

Algoritmi za detekciju plagijarizma

Seminarski rad u okviru kursa
Metodologija stručnog i naučnog rada
Matematički fakultet

Nemanja Subotić, Igor Rodić
suboticnemanja93@gmail.com, igorrodic@gmail.com

9. april 2016.

Sažetak

Sa pojavom modernih tehnologija, pre svega interneta, svet je postao u velikoj meri izložen raznim oblicima zloupotreba. Jednu od tih zloupotreba predstavljaju plagijarizmi. Zbog ogromne količine podataka koja je dostupna velikoj većini ljudske civilizacije, lakše je nego ikada naći potrebne informacije o bilo čemu što nas zanima, ali to sa sobom nosi i rizike. Jer skoro sa istom lakoćom bilo ko može te informacije da iskopira (preformuliše, ukrade ideju) i prezentuje kao svoje delo. Prirodno je potražiti odgovor na ovo pitanje u kompjuterskim algoritmima, jer bi taj zadatak za čoveka bio previše obiman. Shodno tome, razvijeni su mnogi algoritmi za detekciju raznih vrsta plagijarizama, koji imaju za cilj automatizovanje procesa otkrivanja plagijata. Njima ćemo se detaljnije baviti u ovom radu.

Sadržaj

1	Pomoc	2
2	Uvod	2
3	Ideje i pristupi	2
4	Sakrivanje plagijarizma	3
4.1	Tekst	3
4.2	Programski kod	3
5	Implementacija algoritma za detekciju plagijarizama tekst	3
6	Implementacija algoritma za detekciju plagijarizama programskog koda	3
7	Zaključak	4
	Literatura	4
A	Dodatak	4

1 Pomoc

Ovde pišem uvodni tekst. Ovde pišem uvodni tekst. Ovde pišem uvodni tekst. Ovde pišem uvodni tekst.

Primer 1.1 *I tabele treba da budu u svom okruženju, i na njih je neophodno referisati se u tekstu. Na primer, u tabeli 1 su prikazana različita poravnanja u tabelama.*

Tabela 1: Različita poravnanja u okviru iste tabele ne treba koristiti jer su nepregledna.

centralno poravnanje	levo poravnanje	desno poravnanje
a	b	c
d	e	f

2 Uvod

U ovom radu, pored detekcije plagijarizama teksta, bavićemo se i detekcijom plagijarizama programskog koda, koji je za nas informatičare posebno interesantan.

3 Ideje i pristupi

Prva stvar o kojoj treba diskutovati jeste klasifikacija sistema za detekciju plagijarizama. Ona je, prirodno, za svakoga drugačija. Osnovna podela, prema Mozgovoy-u deli pomenute sisteme na dve podgrupe, na sisteme koji prave "otisak prsta" (eng. *fingerprint*) programa i na sisteme koji porede sadržaj.

"Otiska prsta" programa predstavlja kratku sekvencu bajtova koja karakterizuje duži fajl. On može da se dobije primenom heš (eng. *hash*) funkcije na fajl, ali obično se koristi niz numeričkih atributa (prosečan broj reči po liniji, prosečan broj ključnih reči itd.), i zatim se pomoću funkcije razzdaljne računa stepen različitosti dva fajla. Ova tehnika se danas retko koristi jer čak i male promene mogu da rezultuju potpuno drugačijim "otiscima".

Poređenje sadržaja predstavlja osnovu za veliku većinu sistema za detekciju plagijarizama. Ono je generalno zasnovano na Manberovoj definiciji sličnosti i deli se na dve podvrste. One su poredjenje stringova i drveća parsiranja.

Definicija 3.1 *Manberova definiciji sličnosti: Kažemo da su dva fajla slična ako sadrže značajan broj istih podniski koje nisu prekratke.*

Algoritmi za poredjenje sadržaja funkcionišu po istom principu:

```
FOR EACH collection file F
  FOR EACH collection file G, F  $\neq$  G
    Calculate similarity between F and G
```

dok se funkcija za određivanje sličnosti menja. Kada pričamo o poređenju stringova pod time podrazumevamo poredjenje fajlova, koji se tretiraju kao veliki stringovi. Logično, ovaj način poredjenja ne čuva strukturu fajlova, kao ni programskog koda. Algoritmi koji koriste ovaj način detekcije su: FPDS, koji računa sličnost po formuli:

$$\text{sim}(F, G) = \text{MatchedTokens}(F, G) / \text{TotalTokens}(G)$$

YAP (jedan od prvih algoritama ove vrste, koristi poređenje liniju-po-liniju Levenštajnovim rastojanjem), RKR-GST (pravi pokrivač nepreklapajućih stringova koji sadrže maksimalan mogući broj tokena iz oba fajla, algoritam je NP kompletan pa zavisi od uspešnih heuristika, npr. duže podniske su vrednije od kraćih). Drveta parsiranja sa druge strane očuvavaju strukturu fajla, kako tekstualnog (poglavlja, pasusi itd.) tako i programskog (klase, funkcije itd.). Prednosti ovog metoda poređenja još nisu u potpunosti iskorišćene, jedan od algoritama koji to pokušava je Sim utility. On koristi poređenje stringova, ali ne nad celim fajlovima, već nad njihovim drvetima parsiranja, tj. parser prethodi algoritmu poređenja stringova.

4 Sakrivanje plagijarizma

4.1 Tekst

4.2 Programski kod

Kao što je ranije navedeno, detektovanje plagijarizama programskog koda je oblast koja nas informatičare posebno intrigira. Svako od nas je, u jednom trenutku, svesno ili nesvesno, plagirao tuđ kod. Provera plagijarizma programskog koda je nešto što profesori iz oblasti informatike rade svakodnevno, ali još se nije došlo do precizne definicije šta to tačno čini plagijarizam programskog koda. Plagijarizam programskog koda javlja se kada je kod kopiran i/ili izmenjen bez prethodne konsultacije sa autorom. [2]

Definisani su i nivoi modifikacije programskog koda, koji mogu da pomognu bližem razumevanju pojma, tj. da daju uvid u načine na koje plagijarizam programskog koda može da bude sakriven:

- Izmena komentara
- Izmena belina
- Preimenovanje identifikatora
- Izmena rasporeda blokova koda
- Izmena rasporeda naredbi unutar blokova
- Izmena redosleda operatora/operanada u izrazima
- Izmena tipova podataka
- Dodavanje redundantnih naredbi ili promenljivih
- Menjanje naredbi kontrole toka ekvivalentnim naredbama kontrole toka (while u do-while itd.)
- Menjanje poziva funkcije sa telom funkcije

Navedene modifikacije prevazilaze se pretprocesiranjem, konkretno tokenizacijom i parametrizovanim poklapanjem. Tokenizacija pretvara programski kod u niz tokena, time ga spuštajući na gradivni nivo (predstavljajući promenljive, pozive funkcija, naredbe itd.) i eliminišući većinu načina skrivanja. Složenost procesa tokenizacije je linearna, stoga ona ne utiče na kompleksnost finalnog algoritma, međjutim, samim procesom mogu da se izgube važne informacije o sličnostima dva programa. U cilju

očuvanja te sličnosti tokenizator se koristi zajedno sa parametrizovanim poklapanjem. Parametrizovano poklapanje smatra dva koda identičnim ako jedan može da se dobije iz drugog primenom konačnog broja zamenja identifikatora.

5 Implementacija algoritma za detekciju plagijarizama tekst

Stopword n-grams.

6 Implementacija algoritma za detekciju plagijarizama programskog koda

Diff.

7 Zaključak

Zaključak.

Literatura

- [1] Georgina Cosma. *An Approach to Source-Code Plagiarism Detection and Investigation Using Latent Semantic Analysis*. University of Warwick, Department of Computer Science, Coventry, UK, 2008.
- [2] A. MacLeod F. Culwin and T. Lancaster. *Source code plagiarism in U.K H.E computing schools, issues, attitudes and tools*. Technical Report SBU-CISM-01-02, South Bank University, London, 2001.
- [3] M. Luck M. Joy. *Plagiarism in Programming Assignments*. IEEE Transactions on Education, 1999.
- [4] M. Mozgovoy. *Enhancing Computer-Aided Plagiarism Detection*. Department of Computer Science, University of Joensuu, Joensuu, Finland, 2007.

A Dodatak

Pored ovog rada, u prilogu se nalaze i izvorni kodovi algoritama koji su implementirani.