programskim kodovima, još je uvijek dvojbeno. Nekolicina autora navodi da klonovi dovode do širenja i propagiranja pogrešaka u kodu, unošenja ranjivosti u kod, kršenja softverskih licenci te do problema s održavanjem softvera, dok neki drugi sugeriraju da klonovi nisu štetni i čak mogu biti korisni. Nepažljivo kloniranje koda iz jednog projekta u drugi može uzrokovati navedene probleme i to se također događa u kontekstu internetskih foruma kao što je Stack Overflow. Kloniranje koda ima i nekoliko primjena kao što su detekcija plagijata, traženje podrijetla izvornog koda i prepoznavanje kršenja softverskih licenci [7].

Dva fragmenta programskih kodova su klonovi ako su dovoljno slični prema određenoj definiciji sličnosti. S obzirom na otvorenu interpretaciju definicije sličnosti, postoje različiti alati za otkrivanje klonova i detektori plagijata, napravljeni na temelju različitih mjerenja sličnosti [7]. U ovom radu koristi se jedna od definicija sličnosti koja je najčešća, to jest široko prihvaćena, a dijeli klonove koda na 4 tipova sličnosti:

- klonovi tipa 1 (T1),
- klonovi tipa 2 (T2),
- klonovi tipa 3 (T3) i
- klonovi tipa 4 (T4) [10].

2.1. Klonovi tipa 1 - T1

Klonovi T1 su gotovo identični fragmenti programskih kodova, točnije to su fragmenti koji se razlikuju jedino po broju i rasporedu razmaka unutar jedne ili između više linija i po komentarima [10]. U nastavku, u ispisima 2.1 i 2.2 prikazana su dva fragmenta programskog koda koji prikazuju klonove tipa 1.

Ispis 2.1: Originalni fragment koda

```
1  public static void main(String[] args) throws IOException {
2        File input = new File("./file.txt");
3        BufferedReader reader = new BufferedReader(
4                                    new FileReader(input));
5        String line;
6        long pairsCount = 0;
7
8        while ((line = reader.readLine()) != null) {
9            String[] split = line.split(",");
10
11           if(!split[0].equals(split[2])) {
12               pairsCount++;
13           }
14       }
15
16       System.out.println(pairsCount);
17   }
```

Ispis 2.2: Klonirani fragment koda tipa T1

```
1  public static void main(String[] args) throws IOException {
2  File input=new File("./file.txt");
3
4  BufferedReader reader = new BufferedReader(new FileReader(input));
5  String line;long pairsCount=      0;
6
7
8  while((line=reader.readLine())!=null){
9  String[] split=line.split(",")
10 if(!split[0].equals(split[2])){
11 pairsCount++;}}
12
13
14 //printing result
15 System.out.println(pairsCount);}
```

Prikazani kodovi razlikuju se po broju praznih redaka, broju razmaka unutar redaka i po komentarima. U ispisu 2.1 kod je formatiran u više razina odnosno linije su uvučene različitim veličinama praznina. Programski kod u ispisu 2.2 ne sadrži uvučene linije, točnije sve linije započinju u istoj razini. Također, ispis 2.2 sadrži komentar u liniji 14, dok ispis 2.1 uopće ne sadrži komentare. Uz to, ispisi se razlikuju u rasporedu linija. Primjerice, linije 5 i 6 iz ispisa 2.1 zapisane su zajedno u jednom retku u programskom kodu prikazanom u ispisu 2.2.

2.2. Klonovi tipa 2 - T2

Klonovi tipa T2 sadrže razlike koje su karakteristične za klonove tipa T1 i uz to se dodatno razlikuju po imenima varijabli odnosno identifikatora te po vrijednostima zapisanim u konstantama ili varijablama [10]. Primjeri klonova tipa T2 prikazani su u nastavku, u ispisima 2.3 i 2.4.

Ispis 2.3: Originalni fragment koda

```
1  public static void main(String[] args) throws IOException {
2        File inputFile = new File("./file.txt");
3        BufferedReader reader = new BufferedReader(
4                                    new FileReader(input));
5        String line;
6        long counter = 0;
7
8        while ((line = reader.readLine()) != null) {
9            String[] split = line.split(",");
10
11           if(!split[0].equals(split[2])) {
12               pairsCount++;
13           }
14       }
15       System.out.println(pairsCount);
16   }
```

Ispis 2.4: Klonirani fragment koda tipa T2

```
1   public static void main(String[] args) throws IOException {
2         File inputFile = new File("./files/file.txt");
3         BufferedReader reader = new BufferedReader(
4                                       new FileReader(inputFile));
5         String line;long pairsCount=0;
6
7
8
9  //loop that counts lines with different elements on 0th and 2nd index
10        while ((line = reader.readLine()) != null) {
11        String[] array = line.split(",");
12        if(!array[0].equals(array[2])) {counter++;}
13        }
14        //printing result
15        System.out.println(counter);
16    }
```

Kodovi prikazani u ispisima 2.3 i 2.4 razlikuju se po broju praznih redaka, razmaka i po komentarima kao i programski kodovi prikazani u ispisima 2.1 i 2.2, no još se dodatno razlikuju po nazivima i vrijednostima varijabli. U linijama označenim brojem 2 u oba ispisa, nalazi se varijabla istog tipa i istog imena, no pridružene su im različite vrijednosti. Također u ispisu 2.3 u liniji 6 inicijalizirana je varijabla tipa `long` i imena `counter`, a u ispisu 2.4 ta se varijabla nalazi u liniji 5 i pridjeljeno joj je ime `pairsCount`.

2.3. Klonovi tipa 3 - T3

Klonovi koda tipa T3 su sintaksno slični fragmenti koji se razlikuju po rasporedu linija koda. Klonirani se fragment razlikuje od originalnog po višku linija odnosno po novim, dodanim linijama, po modificirani i/ili uklonjenim, to jest izbrisanim linijama koda. T3 klonovi također sadrže razlike prijašnjih tipova, T1 i T2 klonova [10]. U ispisima 2.5 i 2.6 prikazani su primjeri klonova tipa 3.

Klonovi T3 se često nazivaju i skorim promašajima (engl. *near-miss clones*) odnosno skorim duplikatima (engl. *near-duplicates*) iz razloga što su fragmenti koda podvrgnutim manjim ili većim sintaksnim izmjenama i na taj se način puno teže otkrivaju i detektiraju kao klonovi. Smatra se da je takvih klonova najviše i najznačajniji su kod detekcija klonova između velikih (engl. *large-scale*) projekata [10].

Ispis 2.5: Originalni fragment koda

```
1   public static void main(String[] args) throws IOException {
2         File input = new File("./file.txt");
3         BufferedReader reader = new BufferedReader(
4                                       new FileReader(input));
5         String line;
6         long pairsCount = 0;
7
8         while ((line = reader.readLine()) != null) {
9             String[] split = line.split(",");
10
11            if(!split[0].equals(split[2])) {
12                pairsCount++;
13            }
14        }
15        System.out.println(pairsCount);
16    }
```

Ispis 2.6: Klonirani fragment koda tipa T3

```
1   public static void main(String[] args) throws IOException {
2         File inputFile = new File("./files/file.txt");
3         BufferedReader reader = new BufferedReader(
4                                       new FileReader(inputFile));
5         String line;
6         long counter=0;
7         ArrayList<String> lines=new ArrayList<>();
8         while (!((line=reader.readLine())==null)) {
9         lines.add(line);
10            String[] split = line.split(",");
11            if(!split[0].equals(split[2])) counter+=1;
12        }
13    }
```

Programski kodovi prikazani u ispisima 2.5 i 2.6 sadrže razlike karakteristične za klonove tipa T1 i T2 - razlike u imenima i vrijednostima varijabli, razlike u broju i rasporedu praznina te razlike u komentarima. U odnosu na kod iz ispisa 2.5, u ispisu 2.6 dodane su linije označene brojem 7 i 9, modificirane su linije 8 i 11, a linija 15 iz ispisa 2.5 je u potpunosti izbrisana.

2.4. Klonovi tipa 4 - T4

T4 tipovi kloniranih kodova označavaju sintaksno različite fragmente koda koji implementiraju istu funkcionalnost, ali različitom logikom [10]. U ispisima 2.7 i 2.8 prikazane su dvije implementacije funkcije koja prva dva cijela broja i vraća njihov umnožak. Fragmenti prikazani u ispisima su primjeri klonova T4 jer oba implementiraju istu funkcionalnost, no sintaksno su potpuno različiti.

Ispis 2.7: Originalni fragment koda

```
1  public int multiply(int a, int b) {
2        return a*b;
3    }
```

Ispis 2.8: Klonirani fragment koda tipa T4

```
1  public int getProductFrom(int a, int b) {
2        int result = 0;
3        for (int i = b; i >= 0; b--) {
4            result +=a;
5        }
6        return result;
7    }
```

U nastavku rada naglasak će biti klonovima tipa 3, a još će se spominjati klonovi tipa 1 i tipa 2.

2.5. Klonovi kodova u Internetu

Isječci kodova koji su kopirani iz softverskih sustava na internetska web mjesta, poput *Stack Overflow*-a, i obrnuto, nazivaju se *internetski klonovi koda* ili *klonovi kodova u internatu* (engl. *online code clones*). Postoje dva smjera u nastajanju internetskih klonova:

- kod je kloniran iz lokalnog projekta na neki web portal kao primjer, ili
- kod se klonira s internetskog portala u lokalni projekt za postizanje određene funkcionalnosti ili ispravljanje pogreške.

Nažalost, internetske klonove teško je pronaći jer je prostor za pretraživanje enorman i više nije ograničen samo na lokalne repozitorije kodova. Odgovori na *Stack Overflow*-u često kloniraju kod s drugih lokacija poput osobnih projekata, projekata poduzeća i projekata otvorenog koda, kako bi Stack Overflow stajao kao rješenje ili

nadopuna rješenja. Aktivnost kloniranja koda na *Stack Overflow*-u očigledno je korisna, s obzirom na popularnost *Stack Overflow*-a i njegov utjecaj na razvoj softvera [7].

3. Prikupljanje i priprema podataka

U svrhu proučavanja opsega kloniranja kodova između *Stack Overflow*-a i *GitHuba*-a, prikupljeni su podaci s obje platforme. U ovom poglavlju govori se o načinu prikupljanja podataka i njihove pripreme za daljnju obradu. Ovo istraživanje oslanja se na sve kodove s portala *Stack Overflow* dostupne na dan 2. ožujka 2020. godine i na 4029 Java projekta skinutih s *GitHuba*-a.

3.1. *Stack Overflow*

Svi javno dostupni podaci sa *Stack Overflow*-a, uključujući i sva javna pitanja i odgovore moguće je preuzeti sa *Stack Exchange* internetske stranice [1]. Podaci sa *Stack Exchange*-a dostupni su u obliku snimke memorije. Snimka memorije *Stack Overflow*-a objavljuje se na *Stack Exchange* stranici svaka 3 mjeseca [12], a podaci koji se koriste u sklopu ovoga rada, odnosno snimka memorije objavljena je na dan 2. ožujka 2020. godine. Također, podaci su objavljeni pod *Creative Commons* licencom što znači da su podaci dostupni za dijeljenje i slobodno korištenje [1]. Snimak memorije sadrži XML datoteku u kojoj su sadržana sva pitanja i svi odgovori koji su javno dostupni na stranicama *Stack Overflow*-a od srpnja 2008. godine pa sve do dana kada je objavljena zadnja verzija snimke memorije [12]. U dodatku A na kraju rada prikazan je dio XML datoteke u kojoj se detaljno može vidjeti njezina struktura. U tom isječku XML datoteke prikazano je jedno pitanje i jedan odgovor. Pitanja i odgovori razlikuju se po vrijednosti parametra `PostTypeId`. Vrijednost parametra `PostTypeId` za pitanja je 1, a odgovori imaju vrijednost identifikatora 2.

Kako bi iz XML-a bili filtrirani odgovarajući podaci odnosno isječci kodovi pisani u programskom jeziku Java, napravljeno je sljedeće. Svaki je redak XML dokumenta serijaliziran u odgovarajuću Java klasu, `Question` - pitanje ili `Answer` - odgovor. Izvorni kod i popis bitnih parametara klasa `Question` i `Answer` vidljivi su u ispisima 3.1 i 3.2. Nadalje, ako redak sadrži `PostTypeId` parametar s vrijednosti 1, u tom se retku potom još dodatno gledalo sadrži li parametar `Tags` oznaku `java` među

svim navedenim oznakama dostupnim u `Tags` dijelu XML retka. Na taj su način serijalizirana sva pitanja koja sadrže `java` oznaku jer to za većinu odgovora označava da se u njima nalazi ponuđeno rješenje u obliku Java programskog koda. Svi redci koji sadrže `PostTypeId` parametar s vrijednosti 2 serijalizirani su u klasu `Answer`, bez dodatnih filtriranja. Svi serijalizirani podaci potom su spremljeni u lokalnu *PostgeSQL* bazu podataka radi lakšeg rukovanja.

Ispis 3.1: Izvorni kod klase `Question`

```
1  public class Question {
2
3    public int Id;
4    public int AcceptedAnswerId;
5    public String Tags;
6
7    public Question() {
8    }
9  }
```

Ispis 3.2: Izvorni kod klase `Answer`

```
1  public class Answer {
2
3    public int Id;
4    public String Body;
5    public String Code;
6
7    public Answer() {
8    }
9  }
```

Dohvat odgovarajućih *Stack Overflow* odgovora iz baze rađen je koristeći dva kriterija. Skoro svako pitanje na Stack Overflow-u ima više odgovora, a među njima jedan prihvaćeni odgovor kojeg je osoba koja je postavila pitanje označila kao rješenje njegova problema. S obzirom na to da prihvaćeni odgovori sadrže rješenje nečijeg problema, prvi kriterij za odabir odgovora je taj da odgovor mora biti prihvaćen. Pretpostavka je da ako je odgovor označen kao rješenje nečijeg problema, vjerojatnije je da će ta osoba ili bilo koja druga prije kopirati takav odgovor u svoj kod nego isprobavati ostale odgovore. Štoviše, ti su odgovori smješteni odmah ispod postavljenog pitanja pa su stoga uočljiviji od drugih što dovodi do nove pretpostavke kako će te odgovore posjetitelji prije kopirati i isprobati u svom kodu nego ostale. Sljedeći kriterij je duljina isječka koda. Programski se kod u odgovorima nalazi između `<code>...</code>` XML

oznaka. Duljina takvog isječka mora biti dulja od šest linija - šest linija uključivo, jer je to najmanja duljina fragmenta koda uobičajena pri otkrivanju klonova [7]. Drugim riječima, pri filtriranju java programskih kodova, iz dobivene baze su traženi odgovori čiji su identifikatori sadržani u `AcceptedAnswerId` XML poljima pitanja i čija je duljina kodova unutar `<code>` oznaka jednaka ili dulja od 6 linija. Iz svakog takvog pronađenog odgovora, fragmenti koda spremljeni su u posebne java datoteke. Iako su odgovori stajali uz pitanja koja u sebi sadrže `java` oznake u `Tags` poljima, i dalje su postojali fragmenti koda koji su bili dijelovi prihvaćenih odgovora, no nisu sadržavali Java kod. Takve su datoteke izbačene dodatnom provjerom pomoću regularnih izraza i nasumičnom ručnom provjerom. Regularni izrazi provjeravali su sljedeće uvjete :

- postojanje niza znakova `"var "`, `"def "`, `"std::"`, `"/>"`, `"</"` ili `"END;"`) i
- počinje li linija koda nizom znakova `"namespace "`, `"<h:"`, `"<p:"`, `"<f:"`, `"<s:"`, `"#include"`, `"dependencies {"`, `"<?xml version= "1.0"`, `"<!DOCTYPE html PUBLIC"`, `"#define"`, `"using "` ili `"<script type"`.

Datoteke koje su sadržavale neke od tih znakova, pregledane su ručno i po potrebi izbačene. S obzirom da je većina isječaka kodova sa *Stack Overflow*-a nepotpuno, odnosno fragmenti mogu sadržavati kod nepotpunih klasa ili nepotpunih dijelova metoda, za prepoznavanje Java kodova nisu korišteni alati koji na temelju kompajliranja određuju radi li se o validnom Java isječku nego su korištni regularni izrazi. Regularni izrazi provjeravali su postojanje sekvenci karakterističnih za druge programske jezike.

Konačno, dobiveno je 349,791 datoteka s Java fragmentima koji zajedno sadrže 6,044,193 linija Java izvornog koda (engl. LOC - Lines Of Code).

3.2. *GitHub*

Podaci s *GitHub*-a dobiveni su uz pomoć *GitHub* API-ja [2] odnosno korištenjem alata *github-clone-all*. *github-clone-all* je skripta pisana u programskom jeziku Go koja pretražuje repozitorije zadane određenim upitom pomoću službenog *GitHub*-ovog REST API-ja za pretraživanje i klonira pronađeni skup repozitorija odnosno *GitHub* projekata. Zbog ograničenja *GitHub* API-ja za pretraživanje, svakim je pozivom moguće dohvatiti maksimalno 1000 traženih podataka [9]. Također, alat vraća najpopularnije repozitorije koji zadovoljavaju neki upit. U ispisu 3.3 nalazi se primjer jednog poziva alata i predanog upita. Ovim će se pozivom dohvatiti 1000 projekata koji su

pisani programskim jezikom Java i koji sadrže manje od 1 zvjezdice, odnosno repozitoriji koji još nemaju ocjena - najmanje popularni projekti.

Ispis 3.3: Primjer upita za *github-clone-all* skriptu

```
./github-clone-all -extract '(\.java)$' 'language:java stars:<1'
```

S obzirom da će više istih poziva vratit gotovo iste podatke, odlučeno je napraviti nekoliko različitih upita za dohvat projekata. Tako su traženi općenito najpopularniji repozitoriji - projekti s najviše zvjezdica, repozitoriji s 0 zvjezdica odnosno najmanje popularni prokekti te projekti koji se nalaze između maksimalnog i minimalnog broja zvjezdica, a to su projekti primjerice s do 10 zvjezdica i projekti s maksimalno 1000 zvjezdica. U tablici 3.1 prikazane su veličine setova podataka odnosno broj projekata koji su dohvaćeni za svaki od navedenih upita.

ocjena - broj zvjezdica	broj projekata
najpopularniji repozitoriji	1410
0 zvjezdica	1060
>=10 zvjezdica	662
>=1000 zvjezdica	897

Tablica 3.1: Veličine setova podataka dohvaćenih s *GitHub*-a

Kao rezultat korištenja alata *github-clone-all*, ovom je metodom dohvaćeno ukupno 4029 javno dostupna Java projekta koji zajedno sadrže 1,221,907 Java datoteka i ukupno 104,721,555 linija koda. U tablici 3.2 zajedno su prikazane brojke vezane uz sve prikupljene podatke.

Sav programski kod koji je korišten za prikupljanje i pripremu podataka, dostupan je na *GitHub*-u u obliku javnog repozitorija [8].

set podataka	ukupan broj Java datoteka	ukupan broj linija koda
Stack Overflow	349,791	6,044,193
GitHub	1,221,907	104,721,555

Tablica 3.2: *Stack Overflow* i *GitHub* setovi podataka

4. Alat za detekciju klonova - *SourcererCC*

Unatoč desetljećima aktivnih istraživanja, primjetan je nedostatak alata za otkrivanje klonova koji se mogu primijeniti na velike (engl. *large-scale*) projekte s iznimno velikim brojem datoteka s izvornim programskim kodom i uz to da su primjenjivi za otkrivanje skorih duplikata odnosno na kodove u kojima su se dogodile značajne aktivnosti uređivanja i mijenjanja izvornog koda. Otkrivanje takvih klonova koristi se primjerice za otkrivanje sličnih mobilnih aplikacija i programskih projekata, otkrivanje kršenja licenci, reverzno inženjerstvo, pronalaženje podrijetla izvornih kodova i za brže pretraživanje kodova unutar velikih projekata. Također detekcija klonova tipa T3 omogućava znanstvenicima značajnije proučavanje kloniranja u velikim softverskim ekosustavima ili u razvojnim zajednicama otvorenog koda poput primjerice *GitHub*-a. Iz navedenih razloga i za takva istraživanja potrebni su alati koji su skalabilni i kada je riječ o stotinama milijuna linija izvornog programskog koda [10].

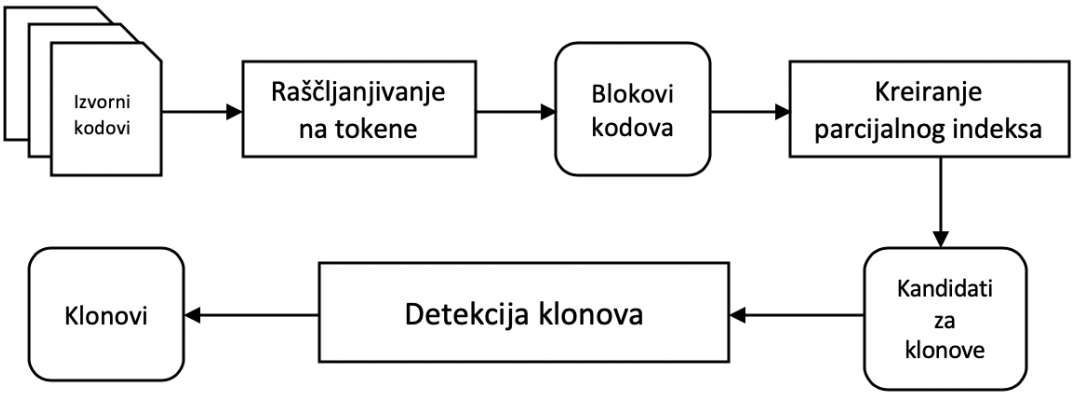
SourcererCC jedan je od rijetkih alata koji nema značajnijih problema sa skalabilnošću kada su u pitanju velike količine podataka i pri tome se može izvršavati na standardnom osobnom računalu [10]. *SourcererCC* je alat za otkrivanje klonova koda koji se temelji na korištenju tokena i radi neovisno o programskom jeziku. Također ima podršku za rad s više razina preciznosti, a to su :

- otkrivanje klonova na razini datoteka,
- na razini metoda,
- na razini linija ili
- na razini blokova koda.

Može otkriti klonove tipa T1, T2 pa sve do tipa T3 odnosno može otkriti klonove s dodanim, uklonjenim i/ili izmijenjenim linijama koda [7]. *SourcererCC* uspoređuje blokove kodova koristeći jednostavnu i brzu strategiju koje se naziva *strategija vreće tokena* (engl. *bag-of-tokens*) koja lako pronalazi klonove tipa T3. Kandidate za klo-

nove pronalazi metodom *parcijalnog invertiranog indeksa* (engl. *partial inverted index*). Kako bi smanjio veličinu indeksa, *SourcererCC* koristi heuristiku za filtriranje koja drastično smanjuje broj potrebnih blokova koji se uspoređuju pri otkrivanju klonova. Konačno, sve se to odražava na performanse alata, stoga *SourcererCC* ima veliku preciznost i odziv pa je konkurentan ostalim alatima za detekciju klonova [10].

Na slici 4.1 prikazan je pojednostavljeni proces detektiranja klonova i njegovi koraci, a u sljedećem potpoglavlju je detaljnije opisan algoritam rada *SourcererCC*-a. Prvi korak koji priprema podatke za proces detekcije klonova je proces raščlanjivanja kodova na skupa tokena. Taj je proces opisan u poglavlju 5.1 u sklopu obrade konkretnih podataka koji se koriste u ovom diplomskom radu.



Slika 4.1: SourcereCC

4.1. Algoritma za otkrivanje klonova

Algoritam za otkrivanje klonova radi u dvije osnovne faze :

- kreiranje parcijalnog indeksa i
- detekcija klonova.

4.1.1. Faza kreiranja parcijalnog indeksa

U fazi stvaranja parcijalnog indeksa, *SourcererCC* analizira blokove koda iz izvornih datoteka i raščlanjuje ih na tokene jednostavnim parserom koji prepoznaje semantiku blokova i tokena karakteristične za određeni programski jezika. Od blokova koda izrađuje invertirani indeks kojim preslikava tokene s određenim blokovima koda koji

ih sadrže. Za razliku od pristupa koji koriste neki drugi alati za detekciju klonova, *SourcererCC* ne stvara indeks od svih tokena koji se nalaze u kodnim blokovima, nego umjesto toga koristi heuristično filtriranje kako bi izgradio parcijalni indeks, ali ne koristeći sve tokene nego samo neke od podskupina tokena iz svakog bloka. Izgrađuje se indeks samo od podskupine tokena jer *SourcererCC* koristi svojstvo koje govori da se pri usporedbi dva skupa tokena ne trebaju uspoređivati svi tokeni iz skupova nego samo njihovi određeni podskupovi i ukoliko se ti podskupovi ne podudaraju, blokovi koda nisu klonovi. Ovo svojstvo ne samo da štedi memorijski prostor, već i omogućava brže pretraživanje zbog manjeg indeksa [10].

4.1.2. Detekcija klonova

U fazi otkrivanja klonova, *SourcererCC* iterira kroz sve blokove kodova te za svaki blok pronalazi njegove kandidate za klonove pomoću stvorenog indeksa. Sljedeći korak je heuristično filtriranje kojim se za ispitivanje indeksa kandidata koriste samo indeksi tokena unutar istog podbloka, što smanjuje broj ukupnih kandidata za klonove. Nakon što su kandidati pronađeni, *SourcererCC* koristi novo heuristično filtriranje koje se odnosni na poredak tokena unutar bloka koda i na taj se način mjere gornje i donje granice sličnosti između bloka koda i njegovog kandidata. Kandidati čija gornja granica pada ispod praga sličnosti odmah se uklanjaju bez daljnje obrade. Isto tako, kandidati se prihvaćaju ako je njihova donja granica veća ili jednaka zadanom pragu sličnosti. To se ponavlja sve dok se ne pronađu klonovi za svaki blok koda. *SourcererCC* također koristi svojstvo simetrije kako bi izbjegao otkrivanje istog klona dva ili više puta [10].

5. Obrada podataka

Postoji poprilično velik broj dostupnih alata za detekciju klonova koda. Za prikupljanje i usporedbu pronađenih informacija o alatima uloženo je puno vremena i uz to je bilo nemoguće isprobati sve ili barem većinu tih alata nad pripremljenim podacima kako bi se utvrdilo koji je od njih optimalan za rješavanje problema, stoga je za ovu studiju odabran samo jedan alat - *SourcererCC*. Također, postoji niz ograničenja u pogledu odabira alata za otkrivanje klonova. Glavno se ograničenje javlja zbog prirode isječaka kodova prikupljenih sa *Stack Overflow*-a iz razloga jer su većina njih nepotpuni fragmenti kodova, točnije nepotpune Java klase ili samo dijelovi metoda [7]. Dakle, detektor mora biti dovoljno fleksibilan za obradu isječaka kodova koji se ne mogu kompajlirati zato što nisu kompletni blokovi. Štoviše, budući da se količina podataka koja se mora obraditi nalazi u ljestvici od oko milijun linija koda, kao što je prikazano u tablici 3.2, alat za detekciju klonova mora biti dovoljno skalabilan odnosno mora uspjeti u razumnom - što kraćem vremenskom roku pronaći klonove.

Uzimajući u obzir sve navedene zahtjeve, za detekciju klonova odabran je alat *SourcererCC* i to najviše zbog velike brzine i zbog dobrih performansi prilikom obrade velike količine podataka. *SourcererCC* ima podršku za rad s više razina preciznosti, a to su otkrivanje klonova između datoteka, metoda, linija ili blokova, i to u bilo kojem programskom jeziku. U ovom radu koristi se preciznost otkrivanja klonova na razini blokova koda napisanih u programskom jeziku Java.

Za obradu podataka korišten je artefakt u obliku virtualnog stroja koji je dostupan na službenoj *GitHub* stranici *SourcererCC*-a [5]. Virtualni stroj sadrži sve potrebne upute i postavke koje su korisniku potrebne kako bi proveo detekciju klonova. Također, u virtualnom stroju su dostupna i dodatna objašnjenja svih potrebnih koraka u izvođenju procesa otkrivanja klonova. Konačno, stvaratelji *SourcererCC*-a preporučuju korištenje virtualnog stroja kao najlakšeg načina za analizu klonova koda.

Na kraju poglavlja prikazane su tablice koje sadrže neke od konfiguracijskih parametara korištenih pri obradi i usporedbi podataka. Nisu opisani svi mogući konfiguracijski parametri jer ih ima puno, a i nisu svi relevantni za usporedbu kakva je

provedena u ovom radu. U sljedećim potpoglavljima detaljnije će biti opisana svaka od korištenih konfiguracija. Jasno je kako različite konfiguracije imaju različite učinke na rezultate detekcije kloniranja i da je važno pronaći što optimalniju konfiguraciju. No s obzirom na ogromnu količinu skupova podataka te velik broj različitih kombinacija konfiguracijskih parametara i nemogućnost provjere svake od kombinacija, pri detekciji klonova korištena je standardna konfiguracija koju su tvorci alata inicijalno postavili u virtualnom stroju kao što je prikazano u tablicama 5.1, 5.2, 5.3.

5.1. Raščlanjivanje kodova na skup tokena

SourcererCC je detektor klonova koda koji se bazira na tokenizaciji izvornog koda što znači da izvorni kod prije početka usporedbe mora proći kroz početni korak obrade, a to je korak raščlanjivanja na skup tokena [5].

U tablici 5.1 vidljiva su prva dva parametra koja je potrebno podesiti, a odnose se na performanse detektora. Prvi parametar - *N_PROCESSES* odnosi se na broj željenih procesa koji će se koristiti za raščlanjivanje koda na tokene, a *PROJECTS_BATCH* označava veličinu skupa projekata koje će svaki proces obrađivati [5]). Sljedeći se konfiguracijski parametri odnose na postavke programskog jezika i oni su prikazani u tablici 5.2. *SourcererCC* uklanja sve komentare, razmake i praznine iz izvornih kodova kako bi se povećala točnost otkrivanja klonova [4]. Parametri *comment_inline*, *comment_open_tag*, *comment_close_tag* odnose se na znakove koji u ciljanom programskom jeziku označavaju početak odnosno kraj komentara. Konačno, *File_extensions* parametar označava ekstenziju datoteka s izvornim kodom za određeni programski jezik.

Rezultat koraka u kojem se izvorni kodovi raščlanjuju na tokene su sljedeće tri mape s podacima :

- *bookkeeping_projs*,
- *files_stats* i
- *files_tokens*.

Mapa *bookkeeping_projs* sadrži istoimenu datoteku s popisom obrađenih projekata. Svaka linija odnosno obrađeni projekt ima sljedeći format :

- identifikator projekta,
- putanja do projekta
- *URL* projekta.

Izgled konkretne datoteke dobivene za dva projekta korištena u ovom radu prikazan je na ispisu 5.1. Projekt s identifikatorom 1 je projekt u kojemu se nalaze svi kodovi sa *Stack Overflow*-a, dok projekt s identifikatorom 2 su svi kodovi prikupljeni s portala *GitHub*.

Ispis 5.1: Ispis obrađenih projekata

```
1,"/home/oopsla/Desktop/Diplomski␣podaci/StackOverflowFiles.zip",
"NULL"
2,"/home/oopsla/Desktop/Diplomski␣podaci/GitHubFiles.zip","NULL"
```

Mapa *files_stats* sadrži istoimenu datoteku s popisom različitih statistika. Svaka linija ima sljedeći format :

- identifikator projekta,
- identifikator datoteke unutar projekta,
- putanja do datoteke,
- *URL* projekta,
- sažetak datoteke,
- veličinu datoteke u bajtovima,
- ukupan broj linija u datoteci,
- broj linija koda bez komentara (engl. *Lines of code - LOC*) i
- broj linija koda bez komentara i praznina (engl. *Source Lines of code - SLOC*).

Prikaz jednog stvarnog retka iz datoteke koju sadrži mapa *files_stats* prikazan je na ispisu 5.2. Datoteka *week3_2.java* nalazi se u projektu s identifikatorom 2 odnosno u projektu koji sadrži kodove s *GitHub*-a i dio je javnog repozitorija *week3* od korisnika *sunyangg*.

Ispis 5.2: Statistike o obrađenim datotekama

```
  2,9,"/home/oopsla/Desktop/Diplomski␣podaci/
␣␣GitHubFiles.zip/GitHubFiles/sunyangg/week3/
␣␣week3\_2.java","NULL/epos/sunyangg/week3/week3\_2.java",
  "31a724c6f91deb9a4f2625172d60b1d7",1276,50,48,46
```

Mapa *files_tokens* sadrži istoimenu datoteku s popisom raznih statistika pronađenih tokena. Svaki redak ima sljedeći format :

- identifikator projekta,
- identifikator datoteke unutar projekta,

- ukupni broj tokena,
- ukupni broj jedinstvenih tokena,
- sažetak tokena,
- popis tokena i njihovih frekvencija pojavljivanja u obliku : @#@token1@@::@@ frekvencija pojavljivanja tokena1,token2@@::@@frekvencija pojavljivanja tokena2, ...

Primjer jednog takvog retka vidi se na ispisu 5.3, a na ispisu 5.4 je izvorna Java datoteka na temelju koje je nastao navedeni redak.

Ispis 5.3: Ispis statistika o tokenima

```
1,48,48,24,832eb6b7d232bf4a3d5221d155f11c1b@#@arr@@::@@6,
return@@::@@2,StringBuilder@@::@@2,System@@::@@1,else@@::@@1,
static@@::@@1,sb@@::@@6,null@@::@@1,twoDimArrToString@@::@@1,
append@@::@@4,if@@::@@1,obj@@::@@2,String@@::@@1,for@@::@@2,
Object@@::@@3,n@@::@@1,0@@::@@2,length@@::@@4,toString@@::@@1,
println@@::@@1,new@@::@@1,out@@::@@1,public@@::@@1,objArr@@::@@2
```

Ispis 5.4: Fragment koda korišten za raščlanjivanje na tokene

```
1  public static String twoDimArrToString(Object[][] arr) {
2      System.out.println(arr.length);
3      StringBuilder sb = null;
4      if (arr.length > 0) {
5          sb = new StringBuilder(arr.length * arr[0].length);
6      } else {
7          return "";
8      }
9
10     for (Object[] objArr : arr) {
11         sb.append("[");
12         for (Object obj : objArr) {
13             sb.append(obj).append(", ");
14         }
15         sb.append("]\n");
16     }
17     return sb.toString();
18 }
```

Identifikator projekta i identifikator datoteke uvijek upućuju na isti par projekta i datoteke izvornog koda - oni služe kao primarni ključ. Dakle, redak u datoteci

files_stats koji započinje s (1,1) predstavlja istu datoteku kao i redak u datoteci *files_tokens* koji započinje s (1,1), a identifikator projekta dolazi iz zapisa o projektima u datoteci *bookkeeping_projs* čiji redak počinje s identifikatorom 1. Broj redaka u *bookkeeping_projs* datoteci odgovara ukupnom broju analiziranih projekata, broj redaka u datoteci *files_stats* isti je kao i u datoteci *files_tokens* odgovara ukupnom broju datoteka dobivenih iz projekata [5].

Konačno, za svaki *GitHub* projekt pokrenuta je detekcija klonova na čitavom skupu *Stack Overflow* kodova.

5.2. Pronalazak klonova

Prije nego se pokrene proces pronalaska klonova, potrebno je postaviti parametar postotka sličnosti - *threshold*, koji je prikazan u tablici 5.3. Brojka 8 označava da će dva fragmenta koda biti klonovi ako se podudaraju u minimalno 80 posto linija, 7 označava postotak od 70 posto sličnosti, i tako dalje. Standardna postavka ovog parametra je 80 posto sličnosti i taj parametar nije mijenjan.

Rezultat odnosno dobivene informacije predstavljaju popis parova identifikatora datoteka koje su klonovi. Ti identifikatori odgovaraju identifikatorima dobivenim u fazi tokenizacije. Primjer izlaza prikazan je u ispisu 5.5. U ovom slučaju otkriveni klonovi su primjerice (2,350638) i (1,244904). Da bismo znali koje su to datoteke, potrebno je pogledati datoteku *files_stats* u istoimenoj mapi i potražiti retke s jedinstvenim identifikatorima 1,244904 i 2,350638.

Ispis 5.5: Popis parova klonova koda

```
2,350638,1,244904
2,350641,1,244904
2,350645,1,244904
2,352069,1,205981
2,354392,1,244904
2,356109,1,184788
2,356109,1,117433
2,356120,1,184788
2,356120,1,343103
2,356120,1,157054
```

Primjer jednog para klonova koje je detektirao alat *SourcererCC* priloženi su u dodatku B i dodatku C.

Naposljetku, analiziran je izvještaj detekcije klonova i nasumično su ručno pregle-

dani neki od prijavljenih klonova kako bi se provjerila njihova ispravnost.

5.3. Konfiguracijski parametri

naziv parametra	vrijednost
N_PROCESSES	1
PROJECTS_BATCH	2

Tablica 5.1: Parametri za određivanje performansi

naziv parametra	vrijednost
comment_inline	//
comment_open_tag	/*
comment_close_tag	*/
File_extensions	.java

Tablica 5.2: Parametri postavke programskog jezika

naziv parametra	vrijednost
threshold	"$3:-8"

Tablica 5.3: Parametar za postavljanje praga sličnosti

6. Rezultati

Rezultati ovog istraživanja pokazuju sljedeće. Od ukupno 1,221,907 Java datoteka prikupljenih s *GitHub*-a, njih svega 27,745 imaju klona među 349,791 Java datoteka prikupljenih sa *Stack Overflow*-a. Drugim riječima, u 2,27 posto datoteka s *GitHub*-a pronađeno je podudaranje od 80 posto ili više s nekim od fragmenata Java koda objavljenih na *Stack Overflow*-u, odnosno, detektirano je kopiranje programskog koda. Na *GitHub*-u je dostupno cca. 106 tisuća javnih repozitorija koji sadrže Java projekte. Zbog ograničenog memorijskog prostora računala korištenog za izradu ovog istraživanja, s *GitHub*-a je preuzeto i obrađeno 4029 Java projekta što čini oko 4 posto ukupnog broja dostupnih repozitorija.

Pretpostavka je da postotak klonova uvelike ovisi o odabranim setovima podataka za istraživanje te da je moguće da bi neki drugi set podataka prikupljen s *GitHub*-a rezultirao većim postotkom klonova između *Stack Overflow*-a i *GitHub*-a. To potvrđuju različita istraživanja koja se bave pronalaskom klonova za projekte pisane u programskom jeziku Java. Postotak otkrivenih klonova ovisi o setovima podataka i varira između 5.2 pa čak do 46 posto [4].

7. Zaključak

U uvodu rada iznesena je pretpostavka da programeri za vrijeme svoga rada kopiraju tuđe, odnosno već napisane fragmente programskog koda u svoje aplikacije i projekte te ih potom u određenoj mjeri izmjenjuju i prilagođavaju svojim potrebama. Cilj ovog rada bio je istražiti u kojoj se mjeri Java kodovi kloniraju između *Stack Overflow*-a i *GitHub*-a. Za svrhu istraživanja prikupljeno je sveukupno 1,571,698 Java datoteka s obje platforme zajedno.

Rezultati pokazuju kako je dana pretpostavka netočna za preuzeti skup podataka s platforme *GitHub*, odnosno klonovi kodova detektirani su u svega 2,27 posto obrađenih Java kodova sa *Stack Overflow*-a i *GitHub*-a. Pronađeno je ukupno 27,745 klonova među 1,221,907 Java datoteka prikupljenih s *GitHub*-a i 349,791 Java datoteka prikupljenih sa *Stack Overflow*-a. S *GitHub*-a je uzet uzorak koji sadrži 4029 Java projekata od čega je 1410 projekata s najvećom ocjenom, 1060 projekata s najnižom ocjenom, 662 projekata s manje od 10 zvjezdice te 897 projekata s manje od 1000 zvjezdica.

Ovaj diplomski rad ostavlja pretpostavku da bi rezultati potencijalno mogli biti bolji za neki drugačiji skup podataka preuzetih s portala *GitHub* od onoga koji se koristi u ovom istraživanju, s obzirom da je s *GitHub*-a obrađeno samo oko 4 posto ukupnog broja javno dostupnih javinih projekata. Pretpostavka se temelji na nekoliko sličnih znanstvenih radova koji su korišteni kao literatura za ovaj rad, a njihovi rezultati odnosno utvrđeni postotak kloniranja varira između 5.2 i 46 posto.

LITERATURA

[1] Internet Archive. Stack exchange data dump. URL https://archive.org/details/stackexchange.

[2] GitHub Developer. Github rest api v3. URL https://developer.github.com/v3/.

[3] GitHub. Github. URL https://github.com/.

[4] Cristina Lopes, Petr Maj, Pedro Martins, Vaibhav Saini, Di Yang, Jakub Zitny, Hitesh Sajnani, i Jan Vitek. Déjàvu: a map of code duplicates on github. *Proceedings of the ACM on Programming Languages*, 1:1–28, 10 2017. doi: 10.1145/3133908.

[5] Mondego. Sourcerercc tutorial. URL https://github.com/Mondego/SourcererCC.

[6] Stack Overflow. Stack overflow. URL https://stackoverflow.com/.

[7] Chaiyong Ragkhitwetsagul, Jens Krinke, Matheus Paixao, Giuseppe Bianco, i Rocco Oliveto. Toxic code snippets on stack overflow. *IEEE Transactions on Software Engineering*, PP:1–1, 02 2019. doi: 10.1109/TSE.2019.2900307.

[8] Lea Rački. Diplomski rad. URL https://github.com/rackilea/diplomski.

[9] rhysd. Clone matching repos on github. URL https://github.com/rhysd/github-clone-all.

[10] H. Sajnani, V. Saini, J. Svajlenko, C. K. Roy, i C. V. Lopes. Sourcerercc: Scaling code clone detection to big-code. U *2016 IEEE/ACM 38th International Conference on Software Engineering (ICSE)*, stranice 1157–1168, 2016.

[11] Margaret-Anne Storey, Christoph Treude, Arie Deursen, i Li-Te Cheng. The impact of social media on software engineering practices and tools. stranice 359–364, 01 2010. doi: 10.1145/1882362.1882435.

[12] B. Vasilescu, V. Filkov, i A. Serebrenik. Stackoverflow and github: Associations between software development and crowdsourced knowledge. U *2013 International Conference on Social Computing*, stranice 188–195, 2013.

[13] Wikipedia. Duplicate code, 2020. URL `https://en.wikipedia.org/wiki/Duplicate_code`.

[14] Wikipedia. Github, 2020. URL `https://en.wikipedia.org/wiki/GitHub`.

POPIS SLIKA

4.1. SourcereCC 15

POPIS TABLICA

3.1. Veličine setova podataka dohvaćenih s *GitHub*-a 13
3.2. *Stack Overflow* i *GitHub* setovi podataka 13

5.1. Parametri za određivanje performansi 22
5.2. Parametri postavke programskog jezika 22
5.3. Parametar za postavljanje praga sličnosti 22

POPIS ISPISA

2.1. Originalni fragment koda 4
2.2. Klonirani fragment koda tipa T1 4
2.3. Originalni fragment koda 5
2.4. Klonirani fragment koda tipa T2 6
2.5. Originalni fragment koda 6
2.6. Klonirani fragment koda tipa T3 7
2.7. Originalni fragment koda 8
2.8. Klonirani fragment koda tipa T4 8
3.1. Izvorni kod klase `Question` 11
3.2. Izvorni kod klase `Answer` 11
3.3. Primjer upita za *github-clone-all* skriptu 13
5.1. Ispis obrađenih projekata 19
5.2. Statistike o obrađenim datotekama 19
5.3. Ispis statistika o tokenima 20
5.4. Fragment koda korišten za raščlanjivanje na tokene 20
5.5. Popis parova klonova koda 21

Detekcija fragmenata koda raspoloživih na portalu Stack Overflow u aplikacijama na GitHubu

Sažetak

Programeri koriste različite portale kao pomoć u svom radu te za vrijeme razvoja programa i njihovog održavanja recikliraju već napisane dijelove programskih komponenata, isječke kodova i programske knjižice. Jedni od najpopularnijih takvih internetskih portala su *Stack Overflow* i *GitHub*. *Stack Overflow* je forum koji se temelji na konceptu pitanja i odgovora, a programeri ga nekad čak preferiraju više od službenih dokumentacija i knjiga kao resursa za traženje programskog rješenja za neki problem. *GitHub* je jedan od najvećih internetskih portala za pohranu programskog koda.

U ovom je radu istražena pretpostavka da programeri za vrijeme svoga rada kopiraju već napisane fragmente programskog koda u svoje aplikacije koje zatim u određenoj mjeri izmjenjuju i prilagođavaju svojim potrebama. Proces korištenja gotovog koda koji uključuje ponovnu upotrebu odnosno kopiranje izvornog koda naziva se kloniranje koda. Uz to se istražuje koliko je kloniranje kodova često i na koji se način mogu detektirati klonirani fragmenti u izvornim kodovima te u kojoj se mjeri isječci kodova dostupni na *Stack Overflow* portalu kopiraju u javno dostupnim projektima na *GitHub*-u.

Prikupljeno je ukupno 1,221,907 Java datoteka s *GitHub*-a i 349,791 Java datoteka sa portala *Stack Overflow*. Podaci su obrađeni alatom *SourcererCC*. *SourcererCC* je alat za otkrivanje klonova koda koji se temelji na korištenju tokena i može otkrivati klonove u bilo kojem programskom jeziku. Može otkriti klonove tipa T1, T2 pa sve do tipa T3 odnosno može otkriti klonove s dodanim, uklonjenim i/ili izmijenjenim linijama koda, a uz to ima veliku preciznost i odziv pa je konkurentan ostalim alatima za detekciju klonova.

U 2,27 posto datoteka s *GitHub*-a pronađeno je podudaranje od 80 posto ili više s nekim od fragmenata Java koda objavljenih na *Stack Overflow*-u, odnosno u njima je detektirano kopiranje izvornog koda. Postotak klonova uvelike ovisi o odabranim setovima podataka za istraživanje te da je moguće da bi neki drugi set podataka prikupljen s *GitHub*-a rezultirao većim postotkom klonova između *Stack Overflow*-a i *GitHub*-a.

Ključne riječi: Stack Overflow, GitHub, Java, kloniranje kodova, SourcererCC, klonovi tipa T1, klonovi tipa T2, klonovi tipa T3, otkrivanje klonova koda.

Detecting code fragments available on Stack Overflow in applications on GitHub

Abstract

Developers use various portals to help with their work, and during program development and maintenance they recycle parts of program components, code snippets, and frameworks. One of the most popular such web portals is Stack Overflow and GitHub. Stack Overflow is a forum-based online portal with the concept of questions and answers, and developers sometimes even prefer it more than official documentation and books as a resource for finding a software solution to a problem. GitHub is one of the largest Internet portals for storing program code.

This paper investigates the assumption that developers copy fragments of program code into their applications during their work, which is then modified and adapted to their needs. The process which involves reusing or copying source code is called code cloning. In addition, this paper investigates how common code cloning is and how cloned fragments can be detected in source code and the extent to which code snippets available on the Stack Overflow portal are copied in publicly available projects on GitHub.

A total of 1,221,907 Java files from GitHub and 349,791 Java files from the Stack Overflow were collected. The data was processed with the SourcererCC tool. SourcererC is a token-based code clone detection tool that can detect clones in any programming language. It can detect clones of type-1, type-2 up to type-3, as well as it can detect clones with added, removed and/or modified lines of code, and in addition, it has high precision and response, so it is competitive with other clone detection tools.

2.27 percent of files from GitHub were found to match 80 percent or more of some of the Java code snippets published on Stack Overflow, and source code copying was detected in them. The percentage of clones largely depends on the data sets selected for research, and it is possible that some other data set collected from GitHub would result in a higher percentage of clones between Stack Overflow and GitHub.

Keywords: Stack Overflow, GitHub, Java, code clones, SourcererCC, clone type-1, clone type-2, clone type-3, code cloning detection.

Dodatak A
Posts.xml

```
<posts>

  ...

  <row
    Id="9"
    PostTypeId="1"
    AcceptedAnswerId="1404" CreationDate="2008-07-31T23:40:59.743"
    Score="1819"
    ViewCount="594316"
    Body="&lt;p&gt;Given a &lt;code&gt;DateTime&lt;/code&gt;representing a
    person’s birthday, how do I calculate their age in years?&lt;/p&gt;&#xA;"
    OwnerUserId="1"
    LastEditorUserId="3956566"
    LastEditorDisplayName="Rich B"
    LastEditDate="2018-04-21T17:48:14.477"
    LastActivityDate="2020-02-29T01:31:43.697"
    Title="How do I calculate someone’s age in C#?"
    Tags="&lt;c#&gt;&lt;.net&gt;&lt;datetime&gt;"
    AnswerCount="61"
    CommentCount="7"
    FavoriteCount="450"
    CommunityOwnedDate="2011-08-16T19:40:43.080" />
  <row
    Id="1404"
    PostTypeId="2"
    ParentId="9"
    CreationDate="2008-08-04T16:50:06.170"
    Score="2075"
    Body="&lt;p&gt;An easy to understand and simple solution.
    &lt;/p&gt;&#xA;&#xA;&lt;pre class=&quot;lang-cs prettyprint-
```

```
    override&amp;quot;&amp;gt;&amp;lt;code&amp;gt;// Save today's date.&amp;#xA;
    var today = DateTime.Today;&amp;#xA;// Calculate the age.&amp;#xA;
    var age = today.Year - birthdate.Year;&amp;#xA;
    // Go back to the year the person was born in case of a
    leap year&amp;#xA;if (birthdate.Date &amp;amp;gt; today.AddYears(-age))
    age--;&amp;#xA;&amp;lt;/code&amp;gt;&amp;lt;/pre&amp;gt;&amp;#xA;&amp;#xA;&amp;lt;p&amp;gt;
    However, this assumes you are looking for the
    &amp;lt;em&amp;gt;western&amp;lt;/em&amp;gt; idea of age and not using &amp;lt;
    ahref=&amp;quot;https://en.wikipedia.org/wiki/East_Asian_age_reckoning
    &amp;quot;rel=&amp;quot;noreferrer&amp;quot;&amp;gt;&amp;lt;em&amp;gt;East Asian reckoning&amp;lt;
    /em&amp;gt;&amp;lt;/a&amp;gt;.&amp;lt;/p&amp;gt;&amp;#xA;"
    OwnerUserId="212"
    LastEditorUserId="5407188"
    LastEditorDisplayName="GateKiller"
    LastEditDate="2019-12-24T08:44:24.803"
    LastActivityDate="2019-12-24T08:44:24.803"
    CommentCount="34"
    CommunityOwnedDate="2011-08-16T19:40:43.080" />

  ...

</posts>
```

Dodatak B

ArcSSLSocketFactory.java - *GitHub*

```
1 public class ArcSSLSocketFactory extends SSLSocketFactory {
2     SSLContext sslContext = SSLContext.getInstance("TLS");
3
4     public ArcSSLSocketFactory() throws KeyManagementException,
5                     UnrecoverableKeyException, NoSuchAlgorithmException,
6                     KeyStoreException {
7         super(null);
8         init();
9     }
10
11    public ArcSSLSocketFactory(KeyStore truststore) throws
12                    NoSuchAlgorithmException, KeyManagementException,
13                    KeyStoreException, UnrecoverableKeyException {
14        super(truststore);
15        init();
16    }
17
18    protected void init() throws KeyManagementException {
19     TrustManager tm = new X509TrustManager() {
20            public void checkClientTrusted(X509Certificate[] chain,
21                                           String authType)
22                    throws CertificateException {
23            }
24
25            public void checkServerTrusted(X509Certificate[] chain,
26                                           String authType)
27                    throws CertificateException {
28            }
29
30            public X509Certificate[] getAcceptedIssuers() {
31                return null;
```

```
32              }
33          };
34          sslContext.init(null, new TrustManager[] { tm }, null);
35      }
36
37      public ArcSSLSocketFactory(SSLContext context)
38                  throws KeyManagementException, NoSuchAlgorithmException,
39                  KeyStoreException, UnrecoverableKeyException {
40         super(null);
41         sslContext = context;
42      }
43
44      @Override
45      public Socket createSocket(Socket socket, String host, int port,
46                                  boolean autoClose)
47                  throws IOException, UnknownHostException {
48          return sslContext.getSocketFactory()
49                           .createSocket(socket, host, port, autoClose);
50      }
51
52      @Override
53      public Socket createSocket() throws IOException {
54          return sslContext.getSocketFactory().createSocket();
55      }
56  }
```

Dodatak C

fragment_395433.java - *Stack Overflow*

```
1 public class MySSLSocketFactory extends SSLSocketFactory {
2     SSLContext sslContext = SSLContext.getInstance("TLS");
3
4     public MySSLSocketFactory(KeyStore truststore)
5                     throws NoSuchAlgorithmException, KeyManagementException,
6                     KeyStoreException, UnrecoverableKeyException {
7         super(truststore);
8
9         TrustManager tm = new X509TrustManager() {
10            public void checkClientTrusted(X509Certificate[] chain,
11                                           String authType)
12                    throws CertificateException {
13            }
14
15            public void checkServerTrusted(X509Certificate[] chain,
16                                           String authType)
17                    throws CertificateException {
18            }
19
20            public X509Certificate[] getAcceptedIssuers() {
21                return null;
22            }
23        };
24
25        sslContext.init(null, new TrustManager[] { tm }, null);
26    }
27
28    @Override
29    public Socket createSocket(Socket socket, String host, int port,
30                               boolean autoClose) throws IOException,
31                    UnknownHostException {
```

```
32          return sslContext.getSocketFactory()
33                              .createSocket(socket, host, port, autoClose);
34      }
35
36      @Override
37      public Socket createSocket() throws IOException {
38          return sslContext.getSocketFactory().createSocket();
39      }
40  }
```