

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 3967

**SUSTAV ZA DETEKCIJU PLAGIJARIZMA
IZVORNOG KODA**

Dino Rakipović

Zagreb, lipanj 2015

Sadržaj

1. Uvod.....	1
2. Pregled postojećih sustava za detekciju plagijata	3
2.1. Sustavi za detekciju plagijata u tekstualnim datotekama	3
2.1.1. Turnitin.....	3
2.1.2. Plagium.....	4
2.1.3. Dupli Checker	5
2.2. Sustavi za detekciju plagijata izvornog koda.....	6
2.2.1. MOSS	8
3. Arhitektura sustava.....	10
4. Korisničko sučelje i upute za korištenje	15
5. Primjeri rada Plagijatora	18
5.1. Prvi primjer.....	18
5.2. Drugi primjer	19
5.3. Treći primjer	20
6. Korištene tehnologije i alati.....	22
6.1. Python.....	22
6.2. Flask	22
6.3. HTML	23
6.4. Bootstrap CSS	23
7. Zaključak	25
8. Literatura	26
9. Sažetak	27

1. Uvod

Internet je najveći javni repozitorij informacija ikada napravljen. Većina tih informacija je dostupna na više od jednog mjesta, često traženu informaciju možemo naći na desetak skoro istih web stranica.

Velika većina dokumenta objavljenih na internetu su kopije ili plagijati drugih dokumenata [1]. Kako plagijati većinom nisu identični originalnom dokumentu, koristeći jednostavne metode usporedbe teško možemo otkriti što je točno plagijat. Plagijate možemo naći u akademskoj i ne-akademskoj zajednici [2]. U ne-akademskoj zajednici najbolji primjer plagiranja tuđeg rada je prepisivanje novinskih članaka. U akademskoj zajednici najčešći primjeri plagiranja su eseji ili referati, ali možemo ih naći gotovo svugdje, uključujući u znanstvenim radovima ili izvornim kodovima.

Cilj ovog rada je detektirati plagijate među izvornim kodovima. Detekcija plagijata se može obavljati ručno ili može biti potpomognuta nekakvim razvijenim programom. U postupku detekcije plagijata ključnu ulogu ima čovjek. Ručna detekcija zahtjeva puno vremena i nepraktična je ako imamo veliku količinu radova. U tu svrhu izrađena je web aplikacija nazvana Plagijator, koja će nam uvelike pomoći u detekciji plagijata. Aplikacija prima listu izvornih kodova, računa pripadne sličnosti među parovima te nam ih vraća. Algoritam koji aplikacija koristi je linearne vremenske složenosti što nam omogućava provjeru jako velikog broja izvornih kodova u malo vremena. Također u ovom koraku aplikacija otpisuje izvorne kodove za koje misli da imaju premalu sličnost, te tako ubrzava postupak detekcije. Plagijati su, kao što je spomenuto, većinom modifikacije nekog originalnog dokumenta, naša aplikacija takve pokušaje vrlo dobro pronalazi i detektira. Svaki vraćeni par izvornih kodova moguće je i pregledati, a aplikacija nam tu uvelike pomaže tako da istim bojama boja linije za koje je otkrila da su slične u oba izvorna koda što uvelike pomaže u brzini i otkrivanju plagijata od strane čovjeka.

U drugom poglavlju dan je pregled postojećih sustava za detekciju plagijata, koji su podijeljeni u dvije skupine, sustave koji detektiraju sličnosti tekstualnih datoteka i sustave koji detektiraju sličnost izvornih kodova. U trećem poglavlju je opisan sustav za detekciju sličnosti izvornih kodova koji je razvijen. U četvrtom poglavlju analizirano je korisničko sučelje te su dane korisničke upute. Peto poglavlje sadrži nekoliko pokušaja plagiranja izvornog koda te izlaze sustava za iste. U šestom su poglavlju opisane tehnologije pomoću kojih je sustav implementiran. Na kraju rada slijedi zaključak te kratki sadržaj.

2. Pregled postojećih sustava za detekciju plagijata

Sustavi za detekciju plagijata uglavnom se dijele u dvije osnovne skupine, to su sustavi za detekciju plagijata u tekstualnim datotekama i sustavi za detekciju plagijata izvornih kodova te ćemo u nastavku poglavlja opisati i upoznati najpoznatije postojeće sustave.

2.1. Sustavi za detekciju plagijata u tekstualnim datotekama

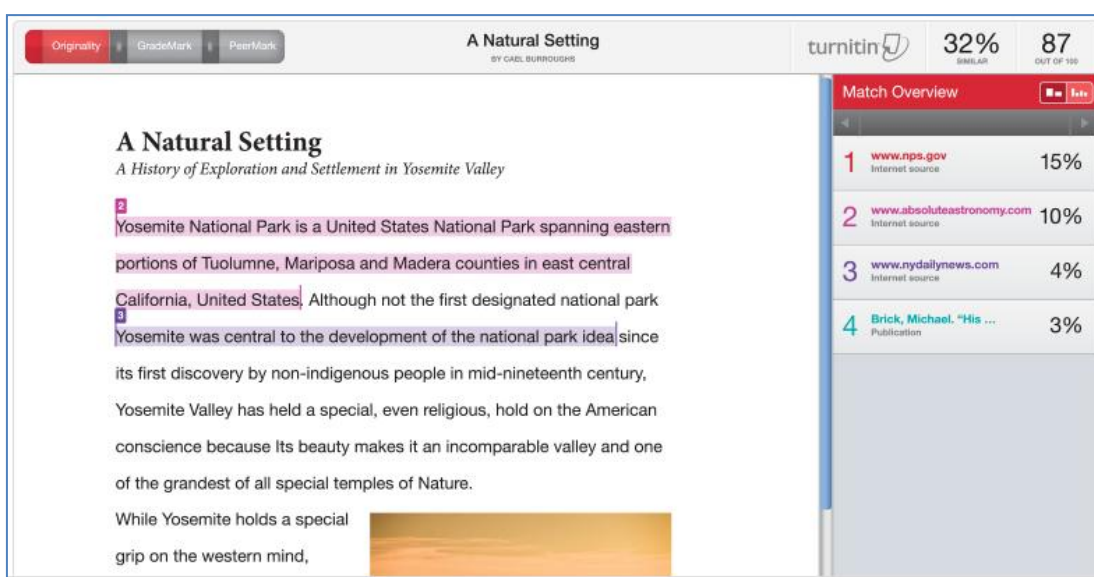
Sustavi specijalizirani za tekstualne datoteke uglavnom implementiraju jedan od dva generička pristupa detekcije, jedan je vanjski pristup dok je drugi unutarnji pristup. Sustavi s vanjskim pristupom uspoređuju sumnjive dokumente s referentnom zbirkom. Referentna zbirka je skup dokumenata za koje možemo reći da su izvorni (originalni). Zadatak sustava je korisniku vratiti sve dokumente koji su slični sumnjivom dokumentu u mjeri koja ja preko neke određene mjere. Tu mjeru određuje tvorac sustava i najlakše ju je eksperimentalno odrediti. Sustavi s unutarnjim pristupom ne uspoređuju dokumente s nekakvim vanjskim referentnim zbirkama već uspoređuju dokumente koje dobiju od korisnika te vraćaju sličnosti među njima.

U nastavku potpoglavlja upoznat ćemo neke od najpoznatijih i najkorištenijih sustava specijaliziranih za detektiranje plagijata među tekstualnim datotekama. Pod tekstualne datoteke spadaju eseji, referati, znanstveni radovi, itd.

2.1.1. Turnitin

Turnitin.com je web stranica na kojoj je moguće provjeriti originalnost nečijeg rada [5]. Specijalizirana je za provjere učeničkih i studentskih radova. Pomoću vjerojatno najveće baze akademskih radova možemo s velikom sigurnošću dobiti što je plagijat, a što ne. Stranica ima sjajan izgled i jako puno mogućnosti koji uvelike pomažu korisnicima. Korisnik dobije listu svih poslanih radova zajedno s postotkom sličnosti tog rada s radovima u bazi. Korisnik zatim možemo dobiti detaljan prikaz svakog poslanog rada. U sredini stranice je originalan rad s

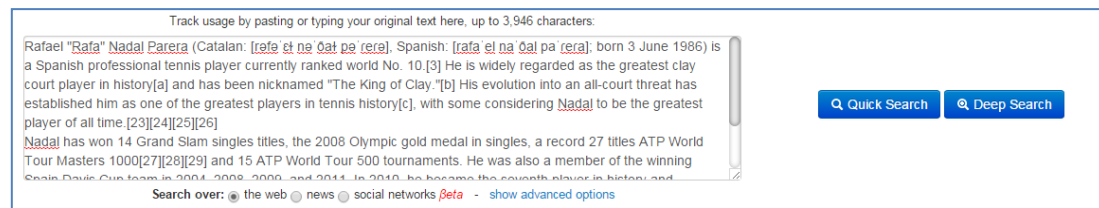
obojenim dijelovima za koje je sustav našao da su slični, svaki obojeni dio ima istu boju kao izvor iz kojeg dolazi. Izvori se nalaze s desne strane, poredani su po postotku te navode iz kojeg izvora dolaze, izvori su web stranice, znanstveni članci ili neki drugi rad. Također klikom na obojeni dio možemo otići na članak iz kojeg je obojeni dio teksta. Sve ovo možemo vidjeti na slici 2.1. Korisnik ima opcije i mijenjati pretragu, može namjestiti da mu se ne izbacuju podudaranja manja od 5 riječi ili 10% ili da se ne bojaju citati nekih članaka. Turnitin je sustav koji naplaćuje svoje usluge.



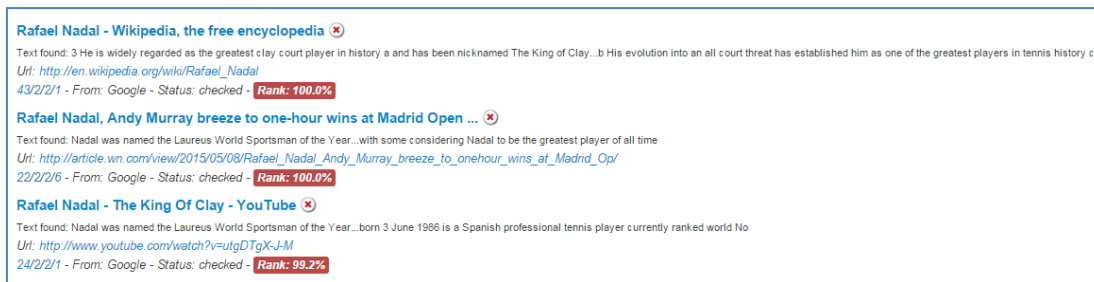
Slika 2.1 Izgled sustava Turnitin

2.1.2. Plagium

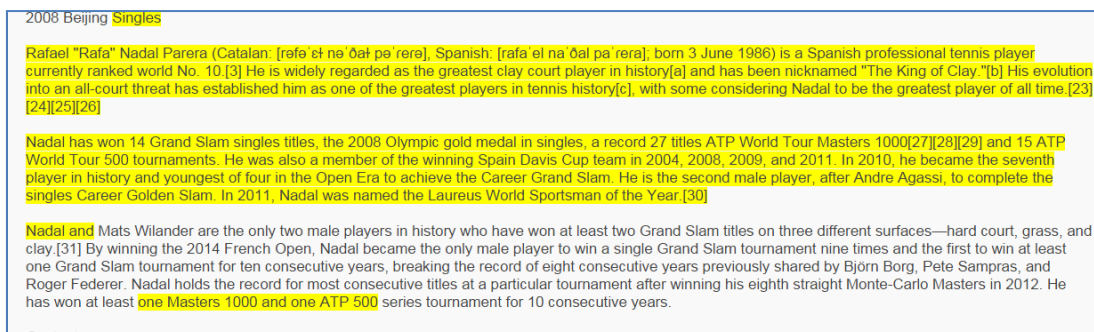
Plagium je sustav za detekciju koji nudi nekoliko mogućnosti [6]. Dostupno je besplatno brzo pretraživanje koje radi jednostavnu pretragu s tekstovima na internetu, novinama te socijalnim mrežama. Možemo vidjeti primjer jednog traženja na slici 2.2. te na slikama 2.3 i 2.4 izlaz ovog sustava. Ovaj način rijetko može s velikom sigurnošću odrediti spada li tekst u kategoriju plagijata. Nadalje Plagium nudi mogućnosti kreiranja računa te s plaćanjem usluge dobijemo mogućnost detaljne pretrage koja je puno bolja od brze besplatne. S korisničkim računom imamo mogućnost uploadati datoteke i tražiti od sustava da ih usporedi i vrati nam njihove sličnosti.



Slika 2.2 Unos teksta u sustav Plagium



Slika 2.3 Rangiranje nađenih stranica po sličnosti



Slika 2.4 Bojanje dijelova stranice koji su jednaki zadanom tekstu

2.1.3. Dupli Checker

Dupli Checker je potpuno besplatan jednostavan sustav za detekciju plagijata među tekstualnim datotekama [7]. Sve što je potrebno za nesmetanu provjeru je besplatna registracija na stranici duplichecker.com. Stranica je dosta jednostavna, možemo unijeti tekst sami ili ga možemo uploadati u obliku datoteke, nakon unesenog teksta i provjere, sustav nam vrati listu stranica na kojima je našao sličnost s našim tekstom te također možemo vidjeti te stranice s obojenim

dijelovima koji su slični ili isti našem tekstu. Primjer pretrage dan je na slikama 2.5 i 2.6.

Please Enter Your Text Below And Press Search:

Nadal and Mats Wilander are the only two male players in history who have won at least two Grand Slam titles on three different surfaces—hard court, grass, and clay.^[31] By winning the 2014 French Open, Nadal became the only male player to win a single Grand Slam tournament nine times and the first to win at least one Grand Slam tournament for ten consecutive years, breaking the record of eight consecutive years previously shared by Björn Borg, Pete Sampras, and Roger Federer. Nadal holds the record for most consecutive titles at a particular tournament after winning his eighth straight Monte-Carlo Masters in 2012. He has won at least one Masters 1000 and one ATP 500 series tournament for 10 consecutive years.

Or Enter Url:

Maximum 1000 words limit per search. Total Words: 121

Or Browse a Docx Or Text File:

Choose File No file chosen

Search Clear

Slika 2.5 Primjer unosa teksta u sustav DupliChecker

Dupli Checker
Free Online Software For Plagiarism Detection

The comparison below was created by DupliChecker
The page below has 141 words matching 100% of the text as highlighted by DupliChecker :
http://en.wikipedia.org/wiki/Rafael_Nadal

Share this page...

Contents
Featured content
Current events
Random article
Donate to Wikipedia
Wikipedia store

Interaction
Help
About Wikipedia
Community portal
Recent changes
Contact page

Tools
What links here
Related changes
Upload file
Special pages
Permanent link
Page information
Wikidata item
Cite this page
Print/export

This name uses Spanish naming customs : the first or paternal family name is Nadal and the second or maternal t

Rafael " Rafa " Nadal Parera (Catalan: [ˈrafaˈet̪ nəˈðal paˈrera] , Spanish: [ˈrafaˈel naˈðal paˈrera] ; born 3 June 1986) is a Spanish professional tennis player currently ranked world No. 10. ^[31] He is widely regarded as the greatest clay court player in history ^[a] and has been nicknamed "The King of Clay." ^[b] His evolution into an all-court threat has established him as one of the greatest players in tennis history ^[c] , with some considering Nadal to be the greatest player of all time. ^{[23][24][25][26]}

Nadal has won 14 Grand Slam singles titles , the 2008 Olympic gold medal in singles , a record 27 titles ATP World Tour Masters 1000 ^{[27][28][29]} and 15 ATP World Tour 500 tournaments. He was also a member of the winning Spain Davis Cup team in 2004 , 2008 , 2009 , and 2011. In 2010, he became the seventh player in history and youngest of four in the Open Era to achieve the Career Grand Slam. He is the second male player, after Andre Agassi , to complete the singles Career Golden Slam. In 2011, Nadal was named the Laureus World Sportsman of the Year. ^[30]

Nadal and Mats Wilander are the only two male players in history who have won at least two Grand Slam titles on three different surfaces—hard court, grass, and clay. ^[31] By winning the 2014 French Open , Nadal became the only male player to win a single Grand Slam tournament nine times and the first to win at least one Grand Slam tournament for ten consecutive years, breaking the record of eight consecutive years previously shared by Björn Borg , Pete Sampras , and Roger Federer. Nadal holds the record for most consecutive titles at a particular tournament after winning his eighth straight Monte-Carlo Masters in 2012. He has won at least one Masters 1000 and one ATP 500 series tournament for 10 consecutive years.

Slika 2.6 Stranica koju je sustav našao s obojenim traženim dijelom teksta

2.2. Sustavi za detekciju plagijata izvornog koda

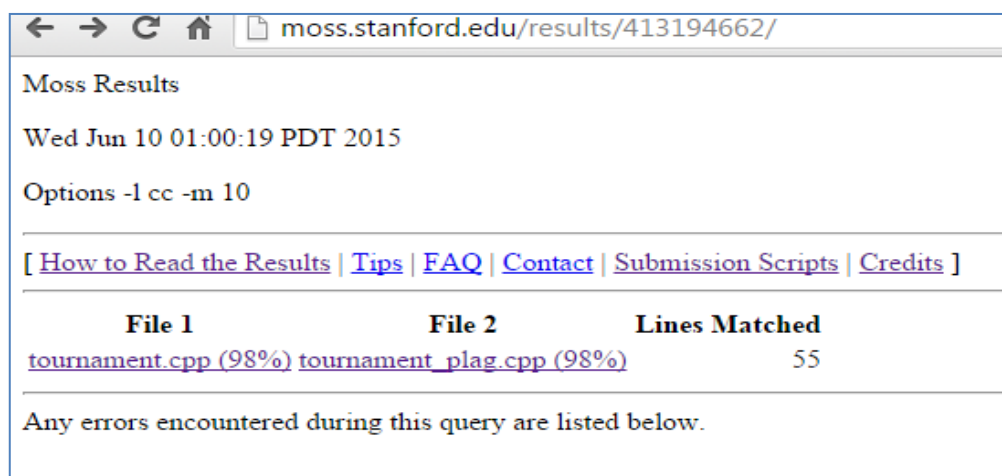
Naglasak rada je na sustavima koji rade s izvornim kodovima te ćemo u ovom potpoglavlju detaljnije pregledati najpoznatije postojeće sustave. Plagijarizam izvornog koda se dosta razlikuje od plagijarizma običnog teksta, izvorni kod se sastoji od varijabli, petlji, naredbi grananja i mnogih drugih aspekata programskih jezika na koje moramo misliti pri detekciji. Također rijetko postoje baze kodova s kojima bi se sumnjivi kodovi mogli usporediti jer se programski zadaci studenata razlikuju iz godine u godinu. Algoritmi za detekciju mogu se podijeliti u nekoliko kategorija s obzirom na način na koji uspoređuju dva izvorna koda.

- Algoritmi koji uspoređuju nizove znakova - traže odgovarajuće segmente jednog koda u drugome (recimo nizove duljine 5). Jako su brzi, ali imaju loše rezultate čim se promijene imena varijabli, ključnih riječi, itd.
- Algoritmi koji uspoređuju simbole - slično kao kod nizova znakova, ali koriste leksički analizator koji dijeli izvorni kod u niz simbola. Tako se gube praznine, komentari što povećava šansu za bolju usporedbu. Većina sustava razvijenih za akademske potrebe radi na ovaj način.
- Algoritmi koji uspoređuju stabla raščlambe - stvaraju i uspoređuju stabla raščlambe, ovo omogućava sličnost na višem nivou, kao naprimjer sličnosti naredbi grananja. Dosta sporiji od prethodno navedenih algoritama.
- Algoritmi koji uspoređuju graf programske zavisnosti - graf programske zavisnosti sadrži stvarni protok programa te omogućava usporedbu na izrazito visokom nivou, loša stvar je velika složenost i sporost.
- Algoritmi koji uspoređuju metriku - metrika računa „uspjeh“ nekih segmenata izvornog koda, kao naprimjer broj petlji i naredbi grananja ili broj različitih korištenih varijabli. Metrika se dosta jednostavno i lagano računa, ali isto tako lagano može doći do problema, dva potpuno različita segmenta mogu imati jednaki „uspjeh“.
- Algoritmi s hibridnim pristupom - to su algoritmi koji kombiniraju više jednostavnijih algoritama, na primjer možemo kombinirati stabla raščlambe s algoritmima usporedbe znakovnog niza.

U nastavku potpoglavlja detaljno je opisan najpoznatiji besplatni sustav za detekciju sličnosti izvornog koda.

2.2.1. MOSS

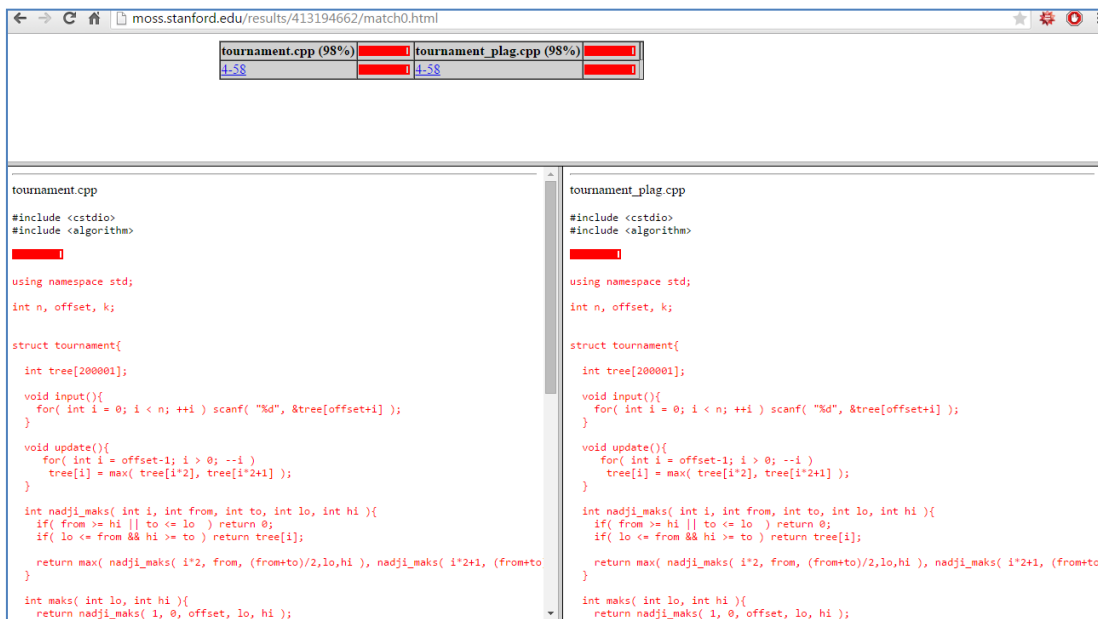
MOSS(engl. Measure of Software Similiarity) [3] je sustav za detekciju plagijata među izvornim kodovima razvijen na Stanfordu. Glavna zadaća Moss-a je detekcija plagijata među studentskim rješenjima programskih zadataka na laboratorijskim vježbama. Može se naravno koristiti i u detekciji ukradenih dijelova software-a među velikim brojem izvornih kodova i programskih knjižnica ili modula koji su kopirani ili malo promijenjeni. Napravljen je još davne 1994. godine te je veoma efikasan u svojoj ulozi. Moss je vjerojatno najkorišteniji sustav za detekciju na fakultetima pa ga tako često koristi i naš fakultet. Međutim, koliko god algoritam na kojem je sustav baziran bio dobar, ne može garantirati da je nešto plagijat, to ipak ostaje na ljudskoj procjeni, kako je već prije spomenuto. Moss koristi algoritam winnowing za traženje digitalnog otiska dokumenta [4]. Moss ne radi na čistoj usporedbi teksta već je svjestan sintakse programskog jezika kojeg uspoređuje. Trenutno Moss radi detekciju među idućim programskim jezicima: C, C++, Java, C#, Python, Visual Basic, Javascript, FORTRAN, ML, Haskell, Lisp, Scheme, Pascal, Modula2, Ada, Perl, TCL, Matlab, VHDL, Verilog, Spice, MIPS assembly, a8086 assembly, HCL2. Korištenje Moss-a je vrlo jednostavno, sustav je web aplikacija kojoj samo pošaljemo listu direktorija, a sustav ostalo odradi za nas. Jedina stvar potrebna za nesmetano korištenje sustava je registracija na isti, potrebno je samo poslati mail sa željenim korisničkim imenom. Nakon obrade svih poslanih datoteka Moss nam ih sortira po sličnosti, broju istih znakovi i broju istih linija te također omogućuje detaljan pogled na dva koda koja je uspoređivao, kao što vidimo na slici 2.7.



← → ↺ 🏠 📄 moss.stanford.edu/results/413194662/		
Moss Results		
Wed Jun 10 01:00:19 PDT 2015		
Options -l cc -m 10		
[How to Read the Results Tips FAQ Contact Submission Scripts Credits]		
File 1	File 2	Lines Matched
tournament.cpp (98%)	tournament_plag.cpp (98%)	55
Any errors encountered during this query are listed below.		

Slika 2.7 Početna stranica Moss sustava

Svi slični dijelovi iz oba koda su obojeni istim bojama te možemo skakati po tim dijelovima kako bi si olakšali pretragu. Također nikad ne boja dijelove koda za koje se očekuje da će biti isti (npr. programske knjižnice). Ovo vidimo na slici 2.8. Korisnik može sam određivati pragove za detekciju ako očekuje da će kodovi biti slični (npr. nekakvi algoritamski zadaci) te zadati koji dio koda očekuje da će biti isti, da ga Moss izbacila iz izračuna.



Slika 2.8 Detaljan pogled na par rješenja

3. Arhitektura sustava

Glavni dio ovog rada bio je razviti sustav koji detektira plagijate među izvornim kodovima. Plagijator trenutno podržava programske jezike C i C++, ali je detekcija programskog jezika koji se uspoređuje napravljena tako da je primjenjiva na skoro bilo koji programski jezik. Rad sustava je zapravo vrlo jednostavan, sustav od korisnika primi izvorne kodove, računa sličnosti te omogućuje korisniku detaljni uvid u parove tih izvornih kodova. Kada sustav primi izvorne kodove korisnika on ih sprema u memoriju na način da je u daljnjem radu dohvat bilo kojeg izvornog koda vrlo jednostavan. Kako primljenih izvornih kodova može biti puno (zbroj njihovih duljina je još veći) algoritam koji koristimo za računanje sličnosti bi trebao biti brz i efikasan. Također vjerujemo da bi algoritam koji detektira plagijate trebao imati iduća svojstva:

1. *Neosjetljivost na nebitne znakove*, nebitni znakovi ne utječu na sličnost dvaju kodova te ih treba ukloniti iz izračuna. Na primjer to su praznine i komentari.
2. *Neosjetljivost na prekratke pogotke*, ne želimo da nam algoritam pronalazi prekratke pogotke, kao naprimjer `int` u programskom jeziku C/C++.
3. *Neosjetljivost na poziciju*, želimo da naš algoritam pronađe pogotke i ako oni nisu na istim pozicijama u oba koda.
4. *Neosjetljivost na promjene imena identifikatora*, jedan od čestih pokušaja plagiranja izvornih kodova je promjena svih identifikatora te želimo da algoritam takve identifikatore pronađe i proglasi istima.

Mjerenje sličnosti u sustavima za detekciju plagijarizma uglavnom se bazira na usporedbi digitalnih otisaka dokumenata [4]. Jedan od algoritama koji traže digitalni otisak dokumenta je winnowing. Winnowing je algoritam koji određuje

digitalni otisak na temelju hash vrijednosti k-grama [4]. Digitalni otisak će zapravo biti niz brojeva. K-gram je slijedni podniz duljine k nekog niza znakova. Uzmimo na primjer niz znakova 'plagijator', 5-grami bi bili plagi, lagij, agija, gijat, ijato, jator. Hash vrijednost se dobije algoritmom Rabin-Karp koji je posebno pogodan za hashiranje k-grama jer traži upravo hash vrijednosti slijednih podnizova. Algoritam o hash vrijednosti novog niza odlučuje uz pomoć dijela starog niza i jednog novog znaka. To nam zapravo znači da sve hash-vrijednosti možemo dobiti u linearnoj vremenskoj složenosti. Winnowing odabire digitalni otisak dokumenta na način da dijeli niz hash vrijednosti u prozore duljine w. U svakom prozoru tražimo najmanju hash vrijednost i odabiremo ju, ako je više najmanjih vrijednosti odabiremo najdesniju. Ako prozor ima istu najmanju vrijednost kao prozor lijevo od njega i pozicija te vrijednosti je manja od pozicije iste vrijednosti u lijevom prozoru, onda tu vrijednost ne odabiremo. Sve odabrane vrijednosti čine digitalni otisak dokumenta.

Algoritam koji koristi Plagijator najprije inicijalizira prefiksno stablo programskog jezika kojeg uspoređuje. Prefiksno stablo je struktura podataka koja nam može u linearnoj složenosti reći pripada li znakovi niz skupu znakovnih nizova od kojeg je stablo konstruirano [8]. Skup znakovnih nizova su zapravo standardne biblioteke i ključne riječi programskog jezika kojeg se uspoređuje. Prefiksno stablo nam služi kako bi brzo mogli znati je li niz znakova identifikator neke varijable. Ako nam stablo kaže da niz znakova ne postoji, niz znakova je identifikator varijable. Algoritam sve identifikatore tada mijenja u jedinstveni identifikator(mi smo se odlučili za slovo 'v'). Nakon ove modifikacije izvornog koda, brišu se sve praznine i komentari. Tada je izvorni kod zapravo samo slijedni niz znakova. Tada pomoću prethodno opisanog Rabin-Karp algoritma računamo hash vrijednosti svih k-grama. Taj niz se dalje šalje winnowingu koji nam vraća digitalni otisak dokumenta. Winnowing koji je implementiran malo se razlikuje od gore opisanog, pošto smo sve identifikatore zamijenili slovom v, postupak bi davao previše zabuna (engl. false positives). Modifikacija je takva da uzmemo svakih n vrijednosti digitalnog otiska te ih slijepimo zajedno u jednu vrijednost, dodajemo kontekst algoritmu. Prethodno opisan postupak imenovali smo winnowing s kontekstom. Uz prethodnu modifikaciju dodana je i još jedna manja, uz svaku vrijednost digitalnog otiska dodan je broj linije u kojem je ta vrijednost nađena. Kada imamo digitalne otiske

svih izvornih kodova možemo odrediti sličnosti među parovima. Korištenjem rječnika(engl. dictionary, map) i malo pametnijom implementacijom usporedbe digitalnih otisaka do svih sličnosti algoritam dolazi u linearnoj vremenskoj složenosti, što algoritam čini izuzetno brzim.

Postoje dvije sličnosti koje algoritam određuje, recimo da je prvi izvorni kod imena h, a drugi imena g, prva sličnost je sličnost h prema g, a druga je sličnost od g prema h. Pomoću te dvije sličnosti računa se funkcija granice (engl. threshold) koja služi za brisanje parova kodova koji nemaju dovoljno veliku sličnost(ispod funkcije granice). Ovo nam, dakako, služi kako bi korisniku olakšali posao, ali i ubrzali naš sustav, ne želimo gubiti resurse na parove koji nisu niti približno slični. Vrijednost ove funkcije vidimo kao mjesto za poboljšanje ovog algoritma.

U ovom trenutku algoritam mora prvi put imati kvadratnu vremensku složenost, mora korisniku predati sve parove izvornih kodova s pripadnim sličnostima. No, možemo primijetiti da smo zbog funkcije granice te parove ipak razrijedili. Također naš algoritam posjeduje sva svojstva koja smo naveli ranije. Svojstvo (1) dobijemo na početku algoritma brisanjem komentara i praznih znakova, svojstvo (2) određuje parametar k unutar winnowinga, svojstvo (3) smo dobili pamćenjem pozicije unutar winnowinga i svojstvo (4) dobijemo korištenjem prefiksnog stabla.

Plagijator također korisniku nudi i detaljan uvid u parove izvornih kodova tako što ih prikaže jednog pored drugog i istim bojama boja jednake dijelove. Bojanje istih dijelova je problem pronalaska najdužeg zajedničkog podniza između dva niza. Naši nizovi su digitalni otisci dvaju kodova. Najduži zajednički podniz tražimo sve dok postoji, uz ogradu da nas ne zanimaju podnizovi manji od duljine m. Najbrži način za traženje najduljeg zajedničkog podniza je sufiksno stablo, sufiksno stablo ga traži u linearnoj vremenskoj složenosti [9]. Kada smo našli podniz, bojamo ga s obje strane jednostavnim dodavanjem HTML tagova.

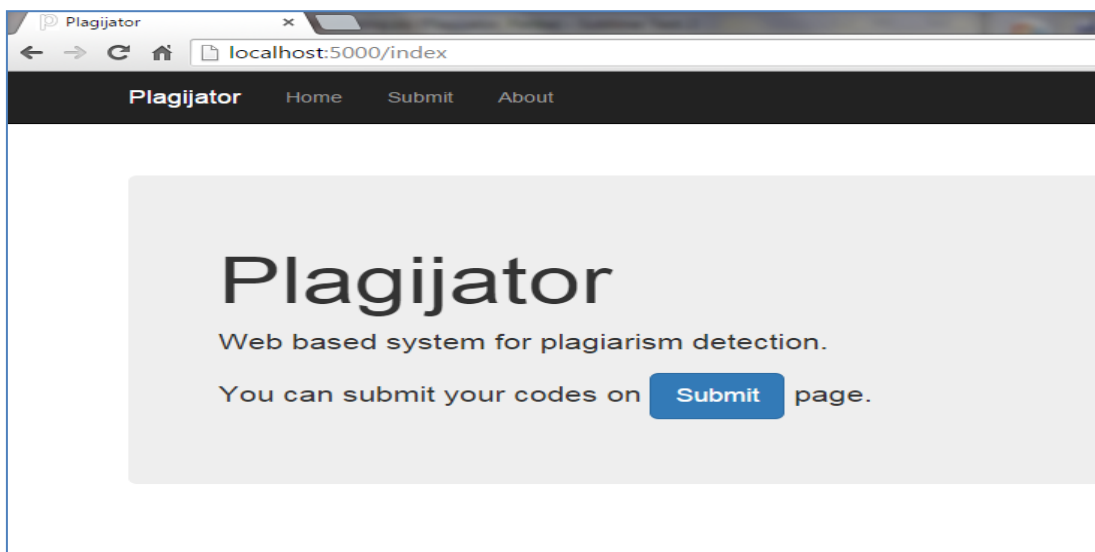
Implementacija aplikacije sastoji se od dva dijela, glavni dio aplikacije napisan u Pythonu te dio aplikacije za prikaz u web pregledniku napisan u HTML-u. U nastavku poglavlja opisani su svi Python moduli i HTML kodovi.

- *coloring.py*, modul koji koristi sufiksno stablo te pomoću njega traži najdulje zajedničke podnizove istih dijelova između dva izvorna koda te boja te dijelove jednakim bojama.

- *data_container.py*, modul koji sprema sve potrebne podatke u memoriju, kao na primjer k-grame, imena kodova, itd. za svaki pojedini izvorni kod. Koristi se za lagani dohvat svih potrebnih podataka.
- *forms.py*, modul u kojem se nalaze sve forme, u ovom slučaju nalazi se forma za upload zip datoteka.
- *index_mapping.py*, modul koji sadrži metode koje računaju digitalni otisak za sve predane izvorne kodove te na temelju njih računa sličnosti među izvornim kodovima.
- *initialisations.py*, modul koji stvara prefiksna stabla za sve podržane jezike te omogućava odabir boja kojim će se slični dijelovi izvornih kodova bojati.
- *language.py*, modul koji predstavlja programski jezik te sadrži sva svojstva tog jezika.
- *main.py*, modul koji pokreće web aplikaciju koristeći ugrađene metode Flask-a za pokretanje servera.
- *prefix_tree.py*, modul koji predstavlja prefiksno stablo jednog programskog jezika te metode za stvaranje i korištenje istog.
- *rolling_hash.py*, modul koji računa hash vrijednosti k-grama koristeći Rabin-Karp algoritam.
- *string_processing.py*, modul koji priprema izvorne kodove za daljnji rad, traži zaglavlja, briše komentare, mijenja identifikatore varijabli u jedan isti identifikator.
- *suffix_tree.py*, modul koji predstavlja sufiksno stablo pomoću kojeg u linearnoj složenosti odgovaramo na pitanje koji je najveći zajednički podniz nizova.

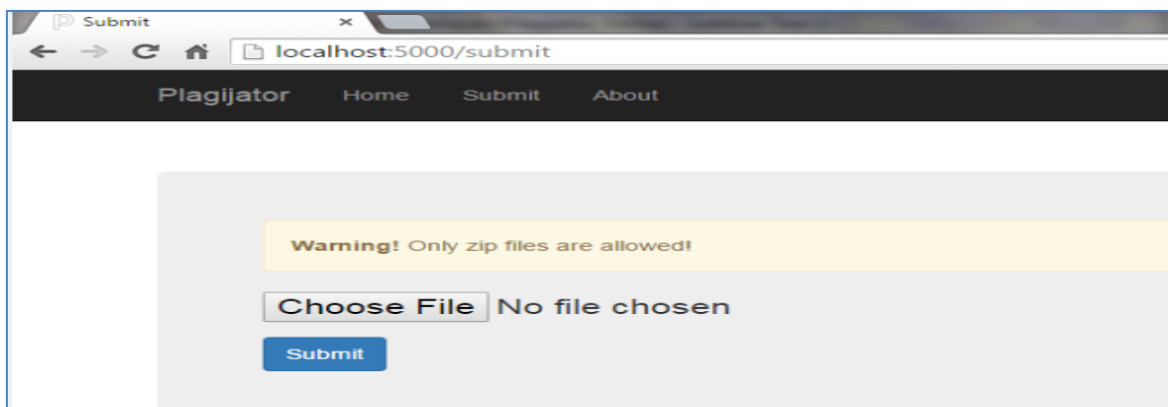
- *views.py*, modul koji koristi HTML kodove za prikaz svih stranica web aplikacije. Svaka metoda ovog modula predstavlja jednu stranicu.
- *winnowing.py*, modul koji koristi algoritam winnowing, dijeli dobivene hash vrijednosti *rolling_hash.py* modula u prozore te računa digitalni otisak.
- *about.html*, HTML kod za stranicu na kojoj je ukratko opisan sustav.
- *base.html*, bazni HTML kod, sadrži izgled kojeg svi ostali HTML kodovi nasljeđuju, to su stvari koje svaka stranica posjeduje, kao na primjer navigacijska ploča(engl. navbar).
- *compare.html*, HTML kod za stranicu na kojoj su prikazana dva koda.
- *index.html*, kod za početnu stranicu sustava.
- *submit.html*, kod za stranicu na kojoj možemo poslati zip datoteku sustavu.
- *uploads.html*, kod za stranicu na kojoj nam je prikazana tablica parova izvornih kodova te njihove sličnosti.

4. Korisničko sučelje i upute za korištenje



Slika 4.1 Početna stranica Plagijatora

Na slici 4.1 prikazana je početna stranica web aplikacije. Odmah možemo primijetiti jednostavnost izgleda iste. Početna stranica nam nudi kratki opis sustava te skok na iduću stranicu klikom na gumb submit koji je obojen plavom bojom. Klikom na taj gumb aplikacija nas preusmjerava na stranicu prikazanu slikom 4.2.

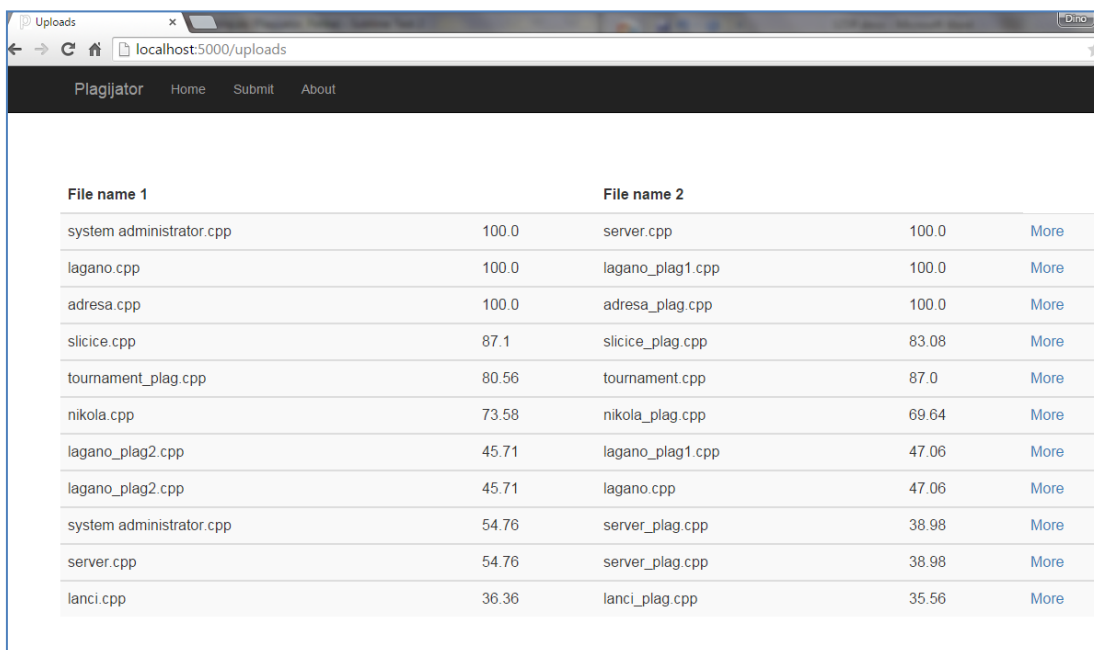


Slika 4.2 Stranica na kojoj šaljemo zip datoteke

Jedina mogućnost koju nam nudi ova stranica je odabir zip datoteke, koja sadrži izvorne kodove koje želimo provjeriti, s našeg računala te slanje istih klikom na gumb submit Plagijatoru. Na stranici također vidimo upozorenje da su dozvoljene jedino zip datoteke, Plagijator je implementiran da zna raditi samo s njima zbog njihove jednostavnosti i male veličine. Zip datoteka bi trebala sadržavati izvorne

kodove jednog problemskog zadatka, jer naravno nema smisla miješati rješenja različitih zadataka te ih uspoređivati. Izvorni kod rješenja može biti napisan u pojedinačnoj datoteci ili kao projekt koji je onda opet potrebno pretvoriti u zip datoteku. Plagijator sam prepoznaje radi li se o pojedinačnim rješenjima ili projektima. Napomena je da se pojedinačna rješenja i projekti također ne miješaju.

Nakon što aplikaciji predamo zip datoteku ona počinje svoj izračun te nam vraća listu parova izvornih kodova te pripadne sličnosti. Pored imena prvog izvornog koda iz para je njegova sličnost prema drugom izvornom kodu para, a pored imena drugog izvornog koda iz para je njegova sličnost prema prvom izvornom kodu para. Ovo možemo vidjeti na slici 4.3. Najbolji primjer zašto su nam potrebne dvije sličnosti je kada na primjer imamo jedan izvorni kod koji se sastoji od samo tri linije i imamo drugi izvorni kod koji se sastoji od 50 linija, ali također sadrži sve tri linije prvog koda, tada prvi kod ima sličnost od 100% prema drugom, dok drugi kod ima sličnost od 6% prema prvom. To nikako ne možemo opisat samo jednim brojem.

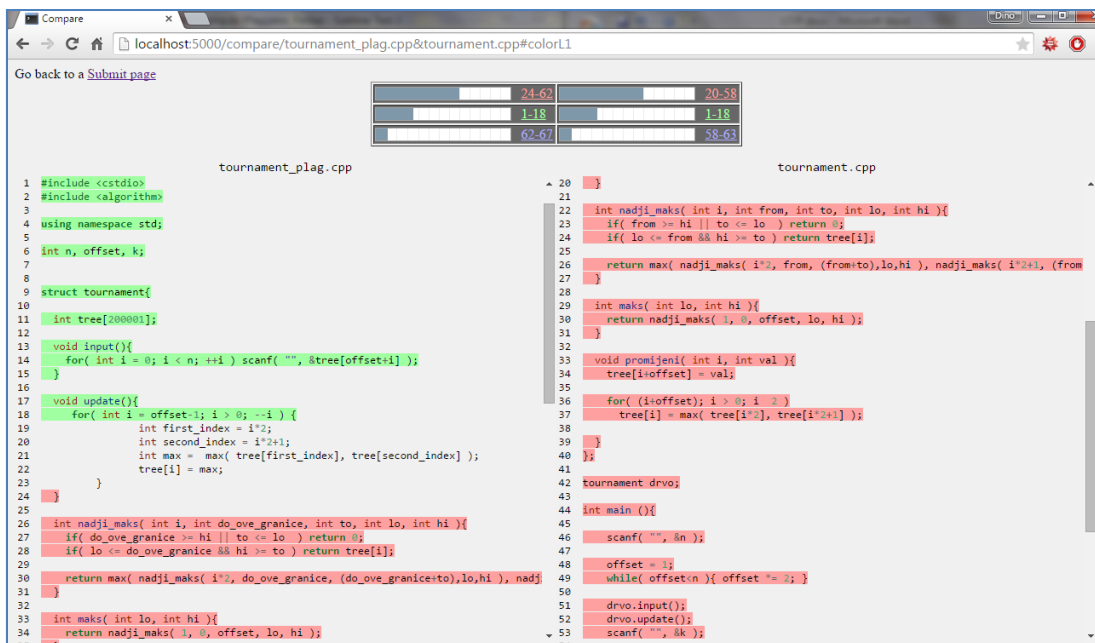


File name 1		File name 2		
system administrator.cpp	100.0	server.cpp	100.0	More
lagano.cpp	100.0	lagano_plag1.cpp	100.0	More
adresa.cpp	100.0	adresa_plag.cpp	100.0	More
slicice.cpp	87.1	slicice_plag.cpp	83.08	More
tournament_plag.cpp	80.56	tournament.cpp	87.0	More
nikola.cpp	73.58	nikola_plag.cpp	69.64	More
lagano_plag2.cpp	45.71	lagano_plag1.cpp	47.06	More
lagano_plag2.cpp	45.71	lagano.cpp	47.06	More
system administrator.cpp	54.76	server_plag.cpp	38.98	More
server.cpp	54.76	server_plag.cpp	38.98	More
lanci.cpp	36.36	lanci_plag.cpp	35.56	More

Slika 4.3 Tablica parova izvornih kodova te pripadne sličnosti

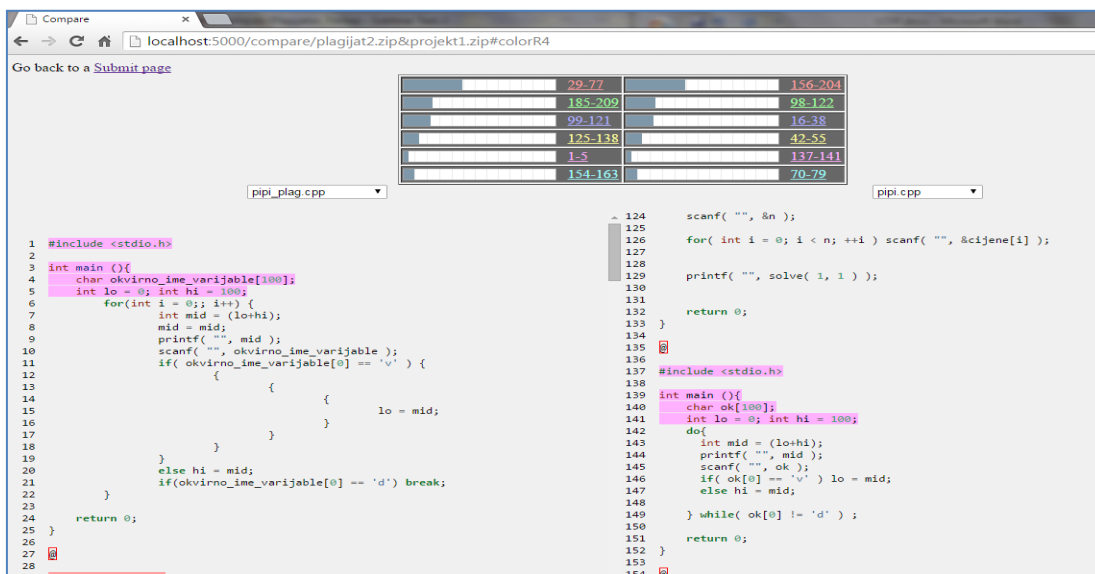
Uz svaki par izvornih kodova s desne strane stoji *More*, klikom na taj link Plagijator nas preusmjerava na stranicu, koju vidimo na slici 4.4, na kojoj možemo detaljno pregledati dva izvorna koda koja nas zanimaju te nam također boja slične dijelove istim bojama što također vidimo na slici 4.4. Zbog lakše navigacije kroz kodove

dodana je tablica svih boja kojim su dijelovi obojeni te klikom na te boje s lijeve strane skaćemo na taj dio izvornog koda s lijeve strane, analogno je s desne strane. Uz boju vidimo koliki udio linija je obojen tom bojom. Tablicu možemo primijetiti također na slici 4.4.



Slika 4.4 Usporedba dva izvorna koda

Kada Plagijator shvati da je korisnik poslao projekte na detekciju, ovu stranica dobije mogućnost lakog prolaska kroz sve datoteke projekta u okviru padajućeg izbornika kojeg vidimo na slici 4.5. Klikom na bilo koje ime skaćemo na taj izvorni kod.



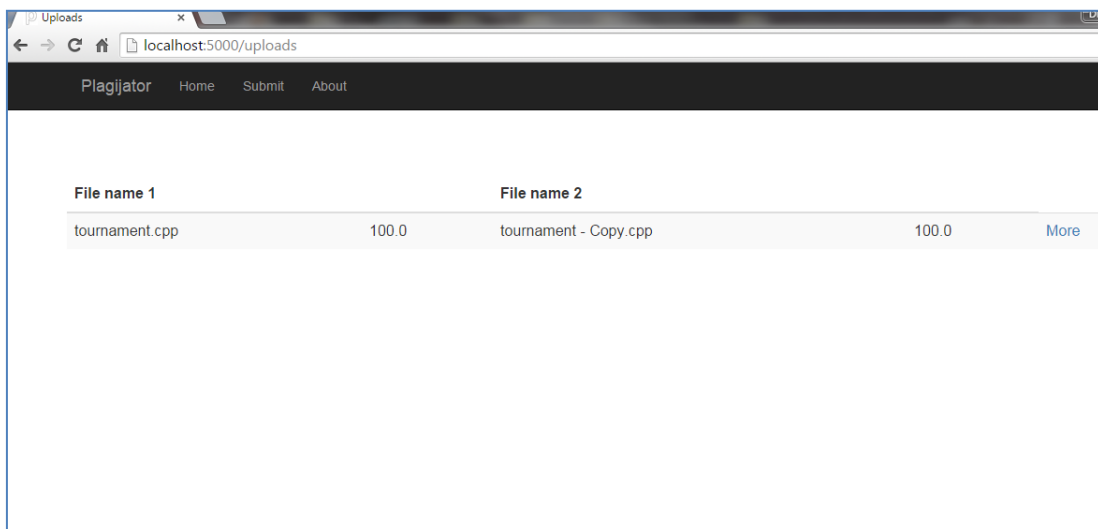
Slika 4.5 Usporedba projekata

5. Primjeri rada Plagijatora

U ovom poglavlju testirat ćemo rad Plagijatora na nekoliko primjera.

5.1. Prvi primjer

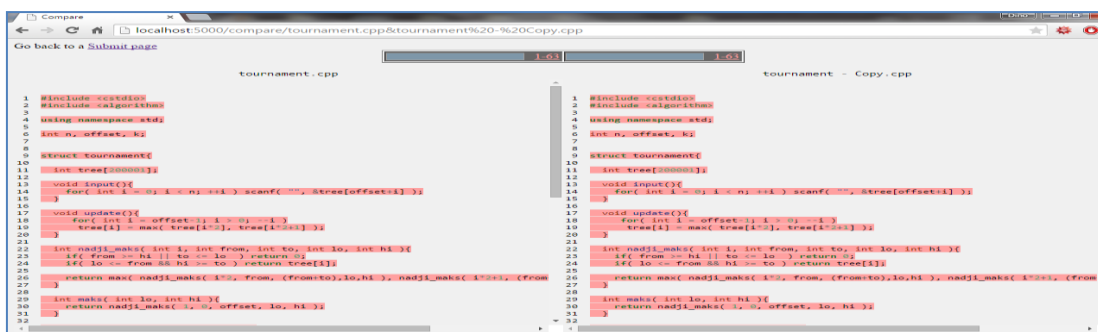
Prvi primjer na kojem je testiran su dva potpuno jednaka izvorna koda, izlaz koji je Plagijator dao prikazan je na slikama 5.1 i 5.2. Možemo primijetiti (slika 5.1) da je Plagijator potpuno skužio da su kodovi jednaki te nam vratio sto postotnu sličnost u oba smjera.



File name 1		File name 2		
tournament.cpp	100.0	tournament - Copy.cpp	100.0	More

Slika 5.1 Sličnosti koje je Plagijator vratio za prvi primjer

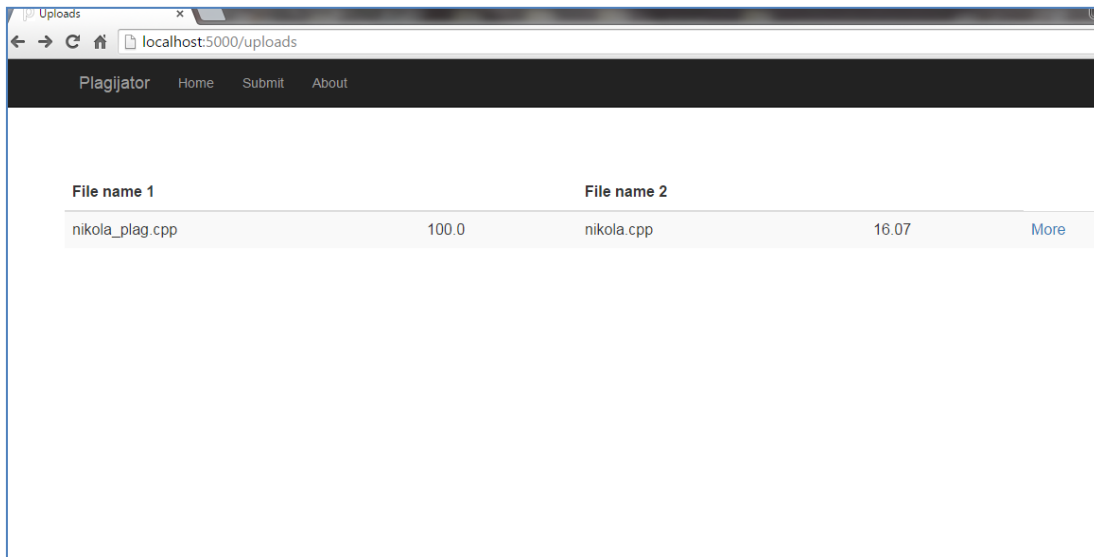
Na detaljnom prikaz (slika 5.2) možemo vidjeti da je Plagijator obojio oba cijela koda istom bojom, što znači da su potpuno jednaki.



Slika 5.2 Usporedba kodova prvog primjera

5.2. Drugi primjer

Drugi primjer na kojem je Plagijator testiran je kada imamo jedan manji i jedan veći izvorni kod, ali su sve linije manjeg sadržane u većem izvornom kodu. Izlaz sustava prikazan je na slikama 5.3 i 5.4

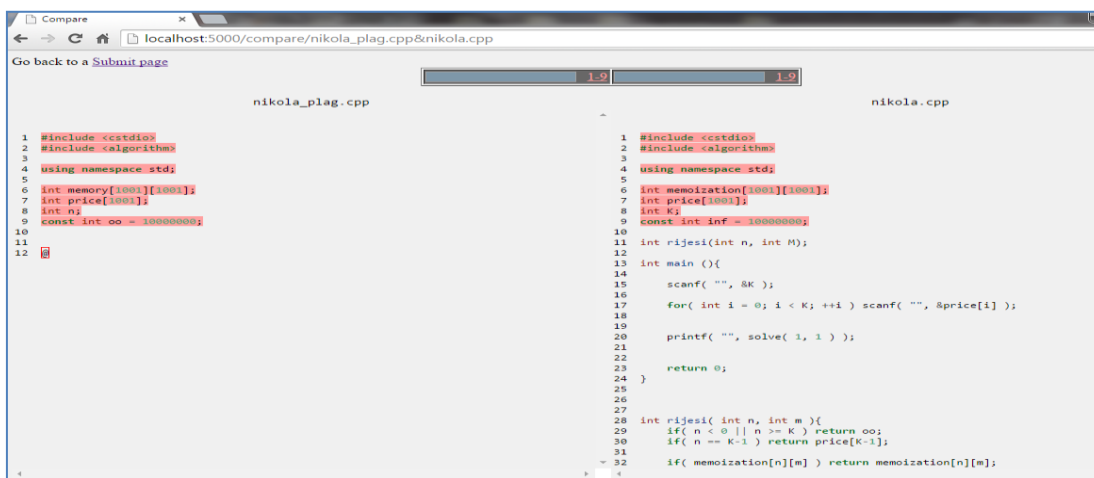


The screenshot shows a web browser window with the URL `localhost:5000/uploads`. The page has a navigation bar with links: Plagijator, Home, Submit, and About. Below the navigation bar, there is a table with two columns: "File name 1" and "File name 2". The table contains one row of data:

File name 1		File name 2		
nikola_plag.cpp	100.0	nikola.cpp	16.07	More

Slika 5.3 Sličnosti koje je Plagijator vratio za drugi primjer

Na slici 5.3 odmah primjećujemo da je jedan kod 100% sličan drugome, a drugi prvome samo 16%, te se možemo pitati zašto je to tako? Odgovor leži u tome da je manji kod potpuno sadržan u velikom te njegova sličnost mora biti 100% prema većem. Manji kod je samo dio većeg koda te je upravo zato njegova sličnost puno manja.



Slika 5.4 Usporedba kodova drugog primjera

Na slici 5.4 možemo vidjeti upravo gore opisano, prvi dio oba koda je identičan te ga je Plagijator kao takvog otkrio i obojio jednakom bojom. Također na slici možemo primijetiti da su imena varijabli izmijenjena, ali ih je Plagijator otkrio kao iste.

5.3. Treći primjer

Treći primjer na kojem je Plagijator testiran je zamjena imena svih varijabli te zamijenjen redoslijed napisanih funkcija. Izlaz sustava dan je na slikama 5.5, 5.6 i 5.7.



The screenshot shows a web browser window with the address bar at 'localhost:5000/uploads'. The page has a dark header with the title 'Plagijator' and navigation links 'Home', 'Submit', and 'About'. Below the header is a table with two columns: 'File name 1' and 'File name 2'. The first row of data shows 'nikola_plag.cpp' in the first column and 'nikola.cpp' in the second column. Between these two file names, the similarity percentage '51.79' is displayed. To the right of the second file name, there is a link labeled 'More'.

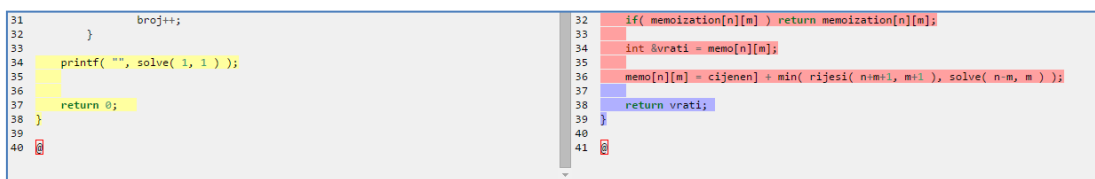
File name 1	File name 2
nikola_plag.cpp	51.79 nikola.cpp

Slika 5.5 Sličnosti koje je Plagijator vratio za treći primjer

Iz slike 5.5 zaključujemo da su oba koda jednako slična jedan drugome i to po 50%, dakle otprilike pola jednog izvornog koda je slično drugom izvornom kodu. Na slici 5.6 možemo uočiti da je Plagijator opet pronašao sve promjene imena varijabli te zaključujemo da je otporan na te pokušaje plagijarizma. Također možemo vidjeti da nije bitno na kojoj su poziciji napisane funkcije, Plagijator ih je našao i obojio jednakim bojama. Dio koji nije obojen su dvije različite programske petlje (for i while petlja) koje rade zapravo jednaku stvar, te uočavamo da Plagijator mehanizam za otkrivanje takvih pokušaja plagijarizma još nema implementiran. To ostavljamo za budući rad.



Slika 5.6 Usporedba kodova trećeg primjera



Slika 5.7 Nastavak usporedbe trećeg primjer

6. Korištene tehnologije i alati

Tijekom implementacije aplikacije korišteno je nekoliko programskih jezika, radni okvir(engl. framework), besplatne programske knjižnice(engl. library) i besplatni alati. U nastavku su navedeni svi glavni alati i tehnologije koji su korišteni tijekom izrade aplikacije. Također je objašnjeno kako doći do tih alata kako bi mogli pokrenuti aplikaciju.

6.1. Python

Kako je aplikacija zamišljena kao web aplikacija bilo je potrebno odabrati programski jezik koji nam nudi izradu web aplikacija kroz svoje radne okvire. Zbog svoje jednostavnosti, lakoće razvoja, ali izuzetnih mogućnosti odabran je Python i to verzija 2.7.

Python je jezik visoke razine te podržava skoro sve najkorištenije programske paradigme, poput objektno orijentirane, imperativne, funkcijske te proceduralne. Također nam nudi široku paletu radnih okvira i programskih knjižnica.

Instalaciju Pythona je jednostavno skinuti s Interneta i instalirati. Instalacija dolazi s ugrađenim IDLE(engl. *Integrated DeveLopment Environment*) . IDLE je razvojna okolina. Također s instalacijom dolazi i Python Shell koji se lako pokreće utipkavanjem ključne riječi *python* u terminalu. Ovo su dva načina kako možemo pisati naše Python programe, no u ovoj aplikaciji smo ipak koristili uređivač teksta, konkretno *Sublime Text Editor 2* radi jednostavnosti. Svi napisani programi jednostavno se prevode unutar terminala tako da prije imena našeg izvornog koda stavimo ključnu riječ *python*.

6.2. Flask

Python nudi nekoliko radnih okvira za izradu web aplikacija. Odabran je Flask zbog svoje jednostavnosti i brzog razvoja.

Flask je mikro radni okvir(engl. micro framework) za izradu web aplikacija. Najnovija stabilna verzija je 0.10.1. Kažemo da je Flask mikro radni okvir jer ne

tjera korisnika na korištenje određenih alata ili programskih knjižnica. Nema apstraktni sloj za bazu podataka, kao ni validaciju formi. Međutim, Flask podržava dodatke koji se dodaju aplikaciji kao da su pisani u Flasku te nam to omogućuje da koristimo baze podataka, validaciju formi i sve ostale potrebne stvari.

Flask se poput velike većine Python knjižnica i radnih okvira može instalirati koristeći alat pip. Pip je alat za jednostavno skidanje i instaliranje Pythonovih knjižnica i radnih okvira. Kako bi ga koristili potrebno ga je skinuti s Interneta te instalirati. Kada imamo spreman pip, u terminal upišemo naredbu `pip install` te nakon toga ime knjižnice ili radnog okvira koji želimo. Za Flask jednostavno upišemo `pip install flask`. Nakon toga spremni smo raditi aplikacije u Flasku. Naravno za nekakve dodatne mogućnosti, kao što je ranije spomenuto trebamo skinuti dodatke, kao npr. `pip install flask-sqlalchemy`(za bazu podataka).

6.3. HTML

HTML (engl. Hyper Text Markup Language) je standardni jezik za stvaranje web stranica. Napisan je u formi HTML elemenata koji se sastoje od tagova zatvorenih u kutnim zagradama(`<html>`). Tagovi najčešće dolaze u parovima kao naprimjer `<div>` i `</div>`. Web preglednici čitaju HTML datoteke te tako stvaraju web stranice. HTML opisuje strukturu svake web stranice te ga to više čini jezikom za označavanje nego programskim jezikom. Omogućava prikaz slika i interaktivnih formi, strukturira tekstove te prihvaća skripte napisane u drugim jezicima poput JavaScript-a.

Kako bi koristili HTML zajedno s Python kodom, Flask koristi template jezik Jinja2. Jinja2 omogućava pisanje Python koda u HTML datotekama pomoću posebne sintakse. Jinja2 dolazi uz Flask te ju nije potrebno posebno skidati i instalirati.

6.4. Bootstrap CSS

CSS (engl. Cascading Style Sheets) je jezik za opis izgleda i formatiranje dokumenta napisanog u jeziku za označavanje, uglavnom HTMLu. Uz HTML i JavaScript, CSS je korijen svih web aplikacija današnjice. CSS je primarno dizajniran kako bi se odvojilo postavljanje boje, fonta i rasporeda od same prezentacije stranice.

U aplikaciji je korišten Bootstrap CSS. Bootstrap je skup već napisanih CSS i JavaScript kodova koji se jednostavno dodaju na web stranicu. Bootstrap stranici daje jednostavan i atraktivan izgled, upravo ono što većina aplikacija i treba. Nudi razne tablice, gumbe, animacije, teme. Kako bi ga dodali na našu stranicu potrebno je skinuti ga s interneta te proučiti primjere korištenja koji su dani na službenoj stranici.

7. Zaključak

U ovom radu opisan je i uspješno implementiran sustav za detekciju plagijarizma među izvornim kodovima, nazvan Plagijator. Također je dan pregled najčešće korištenih sustava za detekciju plagijarizma te su uz svaki navedene kratke upute kako ga koristiti. Plagijator je napravljen kako bi pomogao profesorima, asistentima i ostalom fakultetskom osoblju u otkrivanju pokušaja studentskih prepisivanja.

Plagijator je pokazao izvrsnu detekciju nad većinom najčešćih pokušaja plagiranja, ali postoje stvari koje se još mogu dodati kako bi detekcija bila i uspješnija. Prva stvar koju treba dodati je detektiranje većeg broja jezika te je u kodu ostavljena mogućnost za lako dodavanje programskih jezika. U radu je spomenuta funkcija granice, ona je bitan faktor sustava te ostaje otvoreno pitanje koja funkcija bi bila najbolja za tu zadaću. Spomenute su i programske petlje i kako Plagijator ne zna detektirati izmjenu for petlje u while petlju pa je to jedna od stvari koju bi trebalo dodati u sustav. Ideja kako to napraviti bi možda mogla biti slična kao s varijablama, zamjena svih programskih petlji u jednak oblik. Također moguće je proširiti sustav s bazom podataka u kojoj bi se čuvali svi prethodno poslani izvorni kodovi, te bi se novo poslani kodovi uspoređivali i s njima. Integriranje sustava u postojeće sustave za predaju studentskih rješenja laboratorijskih vježbi te automatska provjera plagijata je još jedna ideja koja bi se mogla ostvariti s postojećim sustavom.

8. Literatura

- [1] Yusof, D. S., The Internet TESL Journal, Vol. XV, No. 2, 2009
- [2] Martin, B., Plagiarism: A Misplaced Emphasis, Journal of Information Ethics, 1994, str. 36-47
- [3] Moss, <http://theory.stanford.edu/~aiken/moss/>, 25.4.2015.
- [4] Aiken, A., Wilkerson D.S., Schleimer, S., Winnowing: Local Algorithms for Document Fingerprinting, 2003
- [5] Turnitin, <http://turnitin.com/>, 27.4.2015.
- [6] Plagium, <http://www.plagium.com/>, 5.5.2015.
- [7] Dupli Checker, <http://www.duplichecker.com/>, 15.5.2015.
- [8] Ramabhadran, S., Ratnasamy, S., Hellerstein J. M., Shenker S, Prefix Hash Tree An Indexing Data Structure over Distributed Hash Tables, 2005
- [9] Suffix Trees, <http://www.cise.ufl.edu/~sahni/dsaaj/enrich/c16/suffix.htm>, 2.6.2015.

9. Sažetak

Sustav za detekciju plagijarizma izvornog koda

Plagijarizam izvornog koda je sveprisutni problem kako u obrazovanju tako i u privredi. U sklopu završnog rada istraženi su i objašnjeni načini na koji funkcioniraju postojeći sustavi za detekciju plagijarizma. Također je osmišljen i projektiran sustav koji može uspješno detektirati plagijate među velikim brojem izvornih kodova (na primjer, studentska rješenja laboratorijskih vježbi) te omogućuje detaljan uvid u pojedine parove rješenja koje sustav smatra sličnima. Sustav je implementiran kao web aplikacija u Flask radnom okviru, koji je dio Python programskog jezika.

Ključne riječi: plagijat, detekcija plagijata, Flask, izvorni kod

Source code plagiarism detection system

Plagiarism of source code is omnipresent problem in education and economy. As part of this paper existing systems for plagiarism detection were explored and explained. Also, system for successful plagiarism detection between large number of source codes(for example, student laboratory assignment solutions) was built. The system allows detailed view of solution pairs for which system thinks are similar. Implementation of application was in Flask framework, which is part of Python.

Keywords: plagiarism, detection of plagiarism, Flask, source code