ALMA MATER STUDIORUM – UNIVERSITA' DI BOLOGNA

CAMPUS DI CESENA

SCUOLA DI SCIENZE

CORSO DI LAUREA IN SCIENZE E TECNOLOGIE INFORMATICHE

TITOLO DELLA RELAZIONE FINALE

PANORAMICA SUI SOFTWARE ANTIPLAGIO

Relazione finale in

Programmazione

Relatore

Prof. Antonella Carbonaro

Presentata da

Alessio Tarabelli

Sessione II

Anno Accademico 2015/2016

INDICE

Introduzione

Negli ultimi anni grazie all'avvento di internet, che in un'apparente annullamento delle distanze fisiche ha reso enormi quantità di informazioni facilmente accessibili, il fenomeno del plagio si è amplificato, toccando, come verrà citato più avanti, ambiti diversi come quelli della politica, dell'economia e i livelli più avanzati della ricerca tecnologica.

Sempre più persone propongono il loro lavoro come originale pur avendolo ripreso da articoli, blog, studi e ricerche trovati in rete.

La materia è da oltre un secolo oggetto di norme nazionali e sovranazionali, ma oggi è opportuno chiedersi se le regole create decenni fa, possano ancora efficacemente tutelare le odierne dinamiche della creatività e dei prodotti della conoscenza e in che modo, le potenzialità dell'informatica, possano piuttosto essere lo strumento idoneo per difendere l'originalità di un'opera e per smascherare ogni tipologia di plagio.

La risposta a questa domanda è insieme un impegno e una sfida, entrambe si intrecciano nella distinzione tra forma e contenuto di un'opera, tra l'individualità del processo creativo e il meccanismo generalizzato delle logiche di mercato. In questo incontro/scontro i programmi informatici studiati per combattere il plagio possono essere una risposta per intraprendere un percorso di tutela e salvaguardia del frutto dell'ingegno umano.

Da tempo le istituzioni, soprattutto le università, sono corse ai ripari per cercare di arginare il fenomeno del plagio.

Data l'impossibilità di effettuare manualmente il confronto tra il testo sotto esame e tutti i testi presenti in rete, è nata l'esigenza di creare dei programmi che facciano questo confronto in modo veloce e completo.

Alcuni software antiplagio sono stati prodotti appositamente per le università, altri si possono trovare su internet disponibili a tutti, altri ancora sono a pagamento ma prodotti da società terze.

In questo testo saranno analizzati diversi software antiplagio sotto diversi aspetti: per esempio, considerando le varie metodologie di rilevamento del plagio, il plagio nelle tesi universitarie, così come negli applicativi, (ricerca quest'ultima ancora più difficile) l'efficacia che hanno questi programmi nel riscontrare tale fenomeno, l'efficienza (cioè il tempo nel quale l'applicativo riscontra il plagio).

Per quanto riguarda le università è necessario considerare la quantità di testi a cui un database ha accesso e che può quindi confrontare con il documento sottopostogli. Per quanto riguarda invece il plagio nel codice sorgente non è possibile un confronto tra codice originale e codice sospetto, ma si deve considerare che un programma sospetto subisce delle trasformazioni per essere analizzato meglio dai software antiplagio essi infatti si attiveranno cercando differenti tipi di plagio in modi diversi.

A conclusione di queste righe introduttive si è scelto di proporre uno stralcio tratto dal codice etico dell'Alma Mater Studio rum – Art. 27 comma 3.

'L'Università non ammette alcuna forma di plagio e disonestà intellettuale, sia essa intenzionale o derivante da condotta negligente o dall'abuso della posizione gerarchica o d'influenza accademica.

Integrano fattispecie di plagio la parziale o totale attribuzione a se stessi o l'appropriazione della titolarità di progetti, idee, risultati di ricerche o invenzioni appartenenti ad altri, nonché l'attribuzione della paternità di un'opera dell'ingegno ad un autore diverso da quello reale. Il plagio include l'omissione e la falsificazione nella citazione delle fonti e prescinde dall'uso della lingua con la quale i prodotti scientifici sono presentati o divulgati.' [(1)]

Qui di seguito vengono riassunti i contenuti trattati nei successivi capitoli.

Capitolo 1: viene presentata l'etimologia della parola plagio e l'evoluzione del concetto di plagio e di diritto d'autore.

Capitolo 2: viene il esposto il quadro normativo nazionale ed europeo in materia.

Capitolo 3: vengono presentati due esempi, uno riguarda l'ex premier Mario Monti e l'altro una denuncia di plagio a carico della Apple; viene inoltre presentato come, una università degli Stati Uniti, ha inteso sensibilizzare docenti e studenti al problema del riconoscimento del diritto d'autore.

Capitolo 4: si espone il pensiero delle università italiane sul fenomeno.

Capitolo 5: si parla del software Compilatio e di come prevenire il plagio.

Capitolo 6: si descrivono le varie tecniche di approccio al problema dal punto di vista testuale.

Capitolo 7: si mettono a confronto le prestazioni dei metodi descritti nel capitolo precedente.

Capitolo 8: si descrivono le caratteristiche che deve avere un software antiplagio per rilevare il plagio nei testi.

Capitolo 9: si parla, nel dettaglio, di Citeplag un programma open-source per il rilevamento del plagio nei testi.

Capitolo 10: è un introduzione al fenomeno del plagio nei software.

Capitolo 11: si descrivono 5 software antiplagio più utilizzati.

Capitolo 12: si presentano i metodi di individuazione che implementano questi software.

Capitolo 13: viene descritta una comparazione tra di essi.

1. Evoluzione storica del plagio

Il termine plagio deriva dal latino plagium (furto, rapimento) che a sua volta deriva dal greco plàgiom (sotterfugio). All'inizio del '500 viene, per la prima volta, citato in un vocabolario di termini latini con un significato che si avvicina molto a quello attuale e cioè 'una persona che ruba libri tenendoli come suoi'.(11)

Attualmente, con il termine plagio, nel diritto d'autore, ci si riferisce 'all'appropriazione, tramite copia totale o parziale, della paternità di un'opera dell'ingegno altrui'. In tale accezione, il termine trova riscontro nell'inglese plagiarism e nel francese e tedesco plagiat.

La Convenzione di Berna, nel 1886, fu la prima convenzione internazionale a stabilire il riconoscimento reciproco del diritto d'autore tra le nazioni aderenti. [16]

Tuttavia, la legge sul diritto d'autore italiana, come le altre leggi dei principali sistemi giuridici occidentali, non contiene la parola 'plagio' nomina invece, il diritto d'autore e, senza definirle, le nozioni di creatività, originalità e contraffazione.

In epoche remote, la produzione delle opere dell'ingegno umano era protetta e finanziata dal mecenatismo, (se pur ad esso sottomessa) e il compenso percepito dall'ideatore prescindeva da ogni legame tra questi e il prodotto.

Nell'evoluzione economica del mercato si è poi '*passati dal mecenatismo, alla vendita*', e quindi alla necessità che i 'fruitori/consumatori', potessero agevolmente riconoscere l'ideatore dell'opera, attraverso l'attribuzione della paternità dell'opera stessa.

La successiva epoca post-industriale coincide poi con il 'declino dell'anonimato degli autori. Citando il nome dell'autore così come si cita quello di un fabbricante si stabilisce un'identità di marchio.... (e si tutela l'opera dell'ingegno)'. (10)

Oggi, nell'epoca postmoderna, dominata dall'informatica e dalle reti di dati, queste nuove tecnologie possono purtroppo contribuire a determinare, la diffusione e l'utilizzo incontrollato del prodotto dell'ingegno umano.

Tuttavia, per quanto riguarda l'informatica è doveroso sottolineare che: '*Il software stesso, bene primario..... dell'era digitale, viene scritto a partire dall'assemblaggio di moduli standardizzati di codice. E la potenza generatrice della comunità dei coders..... risiede proprio in queste pratiche di riutilizzo. Senza la possibilità di attingere al passato, i programmatori sarebbero costretti a duplicare gli sforzi rallentando e, forse, peggiorando i risultati del loro lavoro*.[3]

La realtà è che la società di oggi, come le società del passato, ha bisogno della copia e dell'imitazione così come ha bisogno dell'originalità. La creatività si esprime in entrambe le direzioni. E, ancora, l'insegnamento e l'apprendimento, tramite i quali si pongono le basi per il progresso della conoscenza, non possono fare a meno né dell'imitazione né dell'originalità. Lo stesso mercato oscilla tra queste due forze.' (3)

2. Il plagio nella legislazione

2.1 Quadro normativo italiano

In Italia la legge che per prima ha inteso regolamentare il diritto d'autore è la legge 633/41, poi modificata con la legge 248/00, essa prende in esame le nuove tipologie di diffusione a distanza, come la TV via cavo ecc., oltre a definire e regolamentare il diritto d'autore nello scenario dei moderni contenuti multimediali, 'al fine di combattere la pirateria e la contraffazione, anche quella che si realizza via Internet'.

La normativa italiana così definisce il diritto d'autore (Art. 1 Legge 633/41 e Legge 248/00):

'Tutte le opere dell'ingegno di carattere creativo che appartengono alle scienze, alla letteratura, alla musica, alle arti figurative, all'architettura, al teatro e alla cinematografia qualunque ne sia il modo o l'espressione, formano oggetto del diritto d'autore.

Sono altresì protetti i programmi per elaboratore come opere letterarie ai sensi della convenzione di Berna sulla protezione delle opere letterarie ed artistiche ratificata e resa esecutiva con legge 20 giugno 1978, n. 399.'

L'articolo 2 elenca ulteriormente le opere tutelate, aggiornandole alla luce delle moderne tecnologie e, al comma 8 recita:

'I programmi per elaboratore, in qualsiasi forma espressi purché originali quale risultato di creazione intellettuale dell'autore.

Restano esclusi dalla tutela accordata dalla presente legge le idee e i principi che stanno alla base di qualsiasi elemento di un programma, compresi quelli alla base delle sue interfacce.

Il termine programma comprende anche il materiale preparatorio per la progettazione del programma stesso'.

La legge 248/00 ha inoltre previsto particolari ipotesi di reato per i casi di contraffazione e pirateria informatica aventi ad oggetto anche i programmi per elaboratori.

La violazione delle norme sul diritto d'autore comporta sanzioni anche penali e di particolare gravità, soprattutto se chi utilizza illegittimamente l'opera altrui lo fa con fini di lucro.

Per maggior precisione occorre tuttavia distinguere tra i diritti riconducibili all'utilizzazione economica dell'opera, che potrebbero far capo a un soggetto

diverso dal suo autore (molti programmatori, infatti, sono legati da un rapporto di lavoro con le società software, alle quali spettano quindi tutti i diritti di distribuzione e di utilizzazione economica) e quelli riferiti alla paternità dell'opera, che deve sempre essere riconosciuta al solo autore (art. 2576 c.c.).

2.2 Direttive europee in materia di plagio

Anche l'Unione Europea ha legiferato sul tema della tutela del diritto d'autore, la Direttiva 2001/29/CE datata 22 maggio 2001 si è occupata di armonizzazione di taluni aspetti del diritto d'autore e dei diritti connessi nella società dell'informazione, così come indica l'articolo 1.

'L'armonizzazione proposta contribuisce all'applicazione delle quattro libertà del mercato interno (libertà di circolazione di beni, servizi, persone, capitali n.d.r.) e riguarda il rispetto dei principi fondamentali del diritto e segnatamente della proprietà, tra cui la proprietà intellettuale, della libertà d'espressione e dell'interesse generale'.

L'articolo 6 precisa ulteriormente dicendo:

'.....l'impatto di tali differenze ed incertezze normative diverrà più significativo con l'ulteriore sviluppo della società dell'informazione che ha già incrementato notevolmente lo sfruttamento transfrontaliero della proprietà intellettuale. Tale sviluppo è destinato ad accrescersi ulteriormente. L'esistenza di sensibili differenze e incertezze giuridiche in materia di protezione potrebbe ostacolare la realizzazione di economie di scala per i nuovi prodotti e servizi.....'

L'articolo 11, inoltre raccomanda: 'Un sistema efficace e rigoroso di protezione del diritto d'autore e dei diritti connessi è uno dei principali strumenti in grado di garantire alla creazione e alla produzione culturale europea le risorse necessarie nonché di preservare l'autonomia e la dignità di creatori e interpreti o esecutori'.

La direttiva 2014/26/UE del 26 febbraio 2014 del Parlamento Europeo, intende inoltre fornire indicazioni sulla gestione collettiva dei diritti d'autore e dei diritti connessi e sulla concessione di licenze multiterritoriali per i diritti su opere per l'uso online nel mercato interno.

La direttiva richiede che gli stati membri riconoscano ad autori, artisti, produttori, organismi di diffusione il diritto di produzione, comunicazione e distribuzione delle loro opere, prevede l'istituzione di 'collecting societies', organismi di

gestione collettiva delle royalties a cui i soggetti di diritto possano aderire e da cui ricevere, tempi certi, la distribuzione dei compensi a cui hanno diritto. Questa direttiva dovrà essere recepita dal Parlamento Italiano entro il 10 aprile 2016.

In sintesi ogni opera dell'ingegno appartiene al proprio autore e non è consentito beneficiarne senza il consenso dello stesso.

L'indicazione del copyright, che si trova su molti siti, rafforza e rende esplicita la protezione dell'opera, ma anche in mancanza non ci si deve sentire autorizzati a copiare o riprodurre parti delle opere che si trovano sulla rete.[(12)]

3. Alcuni casi di plagio nell'attualità

3.1 Un esempio di plagio in politica

Un software anti plagio – Turing – è stato reso operativo nell'Università Bocconi di Milano dal 2011: nello stesso anno il suo ex rettore, Mario Monti, che lasciò l'ateneo per assumere la guida del Governo, copiò alcuni passaggi del programma fiscale dalle relazioni di Bankitalia senza citarne la fonte. Un retroscena rivelato dal quotidiano Libero che non ci risulta sia stato smentito.

Usato in chiave politica, un software antiplagio sarebbe stato utile per scoprire che l'ex premier Monti era stato "ispirato" dal giuslavorista Pietro Ichino per esprimere le idee della sua agenda. Nessuna smentita dal professore e Ichino intervistato da Radio 24 ha dichiarato che il suo documento era online da mesi.

Il Turing avrebbe funzionato anche per la neo ministra della Salute Beatrice Lorenzin che, al primo esordio ufficiale a un congresso di medici, ha presentato un programma con parti interamente copiate dal Libro bianco dell'ex ministro, Maurizio Sacconi. Anche in questo caso, la fonte non è stata indicata. (6)

3.2 La Apple accusata di plagio in Cina.

Il gigante informatico e l'azienda cinese che l'ha denunciata sono comparsi oggi per la prima volta, davanti ad un tribunale di Shanghai. La cinese Zhizhen Network Technology sostiene che Cupertino abbia copiato il suo prodotto per creare 'Siri', il sistema di riconoscimento vocale degli iPhone. La Shinshen afferma di aver sviluppato il software nel 2004, e di averlo usato per il suo Robot chiamato 'Xiao i'. Siri è stato messo sul mercato nel 2007. (2)

3.3 Decalogo Anti Plagio Della Northwestern University

Le nuove generazioni dei cosiddetti 'nativi digitali' stanno crescendo nella convinzione che tale libertà sia connaturata al nuovo mondo tecnologico. Sintomatico di ciò è quanto compare sul sito della Northwestern University di Evanston, Illinois USA.

'*In tutti gli ambienti accademici, specialmente nella produzione di testi, costruiamo il nostro lavoro basandoci sulle conoscenze, convinzioni, intuizioni e, soprattutto, sulle parole di altri. Uno scrittore coscienzioso sa sempre distinguere*

chiaramente, in ciò che sta scrivendo, cosa ha imparato da altri da cosa egli stia contribuendo personalmente al lettore di capire. Per evitare il plagio è importante comprendere come attribuire al legittimo proprietario, parole e idee che state usando.' (9)

Sul sito dell'università compare anche una intera e approfondita sezione dedicata a consigli e disposizioni a studenti e insegnanti per evitare episodi di plagio, ma anche e soprattutto con l'intento di educare alla responsabilità.

Il vademecum destinato agli insegnanti viene qui riportato nei punti essenziali, allo scopo di presentare un approccio 'culturale' al problema:

1. Comprendere come mai gli studenti barano: molti studenti, semplicemente non sanno cosa sia il plagio, credono che tutto ciò che sia su Internet sia di pubblico dominio, oppure che sostituendo una parola o ricopiando meno di dieci parole, non sia necessario citare la fonte.
2. Alcuni studenti sanno cosa sia il plagio ma non lo considerano sbagliato
3. Gli studenti sono degli 'economi' naturali e 'cercano la via più breve' per raggiungere risultati;
4. Molti studenti non attribuiscono importanza al processo di apprendimento, ma solo al risultato finale (voto di esame o di laurea);
5. Se si struttura il lavoro di ricerca in modo che le parti intermedie di esso (tema, prime ricerche, prospetto, contorno, brutta copia, la bibliografia, i progetti finali) vengano consegnate a intervalli regolari, gli studenti ricorreranno con meno probabilità a contenuti 'preconfezionati'
6. Gli studenti amano trasgredire le regole: maggiormente condannerete il plagio, più velocemente dovrete attendervelo dai vostri studenti
7. Educate voi stessi contro il plagio, informandovi sugli strumenti informatici e le opportunità offerte dal web, per rilevarlo.
8. Educate i vostri studenti contro il plagio e non date per scontato che lo conoscano anche se, alle vostre domande, faranno gesti di assenso. Mettete bene in chiaro le penalità che verranno applicate se dovessero incorrere in episodi di violazione del diritto d'autore.

9. Discutete con loro sull'opportunità di citare le fonti. Valorizzate il rispetto dovuto agli autori di un'opera e il fatto che, citandone la fonte, essi rivelano di avere efficacemente attinto alla pertinente letteratura scientifica.
10. Usate un dispositivo contro il plagio.

Strategie per rilevare il plagio:

1. Mescolanza di stili
2. Mancanza di riferimenti o citazioni
3. Formattazione inusuale
4. Testo fuori tema
5. Mancanza di segni di attualità e anacronismi
6. Anomalie nel tono di scrittura
7. Anomalie ortografiche e di lessico
8. Pistola fumante (evidenti incongruenze e/o dimenticanze) [(9)]

4. Plagio in ambiente universitario

Per verificare che tesi o relazioni non fossero state plagiate, inizialmente si confrontava il testo sotto esame e molti altri documenti ma questo era dispendioso e richiedeva una ottima dose di memoria, oggi è possibile utilizzare software a questo predisposti, che sono evidentemente molto più efficienti e efficaci.

Una delle pratiche per evitare che il plagio venisse scoperto era quella di sostituire un numero di parole sufficienti ad eludere il software, ed è conosciuta come rogeting.

Le università coinvolte sono numerose, ognuna mette a disposizione alcuni software per la verifica dei testi, sia per gli studenti che per i docenti, qui di seguito riporterò in versione integrale o in sintesi le considerazione di alcune università italiane in merito al fenomeno del plagio:

4.1 Università di Verona

L'Università di Verona, in considerazione della rilevanza sociale della ricerca scientifica, ritiene che i risultati di questa debbano contribuire allo sviluppo e al benessere della comunità; i diritti economici inerenti alla proprietà intellettuale si presumono attribuiti all'Università di Verona nell'ambito di un rapporto reciproco di condivisione degli obiettivi riguardanti l'utilizzazione dei risultati della ricerca tra docenti, ricercatori e personale a vario titolo in essa operante.

Il plagio è la parziale o totale attribuzione di idee, opere, risultanze di ricerche o scoperte altrui a se stessi, a prescindere dalla lingua in cui queste sono ufficialmente presentate o divulgate. Il plagio può essere intenzionale o l'effetto di una condotta non diligente.

L'Università di Verona condanna ogni forma di plagio e pone particolare attenzione ai casi di plagio da parte degli studenti al momento della presentazione di elaborati e di tesi.

L'Ateneo Veronese persegue, allo stesso tempo, i casi di plagio da parte del personale docente nei confronti di elaborati o tesi originali, costituenti il risultato dell'attività di ricerca di laureandi o laureati, dottorandi o dottori di ricerca, specializzandi o specializzati.

Per la protezione della proprietà intellettuale, la promozione del libero accesso alla letteratura scientifica e il divieto di plagio si fa riferimento a quanto contenuto negli appositi Regolamenti di Ateneo.[15]

4.2 Università “La Sapienza”

Le presenti linee guida si prefiggono l’obiettivo di rendere noti i comportamenti che ne determinano la configurazione e le possibili conseguenze che ne derivano, sia in linea generale, che nello specifico nell’ambito universitario.

Ogni comunità accademica, tenuta in alta considerazione la rilevanza sociale della ricerca scientifica, ritiene che i relativi risultati debbano contribuire allo sviluppo e al benessere della collettività, garantito anche attraverso la tutela della proprietà intellettuale nei modi previsti dalla normativa vigente. Viene, pertanto, condannato il plagio in tutte le sue possibili manifestazioni e si invitano tutti i soggetti interessati a far sì che le attività accademiche di rilievo scientifico e di ricerca indichino specificamente il contributo dei singoli componenti.

E’ dunque necessario che, nel raggiungere i propri risultati scientifici e di ricerca, lo studente operi secondo integrità, onestà, professionalità, libertà.

Costituisce plagio lo sfruttamento totale o parziale dell’idea altrui espressa attraverso elementi caratterizzanti simili.

Il plagio può essere intenzionale o conseguente a una condotta non diligente e consiste, quindi, nell’illegittima appropriazione, presentandola come propria, dell’altrui opera intellettuale.

Il plagio può essere riscontrato anche in un semplice lavoro riepilogativo ed espositivo (le cd. tesi compilative), laddove manchi quello sforzo di ripensamento delle problematiche altrui, l’espressione personale nell’elaborato nonché, ad ogni modo, i riferimenti dei testi scritti da cui si prende spunto per la presentazione di una tesi affermata come propria.

Si ricorda, pertanto, agli studenti che ogni elaborato prodotto durante il percorso universitario di studi (prove scritte d’esame, tesi triennali e magistrali, tesine, ecc) non deve essere in nessuna sua parte frutto di plagio.

Esempi di plagio sono:

- Frasi copiate senza indicare la fonte;
- Frasi scritte da autori non virgolettate;
- Parafrasi di un testo (anche breve) senza indicarne la fonte;
- Frasi, paragrafi, pagine e testi copiati da colleghi;

- Parti di testi scritti e di pagine web riprese da siti internet senza che ne venga indicata la fonte.

Con la proliferazione di materiale reperibile sul Web, il plagio è diventato il c.d. facile "copia e incolla", che si è cercato di rinvenire attraverso metodi differenziati.(14)

Ciò può avvenire per esempio:

a) Mediante l'utilizzo di applicazioni online gratuite che non richiedono abbonamenti o iscrizioni per controllare documenti di tipo elettronico. Un buon verificatore di plagio fornirà una funzione di confronto dei testi e permetterà di vedere quali porzioni di testo sono state copiate.
b) Mediante l'utilizzo di alcuni dei più famosi motori di ricerca, attraverso cui, inserendo le frasi o i brani di testo, questi effettuano una ricerca e un confronto nei loro database.
c) Mediante servizi su abbonamento, sempre più popolari tra gli insegnanti, volti alla prevenzione dei casi di plagio.

E' assolutamente lecito utilizzare le opere di ingegno altrui per essere impiegate nelle proprie argomentazioni oppure per sostenere elaborazioni di nuove prospettive o per confutare le tesi precedentemente riportate, purché i testi utilizzati vengano impiegati secondo regole precise.

Quando si utilizzano le opere scritte da terzi, occorre, infatti, includere sempre i riferimenti dell'opera consultata e del suo autore, tramite una citazione diretta nel testo, o in nota, o tramite la bibliografia.

Come detto, esistono strumenti di supporto informatico che possono aiutare lo studente ad evitare il plagio mediante una preventiva verifica dei testi o parti di esso riportati all'interno dell'elaborato.

In ogni caso, l'essenziale è mantenere l'originalità dei testi utilizzati come fonte, che viene tutelata con il richiamo dell'opera utilizzata del suo autore e i dati riferibili alla pubblicazione.

Ogni qualvolta si vogliano utilizzare idee di altri autori ai fini del proprio scritto, deve sempre essere citata la fonte. Esistono diversi modi per citare i riferimenti bibliografici.

Nel caso in cui si presentino le idee altrui citando esattamente le stesse parole dell'autore, occorre usare sempre le virgolette e includere il riferimento bibliografico con una nota a piè pagina o una nota di chiusura di pensiero (tra parentesi), in cui viene riportata la fonte, specificando anche la pagina in cui è reperibile il pensiero citato, gli estremi dell'edizione ed eventualmente della traduzione.
Anche nel caso si riporti il pensiero di un autore, senza adoperare però le stesse parole da esso utilizzate, è necessario citare il riferimento con le medesime modalità.

La bibliografia deve riportare tutte le opere consultate al fine di redigere la tesi/tesina, sia nel caso in cui tali opere siano state citate oppure semplicemente utilizzate per costruirne l'argomentazione.
Per ogni riferimento devono essere indicati:
a) autore/i o curatore/i;
b) anno di pubblicazione;
c) titolo;
d) rivista (se si tratta di articolo) o Volume (se si tratta di capitolo) in cui il contributo è stato pubblicato;
e) pagine, nel caso di articolo su rivista o capitolo di libro;
f) editore o edizione;
h) traduzione, nel caso di opera scritta in altra lingua ma a cui si faccia riferimento in italiano. (14)

4.3 Università Bocconi di Milano

La Bocconi è impegnata a combattere ogni forma di plagio. In particolare, il lavoro finale e la tesi sono le attività conclusive di un corso di studi e rappresentano lo strumento con cui lo studente dimostra la sua maturità e la sua capacità di coordinare concetti e nozioni appresi durante gli anni di studio.

E' indispensabile che i lavori prodotti dallo studente siano il risultato di un contributo personale, che sia prestata particolare attenzione alle citazioni e che non vi siano copiature di testi recuperati da altre fonti. A questo riguardo l'Università Bocconi si è dotata di un software specifico adatto alla verifica di eventuali copiature o di un uso improprio delle citazioni.

Si invitano pertanto gli studenti a prendere visione della corretta modalità per effettuare citazioni, indicata nelle guide per la realizzazione dei lavori finali dei trienni, Giurisprudenza e Laurea Magistrale.[13]

5. Come e perché prevenire il plagio

5.1 COMPILATIO

Compilatio è uno dei programmi attualmente più utilizzati da docenti e anche dagli studenti per verificare se i loro testi sono ammissibili.

Secondo il punto di vista dei produttori del programma <<Gli studenti sono spesso valutati dagli elaborati che dovrebbero portare i loro autori a cercare conoscenze, acquisirle, appropriarsene, e restituirle tramite scritti.

Internet è ad oggi la fonte di documentazione privilegiata degli studenti. Purtroppo, questo mezzo facilita anche il riutilizzo dei documenti trovati senza effettuare alcuna modifica, grazie alla funzionalità "copia-incolla". Capita spesso che uno studente non faccia apparire, segnalandola come citazione, un brano preso da un altro autore. Peggio, alcuni studenti si appropriano senza vergogna della paternità di paragrafi interi di testo copiato da altri documenti. Tali pratiche costituiscono un atto di plagio.

Anche se 4 studenti su 5 ammettono di ricorrere al "copia-incolla" nei propri compiti, non tutti hanno un'anima imbrogliona. Gli studi fatti da Compilatio in partnership con numerose università ed istituti superiori, permettono di spiegare questo massivo ricorso al plagio: oltre alla mancanza di valore etico, ciò che seduce gli studenti è la facilità che Internet offre, di trovare e riutilizzare i testi. Ogni individuo tende a prendere la via più facile per finalizzare il proprio compito. La stessa cosa succede nel caso in cui uno studente cerchi di finire l'elaborato il prima possibile. Ad oggi, nulla frena l'uso del 'copia-incolla', che gli studenti riconoscono comunque come pratica proibita e punibile.>>[(4)]

Per concludere ci sembra che i progettisti di Compilatio, intendano suggerire che gli studenti dovrebbero essere maggiormente coinvolti nell'azione di prevenzione del plagio e responsabilizzati.

Questo programma è utilizzato in diverse università (Bocconi di Milano, La Sapienza, Ca' Foscari, università di Verona, università di Cagliari, eccetera), lo possiamo trovare in tre formati diversi (magister, studium, copyright):

Magister fornisce le seguenti informazioni:

- Percentuale globale di similitudini
- Parti del documento ritrovate in modo identico
- Fonti usate nel documento

Studium, invece, fornisce solo il primo tipo e il terzo tipo di informazioni.

La prima versione è mirata alle esigenze degli insegnati la seconda a quelle degli studenti, invece la versione copyright è indicata per professionisti, allo scopo di verificare l'originalità del loro lavoro.

Compilatio.net, nella fase di confronto dei testi, utilizza la tecnica di *string matching*, quando si carica il proprio documento sulla piattaforma online messa a disposizione degli utenti registrati, esso viene messo a confronto con i dati presenti su Internet e con il database presente nel sistema.

Compilatio supporta diversi formati:

- Testo ".txt"
- Adobe Acrobat ".pdf"
- Testo arricchito ".rtf"
- Trattamento testo ".doc",".docx",".odt"
- Foglio Elettronico ".xls",".xlsx"
- Proiezione di diapositive ".ppt",".pptx"
- File ".html"
- File ".php"
- File ".asp"

<<Oltre alle basi dati, il software COMPILATIO.NET funziona come un meta-motore, cioè interroga i diversi strumenti di ricerca e centralizza i risultati ottenuti. Per trovare una fonte, COMPILATIO.NET ripercorre lo stesso cammino compiuto dall'autore del documento analizzato.>>[(5)]

Se l'autore ha saputo trovare il documento su Internet, anche COMPILATIO lo troverà.

Le informazioni relative a questo software, sono scarsamente accessibili, sia dal punto di vista qualitativo che quantitativo, in quanto esso è protetto da copyright e pertanto tutelato dalla legge sul diritto d'autore.

6. Metodologie di approccio al problema

Due possono essere le maniere di analizzare un documento sospetto a livello di testo: in modo **esterno** e in modo **intrinseco** quindi il programma che si utilizza per analisi deve implementare uno di questi due approcci.

I sistemi di rilevamento esterni confrontano il testo da esaminare con un gruppo di documenti che sappiamo, con sicurezza, essere originali.

Sulla base di alcuni criteri di somiglianza predefiniti, il compito del sistema di rilevamento è quello di recuperare i documenti che contengono un grado di somiglianza superiore ad una soglia precedentemente indicata al documento sospetto. I sistemi intrinsechi non comparano il documento con dei testi esterni ma mirano a riconoscere, come indicatore di potenziale plagio, i cambiamenti nello schema di scrittura, i sistemi non sono però in grado di identificare il plagio in modo affidabile quindi è necessario l'apporto umano. Le somiglianze sono calcolate su dei modelli predefiniti e potrebbero portare a dei falsi positivi.

L'immagine seguente classifica tutti gli approcci per il rilevamento del plagio nei testi suddividendoli nei due tipi precedentemente discussi.

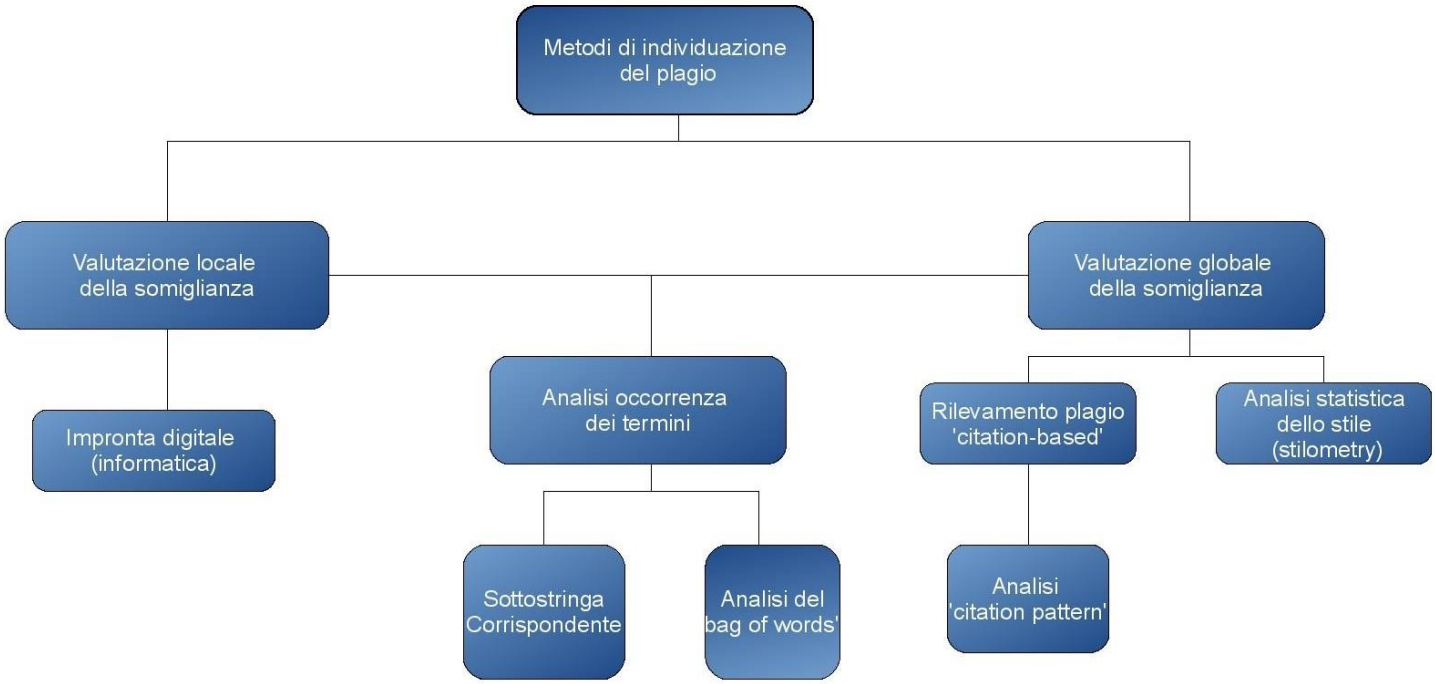


Grafico 1 [17]

6.1 Fingerprint(impronta digitale)

È il metodo più utilizzato attualmente. Questo metodo seleziona delle sottostringhe di testo che andranno a comporre l'impronta digitale di quel documento, questa operazione viene eseguita sia per il documento sospetto che per ogni altro documento con il quale quest'ultimo viene confrontato.
Il fingerprint si basa su delle funzioni *hash* che possono portare ad avere dei duplicati se si generano impronte troppo piccole, perciò devono essere opportunamente dimensionate senza tralasciare il fatto che, se una impronta è troppo grande, richiederà più tempo per il confronto con le altre.
Quando le sottostringhe di due impronte a confronto si somigliano in una quantità superiore ad una certa soglia predefinita allora è possibile che sia stato rilevato un plagio.
Questo metodo è classificato come locale o intrinseco.

6.2 Substring matching

È un altro metodo molto diffuso in informatica. Consiste nel confrontare delle stringhe, del testo sotto esame, con delle stringhe di altri testi e verificare se ci sono delle corrispondenze. Questo metodo utilizza la struttura dati degli alberi di suffisso che serve per risolvere in tempo lineare il problema del matching delle stringhe ciononostante non è adatto per un confronto con una raccolta di documenti numerosa perché porterebbe ad un costo computazionale troppo grande. Substring matching è un metodo sia locale che globale.

6.3 Bag of words

È un modello utilizzato anche in altri campi dell'informatica. I documenti confrontati vengono considerati esaminando le parole che hanno al loro interno senza ordine, le parole di un testo vengono inserite in un vettore in maniera casuale così viene effettuato anche per l'altro, infine si procede al confronto tra i due vettori con la tecnica euristica del "coseno di similitudine", esso misura le similitudini tra i due vettori calcolandone il coseno.
Anche questo è un metodo sia locale che globale.

6.4 Analisi delle citazioni

Questo è l'unico approccio che non si basa sulle somiglianze testuali. Vengono analizzate le citazioni riportate nel testo in esame e i riferimenti testuali per vedere se ci sono dei modelli simili. Questo approccio è piuttosto recente e viene utilizzato per testi scientifici o documenti accademici, ancora non viene adottato da nessun software commerciale. Questo metodo è globale.

6.5 Analisi stilometrica

Questo modello viene utilizzato per accertare lo stile con il quale un certo documento viene scritto e per assegnarne la paternità ad un certo autore quindi può essere utilizzato per determinare il plagio se lo stile di scrittura del testo sospetto fosse uguale o simile a quello di un autore diverso. Per verificare il plagio questo metodo utilizza delle funzioni statistiche.
Anche questo è un metodo è globale.

7. PERFORMANCE

Si valutano ora le prestazioni dei vari metodi di rilevamento del plagio.

Notiamo subito che le prestazioni dipendono dal tipo di plagio presente nel testo, fatta eccezione per l'analisi delle citazioni perché tutti gli altri metodi rilevano somiglianze di tipo testuale.

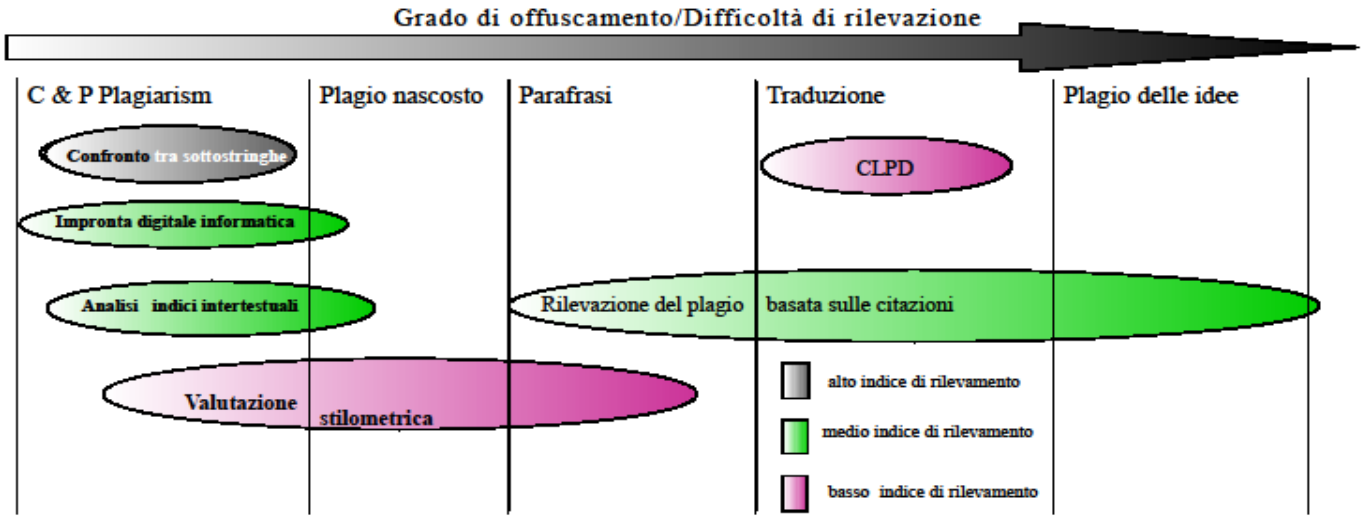


Grafico 2 [17]

È ovvio che la precisione dei metodi di rilevamento diminuisce più il plagio viene nascosto.

Il Substring Matching fornisce buone prestazioni in merito al rilevamento del copia-incolla, soprattutto se questo è poco nascosto, comunque anche altri metodi di tipo globale sono molto efficienti nell'individuare questo tipo di plagio.

I metodi di Fingerprinting e Bags of Words, nella ricerca del copia-incolla, sono inferiori al Substring Matching, ma competitivi, le loro prestazioni variano in base alla perdita di informazioni rispetto al testo originale.

Applicando delle strategie di frammentazione e selezione flessibile, essi sono in grado di rilevare il plagio mascherato meglio del metodo di Substring Matching.
Il rilevamento con il metodo di stilometria non è di tipo testuale come i precedenti ma di tipo linguistico quindi rilevando differenze di stile può essere adottata per il copia-incolla, il plagio mascherato e quello parafrasato.
Il confronto stilometrico rischia di fallire quando vengono confrontati segmenti molto parafrasati al punto da far avvicinare il testo originale allo stile della persona che sta commettendo il plagio o se un testo è stato compilato da molti autori.
L'analisi stilometrica funziona in modo affidabile solo per documenti con una lunghezza superiore alle migliaia di parole o decine di migliaia.
Si stanno svolgendo diverse ricerche per trovare un sistema in grado di rilevare i plagi derivanti dalla traduzione di un testo da una lingua ad un'altra.
Attualmente il CLPD (cross-language plagiarism detection) non è visto come un sistema efficiente, i risultati, infatti, non sono accettabili.
Il metodo basato sull'analisi delle citazioni è in grado di rilevare parafrasi e traduzioni in maniera più efficace rispetto ad altri approcci perché è indipendente dalle caratteristiche testuali. Tuttavia devono essere presenti nel testo sufficienti citazioni per poter far funzionare il sistema correttamente.
Questo metodo, di conseguenza, si limita ai testi accademici. [5]

8. Fattori che caratterizzano un software di rilevamento

Di seguito l'elenco dei fattori che caratterizzano un software di rilevamento:

- **Ambito della ricerca**: internet pubblico con motori di ricerca, database istituzionale, database specifico per il sistema locale
- **Tempo di analisi**: tempo trascorso da quando il documento sospetto è stato inviato a quando si hanno disponibili i risultati
- **Numero totale documenti / quantità documenti elaborati**: numero di documenti che il sistema può elaborare in una unità di tempo
- **Controllo intensità**: quante volte il software fa una richiesta esterna (per esempio ad un motore di ricerca) per cercare di individuare uno dei possibili tipi di plagio
- **Tipo di algoritmo**: gli algoritmi che definiscono come si effettua il confronto tra il testo sospetto e gli altri
- **Precisione e richiamo**: numero di documenti contrassegnati correttamente come plagiati rispetto al numero totale di documenti contrassegnati e al numero totale di documenti effettivamente plagiati. Alta precisione significa che sono stati trovati pochi falsi positivi mentre alto richiamo significa che alcuni falsi negativi non sono stati trovati. [17]

La maggior parte dei sistemi di rilevamento del plagio di grandi dimensioni, utilizzano grandi database interni (oltre ad altre risorse) che crescono con ogni documento che viene inviato per l'analisi.

I software anti-plagio generalmente sono web-based e sono closed-source con delle piccolissime eccezioni, ecco un elenco di alcuni di questi software:

Gratuiti	**Commerciali**
Chimpsky	Attributor
CitePlag	Copyscape
CopyTracker	PlagTracker
eTBLAST	Iparadigms Ithenticate, Turnitin
Plagium	PlagiarismDetect
SeeSources	PlagScan
The Plagiarism Checker	VeriGuide
	Ephorus
	URKUND

CitePlag e CopyTracker sono gli unici programmi open-source.

Analizzeremo nel prossimo paragrafo come funziona CitePlag.

9. Il software CitePlag

Questo programma è sviluppato in Java e si basa sul metodo di analisi delle citazioni. Confronta le citazioni del testo sospetto con le citazioni presenti nei testi del suo database interno. Il software è diviso in quattro componenti:
il parser, il database, il detector, il report generator.
Il parser estrae i dati bibliografici, come citazioni, riferimenti, autori e titoli dal documento e li memorizza nel database. Il database fornisce questi dati al detector che esegue degli algoritmi di analisi su di essi, una volta completata l'analisi manda indietro i risultati al database che li archivierà.
Il report generator recupera i risultati dal database e li rende disponibili per il controllo umano. [8]

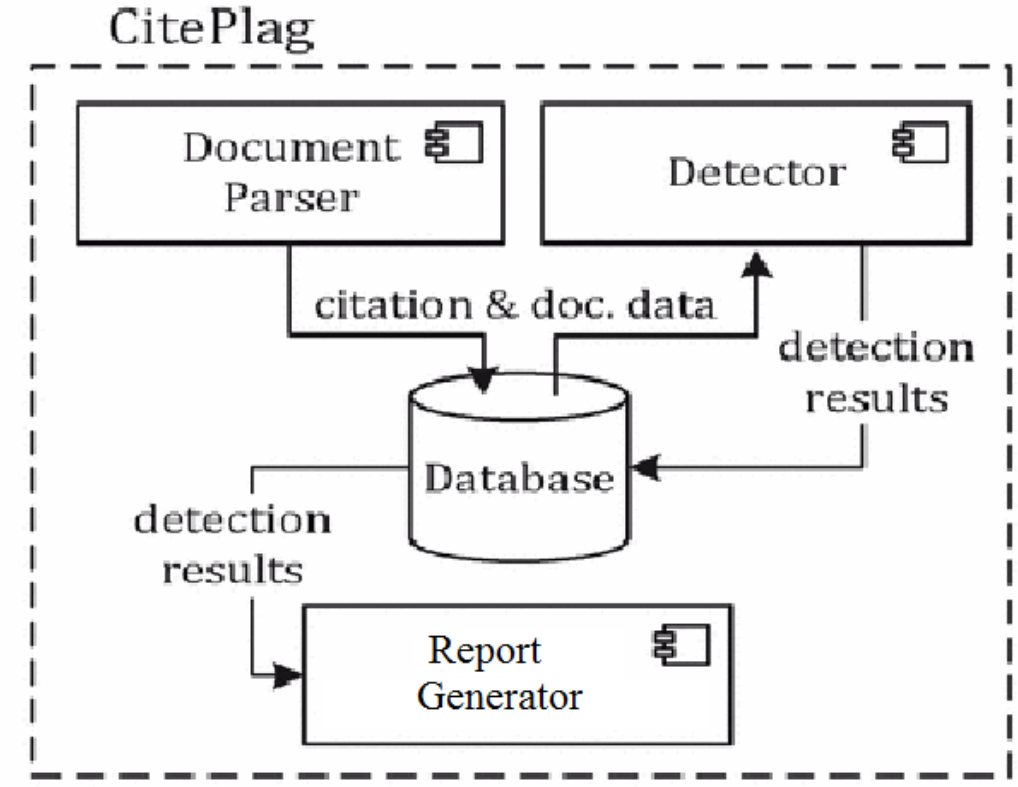


Grafico 3 [8]

9.1 Parser

Il compito del parser è di estrarre tutte le necessarie citazioni e referenze dal documento inserito e importarlo nel database.
La versione attuale di CitePlag è solo in grado di analizzare testi NXML.

Il principale compito nel parser è determinare la posizione esatta delle citazioni all'interno del testo. Il processo di rilevamento dei modelli di citazioni richiede la conoscenza della citazione nel testo. Noi misuriamo questa posizione in termini di caratteri, parole, frasi, paragrafi e sezioni tenendo conto di dove la citazione si presenta. Il parser applica il procedimento standard di Java, per quanto riguarda i testi, per l'acquisizione e il conteggio dei caratteri e valuta il tag corrispondente nel testo NXML per ottenere la posizione della citazione nel paragrafo e nella sezione.

Il testo NXML non fornisce markup per frasi e parole. Quindi, identificare i confini di questi elementi richiede una priorità nel procedimento di estrazione dei dati.

L'obiettivo della fase di pre-procedimento è di includere delimitatori per frasi e parole senza compromettere il markup XML esistente. Per svolgere questo compito si sviluppa un subcomponente indipendente dal parser, il Sentence Word Tagger. Dopo che il Sentence Word Tagger ha identificato i confini di frasi e parole, un secondo sub-componente, il Data Parser estrae tutti i dati rilevanti e li inserisce nel database.

9.2 Sw-tagger

Esso identifica frasi e parole in un testo NXML e le contrassegna con caratteristici delimitatori che non danneggiano la validità del mmarkup XML originale.

L'ambiguità del linguaggio naturale rende impegnativo il riconoscimento delle frasi, parole e altre parti del discorso. Un esempio di massima ambiguità è per esempio il modo di indicare la fine del periodo. I termini specifici di alcuni linguaggi tecnici sono un ulteriore sfida.

La peculiarità dei testi scientifici costringe i ricercatori a regolare le codifiche POS (part of speech) specificatamente per questo campo, al fine di raggiungere una efficiente rilevazione.

A parte per la ottima performance, il rilevatore di frasi del SPToolkit offre un formato che è più facile da integrare con l'altro sub-componente del parser del documento rispetto a OpenNLP o StanfordCoreNLP.

Gli ultimi due strumenti possono elaborare testi XML, tuttavia entrambi scartano i markup originali creano files di output formattati diversamente.

Questo comportamento richiederebbe cambiamenti ai codici sorgenti degli strumenti per produrre un output che includa markup di frasi e parole in aggiunta ai tag originali XML.
SPToolkit fornisce i suoi output come una semplice stringa di oggetti Java che è usabile universalmente.
Si è deciso di incorporare SPToolkit nel parser del documento perché gli altri strumenti testati hanno praticamente la stessa performance di rilevazione, inoltre esso offre sia una migliore performance di lavoro e un buon formato di output.
Per contro SPToolkit non è in grado di elaborare testi XML. Tuttavia noi sostituiamo tutti i tag XML nel documento originale con un'unica stringa e immagazziniamo il contenuto del tag in un indice per essere inserito successivamente.
Per evitare di usare un tag POS di elaborazione intensiva basato sul linguaggio macchina, sono stati adattati e incorporati markup che sono comunemente utilizzati negli strumenti di suddivisione delle parole. Il tagger è stato programmato per marcare parole con testo semplice e annotazione simili a quelli che usiamo per la codifica delle frasi, così essi non interferiscono con il markup originale.
Il tagger ripristina il markup dopo la rilevazione delle frasi e parole attraverso un'ulteriore sostituzione delle stringhe segnaposto, con il contenuto originale del tag preso dall'indice immagazzinato.

9.3 Data parser

Il data parser estrae tutte le informazioni necessarie per una analisi CbPD (citation-based plagiarism detection) da un testo NXML.
Questo compito richiede la valutazione del markup XML originale e il markup del normale testo per frasi e parole che il SW-Tagger ha introdotto nel documento durante la fase di pre-processo.
Il parser deve elaborare tutti i documenti nella loro interezza, perché esso deve leggere ed estrarre i dati da ogni parte del testo corrispondente. Per esempio, i documenti generalmente elencano i metadati, come nome dell'autore e delle riviste, all'inizio del testo, informazioni sulle citazioni si trovano lungo il testo, mentre le referenze si trovano alla fine del testo.

Il compito di estrarre richiede parsing sequenziali di solo lettura di 234.000 documenti. Sono stati implementati i Data Parser seguendo il Simple API for XML (SAX), perché la funzionalità di SAX si adatta bene alle richieste, offrendo una buona velocità di elaborazione.

Il linguaggio Java offre diverse strutture per l'elaborazione XML, oltre a SAX , per esempio il Java API per XML Processing (JAXP) o lo Streaming API per XML (StAX).

SAX è la base, perché al contrario delle altre due strutture impone letture strettamente sequenziali dei documenti, senza interruzioni e non offre la possibilità di modificare i documenti.

Alcuni di questi eventi sono generati dell'incontrare il tag di inizio o di fine del documento o incontrare dei caratteri speciali.

Solo l'applicazione che richiama il parser SAX definisce reazioni per gli eventi che SAX parser riporta.

Questi gestori contengono ed eseguono logica di programmazione in relazione all'evento che ricevono dal SAX parser.

Il gestore dei contenuti è il gestore di callback del parser di dati che estrae dal documento, dati, citazioni, e referenze.

Per molti elementi, come documenti Ids, autore o nome questa estrazione è lineare. Allo stesso modo le citazioni sono facili da analizzare quando il rispettivo testo NXML contiene tag singoli per ogni citazione.

Comunque molti testi presentano svariate citazioni in modo abbreviato, senza offrire markup XML per tutte le citazioni presenti. Per riconoscere queste notazioni abbiamo progettato un controllo aggiuntivo per vedere se le citazioni si verificato in un intervallo fino a 13 caratteri. Se è presente la citazione entro l'intervallo dei 13 caratteri il gestore dei contenuti usa una normale espressione per controllare se il carattere letterario presente tra le citazioni in realtà rappresenta una ulteriore citazione.

Per tenere traccia del conteggio delle parole o frasi, è stato utilizzato il metodo del gestore dei richiami che reagisce a notifiche di eventi per i dati di carattere letterale.

Vengono usate espressioni regolari per riconoscere i markup della frase e della parola presentati nella fase di pre-elaborazione.

Dopo la raccolta di tutti i dati per un elemento, come una citazione o un riferimento, il gestore dei contenuti sottomette l'elemento al database.

9.4 Database

Il database è composto dalle seguenti entità: Citation, Author, Document, Reference, CitPatMatch(modelli di citazioni), CitPatMember(citazioni che formano il modello).
Le entità sono composte dai seguenti attributi:

Citation:

- dbCitId INT(11)
- docPmcId INT(11)
- docRefId VARCHAR(128)
- seqPos SMALLINT(5)
- charac INT(11)
- word MEDIUMINT(8)
- sentence MEDIUMINT(8)
- paragraph SMALLINT(5)
- section VARCHAR(512)
- refFreq MEDIUMINT

Document:

- docPmcId INT(11)
- docPmId INT(11)
- title VARCHAR(1024)
- year SMALLINT(5)
- month TINYINT(3)
- file VARCHAR(255)

Reference:

- dbRefId INT(11)
- docPmcId INT(11)
- docRefId VARCHAR(128)
- docRefPmId INT(11)

- docRefPmcId INT(11)
- docRefMedId VARCHAR(40)
- docRefDoi VARCHAR(255)
- dbRefAuthKey VARCHAR(41)
- dbRefTitKey VARCHAR(41)
- refFreq MEDIUMINT

Author:

- dbAuthId MEDIUMINT(8)
- docPmcId INT(11)
- dcoPmId INT(11)
- lastname VARCHAR(255)
- firstname VARCHAR(255)

CitPatMatch:

- dbPatId BIGINT
- docPmcId1 INT(11)
- docPmcId2 INT(11)
- proc TINYINT
- length SMALLINT
- countScore MEDIUMINT
- CFScore MEDIUMINT

CitPatMember:

- dbPatId BIGINT
- memberNr SMALLINT
- docPmcId INT(11)
- gap SMALLINT(5)
- refFreq SMALLINT
- dbCitId INT(11)

La ripartizione delle entità all'interno delle tabelle e le relazioni fra quelle tabelle seguono una pratica progettazione di database.

Molte tabelle e nome degli attributi sono autodescrittivi. Spieghiamo i nomi che possono non essere intuitivi.

Gli attributi con il prefisso DB rappresentano CitePlag-internal Ids (identificativi interni di CitePlag) che abbiamo assegnato.
Gli attributi con prefisso DOC sono identificativi contenuti nei testi originali NXML.
Gli attributi 'dbRefAuthKey' nella tabella reference sono chiavi che sono state artificialmente basate sui nomi degli autori e sui titoli delle referenze. Queste chiavi vengono usate per un confronto approssimativo tra le referenze, in caso che il documento non indica altri Ids come PMCID o DOI per la referenza specifica.
L'attributo proc, nella tabella CitPatMatch, identifica il la procedura di analisi del modello che identifica il confronto.
L'attributo Lenght nella stessa tabella indica il numero di citazioni che fanno parte del confronto.
L'attributo ContScore e CFScore sono fattori simili per catalogare modelli di citazioni secondo il loro grado di possibilità di plagio.

9.5 Detector

La componente di rilevamento esegue confronti a coppie tra le sequenze di citazione. Essa applica tre diversi modelli di algoritmo di analisy, che sono state per coprire le maggiori forme di plagio.
I tre modelli di algoritmi di analisi per le citazioni sono:
Longest Common Citation Sequence
Citation Tiling
Citation Chunking

LONGEST COMMON CITATION SEQUENCE (LCCS) è un adattamento di una misura di similarità tradizionale per stringa di testo. LCCS consiste nel massimo numero di citazioni che si possono ricavare da una sequenza di citazioni senza cambiare il loro ordine, ma permette di saltare le citazioni che non hanno corrispondenza. Per esempio la sequenza 3, 4, 5 è una sottosequenza di 2, 3, 1, 4, 6, 8, 5, 9. LCCS riconosce l'ordine delle citazioni ma offre flessibilità per far fronte con lievi trasposizioni o lacune se le citazioni non corrispondono.
Misurando le LCC produce punteggi più alti di similarità se un plagiario usa parti più lunghe di un altro testo senza modifiche o solo con minimi cambiamenti della fonte citata.

Questo modello è caratteristico del plagio copia e incolla.

GREEDY CITATION TILING (GCT) è anche un adattamento del noto misuratore di somiglianza per stringhe di testo che il suo inventore specificatamente aveva designato per scopi di rilevamento del plagio.

GCT identifica il più lungo modello per citazioni corrispondenti consecutive. L'algoritmo collega permanentemente singole partite più lunghe nelle sequenze di citazione a confronto e le memorizza come cosiddette lite.

ACTA focalizza le corrispondenze esatte nelle sequenze di citazioni. Infatti le corrispondenze sono dei forti indicatori del potenziale sospetto di somiglianza. ACTA è in grado di trattare con trasposizioni nella sequenza di citazioni che risulta dal rimaneggiamento dei lungi segmenti di testo, che è tipico del plagio Copia&Incolla.

CITATION CHUNKING è un set di procedure che identificano locali modelli di citazioni senza considerare potenziali trasposizioni.

Si definiscono tre strategie per delimitare brani di citazioni. La prima strategia considera solo corrispondenza consecutive. La seconda include confronti di citazioni in uno spezzone – se è minore o uguale a 1, o 1 è maggiore di n che è minore o uguale a s – le citazioni non corrispondenti lo separano dall'ultimo procedimento di confronto di citazioni. La variabile s indica il numero di citazioni nello spezzone sotto esame. La terza strategia include citazioni che capitano all'interno in un intervallo di testo definito.

Si considera che la prima strategia di chunking preferibile è di riprodurre e rilevare modelli di citazioni che risultano da C&P o paragrafi che provengono dall'utilizzo di sinonimi. La seconda strategia di chunking rivela il plagio shake&paste che risulta dalla combinazione di segmenti di testo provenienti da fonti differenti.

Quando si analizzano corpi di testo più lunghi la terza strategia di chunking può rivelare il plagio delle idee. Dopo che ciascuna procedura ha delimitato gli spezzoni di citazione, vengono confrontate tutte le coppie senza considerarne l'ordine all'interno dello spezzone. Il numero delle citazioni corrispondenti è il principale criterio di somiglianza.

Identificare modelli di citazioni è il primo dei due compiti in un procedimento della somiglianza basato sulle citazioni.

Il secondo è elencare i modelli secondo la loro verosimiglianza. Si determinano due principali fattori che aumentano questa verosimiglianza e generano due funzioni di catalogazione per analizzarli:

FUNZIONI DI PUNTEGGIO PER LE CITAZIONI: ci occupiamo del conteggio delle citazioni nei documenti individuali, per esaminare ciò che deve essere considerato per valutare le potenziali citazioni sospette.

Intuitivamente, due documenti A e B che entrambi presentano (per esempio) 200 citazioni è preferibile che compaiano in un modello di corrispondenza di citazioni, che due documenti C e D che presentano tre citazioni ciascuna. Consideriamo quindi modelli di citazioni contenenti un più alto numero di documenti altamente citati e che sono probabilmente rappresentano letteratura standard comunemente citata.

Se un documento è già stato citato, aumenta la probabilità che venga citato ancora. Col tempo, i documenti più altamente citati vanno a costituire una sorta di bagaglio standard per quell'argomento, molti autori lo fanno normalmente.

Quindi l'utilizzo della cosiddetta letteratura standard, di per sé non determina la somiglianza tra due testi.

Al contrario si tiene conto di citazioni raramente utilizzate per ottenere indicatori di maggiore efficacia per potenziali sospetti di somiglianza.

Da questa teoria deve scaturire un conteggio matematico che stabilisca il punteggio. Per semplificare si ipotizza che gli autori citano le fonti indipendentemente gli uni dagli altri.

Questa ipotesi non rispecchia l'effettivo comportamento, ma incorporando questi complessi e correlati fattori in un modello è difficile e allontanerebbe dallo scopo principale.

Si ipotizza che citazioni indipendenti per arrivare ad una elaborazione più lineare anche se con un certo margine di approssimazione che si conta di eliminare nel futuro.

La probabilità che un riferimento r punti un documento X equivale al conto di tutti i riferimenti X nel corpo di testo divisi per la dimensione N del corpo stesso

$$P(r \text{ di } x) = | r \text{ di } x| \ / N$$

Visto che i documenti citati raramente sono più indicativi e possono ricevere un punteggio maggiore, si inverta il tasso di probabilità in questo modo :

N / | r di x|

Ci si attende che i valori più frequentemente citati non diminuiscano in modo direttamente proporzionale al numero di citazioni che queste fonti raggruppano. Quindi consideriamo la radice quadrata del numero totale di referenze di una fonte √ | r di x| come denominatore del nostro punteggio.

Siccome il nostro punteggio deriva dall'analisi della frequenza delle citazioni lo chiamiamo CF-score. Esso, per una citazione c di i che è correlata a un riferimento r di j, che rappresenta il documento fonte X calcola come

CF (c di i (r di j)) N / √ | r di x|

Per calcolare il CF score per un modello di citazioni p di k che consiste in n citazioni c1,c2,…,cn collegati a m riferimenti r di j accumuliamo il punteggio di tutte le citazioni del modello CF(p di k) = ∑ da 1 a q CF(c di i(r di j))

Analogamente calcoliamo il Cf score per un paio di documenti d1 d2 che condividono q corrispondenze di modelli di citazioni p di k, attraverso l'accumulazione dei CF-scores del modello di confronto

CF(d1,d2) = ∑ da 1 a q CF(p di k)

Per semplificare il calcolo del Cf scores per la classifica dei modelli di citazioni, si considera un corpus di 1000 documenti. In questo corpus, 4 documenti A,B,C,D hanno i seguenti conteggi di citazioni:

$|r_A| = 100$, $|r_B| = 50$, $|r_C| = 10$, $|r_D| = 5$.

Si considerino inoltre due coppie di documenti X,Y che X,Z che condividono i seguenti modelli di citazione; X,Y : (A,B) (A,C) e X,Z: (CD)

Il risultante Cfscores per le coppie di documento è calcolato come:

$$CF(X,Y) = CF(p_1(A,B)) + CF(p_2(A,C)) + CF(p_3(B,B))$$
$$= \left(\frac{1{,}000}{\sqrt{100}} + \frac{1{,}000}{\sqrt{50}}\right) + \left(\frac{1{,}000}{\sqrt{100}} + \frac{1{,}000}{\sqrt{10}}\right) = 657.65$$

$$\mathrm{CF(X,Z)} = \mathrm{P}(p_1(\mathrm{C,D})) = \left(\frac{1{,}000}{\sqrt{10}} + \frac{1{,}000}{\sqrt{5}}\right) = 763.44$$

l'esempio mostra che sebbene la coppia X Y condivida più modelli di citazione, il singolo modello che il documento X condivide con il documento Z ha un punteggio maggiore perché compara fonti citate raramente.

CONTINUITY SCORE: il numero e la vicinanza delle citazioni corrispondenti all'interno del modello di citazioni condiviso sono i più importanti fattori che determinano la somiglianza e il grado di attenzione per i modelli individuali.

La nostra previsione su casi reali di plagio, conferma decisamente questa relazione.

Per incorporare questa conoscenza come parte della analisi CbPD abbiamo sviluppato un sistema per calcolare quello che noi chiamiamo un punteggio di continuità per i modelli di citazione. Basato sulla esperienza delle prime ricerche sul plagio, è stato elaborato il seguente punteggio che diventa determinante nel caso di citazioni frequenti e ravvicinate.

All'interno di un modello di citazioni, si esamina ogni citazione corrispondente successiva ad un'altra citazione corrispondete, se dopo tre ripetizioni, si verificano citazioni che non corrispondono, il punteggio del modello dovrebbe aumentare.

L'aumento del punteggio in questo caso dovrebbe essere maggiore di 1 perché non rispecchia un semplice conteggio di citazioni di corrispondenza, le quali sono state registrate separatamente.

Inoltre l'aumento del punteggio dovrebbe essere maggiore se delle citazioni non corrispondenti separano due citazioni successive.

Infine il punteggio dovrebbe aumentare in proporzione al numero di citazioni corrispondenti previste che soddisfano il criterio di avere un massimo di tre citazioni intermedie non corrispondenti separate dalla precedente citazione corrispondente.

Questa caratteristica della funzione riflette l'osservazione che la somiglianza dei modelli di citazione aumentano progressivamente se nel modello si verificano sequenze più lunghe di citazioni corrispondenti.

La formula seguente presenta la definizione formale del continuity-score.

$$Cont.-Score = \sum_{p=1}^{n} \max\{a - 0.25 \cdot \left(p\left(c_M^i\right) - p\left(c_M^{i-1}\right) - 1\right),$$

$$a = \begin{cases} 1 & c_M^1 \\ a+1 & p\left(c_M^i\right) - p\left(c_M^{i-1}\right) \leq 4 \,, i > 1 \\ 1 & p\left(c_M^i\right) - p\left(c_M^{i-1}\right) > 4 \,, i > 1 \end{cases}$$

La formula considera un punteggio base a per citazioni corrispondenti. Per la prima citazione corrispondente nel modello C1m, stabiliamo il punteggio base di 1.

Si incrementa il punteggio base per ogni citazione corrispondente successiva Cim|i >1 nel modello se meno di quattro citazioni non corrispondenti separano CiM dalla precedente citazione corrispondente Ci-1 M.

Esprimiamo questa condizione in termini di posizione sequenziale p(c) delle citazioni.

Per penalizzare le citazioni non corrispondenti all'interno di quelle corrispondenti, viene sottratta una penalità di 0,25 per ogni citazione non corrispondente.

Se quattro o più citazioni non corrispondenti separato due citazioni corrispondenti il punteggio base è fissato di nuovo a 1. In questo caso, l'addendo diventerebbe 0 se quattro citazioni non corrispondenti intermittenti separano le citazioni che non corrispondono oppure negative se ci sono più di quattro citazioni che non corrispondono.

Si è scelto di respingere la possibilità che il punteggio di continuità di un modello possa diventare minore del conteggio delle citazioni corrispondenti nel modello.

Si è arrivati a questo attraverso l'applicazione dell’operatore MAX, che assicura che il minimo punteggio cresca perché ogni citazione corrispondente è pari a 1.

Nell'esempio entrambi i modelli di citazioni contengono otto citazioni corrispondenti.

Questo relativamente elevato numero di citazioni corrispondenti permette a entrambi i modelli di ricevere un punteggio di continuità che supera la lunghezza del modello, che è uguale al conteggio delle citazioni corrispondenti.

Il punteggio di continuità del secondo modello equivale a circa 1,7 volte il punteggio del primo modello. In questo esempio, il punteggio più alto potrebbe segnalare che il secondo modello più probabile che sia una corrispondenza più estesa e quindi è sospetto.

Il primo modello sembra rappresentare tre corrispondenze più piccole quindi meno sospette.

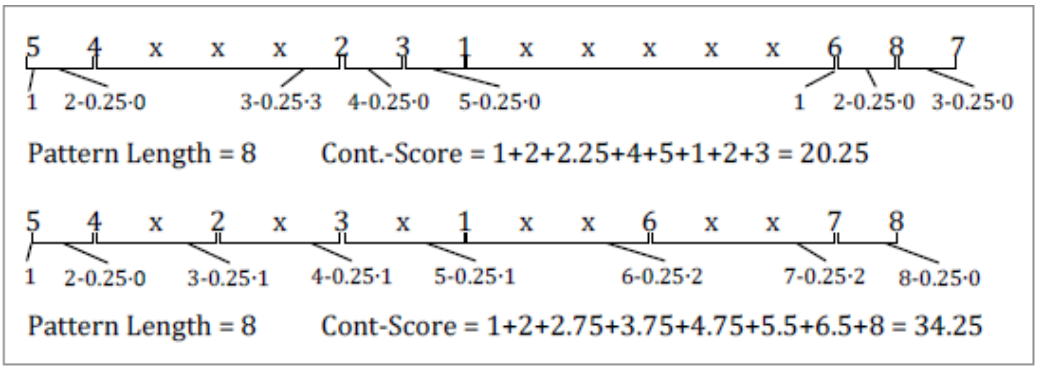


Grafico 4 [8]

DETECTOR IMPLEMETATION: l'immagine sotto presenta i principali componenti del detector usando una notazione del diagramma delle classi del Unified Modelling Language (UML). Abbiamo implementato ogni algoritmo di analisi dei modelli come una classe Java a se stante.

La classe 'CittionPatternChecker' è un fulcro che manifesta le diverse classi di analisi secondo parametri selezionabili e linee di funzionalità come richiedono gli algoritmi di analisi dei modelli, per esempio determinando il set di riferimenti condivisi.

Le altre classi sono implementazioni multi threaded per sottoattività legate a operazioni di input e output nel database CitePlag.

9.6 Report Generator

Attualmente CitePlag ha solo rapporti di funzionalità base, dovuti alla sua natura di prototipo.

Il report generator di CitePlag recupera i risultati della rilevazione dal database e crea dei semplici file di testo per ogni coppia di documenti che risulta avere modelli di citazioni corrispondenti su una soglia utente inserito. L'immagine seguente mostra un esempio di rapporto.

```
Doc1. 59651, Brown: Debate: "How low should LDL
cholesterol be lowered for optimum prevention of
vascular disease?" Viewpoint: "Below 100 mg/dl"

Doc2. 524501, Al Shaer, Choueiri, Suleiman: The pivotal
role of cholesterol absorption inhibitors in the
management of dyslipidemia


Match 1, l= 5

D1, Sec. 1, Sent. 6: [B9] _(0)_ [B10] _(0)_ [B11] _(0)_
[B12] _(0)_ [B13]

D2, Sec. 1, Sent. 9: [B4] _(0)_ [B5] _(0)_ [B6] _(0)_
[B7] _(0)_ [B8]


Match 2, l= 5

D1, Sec. 1, Sent. 6: [B9] _(0)_ [B10] _(0)_ [B11] _(0)_
[B12] _(0)_ [B13]

D2, Sec. 5, Sent. 53: [B4] _(0)_ [B5] _(0)_ [B6] _(0)_
[B7] _(0)_ [B8]
```

Il rapporto contiene i principali metadati (titolo, authors, PMCID) per entrambi i documenti e tutti i modelli di citazione corrispondenti.

Per ogni modello di citazione corrispondente il rapporto stabilisce la lunghezza l della corrispondenza, la sezione 'Sec.' (section) e la frase 'Sent.' (Sentence) per ogni documento nel quale il confronto inizia e la citazione che compone il confronto.

Il rapporto elenca le citazioni corrispondenti racchiuse in parentesi quadre usando la citazione ID del testo NXML originale.

In aggiunta, il rapporto indica il numero delle citazioni non corrispondenti che separano citazioni corrispondenti all'interno di un modello di citazione scritto tra parentesi tonde.

Il generatore di rapporto include il punteggio totale per ogni coppia di documenti, esso è il primo carattere del nome del file per il rispettivo file di report.

Questa convenzione di attribuzione ci permette di usare il file alfabetico del sistema operativo specie per i file di classifica in base alla loro importanza. [8]

10. Plagio nei software

È molto frequente anche il plagio nel codice sorgente dei programmi, esso, per essere individuato, richiede di utilizzare metodi diversi dal plagio nei documenti.
Il plagio del codice sorgente è diverso da quello nei testi perché quando si prendono in considerazione un problema che deve essere risolto da un applicativo, molto probabilmente, non esisterà un altro applicativo in grado di risolvere quello stesso specifico problema, quindi il plagio si presenterà in altre forme. Inoltre se si scrive un programma partendo dalle basi, è difficile integrare in esso del codice esterno.
Possiamo dire che ci sono due tipi di plagio nei software, uno di alto livello e uno di basso livello: quello di alto livello riguarda le similitudini nelle specifiche del codice mentre quello di basso livello riguarda le similitudini nella duplicazione del codice sorgente.
Per esempio in ambito accademico gli studenti sono tenuti a codificare le stesse specifiche, quindi ad alto livello non ci può essere plagio, perciò possiamo considerare solo il basso livello come indice di plagio.
Includere del codice duplicato nel proprio codice può comportare grossi problemi dal punto di vista della manutenzione del software e potrebbe introdurre dei sottili errori, inoltre il codice duplicato può avere effetti negativi sullo sviluppo del programma nel futuro perché potrebbe non essere stato compreso completamente, questo per i programmi di grandi dimensioni.
I motivi per il quale si compie questo tipo di violazione sono molteplici: il copia-incolla è la forma più semplice di plagio del codice sorgente, è un modo veloce di riutilizzare un frammento che si sa essere sicuro e affidabile.
Un'altra metodologia è il forking, cioè prendere un frammento di codice e utilizzarlo per scopi diversi rispetto all'utilizzo che ne faceva il software dal quale è stato copiato.
Un'altra possibile operazione è quela di riutilizzare le funzionalità e la logica di un altro programma se i due hanno finalità simili.
Un altro motivo può essere quello di prendere due programmi e fonderli insieme per ottenere un nuovo programma che faccia al caso nostro.

11. Software anti-plagio

In questo capitolo faremo una panoramica sui software più utilizzati e famosi.

11.1 Jplag

JPlag è stato sviluppato da Guido Malpohl presso l'Università di Karlsruhe.
Nel 1996 è iniziato come progetto di ricerca degli studenti e dopo pochi mesi si è evoluto in un sistema on-line.
JPlag è stato trasformato in un servizio web da Emeric Kwemou e Moritz Kroll.
JPlag converte programmi in stringhe (chiamate token) che rappresentano la struttura del programma, e utilizza un approccio basato struttura per confrontare due token.
JPlag utilizza l'algoritmo “Greedy String Tiling " (Segmentazione di stringhe), come proposto da Michael Wise, ma con differente ottimizzazione per una migliore efficienza.

11.2 Marble

Marble è uno strumento sviluppato nel 2002 presso l'Università di Utrecht. L'intenzione era di creare uno strumento semplice, di facile manutenzione che può essere utilizzato per rilevare casi plagio nei programmi scritti in Java.
Utilizzato sulla raccolta di tutti i programmi presentati nei vari archivi del dipartimento di informatica a Utrecht, Marble è stato determinante per scoprire alcune decine di casi di plagio.
Per motivi di scalabilità è essenziale che lo strumento possa distinguere tra parti di codice sorgente vecchie e nuove, così che le vecchie parti di codice sorgente non possano essere più confrontate tra loro.
Marble utilizza un approccio basato sul confronto delle parti di codice che gli vengono sottoposte. Si inizia dividendo queste parti in più file in modo che ogni file contiene una sola classe. La fase successiva è quella di normalizzazione, per rimuovere i dettagli da i file che possono essere troppo facilmente modificati dagli studenti: una analisi lessicale viene effettuata (implementata in Perl usando le espressioni regolari) che conserva le parole chiave (come di class, for) e nomi di classi e metodi utilizzati di frequente (come String, di sistema, toString).

Commenti, eccessivi spazzi vuoti, le costanti di stringa e le dichiarazioni di importazione vengono semplicemente rimosse, altre parti sono astratte per il loro "tipo". Per esempio, ogni numero esadecimale è sostituito da H e ogni carattere letterale dalla L.

Per ogni file, marble genera due versioni normalizzate: una in cui l'ordine di campi, metodi e classi interne è esattamente come quello del file originale, e uno in cui i campi, i metodi e le classi interne sono raggruppati insieme, e ogni gruppo è ordinato. L'ordinamento viene eseguito in modo euristico.

Ad esempio, i metodi vengono prima ordinati per numero di token, poi per lunghezza totale del flusso di token e infine in ordine alfabetico.

Per essere in grado di estrarre classi interne, metodi e campi della classe Java senza doverle analizzare, Marble prima annota le parentesi graffe nel programma con la loro profondità di annidamento, e divide le classi abbinandole in coppie di profondità di annidamento destra (profondità 0 corrisponde alla classe definizione, profondità da 1 a classi e metodi interne). Conoscere la posizione della parentesi di un metodo di apertura non dà direttamente la posizione di inizio del metodo, ma con la scansione all'indietro per il rilevato del primo punto e virgola o della parentesi di chiusura, è possibile trovarlo.

Le classi interne sono trattate allo stesso modo.

La persona che esegue lo strumento può scegliere di confrontare la versione ordinata o quella non ordinata normalizzata (o entrambi). Visto che l'ordinamento è euristico, piccole modifiche ai metodi in una classe possono cambiare totalmente l'ordinamento dei metodi. E' stato osservato un caso in cui uno studente dopo aver fatto una serie di cambiamenti, non ha effettuato il riordino dei metodi. A causa delle modifiche, gli stessi metodi nella versione originale e la versione plagiata sono finiti in posizioni sostanzialmente differenti, influenzando negativamente il punteggio. Questo è il motivo per cui ha senso confrontare anche le versioni non ordinate.

Il confronto dei file normalizzati viene effettuato usando l'utility "DIFF " di Unix / Linux. Il punteggio viene quindi calcolato dal rapporto tra il numero di linee in cui differiscono l'uno dall'altro e il numero totale di linee risultato nel confronto tra i due file normalizzati.

11.3 MOSS

MOSS è l'acronimo di “Measure Of Software Similarity.”.

MOSS è stato sviluppato nel 1994 a Stanford University da Aiken et al.

È disponibile come webservice e vi si può accedere mediante uno script che può essere ottenuto dal sito web MOSS.

Un account MOSS può essere richiesto via e-mail da moss@moss.stanford.edu.

Lo script di MOSS funziona per le piattaforme Unix / Linux e può funzionare sotto Windows con Cygwin, ma quest’ultimo non è stato testato.

Per misurare la somiglianza tra i documenti, MOSS ne mette a confronto le versioni standard, MOSS utilizza un algoritmo di documento fingerprinting chiamato “winnowing” (vagliatura).

Il fingerprinting è una tecnica, già citata e analizzata nei capitoli precedenti, che divide un documento in sottostringhe contigue, chiamato k-grammi (con k scelto dall'utente).

Un sottoinsieme di tutti gli k-grammi è selezionato come fingerprint del documento.

“Winnowing” (vagliatura) è un algoritmo efficiente per la selezione di questi sottoinsiemi.

11.4 Plaggie

Plaggie è un software di rilevamento del plagio nel codice sorgente utilizzato per esercizi di programmazione Java.

In apparenza e funzionalità, è simile a Jplag, ma ci sono anche aspetti di Plaggie che lo rende molto differente da Jplag, Plaggie deve essere installato in locale e il suo codice sorgente è aperto.

Plaggie è stato sviluppato nel 2002 da Ahtiainen et al alla Helsinki University of Technology. È un applicazioni Java indipendente a riga di comando.

L'algoritmo di base utilizzato per confrontare due file sorgente è lo stesso di Jplag, perché utilizza il “Greedy String Tiling” come descritto in Jplag.

Gli autori ci dicono che non hanno applicato le ottimizzazioni che sono state implementate in JPlag.

Nel file readme di Plaggie, gli autori rendono noti l’elenco degli attacchi non riusciti (modifica dei commenti, cambiare i nomi delle classi, i metodi o variabili), così come una lista degli attacchi riusciti (scambiare linee di codice per separare

metodi e viceversa, l'inclusione di codice ridondante nel programma, cambiando l'ordine di blocchi if-else) e problemi conosciuti (la precisione di Plaggie quando viene usato sul codice della GUI o il codice generato automaticamente).

11.5 SIM

SIM è un programma per il rilevamento della somiglianza tra software per programmi scritti in C, Java, Pascal, Modula-2, Lisp, Miranda, e per il linguaggio naturale.

E 'stato sviluppato nel 1989 da Dick Grune presso l'Università VU Amsterdam. La versione attuale è sim 2.26. Matty Huntjens ha scritto lo script che prende l'output di sim e lo trasforma in un rapporto sul plagio che possa facilmente essere interpretato dall'essere umano.

SIM utilizza un processo di "tokenize", simile a quello descritto per i precendenti software, per rilevare somiglianze nel codice sorgente come prima cosa, poi costruisce una tabella di riferimento che può essere utilizzata per rilevare le migliori corrispondenze tra i file presentati e il testo con il quale devono essere comparati.

Un programma intelligente fa in modo che il tempo di calcolo per la procedura precedentemente descritta rimanga entro limiti ragionevoli.

SIM non è più supportato e non c'è nessuno che si occupi del suo sviluppo quindi il suo codice sorgente è disponibile al pubblico.

Gli autori ci dicono che, questo software di rilevamento plagio è stato sviluppato molto su misura per la situazione al VU University Amsterdam e quindi non molto versatile ed adattabile ad altri casi o situazioni. [(7)]

12. Metodi di individuazione del plagio nei software

In questo capitolo si tratterà degli approcci di individuazione del plagio, ecco un elenco dei principali metodi utilizzati dai software:

- **Strings**: metodo che cerca corrispondenze testuali nei segmenti di codice confrontati. Veloce ma può essere facilmente sviato rinominando gli identificatori
- **Albero sintattico**: costruisce e confronta alberi sintattici questo permette di rilevare somiglianze ad un livello superiore di quello letterale. Per esempio si può confrontare gli alberi delle istruzioni condizionali e rilevare se ci sono delle somiglianze.
- **Programma grafico di dipendenza (PDG)**: un PDG cattura il flusso di controllo di un programma e permette di trovare somiglianze di livello più alto. C'è una maggiore spesa in termini di complessità e tempo di calcolo computazionale.
- **Metriche**: cioè vengono selezionati dei segmenti di codice in base a determinati criteri come il numero di cicli e condizioni o il numero di variabili diverse usate. Questo approccio è poco dispendioso in termini ci costo computazionale e i confronti sono molto rapidi, ma può portare a dei falsi positivi; due frammenti con gli stessi punteggi su di un set di metriche possono fare cose completamente diverse.
- **Approcci ibridi**: per esempio alberi sintattici più alberi dei suffissi possono combinare la capacità di rilevamento degli alberi sintattici con le velocità offerte dagli alberi di suffisso, un tipo di struttura dati string-matching. [17]

13 Comparazione software

I confronti tra gli strumenti di rilevamento del plagio possono essere approssimativamente suddivisi in due categorie:

- confronti che partono dall'esame delle caratteristiche
- confronti che considerano le prestazioni.

I confronti tra le caratteristiche dei vari strumenti di rilevamento affrontano il tema con un approccio qualitativo: vengono prese in esame le proprietà di un programma, i linguaggi di programmazione che supporta, se si tratta di un applicazione locale o di una web, quale algoritmo viene utilizzato per confrontare i file, e così via. Tale confronto è puramente descrittivo e, sulla base di questo confronto, risulta difficile affermare quale di questi strumenti potrebbe essere considerato 'il migliore'.

I confronti tra prestazioni sono paragoni quantitativi, essi, in genere, descrivono alcuni esperimenti eseguiti mediante l'utilizzo di alcuni strumenti. Si confrontano poi i risultati ottenuti, piuttosto che le loro proprietà.

Prima di entrare nel dettaglio è opportuno fare alcune annotazioni sui seguenti strumenti.

Nel rapporto di Jplag, in un confronto testuale a caratteristiche simili a MOSS e yap3, include anche un lungo elenco di cosiddetti "attacchi inutili".

Il sistema "Software Integrity Diagnosis" (SID) è presentato come una continuazione di un progetto generale che si occupa di determinare la misura della somiglianza di due sequenze (se queste sequenze rappresentano genomi, documenti, musica o programmi). Il misura che utilizza SID si basa sulla complessità di Kolmogorov. I risultati di SID vengono confrontati con i risultati di MOSS e JPlag sugli stessi programmi. Gli autori riportano che in molti casi i risultati dei tre strumenti sono simili, ma che sia MOSS che JPlag sembrano essere sensibili all'inserimento casuale di codice ridondante (ad esempio, molte dichiarazioni System.out.println). I risultati della nostra analisi di sensibilità indicano che in effetti MOSS è piuttosto sensibile a numerosi piccoli cambiamenti nel codice sorgente. [(7)]

13.1 Comparazione qualitativa

Ora si presenta un confronto preliminare nelle caratteristiche di base tra i software antiplagio precedentemente presentati cioè MOSS, Jplag, SIM, Plaggie, Marble.

Token: viene utilizzato un analizzatore lessicale (lexer) per convertire il programma in token (parti di codice). Il lexer scarta gli spazi vuoti, commenti e identificatori che rendono l'analisi più efficiente e meno sensibile alla semplice sostituzione del testo finalizzata a camuffare il plagio. La maggior parte dei sistemi di rilevamento del plagio accademico lavorano a questo livello, utilizzando algoritmi differenti per misurare la somiglianza tra le sequenze di token.

Caratteristiche	**Jplag**	**Marble**	**MOSS**	**Plaggie**	**SIM**
Linguaggi supportati	6	1	23	1	5
Estensibilità	No	No	No	No	Si
Presentazione dei risultati (1-5)	5	3	4	4	2
Usabilità (1-5)	5	2	4	3	2
Esclusione dei modelli di codice	Si	No	Si	Si	No
Esclusione dei piccoli file	Si	Si	Si	No	No
Confronto storico	No	Sì	No	No	Sì
Presentazione o file valutazione	Presentazione	File	Presentazione	Presentazione	File
Locale o web	Web	Locale	Web	Locale	Locale
Open source	No	No	No	Sì	Sì

Tabella 1

In questa tabella confrontiamo i software su dieci caratteristiche diverse:

1. **Linguaggi supportati**, cioè linguaggio che il programma può analizzare per verificare la presenza di plagio

2. **Estensibilità**: va di pari passo con i linguaggi supportati perché indica se uno strumento ha la capacità di adattarsi ad altri linguaggi di programmazione e se possa essere adatto per valutare altri linguaggi.
3. **Presentazione dei risultati**: indica con un numero da 1 a 5, in cui 1 è scarso e 5 è ottimo, se il rapporto finale dopo l'analisi è facilmente leggibile dall'utente che ha utilizzato lo strumento e se il rapporto è esaustivo. Una buona presentazione dei risultati dovrebbe contenere almeno i seguenti elementi: riassunto dei dati, risultato dei confronti e strumenti di confronto per evidenziare i punti dove è più probabile il plagio
4. **Usabilità**: cioè la facilità di utilizzo di uno strumento (anch'essa espressa con un numero da 1 a 5)
5. **Esclusione dei modelli di codice**: indica se il programma è in grado di eliminare dal confronto quelle parti di codice che sono in comune perché rappresentano una base comune per tutti per il problema che applicativo deve risolvere, quindi sono citazioni non plagio.
6. **Esclusione dei piccoli file**: sono file contenenti solo metodi setter e getter o attributi quindi tutti quei file che sono di supporto al programma.
7. **Confronti storici**: si intende la capacità di uno strumento di confrontare un nuova serie di osservazioni con i contributi provenienti da programmi più vecchi
8. **Presentazione o file di valutazione**: vale a dire quando, in una presentazione, tutti i dati raccolti su ogni confronto sono sintetizzati in un unico file mentre nell'altro caso si hanno tanti file per ogni confronto effettuato, i file che fanno riferimento a ogni singolo confronto contengono solo i dati ad esso riferiti;
9. **Locale o web**: alcuni di questi programmi possiamo trovarli sono sul web altri devono essere installati sul PC per poter essere eseguiti (locale).
10. **Open source**: significa che il codice del programma è disponibile a tutti e ognuno lo può modificare come vuole per adattarlo al suo caso.

13.2 Comparazione delle performance

In questa sezione si descrivono due esperimenti che sono stati effettuati per misurare le prestazioni dei software dal punto di vista quantitativo.

Il primo di questi è un'analisi di sensibilità, il secondo è un confronto top-n.
Nella analisi di sensibilità prendiamo due classi Java, la versione refactoring (riscritta) viene presentata in diciassette modi differenti che saranno confrontati, con la versione originale, in modi diversi per ogni strumento. In questo modo ci attendiamo di ottenere una certa comprensione dei punti deboli e dei punti di forza di ogni strumento.
Dal momento che abbiamo visto in precedenza come gli esperimenti di refactoring possano cambiare di molto i punteggi, abbiamo ampliato il nostro esperimento di sensibilità anche considerando gli effetti di una combinazione di refactoring. Nel nostro ultimo esperimento, il confronto top-n, si eseguono i programmi su una raccolta di altri software (in cui sono noti alcuni casi di plagio), e si considera la top-10 dei punteggi più alti tra le coppie proposte per ogni strumento.
Quindi, questo esperimento confronta le parti di codici sorgenti piuttosto che i file. Ci sono diversi motivi per cui è necessario fare questo confronto top-n. Prima di tutto, quando si vuole determinare l'accuratezza di uno strumento, è necessario riuscire a farlo funzionare su una vasta collezione di osservazioni in cui si hanno numerose occorrenze di casi evidenti, meno evidenti, e altamente evidenti di refactoring e di plagio nascosti. Poi si verifica manualmente se lo strumento ha trovato un caso di effettivo plagio. Ovviamente, per le collezioni di grandi dimensioni, è un compito che richiede molto tempo.
Se, tuttavia, si è in grado di eseguire diversi strumenti su questa stessa serie di osservazioni, possiamo utilizzare le informazioni aggiuntive che altri strumenti forniscono per determinare più facilmente se il nostro strumento è preciso.
Una seconda ragione per cui abbiamo incluso il confronto top-n è la seguente: nell'esperimento di sensibilità, ciascun utensile restituisce un punteggio per ciascuna coppia costituita da un file originale e un file refactoring. Possiamo normalizzare il punteggio per il rilevamento delle somiglianze tra i due file da un massimo di 100, e un minimo di 0. Se uno strumento da un punteggio di 100 per una versione refactoring nel confronto con l'originale, si considera lo strumento insensibile a quel particolare refactoring.
Tuttavia, è difficile valutare le conclusioni degli altri punteggi nella gamma da 0 a 100, perché ogni strumento ha una sua modalità di calcolo del punteggio: un punteggio di 90 per uno strumento non significa necessariamente la stessa cosa di un punteggio di 90 per un altro strumento.

Confrontando il top-n di tutti gli strumenti questo problema è stato eliminato.
Infine, si evidenzia che è molto facile costruire uno strumento che segnali sempre il 100 per qualsiasi confronto tra due file. Tale strumento è ovviamente poco sensibile ed è destinato a dare una pessima performance nel confronto top-10.

13.3 Confronto di sensibilità

Questa paragrafo descrive la parte del set-up di questo esperimento che è comune per entrambi i test “singole refactoring” e il “refactoring combinato”.

Set-up comune

Per confrontare la sensibilità degli strumenti al singolo cambiamento, si è effettuato un esperimento in cui si sono create diciassette differenti versioni del programma "Animated Quicksort" (e una versione combinata per l'esperimento di sensibilità combinata). Questo metodo è stato utilizzato in un corso di programmazione distribuita presso l'Università di Utrecht. La versione a portata di mano è composta da cinque Java file: QSortAlgorithm.java, QSortApplet.java, QSortView.java, QSortObserver.java, QSortModel.java.
Di questi file, gli ultimi tre mostrano solo piccole differenze o dovrebbero essere molto simili per natura. Pertanto, per entrambi gli esperimenti di sensibilità, prendiamo in considerazione solo QSortAlgorithm.java e QSortApplet.java.
Consideriamo diciassette strategie che possono essere utilizzate per cercare di mascherare un programma plagiato. Le modifiche in questa tabella conservano la semantica del programma ma cambiano il suo aspetto.
Inoltre, le modifiche non richiedono una conoscenza approfondita del programma e non richiedono molto tempo per essere implementate.
Gli studenti che plagiano spesso non hanno le conoscenze o il tempo per costruire un programma dalle basi, il che rende queste modifiche dei pretesti efficaci per mascherare il plagio.
Le modifiche dalla 12 alla 17 si basano sulla funzionalità di refactoring offerte dal popolare programma Eclipse, che è noto per essere utilizzato da un molti studenti.

13.4 Sensibilità al singolo cambiamento

Di seguito viene descritto il set-up per questo esperimento particolare, il modo in cui presentiamo i risultati, e infine dobbiamo interpretare questi risultati.

Indichiamo la versione originale del programma come “versione 0” (vale a dire, c'è una versione 0 per entrambi QSortApplet.java e QSortAlgorithm.java). Creiamo diciassette diverse versioni di ciascuno dei due programmi, utilizzando le refactoring descritti in Tabella 1. Le versioni risultanti sono numerate secondo i numeri dei rifattorizzazione nella tabella 1, in ogni versione una sola modifica viene applicata. In questo modo ci proponiamo di essere in grado di valutare la sensibilità degli strumenti nelle differenti strategie di modificazione.

Poiché l'analisi di sensibilità è una per file, a volte bisogna creare in modo esplicito due directory, una che contiene la versione 0 e l'altra che contiene tutte le versioni dalla 1 alla 17, al fine di garantire che lo strumento in questione restituisca un punteggio per ciascuna delle coppie “versione 0, versione i” dove per ‘i’ si intende tutte le versioni dalla 1 alla 17.

Versione	**Descrizione**
0	Versione originale del programma
1	Piccole modifiche ai commenti e al layout
2	Modificare il 25% dei metodi
3	Modificare il 50% dei metodi
4	Modificare il 100% dei metodi
5	Modificare il 50% degli attributi delle classi
6	Modificare il 100% degli attributi delle classi
7	Cambiare il codice della GUI
8	Cambiare gli import
9	Cambiare il testo e i colori della GUI

10	Rinominare le classi
11	Rinominare le variabili
12	Eclipse - pulire la funzione: accesso ai membri con il comando “this” per i campi e i metodi
13	Eclipse - pulire la funzione: stile del codice (usare blocchi come if/while/for/do, usare le parentesi sulle codizioni)
14	Eclipse – generare codice hash e funzioni equivalenti
15	Eclipse – esternare le stringhe
16	Eclipse – estrarre classi interne
17	Eclipse – generare un getter e un setter per ogni attributo

Tabella 2

Presentazione dei risultati

Per ogni versione modificata, le cifre mostrano sei valori di similarità.

Il valore primo è il valore di somiglianza calcolato utilizzando l'utilità DIFF2 di Unix direttamente sul codice sorgente. Questo serveper quantificare gli effetti delle modifiche fatte.

Il valore viene calcolato come segue:

$$\text{somiglianze} = 100 - \frac{100 * \text{ numero di linee differenti } (\texttt{diff})}{\text{numero di righe del file1} + \text{numero di righe del file2}}$$

Si noti che è importante utilizzare DIFF con l’opzione -w per ignorare le linee che differiscono nella quantità e / o nella posizione degli spazi vuoti.

I cinque valori successivi sono i valori di somiglianza calcolati dai cinque strumenti di rilevamento del plagio che noi abbiamo considerato. Per Marble la selezione del valore di somiglianza è semplice, poiché per ogni coppia di file Marble produce un singolo punteggio di somiglianza.

Altri strumenti, però, restituiscono due punteggi di somiglianza per ogni confronto tra coppie di file. Un punteggio indica la percentuale di linee nel file A che sono state trovate essere simili alle linee nel file B, e viceversa, un punteggio che indica la percentuale di linee nel file B che sono simili alle linee nel file A. Per tutti gli

strumenti che utilizzano questo approccio che abbiamo scelto di tenere in considerazione il punteggio massimo tra i due punteggi di somiglianza.

Interpretazione dei risultati

Non sappiamo sempre esattamente quale funzione uno strumento utilizza per calcolare i propri punteggi.

Pertanto, quando il punteggio dello strumento X supera quello dello strumento Y per una determinata versione, questo non significa necessariamente che strumento Y sia più sensibile a questo tipo di attacco rispetto allo strumento X.

Inoltre, un modo di utilizzare uno strumento di rilevamento del plagio è di eseguirlo su una serie di osservazioni, e quindi controllare le coppie sospettate di plagio dallo strumento, a partire dalla coppia con il punteggio più alto, quindi il secondo più alto, e così via, fino a quando non vengono trovati dei seri candidati. In questo modo, i punteggi esatti non sono importanti.

Come accennato prima, dobbiamo prendere in considerazione un punteggio di 100 che indica l'insensibilità dello strumento per il refactoring. Inoltre, un punteggio molto basso rispetto al corrispondente punteggio di 'DIFF' e un punteggio dello strumento significativamente inferiore rispetto ai corrispondenti punteggi di tutti altri strumenti sono incentivi per ulteriori considerazioni.

Dai grafici nelle figure 1 e 2 possiamo osservare diverse cose interessanti.

Per ogni versione e per ogni strumento, abbiamo scelto tra i due risultati quello con il punteggio più elevato.

Ad esempio per QSortAlgorithm.java, le versioni 7, 8, 9, 15 e 16 danno come risultato 100 sia per DIFF che per ogni altro strumento, perché quelle refactoring non influiscono in maniera particolare su quel file, tuttavia, fanno causare alcuni cambiamenti nella QSortApplet.java, a motivo di ciò per queste versioni si sceglie i valori di figura 1.

Si comincia con un'osservazione generale sulle due cifre. Quattro dei cinque strumenti spesso danno come punteggio 100, e anche RIFF lo fa di tanto in tanto. MOSS, tuttavia, mai da un punteggio pari a 100. Dal momento che anche questo avviene per tutti i file, questa potrebbe essere una caratteristica intrinseca di MOSS, e tendiamo a considerare i punteggi come 98 e 99 come se fossero 100.

Prima di tutto, in entrambe le figure possiamo vedere, per la versione 1, che tutti i programmi sono insensibili ai cambiamenti nel layout e commenti (in senso stretto, MOSS non è completamente insensibile).

Le barre per le versioni 2, 3 e 4 mostrano la sensibilità ai cambiamenti dei programmi nella modifica dei metodi.
Solo Marble è completamente insensibile alle variazioni nei metodi. In Figura 2 osserviamo che JPlag, MOSS, Plaggie e SIM sono notevolmente sensibili alle modifiche apportate ai metodi. Nella stessa figura, questi strumenti mostrano anche una leggera diminuzione del punteggio di somiglianza quando cambiamo la posizione degli attributi di classe.
Nessuno degli strumenti è completamente insensibile alle variazioni della posizione degli attributi della classe (versioni 5 e 6).

La figura 1 mostra che tutti gli strumenti, tranne Marble, hanno un punteggio che è significativamente inferiore rispetto a quello di DIFF quando il codice della GUI viene modificato.

Le modifiche applicate nelle versioni 8, 9, 10, e 11 sono completamente inefficaci per tutti gli strumenti in entrambi file (con una piccola eccezione per JPlag e MOSS). Queste modifiche riguardano la ridenominazione delle variabili e delle classi e la modifica delle importazioni.

Le modifiche nelle versioni 12 sembrano causare una certa confusione per Marble, e molta confusione per MOSS e SIM.
JPlag e Plaggie sono insensibili a questo tipo di refactoring. Per MOSS questo corrisponde ai risultati riportati nello studio comparativo per SID.

Quasi lo stesso vale per la versione 13. Però qui il punteggio di Plaggie scende e raggiunge quello degli altri programmi.

Sia JPlag che Marble sembrano peggiorare alla versione 14: i loro punteggi sono inferiore a DIFF. Tuttavia, MOSS, SIM, e sorprendentemente, dal momento che è molto simile a Jplag, Plaggie si comporta abbastanza bene con questa versione.

Per quanto riguarda la versione 15, nessuno degli strumenti è insensibile, ma il punteggio di JPlag e Marble supera quello di DIFF, mentre gli altri tre hanno punteggio più basso.

JPlag, Plaggie e SIM sono insensibili al refactoring dalla versione 16, mentre i risultati di Marble e MOSS non sono chiari. Tuttavia, il punteggio di Marble è più alto rispetto a quello di DIFF in entrambe le figure.

Nella versione 17 (generazione di getter e setter): nessuno degli strumenti è completamente insensibile, ma Marble e JPlag sembrano avere qualche problema (punteggio inferiore a DIFF nella figura 2).

Ad una prima occhiata si potrebbe essere tentati di concludere che, anche se ci sono alcune differenze di valutazione, tutti i tentativi di mascherare il plagio da parte delle differenti tecniche di modifica sarebbero stati trovati da tutti gli strumenti. Dopo tutto, gli strumenti restituiscono un punteggio piuttosto alto di somiglianza per la maggior parte delle versioni. Si noti, tuttavia, che la validità della dichiarazione precedente dipende dal tasso con cui gli strumenti valutano i casi di non plagio. Ad esempio, potrebbe essere che gli strumenti valutano coppie non plagiate con valori simili come abbiamo visto in questo esperimento. Questo dovrebbe essere verificato in un esperimento.
Il confronto top-n nel prossimo capitolo può fornire informazioni relative a questa funzione.
In questo momento possiamo dire, in base alla nostra esperienza con i differenti strumenti durante i nostri esperimenti, che la maggior parte dei software sembrano valutare coppie non simili con dei punteggi significativamente inferiori rispetto ai punteggi che abbiamo visto in questo esperimento di sensibilità.

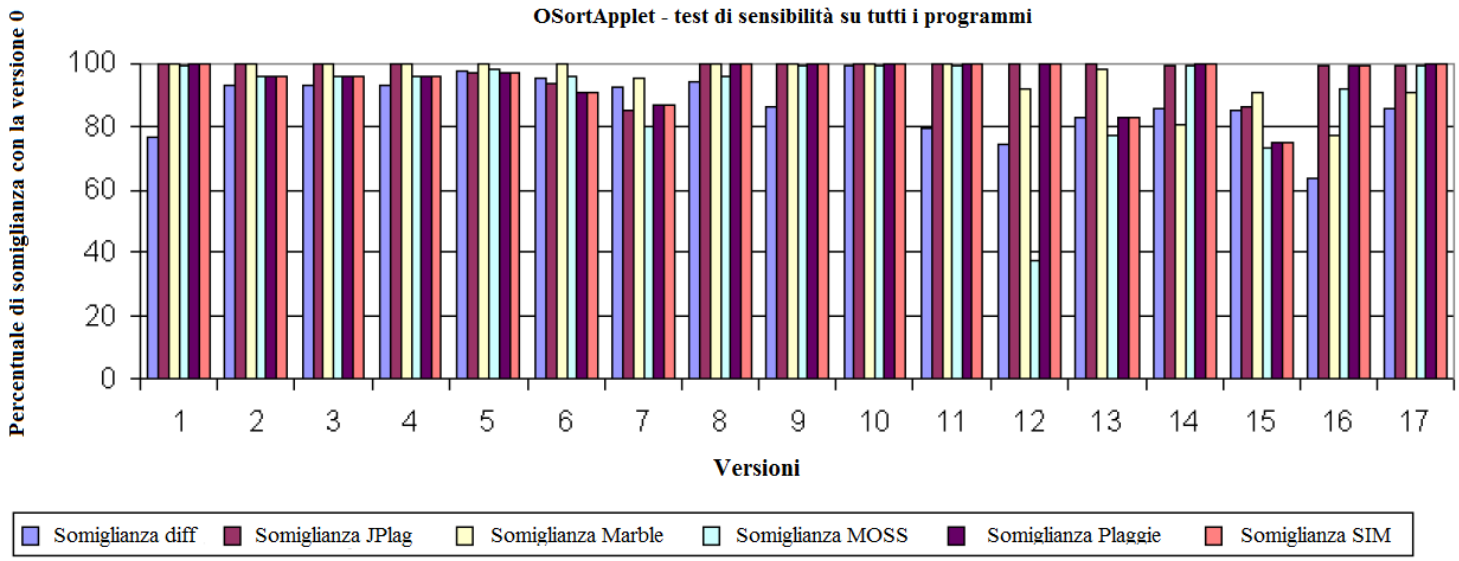


Figura 1 [7]

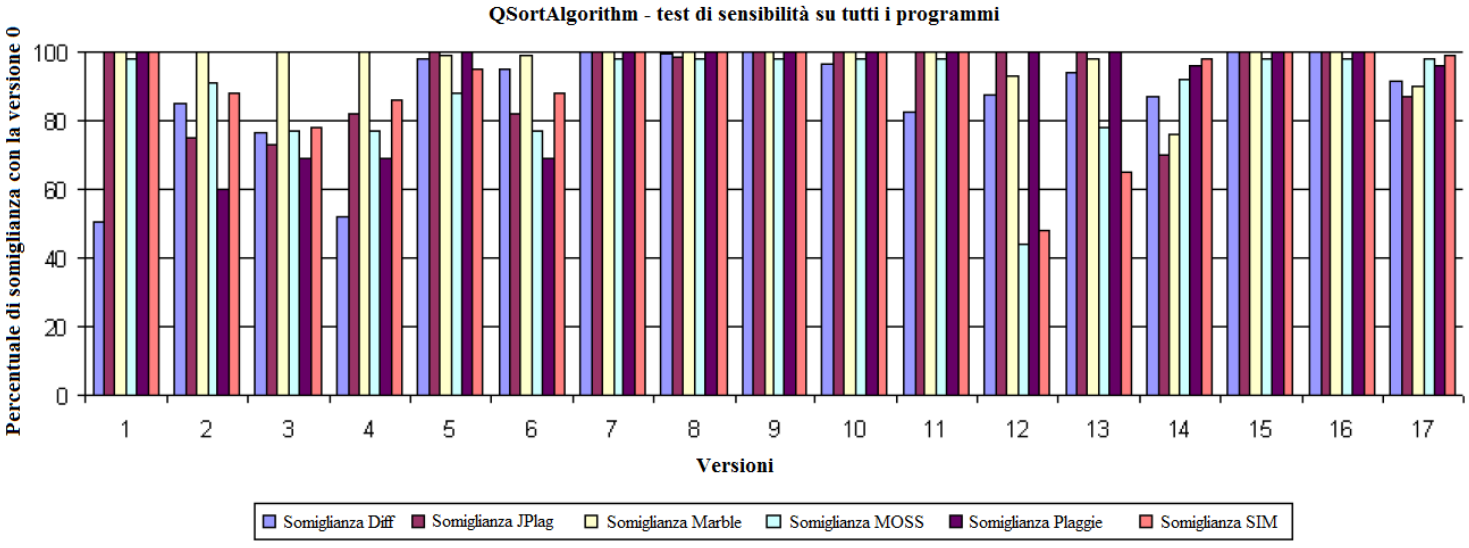


Figura 2 [7]

13.5 Sensibilità alle modifiche combinate

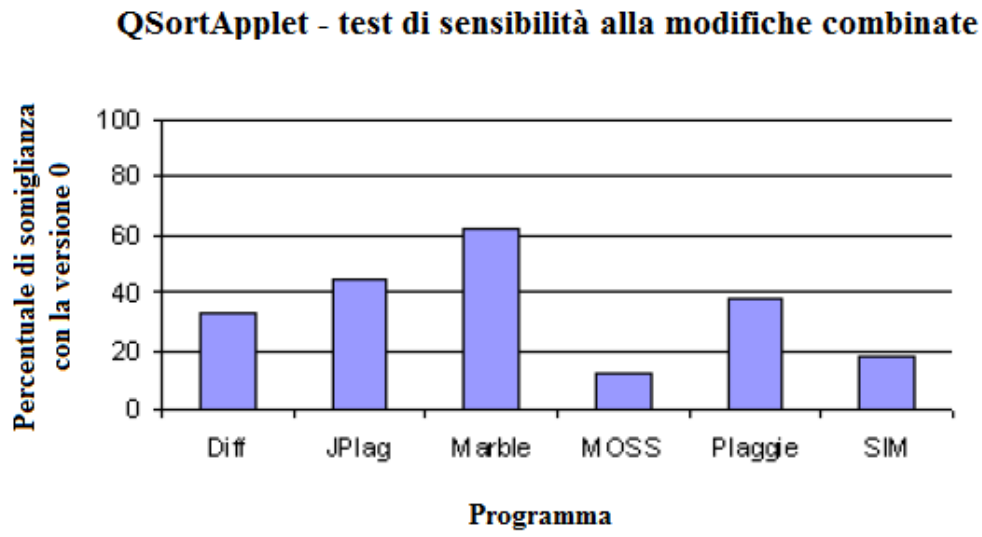
Presentiamo qui l'impostazione e risultati della seconda parte dell'esperimento sensibilità.

Impostazione e presentazione dei risultati

Per fare un passo avanti rispetto all'esperimento precedente, abbiamo unito diverse modifiche, presentate precedentemente, (in particolare la 4, la 6, la 11 e la 17) per vedere se siamo in grado di sfuggire al controllo effettuato da questi programmi.
Queste modifiche hanno dimostrato di essere più efficaci sulla la maggior parte degli strumenti. Inoltre, le modifiche 11e 17 sono molto facili applicare in quanto possono essere eseguite automaticamente dal Eclipse IDE.
I risultati in figura 3 mostrano i punteggi per QSortApplet.java e QSortAlgorithm.java.

Interpretazione dei risultati

Dai risultati della figura 3 si osserva che i punteggi di somiglianza sia di MOSS che di SIM sono significativamente in caduta per tutti i file. Marble, JPlag e Plaggie mostrano una flessione più contenuta nel punteggio di somiglianza.
Per essere in grado di rilevare il plagio durante la ricerca solo un file ha bisogno di finire in alto nella classifica.
Dall'esperienza con Marble si può dire però che un punteggio di 64 per QSortAlgorithm.java finirà abbastanza in alto nella lista dei risultati.



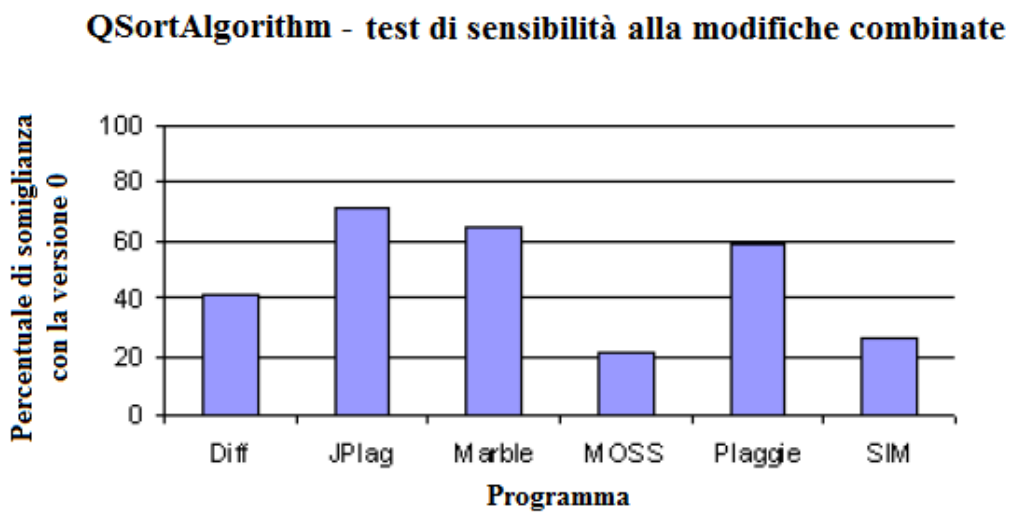


Figura 3 [7]

13.6 Confronto Top-n

Un'altra idea per confrontare le prestazioni dei programmi è mediante il confronto dei top-n dei risultati degli strumenti. In questo paragrafo si descrivono i risultati di questo esperimento per n = 10.

Si consideri la top 10 delle coppie di punteggi più alte per ogni strumento. Questo ci dà un minimo di 10 (n = 10 perché assumendo che ci sono 10 coppie che sono abbastanza simili da essere rilevate da almeno un programma) a un massimo di 50 (poiché consideriamo cinque strumenti) coppie per programmi, per queste coppie abbiamo cercato di capire (1) se pensiamo che i programmi sono molto simili, e (2) se la somiglianza può essere causa di plagio.

Se consideriamo tutte queste coppie altamente simili insieme, in seguito gli strumenti che hanno la maggior parte di queste coppie nella loro top 10, e poi i casi che costituiscono plagio reale allora esse sono considerate accettabili.
Va notato, tuttavia, che per quanto un programma possa effettuare un ottimo confronto, il suo risultato dipende fortemente dai modi con cui gli studenti hanno cercato di nascondere il plagio. Ad esempio, se una tecnica X non viene mai usata per nascondere il plagio nella nostra serie di osservazioni, poi i risultati sono tutti uguali; noi considereremo due programmi Y e Z con la stessa precisione, anche se potrebbe accadere che Y può rilevare la presenza di un tentativo di mascheramento del plagio con X mentre Z può risultare inutile.
Quando un programma ha una lista di coppie con punteggio elevato, e gli elementi di queste coppie risultano non essere simili, allora questo è un chiaro segno che qualcosa può essere migliorato, in particolare se la lista delle coppie degli altri strumenti non presenta quella combinazione.

Impostazione

Per il confronto top 10 si usa un approccio differente rispetto a quello che abbiamo usato per l'esperimento di sensibilità. All'Università di Utrecht, viene utilizzato un database piuttosto ampio di osservazioni per applicare Mandelbrot.
Quindi non hanno fatto altro che sottoporre ai programmi questo insieme di documenti nel quale cercare di identificare un qualche tipo di plagio, avendo la conferma che tra questi si nascondesse qualche caso di plagio.

Presentazione dei risultati

Tabella 2 presenta i risultati del confronto top 10. Per ciascuno dei cinque strumenti, abbiamo guardato le 10 paia di osservazioni con la più alta somiglianza.
Va notato che tale confronto è stato effettuato convertendo l'output dei file dei vari programmi per avere un formato che possa essere comparato.
È essenziale avere qualche programma di supporto, soprattutto se si vuole rifare l'esperimento in seguito per altre serie di osservazioni.
I programmi di conversione rivelano anche che i punteggi dei vari programmi sono trattati in modo uniforme, e documentano come sono state effettuate le modifiche.

Questo ammonta a un totale di 28 differenti coppie, che sono tutte presentate nella tabella 2, in ordine sparso. Le righe numerate dall'1 al 28 sono state aggiunte solo per facilitarne la consultazione.

Inoltre, abbiamo scelto di fornire alla coppia in questione una nostra denominazione, invece di mostrarle in forma anonima.

Sei differenti denominazioni appaiono nella tabella:

- Il termine 'plagio' vuol dire che nella coppia (la coppia che presenta quel termine nel suo nome) è stata confermata la presenza di plagio.
- Il termine `riutilizzo 'è sinonimo di un riutilizzo legittimo dello stesso programma (dallo stesso studente, in anni di corso differenti, in modo che non ci sia nessun plagio)
- Ci sono tre coppie (righe 21, 25 e 28) denominate `stessa osservazione ': infatti queste consistono di osservazioni identiche (senza plagio). Solo SIM trovato queste coppie, il che non è molto sorprendente, poiché SIM in origine è uno strumento di rilevamento della clonazione.
- Infine, `file piccolo 'indica che la somiglianza deriva dal fatto che i file in questione sono solo molto piccoli (niente plagio).

La tabella 3 contiene coppie rilevanti (denominate `plagio ',` riutilizzo' o `simile ') tutte queste coppie sono davvero molto simili e dovrebbero essere studiate. Ci sono 4 coppie irrilevanti (denominate `stessa osservazione ' o `piccoli file') che non appartengono in alcun top-10. Le altre 11 coppie (`falso allarme') non si trovano nella top 10 ma ci dovrebbero stare.

Le righe 1, 2 e 17 sono collegate: la coppia nella riga 1 consiste nella coppia (A, C) quella nella riga 17 consiste nella coppia (B, C), dove A e B sono nuove osservazioni, mentre C è un osservazione presentata da un allievo differente un anno prima.

La riga 2 rappresenta la coppia (A, B). Gli errori di analisi sono causati da B che inizia l'analisi in maniera non corretta.

Ogni colonna mostra, per lo strumento menzionato nell'intestazione di colonna, i punteggi della sua top 10, più i punteggi dei restanti 18 coppie.

La ragione per cui Marble, a differenza degli altri mostra un punteggio per ogni riga (ad eccezione di quelli contrassegnati come `stessa osservazione ', dal momento che questi sono confrontati solo da SIM), è che in questa particolare

esecuzione Marble genera un punteggio per ogni coppia che viene confrontata, mentre gli altri strumenti generano punteggi fino a una certa soglia.

- Le denominazioni `falso allarme ' e ' simili ' hanno solo una sottile comune differenza: una coppia chiamata ‘simile’ contiene alcuni file che sono davvero simili, ma non plagiati, mentre` falso allarme' significa che il file non erano nemmeno simile (e anche non plagiato).

	Jplag	**Marble**	**MOSS**	**Plaggie**	**SIM**
1 plagio		86	46	60	30
2 plagio	Errore analisi	82	49	Errore analisi	
3 plagio	94	94	72	89	70
4 falso allarme		43		94	4
5 falso allarme		44		94	96
6 falso allarme		44		94	
7 falso allarme		44		94	
8 riutilizzo	95	92	60	92	61
9 simile	100	84	87	100	100
10 riutilizzo	100	100	84	100	
11 falso allarme		44		94	
12 falso allarme	84	56		65	
13 falso allarme	84	52		69	
14 falso allarme	85	57		69	
15 riutilizzo	100	100	73	100	100
16 falso allarme		47		94	

17 plagio	Errore analisi	92	71	Errore analisi	73
18 falso allarme		28			83
19 plagio	84	89	49	92	56
20 falso allarme	86	54			
21 stessa osservazione					100
22 simile	79	94		90	43
23 file piccolo		31			97
24 riutilizzo	86	89	80	97	75
25 stessa osservazione					100
26 riutilizzo	83	61	63	79	63
27 simile	85	75		82	49
28 stessa osservazione					100

Tabella 3

Interpretazione dei risultati

Da questi risultati possiamo osservare che i top 10 di JPlag, Marble e MOSS sono abbastanza simili, mentre i top 10 di Plaggie e SIM differiscono di parecchio dagli altri tre. Qui di seguito, descriviamo i top 10 più in dettaglio.

Prestazioni Jplag: Per molti file, non vi è alcun risultato per JPlag, la cui causa è ancora ignota e deve essere indagata. Dei 5 casi di plagio, JPlag non riesce ad analizzarne due a causa di errori di analisi e uno perché non c'è nessun punteggio. Gli altri due hanno punteggi abbastanza alti.

Prestazioni Marble: Marble fa abbastanza bene, ottenendo un risultato per quasi tutti i casi di plagio, “riutilizzo“ e ”simili” danno punteggi di 82 o superiore (le due eccezioni sono le righe 26 e 27, ma ottiene dei punteggi anche per le categorie irrilevanti `falso allarme ', `stessa osservazione 'e` file piccoli').

Prestazioni MOSS: MOSS da dei risultati per tutti tranne che per due coppie che saranno sottoposte ad ulteriori indagini. A volte i punteggi di queste coppie sono piuttosto bassi, ma, come discusso in precedenza, questo potrebbe essere solo una proprietà di MOSS. In altre parole, il punteggio di soglia di MOSS potrebbe essere solo inferiori a quelli degli altri strumenti.

Prestazioni Plaggie: Plaggie restituisce per molti casi di “falso allarme” un punteggio molto alto. Inoltre (o di conseguenza), diversi casi che dovrebbero essere investigati finiscono più in basso della top 10 a causa dei risultati ottenuti sulle prove di “falso allarme" (soprattutto i plagi nelle righe 1, 3 e 19). Plaggie non riesce ad analizzare due casi di plagio a causa di errori di analisi.

Punteggi prestazioni SIM: SIM sembra piuttosto imprevedibile, si veda ad esempio la gamma di punteggi per i casi di “plagio”,` riutilizzo' e `simile ', che è 30 (per plagio) e 100 (per “simile” e `ripresentazione '). Inoltre, vi è un 'falso allarme' nella riga 5 con il punteggio più alto di 96 (nota, tuttavia, che questo è l'unico `falso allarme 'che è stato ritenuto abbastanza importante da SIM.
Questo caso dovrebbe essere studiata ulteriormente.
Posizioni 1, 2 e 3 dalla top 10 di SIM contengono “stesse osservazioni” (righe 28, 25 e 21, rispettivamente). La linea 21 in realtà appare anche sulla posizione 9 di questa top 10. Inoltre, la riga 9 appare su posizioni 5 (punteggio 100) e 6 (punteggio 99, non menzionato nella tabella 4). Infine, la riga 23 ('piccoli file') è in posizione 7. Riassumendo, SIM tende a calcolare i punteggi più alti di somiglianza per i casi che chiaramente non appartengono alla top-10. Principalmente, questo è causato dal fatto che SIM confronta una osservazione con se stesso. Questo non è così sorprendente, poiché SIM stessa è uno strumento di rilevamento della clonazione, che garantisce il confronto di se stesso. Un altro motivo per cui la top 10 contiene coppie irrilevanti è il fatto che SIM non prevede la facoltà di escludere file sotto una certa dimensione.

Confronto mancanze per SIM e MOSS: La coppia nella riga 10 non ha risultato per SIM, e la riga 22 e 27 non ha risultati per MOSS. Tuttavia, entrambe le righe hanno chiaramente bisogno di uno sguardo più attento (anche se entrambi si è

rivelato non essere plagiate). Forse questo è un bug dei rispettivi programmi o un errore nell'uso degli stessi per l'esperimento.
Ciò richiede uno sguardo più da vicino a SIM e MOSS per vedere cosa esattamente sta succedendo.

Errori di analisi: ora analizzando gli errori nelle osservazioni provenienti dalle righe 2 e 17 possiamo capire come questi due casi di plagio vengano rilevate da entrambe JPlag e Plaggie. Questi errori di analisi non sono stati causati dagli strumenti, ma dalle osservazione che sono state controllate dagli stessi.
Fortunatamente entrambi gli strumenti tengono un registro in cui sono menzionati questi due casi. Tuttavia, è possibile presentare intenzionalmente un file che è sintatticamente incorretto. Piccoli errori, per esempio (volutamente) dimenticare una parentesi chiusa, sono di solito corretti da chi controlla l'osservazione. Per garantire che questo tipo di trucco non possa aiutare a mascherare il plagio, si potrebbe decidere di accettare la compilazione del programmi solo per la classificazione.

13.7 Efficienza a runtime

L'efficienza può essere un problema importante per i programmi di rilevamento del plagio. I programmi sono spesso eseguiti su un gran numero di osservazioni, e queste osservazioni possono contenere più file. Anche se non sono stati effettuati esperimenti di temporizzazione precisi possiamo almeno dire che per gli esperimenti che abbiamo fatto, abbiamo incontrato gravi problemi con l'efficienza di tutti i programmi presi in considerazione.
Il tempo di esecuzione variare da alcuni minuti a circa 30 minuti.
Noi pensiamo che un tempo di esecuzione di 30 minuti è ancora accettabile ai fini di rilevare il plagio in un ambiente accademico.
Però, tali strumenti non possono essere accettabili per l'uso in un ambiente con più stretti vincoli di tempo. Si noti inoltre che i programmi offrono un servizio che può essere molto veloce, ma se vi capita di verificare la presenza di plagio esso può richiedere un certo tempo prima di produrre dei risultati.

Conclusioni

In questo lungo discorso sul plagio e sui software che lo possono identificare, sia di tipo testuale che nel codice sorgente degli applicativi, abbiamo potuto constatare come la battaglia per contrastare questo fenomeno sia ancora all'inizio, Tale situazione è dovuta, da un lato, ad un discorso di disinteresse verso il problema da parte degli organismi preposti, dall'altro lato perché gli approcci ad oggi più utilizzati e sviluppati non ci permettono di individuare con certezza la maggior parte degli episodi di plagio, ma spesso danno dei falsi positivi (di più nel caso dell'analisi del codice sorgente rispetto a quella testuale).

In ambito universitario Compilatio è il software più diffuso ed utilizzato, si tratta di un software sviluppato da un privato che offre un servizio più completo e preciso possibile.

Gli altri programmi di analisi del testo sono sviluppati anch'essi da aziende private di tutto il mondo, essendo questo un problema comune, ma appunto perché sono sviluppati da privati il loro codice è tutelato e non accessibile, quindi non è possibile analizzarli e spiegarne la logica e il funzionamento.

Ci sono dei programmi open-source come CitePlag (del quale ho discusso), ma sono un po datati inoltre, proprio per il fatto che il loro codice fornisce una base che può essere modificata da chiunque, sono soggetti ad adattamenti frammentari, effettuati in base a casi particolari e senza una visione di insieme tesa ad una costante evoluzione.

I software di rilevamento del plagio di tipo testuale sono prevalentemente di tipo Substring matching e di analisi delle citazioni perché sono quelli che ottengono i migliori risultati in breve tempo mantenendo una efficienza elevata.

I software che rilevano il plagio nel codice sorgente sono moltissimi e più complessi da implementare rispetto a quelli testuali, anche in questo caso la maggior parte di questi programmi non è open-source gli unici software aperti sono Plaggie e SIM.

Confrontando i programmi trattati nell'ultimo capitolo possiamo fare delle considerazioni.

Abbiamo confrontato questi programmi attraverso le dieci caratteristiche principali e abbiamo confrontato la performance di un'analisi di sensibilità su un gruppo di codici sorgenti intenzionalmente plagiati in diverse maniere e facendo

anche un confronto top n, in questo caso, esaminiamo i primi 10 risultati per ogni strumento rispetto ai risultati degli altri.

I risultati del confronto ci danno una buona comprensione dei punti di forza e di debolezza dei differenti programmi.

I risultati dei due confronti possono essere riassunti come segue:

- Molti programmi sono sensibili a numerosi piccoli cambiamenti.

- Tutti i programmi denotano efficienza in presenza di singole modifiche che vengono apportare al programma, ma molti programmi hanno dei punteggi piuttosto bassi quando si è in presenza di modifiche combinate e si comportano peggio di DIFF che ottiene risultati migliori.

- Un risultato sorprendente del confronto top 10 è che i top 10 di JPlag, Marble e MOSS sono abbastanza simili, mentre i top 10 di Plaggie e SIM differiscono parecchio.

- Lungo la strada abbiamo scoperto alcuni casi in cui sarebbe necessario un esame più dettagliato del comportamento dei programmi in questione.

BIBLIOGRAFIA/SITOGRAFIA

1) Alma Mater Studiorum - Codice etico - Art. 27 comma 3. emanato con DR. n. 1408/14 del 01/10/2014 BUR Supplemento Straordinario n. 93 del 31/10/2014) www.normateneo.unibo.it
2) ANSA Redazione internazionale - Cina, Apple accusata di plagio per Siri – 27 marzo 2013 – ANSA – Roma www.ansa.it
3) Caso Roberto – Plagio, diritto d'autore e rivoluzioni tecnologiche – Trento – Università degli studi di Trento Dipartimento scienze giuridiche – 2011
4) Compilatio – Perché prevenire il plagio 2014 Saint Félix France - compilatio.net
5) Compilatio – Software anti plagio FAQ 2014 Saint Félix France - compilatio.net
6) Di Cesare Loredana, Testi, discorsi e programmi copiati: esiste il software anti plagio – 27 luglio 2013 www.ilfattoquotidiano.it
7) Hage Jurrian, Domain specific Type error diagnosis – Techincal Report giugno 2014 . Department of Information and Computing Sciences Utrecht University, Utrecht, The Netherlands www.s.uu.nl/research
8) Meuschke, Gipp, Bretlinger, Citeplag: a citation based plagiarism detection system prototype Conference 2012 – plagiarismadvice.org
9) Northwestern University, Northwestern's "Principles Regarding Academic Integrity" Evanston, Illinois USA settembre 2015 - www.northwestern.edu
10) POSNER R.A., *Il piccolo libro del plagio*, Roma, 2007
11) Setti Raffaella - Plagio, origine e significati antichi e moderni – Firenze 22 dicembre 2009 www.accademia della crusca.it
12) studiolegaleonline.net – Milano 2014
13) Università Bocconi, Honor Code Cosa succede se.... . Milano – 6 ottobre 2014 – unibocconi.it
14) Università di Roma La Sapienza, Linee guida sul plagio – Roma novembre 2014 www.we.uniromal.it
15) Università degli Studi di Verona, Codice etico Verona www.univr.it
16) Wikipedia Enciclopedia libera, Il plagio nel diritto d'autore – it.wikipedia.org
17) Wikipedia Enciclopedia libera, Plagiarism detection -en.wikipedia.org