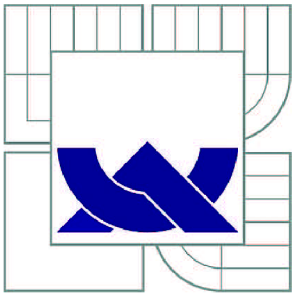
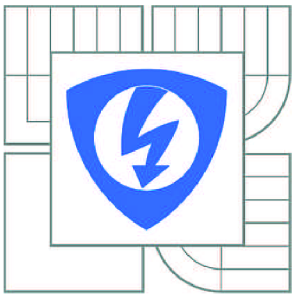


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV BIOMEDICÍNSKÉHO INŽENÝRSTVÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF BIOMEDICAL ENGINEERING

DETEKCE PLAGIÁTŮ PROGRAMOVÝCH KÓDŮ

PLAGIARISM DETECTION OF PROGRAM CODES

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

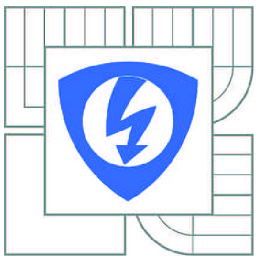
AUTOR PRÁCE — Bc. ANEŽKA NEČADOVÁ

AUTHOR

VEDOUCÍ PRÁCE — Ing. MARTIN VÍTEK, Ph.D.

SUPERVISOR

BRNO 2015



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav biomedicínského inženýrství

Diplomová práce

magisterský navazující studijní obor
Biomedicínské a ekologické inženýrství

Studentka: Bc. Anežka Nečadová ***ID:*** 128130
Ročník: 2 ***Akademický rok:*** 2014/2015

NÁZEV TÉMATU:

Detekce plagiátů programových kódů

POKYNY PRO VYPRACOVÁNÍ:

1) Seznamte se podrobně s definicí plagiátorství z pohledu programových zdrojových kódů a nastudujte možnosti současných metod jeho detekce. 2) Vytvořte vlastní databázi vhodnou pro testování plagiátorství. 3) Nalezněte příznaky pro detekci plagiátorství a v prostředí Matlab separátně otestujte jejich vhodnost na vytvořené databázi. Dosažené výsledky diskutujte. 4) Realizujte detektor plagiátorství založený na kombinování vhodných příznaků. 5) Navržený detektor otestujte a dosažené výsledky statisticky zpracujte. 6) Program opatřete vhodným grafickým uživatelským rozhraním.

DOPORUČENÁ LITERATURA:

[1] CHÝLA, R. Detekce plagiátorství. Ikaros [online]. 2009, roč. 13, č. 2. Dostupné z: http://ikaros.cz/node/5253
[2] SI, A., H.V. LEONG a R.W.H. LAU. CHECK: A Document Plagiarism Detection System. In Proceedings of ACM Symposium for Applied Computing. February 1997, s. 70–77.

Termín zadání: 9.2.2015 ***Termín odevzdání:*** 22.5.2015

Vedoucí práce: Ing. Martin Vítek, Ph.D.
Konzultanti diplomové práce:

prof. Ing. Ivo Provazník, Ph.D.
Předseda oborové rady

UPOZORNĚNÍ:

ABSTRAKT

Práce uvádí definice pojmu plagiátorství a zaměřuje se především na řešení tohoto problému na akademické půdě. Hlavním tématem je detekce plagiátu. Jsou zde rozebrány jednotlivé kroky procesu detekce a zvláštní pozornost je věnována detekci plagiátu programových kódů. Práce se pozastavuje nad spolehlivostí detekčních nástrojů a snaží se rozdělit metody detekce plagiátů do základních skupin. Jedna kapitola je věnována metrikám pro porovnávání souborů. Zmíněny jsou dva dostupné nástroje pro detekci plagiátů. V poslední kapitole je rozebrán vlastní návrh programu pro detekci plagiátu programových kódů a graficky zhodnoceny výsledky aplikace detektoru na databázi studentských prací.

KLÍČOVÁ SLOVA

Detekce plagiátu, plagiátorství, metrika, zdrojový kód, Matlab

ABSTRACT

This semestral thesis presents definition of plagiarism and focuses primarily on solving this problem in academic world. The main topic is the detection of plagiarism. It is discussed the various steps of the detection process and special attention is given to plagiarism detection of program codes. The work mentions question of the reliability of detection tools and divides the plagiarism detection methods into basic groups. One chapter is devoted metrics for comparing files. Mentioned are two tools available to detect plagiarism. In the last chapter is analyzed own draft program for plagiarism detection of program codes. The detector was applied to a database of student's works, and the results were plotted.

KEYWORDS

Plagiarism detection, plagiarism, metric, source code, Matlab

NEČADOVÁ, A. *Detekce plagiátů programových kódů*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2015. 51 s., 2 s. příloh. Vedoucí diplomové práce Ing. Martin Vítek, Ph.D.

PROHLÁŠENÍ

Prohlašuji, že svoji diplomovou práci na téma Detekce plagiátu programových kódů jsem vypracovala samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušila autorská práva třetích osob, zejména jsem nezasáhla nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědoma následků porušení ustanovení § 11 a následujících zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

V Brně dne ....

(podpis autora)

PODĚKOVÁNÍ

Děkuji vedoucímu diplomové práce Ing. Martinu Vítkovi, Ph.D., za účinnou metodickou, pedagogickou a odbornou pomoc a další cenné rady při zpracování mé diplomové práce.

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

ÚVOD

Téma plagiátorství se v dnešní době internetu a elektroniky skloňuje ve všech odvětvích stále častěji. Vzhledem k tomu, že lze mnohem snadněji získávat informace a využívat různé online zdroje dokumentů či dokonce hotových prací, je nasnadě věnovat se možnostem rozpoznávání plagiátů a neautorských prací.

Cílem této práce je přiblížit pojem plagiátorství a rozebrat problematiku z technického pohledu detekčních nástrojů. Cílem praktické části je nalezení příznaků pro detekci plagiátorství programových kódů a v prostředí Matlab otestovat vhodnost na vytvořené databázi. Následně realizovat detektor založený na kombinování vhodných příznaků a opatřit ho grafickým uživatelským rozhraním. Navržený detektor otestovat a výsledky statisticky zpracovat.

V předkládané diplomové práci je na úvod rozebrána definice plagiátu a jsou zde uvedeny příklady plagiátorství studentských prací. Okrajově je zmíněna i prevence tohoto problému.

V další kapitole je popsán proces detekce plagiátu a základní rozdělení detekčních nástrojů. Zároveň práce neopomíjí témat spolehlivosti těchto nástrojů a nechybí zde ani ukázka dvou dostupných programů JPlag a CodeMatch.

Samotná podkapitola je věnována dvěma metodám, a to metodě počítání atributů a metodě porovnávání řádků, které slouží k detekci plagiátu programových kódů. Podkapitola dále zahrnuje rozbor základních metrických vzorců.

Třetí část práce se věnuje mnou navrženému programu, který se snaží detekovat plagiát zdrojového kódu na základě počítání atributů a porovnávání řádků. Tento program byl vytvořen v prostředí Matlab a je prioritně určen pro porovnávání studentských projektů odevzdávaných v rámci předmětů zabývajících se výukou programování rovněž v programu Matlab.

Testování mnou vytvořeného programu bylo provedeno nejprve separátně pro jednotlivé příznaky na uměle vytvořené databázi. Následně byl naprogramován detektor, který vyhodnocuje podobnost na základě kombinace všech příznaků. Ten byl aplikován na reálnou databázi studentských prací. Výsledky jsou přehledně přeneseny do grafů a tabulek a na jejich základě byla provedena podrobná diskuze. Přiloženy k náhledu jsou i konkrétní úseky porovnávaných dvojic kódů.

V další kapitole je popsáno zpracování grafického uživatelského rozhraní, kterým je program pro detekci opatřen, neboť je důležité mít na paměti, že uživatel často

nehodnotí pouze funkčnost programu, ale i jeho uživatelskou přístupnost.

1 PLAGIÁTORSTVÍ

Před samotným hlavním tématem, které se zabývá detekcí plagiátů, je zde definován pojem plagiátorství a to, kde se s ním lze setkat. Tato úvodní kapitola se dále zaměřuje na plagiáty v akademické prostředí a boj s nimi.

1.1 Definice

Slovo plagiátorství se vyskytuje v různých prostředích a souvislostech už od pradávna. Je tedy vhodné hned na úvod vysvětlit, co tento pojem znamená. Slovo je odvozeno z latinského výrazu *plagiarius*, jehož doslovný překlad je *únosce*. V dnešní době je slovo plagiát používáno ve významu opisování, vydávání cizí práce za vlastní, neuvedení zdrojů, ze kterých práce čerpá, nebo publikování cizích myšlenek bez zmínění pramenů. Následuje výčet příkladů konkrétních definic.

„Norma ČSN ISO 5127-2003 definuje plagiát jako představení duševního díla jiného autora půjčeného nebo napodobeného v celku nebo z části jako svého vlastního.“ [10]

V otevřené encyklopedii Wikipedie lze nalézt jednoduchou definici, že „plagiát je umělecké nebo vědecké dílo, jež někdo jiný než skutečný autor neprávem vydává za své“. [19]

Ústřední knihovna ČVUT na svých internetových stránkách uvádí, že „za plagiátorství se považuje nejen úmyslné okopírování (ukradení) cizího textu a jeho vydávání za vlastní, ale i nedbalé citování, neúmyslné opomenutí citace některého využitého zdroje a myšlenky či nedostatečná práce s původním textem (nedostatečná parafráze, kompilace původního textu).

Plagiátorství se dopouští nejen ten, kdo takovýmto způsobem neoprávněně cizí text využije, ale i osoba, která poskytuje služby, jež plagiátorství přímo umožňují, popřípadě k němu nabádají. Tedy strana, která tyto texty zdarma či za úplatu produkuje a/nebo poskytuje.“[11]

Podle zahraničních výukových webových stránek ics.heacademy.ac.uk „je plagiátorství úmyslná nebo neúmyslná reprodukce (kopírování, přeformulování, parafrázování, přizpůsobení atd.) práce, která byla vytvořena jinou osobou (osobami), bez řádného uvedení zdroje.

Povolení takové reprodukce, ať už úmyslně nebo z nedbalosti, může rovněž

představovat plagiátorství.

Práce, které mohou být plagiát, zahrnují: slova (jazyk), nápady, poznatky, spisy, grafická znázornění, počítačové programy, diagramy, grafy, ilustrace, kreativní práce, informace, přednášky, tištěné materiály, elektronické materiály, nebo jakýkoli jiná původní díla vytvořená někým jiným.“[3]

1.1.1 Autoplagiátorství

Speciální a neméně závažný typ plagiátorství je autoplagiátorství. Tedy opětovné použití své vlastní práce v práci nové, bez upozornění na tuto skutečnost. I když jde o dílo jednoho autora, musí se na něj odkázat citací. Na první pohled se to nemusí zdát zavrženíhodné, avšak při představě situace, kdy student ve své dizertační práci použije značnou část své diplomové práce bez řádného citování, dá se to chápat jako snaha získat druhý titul za stejnou práci. To už se za opominutelný problém pokládat nedá.

1.2 Příklady studentského plagiátorství

Plagiátorství se prvotně řeší v akademické sféře, proto jsou v této kapitole uvedeny příklady plagiátorství u studentů.

1.2.1 Příklady z plagiátorství studentských esejí

V odevzdávané práci psané v přirozeném jazyce je nutné uvést zdroj a autorství pasáží, které student převzal, a není jejich autorem. Je nutné použít uvozovky a správné odkazování. Zde jsou uvedeny některé příklady plagiátorství v esejích:

- Kopírování práce (nebo její části) jiného studenta a prohlašování jí za vlastní. Student, který umožní kopírování vlastní práce, je za plagiátorství také zodpovědný.
- Kopírování materiálu z jednoho nebo z více zdrojů, aniž by byly správně uvedeny všechny zdroje.
- Parafrázování textu z jednoho nebo z více zdrojů bez správného označení a uvedení zdroje.
- Krádež práce někoho jiného a její předložení jako své vlastní.
- Zadat někomu jinému, ať napíše práci nebo ji koupit z on-line databáze prací.
- Znovupoužití práce z již v minulosti odevzdané.

- Spolupráce s jedním nebo více studenty při vytváření úkolu, pokud to není povoleno.

1.2.2 Příklady studentského plagiátorství v počítačových programech

Studenti jsou povinni uvést zdroj a autorství zdrojového kódu, který nebyl původně napsaný jimi, v zadání zdrojového kódu, v rámci programu zdrojového kódu formou komentářů nebo v příslušné dokumentaci. Zde jsou některé příklady z plagiátorství ve zdrojových kódech:

- Reprodukce/kopírování zdrojového kódu, aniž by byl změněn, bez řádného uvedení zdroje.
- Minimálně nebo mírně upravený zdrojový kód napsaný někým jiným, aniž by byl uveden autor.
- Převod části nebo celého zdrojového kódu někoho jiného v podobném programovacím jazyce.
- Použití kódu vygenerovaného softwarem pro tvorbu kódu bez uvedení této skutečnosti.
- Zaplacení někomu za vytvoření programu.
- Spolupráce s jiným studentem na tvorbě programu, pokud to není povoleno.

1.3 Prevence

Přestože se práce zabývá detekcí, je považováno za vhodné uvést i možnosti prevence plagiátorství, které by na akademické půdě dopomohli předcházet problémům spjatým s tímto tématem. Podle zahraničních výukových webových stránek ics.heacademy.ac.uk [3] je nutné, aby univerzity dodržovaly několik základních bodů:

- Sami se vzdělávaly v tom, co je plagiátorství.
- Obeznámily se a dodržovaly své univerzitní předpisy o plagiátorství.
- Vzdělávaly své studenty v tom, co je plagiátorství, a zajistily, aby pochopili důsledky z plagiátorství. Ujistily se, že studenti chápou otázky spojené s podváděním, nekalými praktikami (tj. nevhodnou spoluprací), podvodem a padělaním.
- Vytvářely takové zadání prací, které by ztěžovalo studentům vytvářet plagiáty.

- Informovaly studenty o tom, co bude a co nebude považováno za plagiát v konkrétních úkolech.
- Používaly internetové zdroje a nástroje pro detekci plagiátorství.
- Dokumentovaly komunikaci se studenty, což v případě odhalení plagiátorství může sloužit jako důkaz u soudu.
- Při odhalení plagiátu vždy podnikly patřičné kroky.

2 DETEKCE PLAGIÁTORSTVÍ

Hlavním tématem práce je detekce plagiátu. Tato kapitola má za úkol uvést do problematiky tohoto tématu. Rozebírá se zde proces detekce, rozdělení detekčních nástrojů a zvláště se zaměřuje na detenci plagiátů programových kódů.

2.1 Vodítka k podezření z plagiátorství

Ne vždy se aplikuje na každou odevzdanou studentskou práci některý z nástrojů pro detekci plagiátorství. Většinou se využívá až u případu, kdy vyučující nabyde určitého podezření o její nekorektnosti. Zde je nutné předložit výčet bodů, které mohou vést k tomuto podezření a utvrdit vyučujícího v rozhodnutí se využít určitý detektor plagiátů.

- Neobvyklé formátování, jako jsou změny v okrajích, podivné odstupy, nekonzistentní velikost a barva písma, podivné číslování stránek.
- Změny ve stylu psaní a chybná syntax textu. Například smíšené styly odstavců můžou být známkou toho, že student pouze zkopíroval a vložil materiál.
- Změny v pravopisném stylu, či dokonce použití Slovenštiny u českého studenta a naopak.
- Oddíly nebo věty, které spolu nesouvisí, a nenavazují na sebe.
- Použití nespisovné nebo rozšířené slovní zásoby.
- Smíšené citační styly.
- Chybějící odkazy nebo uvozovky. Použití uvozovek u citací, které neodkazují na seznam použité literatury.
- Použití starých neaktuálních informací. Uvedení odkazů na nefunkční webové stránky může být známka toho, že práce byla napsána někdy v minulosti.
- Změny v kvalitě práce studenta v různých částech zadání nebo v různých jednotlivých úkolech.
- Vodoznak prozrazuje, že práce byla vytištěna na internetu.

2.2 Spolehlivost detekčních nástrojů

Je velice důležité zdůraznit, že sebedokonalejší program k jednoznačné detekci plagiátu pouze dopomůže. Konečný rozsudek, zda se jedná o plagiát, musí vždy pronést fyzická osoba. Úkolem detekčních nástrojů je pouze upozornit na určité znaky, které by mohly o plagiátorství svědčit. Například při detekování velké shody s jinou prací se může jednat o případ, kdy sice byla použita část cizí práce, ale přitom řádně ocitovaná, a tedy se o plagiátorství nejedná.

Nejen že nelze s určitostí říct, že nástrojem označený soubor je plagiát, ale i naopak, pokud program dokument za plagiát neoznačí, neznamená to, že se o plagiát nejedná. Doposud například neexistuje žádný nástroj, který by byl schopen odhalit plagiát vzniklý překladem dokumentu z cizího jazyka, i kdyby byl překlad doslovný. Odhalit nelze ani plagiát, u kterého neznáme zdroj, z kterého bylo čerpáno.

2.3 Proces detekce plagiátorství

Obecně můžeme rozdělit proces detekce plagiátorství do čtyř kroků:

- sběr,
- analýza,
- potvrzení,
- vyšetřování.

V první fázi je třeba vytvořit elektronickou databázi prací. Práce jsou však většinou nutné přepracovat do vhodného jednotného formátu.

V dalším kroku jsou dokumenty porovnávány buď mezi sebou navzájem ve vytvořeném shromaždišti prací, či se soubory získanými z webu. Výstupem jsou buď dvojice nápadně podobných prací, nebo jednotlivé detekované dokumenty s podezřením na plagiát.

Následuje fáze verifikace neboli potvrzení, zda analýzou zvolené dokumenty jsou opravdu plagiáty. Jak již bylo zmíněno výše, i za použití sebedokonalejší analyzační techniky musí být při řešení detekce plagiátorství vždy přítomna osoba, která provede závěrečný verdikt s ohledem na možné disciplinární opatření.

V posledním kroku se započne vyšetřování a určí se rozsah údajného pochybení. Tato fáze zahrnuje také proces určení viníka a případné sankce.

2.4 Rozdělení

Postupem času byly vyvinuty různé detekční nástroje, které se od sebe mohou v různých ohledech lišit. V této kapitole jsou rozděleny do jednotlivých skupin, které jsou následně podrobněji rozebrány. Pro toto rozdělení platí, že není striktní, jednotlivé skupiny se mohou navzájem prolínat, je nutné brát v potaz také to, že není jediné a uvedené okruhy se mohou dále podrobněji dělit a třídit dle stanovených hledisek. Vzhledem k praktičnosti této práce byl zvolen pouze základní, méně podrobný výčet, který si klade za cíl jasné uvedení do problematiky.

Pro přehlednost je nejprve uveden výčet, kterým skupinám je zde věnována pozornost.

1. Rozdělení podle způsobu detekce plagiátů:
   - nástroje pracující s vnitřním obsahem,
   - detekce na základě značkování.
2. Rozdělení podle způsobu porovnávání:
   - intrakorpální,
   - extrakorpální.
3. Rozdělení podle typu dokumentů:
   - detekce plagiátorství volného textu,
   - detekce plagiátorství programových kódů.

2.4.1 Způsob detekce

Podle způsobu detekce plagiátu se rozlišují nástroje pracující s vnitřním obsahem nebo na základě značkování.

Známým příkladem druhého typu nástroje je zavedení vodoznaku. Méně pozorný plagiátor je tak bezpečně detekován a jasný je i zdroj, ze kterého čerpal. Dále se zavádí neviditelné značkování do textu v podobě drobných změn šířky mezi řádky či odstavci, rozložení textu na stránce a podobně. Způsob je podrobněji rozebrán dále.

Převažující způsob detekce pracuje s vlastním obsahem dokumentu. Tyto nástroje se dále rozdělují podle způsobu porovnávání, který je uveden v kapitole 2.4.2.

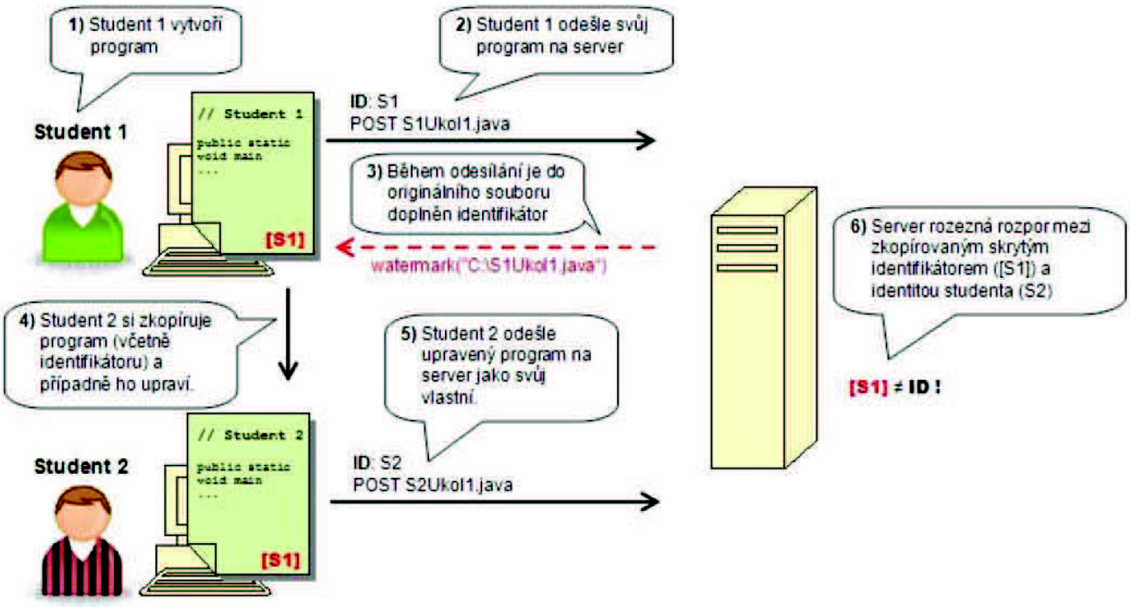
Detekce na základě značkování

Metoda vychází z principu opatření dokumentu vodoznakem. Když je vodoznak zjevný,

funguje spíše jako prevence. Například jméno či jiný identifikační údaj napsaný diagonálně v pozadí za textem. Zavést identifikační údaje do textu však lze i takřka neznatelně, a to pomocí netisknutelných znaků jako jsou mezery, tabulátor nebo různé velikosti řádkování.

V praxi se to realizuje tak, že při odevzdávání studentské práce do elektronického systému jsou do ní utajeně přidány identifikační údaje, které jsou zároveň zapsány i přímo do originálního souboru na disku studenta. Identifikační údaje jsou zakódovány právě pomocí bílých znaků (white space). Jako identifikační údaj se nabízí použít například přihlašovací údaje do elektronického systému, kde se práce odevzdává.

Pokud student poskytne svoji již odevzdanou a tedy označkovanou práci někomu jinému a ten ji pak odevzdá do systému, byť značně upravenou, je identifikace plagiátora a zároveň i poskytovatele práce jednoznačná. Proces metody názorně ukazuje obrázek 1.



Obrázek 1: Proces detekce na základě značkování [8]

S postupem času však začíná být patrné, že metoda má spoustu nedostatků. Hlavním opatřením, které je nutné dodržet, je utajení tohoto nástroje. Při jeho prozrazení je naprosto jednoduché ho obejít. Ale i při nevědomosti o přítomnosti této detekce stačí, aby autor poskytl jinou než odevzdávanou verzi práce, byť její přesnou kopii vytvořenou ještě před odevzdáváním, a plagiátor ani poskytovatel nebude odhalen.

Naproti tomu má však tento způsob detekce plagiátorství jedinečné výhody.

Nejedná se totiž o pouhé upozornění na podezřelé shody, ale s naprostou přesností odhaluje plagiát. Navíc jeho výstupem je odhalený plagiátor i poskytovatel práce. Oba účastníci jsou odhaleni okamžitě a není třeba čekat na všechny odeslané úkoly pro vytvoření korpusu pro porovnávání.

2.4.2 Způsob porovnávání

Nástroje pracující s vnitřním obsahem dokumentu analyzují vždy určitou množinu dokumentů, tzv. korpus. Podezřelé shody mohou hledat porovnáváním souborů mezi sebou (tzv. intrakorpální nástroje) nebo porovnáváním balíku souborů s externí databází (tzv. extrakorpální nástroje).

Intrakorpální nástroj zpracovává dokumenty jen v rámci korpusu, a tedy nedokáže odhalit plagiátorství z jiných zdrojů. Je to však jednoduchý a rychlý způsob porovnávání a může upozornit na spolupráci dvojice či skupiny autorů.

Extrakorpální nástroje neporovnávají soubory mezi sebou v daném korpusu, ale například se školní databází nebo s internetem. Výhodou je větší množina zdrojů, a tím i větší šance na odhalení plagiátu. Tato výhoda se však stává i nevýhodou ve smyslu většího počtu dat pro zpracování a s tím i rostoucí časové náročnosti analýzy.

Další rozdělení, které se s tímto dělením prolíná, rozlišuje softwarové a webové nástroje.

Webové nástroje mohou pracovat online a zahrnují nejen programy pracující s online databází, ale i používání internetového vyhledávače. Uživatel může zkopírovat a vložit podezřelé fráze převzaté z práce studentů do vyhledávače ve snaze nalézt online materiál, který obsahuje podezřelé fráze.

Softwarové nástroje pracují buď jen v rámci korpusu, nebo s určitou interní databází.

2.4.3 Typ analyzovaných dokumentů

Nástroje pro detekci lze obecně rozdělit podle toho, s jakými dokumenty pracují. Základní členění by mohlo být na texty v přirozeném jazyce a zdrojových kódech. Jednotlivé nástroje používají odlišný způsob zpracování.

U souborů s volným textem se dále řeší, v jakém jsou formátu (PDF, TXT apod.) Detekční nástroj buď pracuje s určitým typem, nebo je sám schopen převádět dokumenty do jednotného formátu. Typický je převod do čistého textu bez formátování.

U pokročilých metod se lze setkat s určitou snahou o optimalizaci textu pomocí

specializovaného slovníku. Nabízí se převod ohebných slov na základní tvar, práce se synonymy a podobně.

U zdrojových kódů tento problém víceméně odpadá, neboť jsou v naprosté většině ve formě běžných textových souborů. Nástroje se zaměřují na strukturu konkrétních programových jazyků. Je tedy žádoucí na každý odlišný jazyk implementovat vlastní modul. Podrobněji se tomuto tématu věnuje kapitola 2.6.

2.5 Metrika

Součástí každého detekčního nástroje je metrika, která zhodnocuje úroveň výskytu podezřelých znaků ve zkoumaném dokumentu nebo udává podobnost testovaných souborů. Vyjadřuje míru podezřelosti, že jeden dokument je plagiátem druhého.

Vhodná metrika pro použití v určitém programu se volí podle několika hledisek. Jedním hlediskem je *rozměr*, neboli velikost zpracovávané množiny dokumentů, podle kterého se rozlišují metriky singulární, párové a multidimenzionální.

Singulární metrika pracuje jen nad jedním dokumentem. V praxi může jít o různá stylometrická vodítka například počet a frekvence konkrétního slova, průměrná délka věty a podobně.

Převažující jsou metriky párové. Vyjadřují míru podobnosti dvou prací. Jedná se například o počet shodných slov, počet řádků, velikost souboru v bytech a jiné.

Multidimenzionální metriky pracují s celým korpusem prací a jejich úkolem může být nalezení shluku podobných dokumentů a následné odhalení skupinky autorů, kteří od sebe opisovali.

U párové metriky se dále řeší *symetričnost*. Výstupem symetrické metriky je hodnota pro oba porovnávané dokumenty stejná a nesymetrická metrika ohodnotí každý dokument jinou hodnotou.

Základní jednotkou pro metrické hodnocení může být slovo, řádek, věta, odstavec a podobně. V praxi se pro výpočet využívají standardní množinové operace sjednocení a průnik. Pro nastínění problematiky je stanoveno, že $J(X)$ je množina základních jednotek dokumentu X.

2.5.1 Symetrická metrika

Jedná se o párovou metriku, kde jsou porovnávány dva dokumenty ve vzorci označené A a B. Vzorec pro symetrickou metriku je nazýván podobnost (resemplance) a používá se zápis:

$$res(A,B) = \frac{|J(A) \cap J(B)|}{|J(A) \cup J(B)|}. \quad (2.5.1)$$

Výsledkem je číslo v množině <0;1>. Při porovnávání stejných dokumentů se výsledek rovná jedné, a pokud není nalezena žádná podobnost, výstupem metriky je nula. Základní vlastností je symetričnost:

$$res(A,B) = res(B,A). \quad (2.5.2)$$

Tato symetričnost platí i pro jednodušší vzorec, který se oproti předchozímu liší ve jmenovateli, kde je místo absolutní hodnoty počtu prvků sjednocení pouze součet prvků obou množin a je definován:

$$res(A,B) = \frac{|J(A) \cap J(B)|}{|J(A)| + |J(B)|}. \quad (2.5.3)$$

Výsledkem je číslo v množině <0;0,5>. To lze jednoduše vyřešit vynásobením dvěma pro následné vyjádření v procentech, které je často žádoucí.

Nevýhodou obou metrických vzorců je, že s rostoucí velikostí jednoho souboru (rozdílem jejich velikostí), klesá výsledné procento, a tedy i podezření z plagiátu.

2.5.2 Asymetrická metrika

Výše popsanou nevýhodu nestejné velikosti souborů řeší asymetrická metrika. Základní vzorec je nazýván obsah (containment) a je definován:

$$con(A,B) = \frac{|J(A) \cap J(B)|}{|J(A)|}. \quad (2.5.4)$$

Výsledkem je opět číslo v množině <0;1>. Pokud se výsledek rovná jedné, znamená to, že dokument *A* je celý obsažen v dokumentu *B*. Při nulovém výsledku nemají dokumenty žádný společný obsah. Zde tedy může platit:

$$con(A,B) \neq con(B,A). \quad (2.5.5)$$

2.6 Detekce plagiátů programových kódů

V předešlých kapitolách je obecně popsána problematika ohledně detekce plagiátu. Tato kapitola se zaměřuje na postupy využívané především při analýze programových kódů.

2.6.1 Předzpracování

Před samotnou analýzou a aplikací vybraných metod se provádí předpracování

zdrojového kódu v podobě několika základních kroků, které zjednoduší práci s obsahem dokumentu. Příkladem může být:

- převod celého kódu na malá písmena,
- odstranění prázdných řádků, mezer a odsazení,
- odstranění komentářů,
- nahrazení proměnných jednotnými názvy pro všechny porovnávané dokumenty.

2.6.2 Metoda počítání atributů

Pouze u zdrojových kódů se využívá metoda počítání atributů, která hledá podobnost na základě počtu určitých prvků. Nabízejí se slova, která se v určitém programovém jazyce typicky vyskytují. Pro program Matlab to mohou být například for, while, if, end, plot a podobně. Atributem může být i počet řádků nebo velikost dokumentu v bytech.

2.6.3 Metoda porovnávání řádků

Důkladnější a časově náročnější metoda porovnávání řádků je oproti předešlému způsobu nezávislá na druhu programovacího jazyka. Není potřeba znát typické prvky kódu, ale dá se říct, že metoda je univerzální.

Po odstranění mezer se porovnávají řádky jednoho dokumentu s řádky druhého dokumentu. Výsledky porovnávání nabírají hodnot 0 a 1 a jejich četnost je pak použita ve vhodné metrice pro vyhodnocení procentuální podezřelosti z plagiátu.

2.7 Nástroje

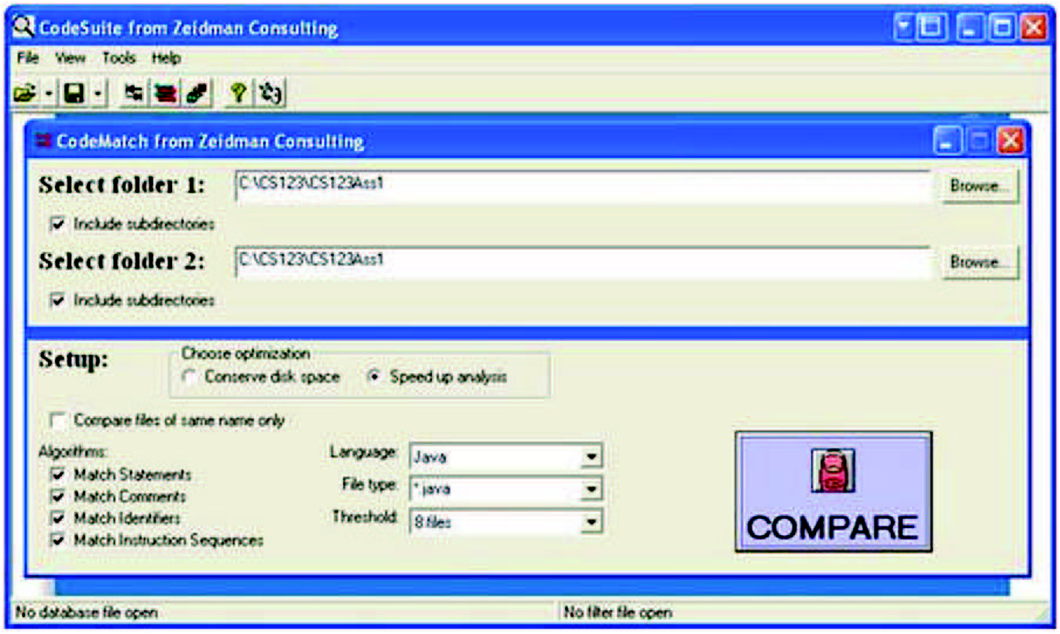
Zde jsou uvedeny některé dostupné nástroje pro detekci plagiátorství zdrojových kódů, jejich hlavní rysy, výhody a nevýhody. Vycházelo se především ze zahraničních výukových webových stránek ics.heacademy.ac.uk [3].

2.7.1 CodeMatch

Komerční nástroj CodeMatch je součástí softwaru CodeSuite od firmy Zeidman Consulting a je vybaven sofistikovanými algoritmy pro detekci plagiátorství zdrojových kódů. Pro názornost je zde uveden příklad aplikace programu na 51 zdrojových kódů s názvy 1.java až 51.java umístěných ve složce CS123Ass1.

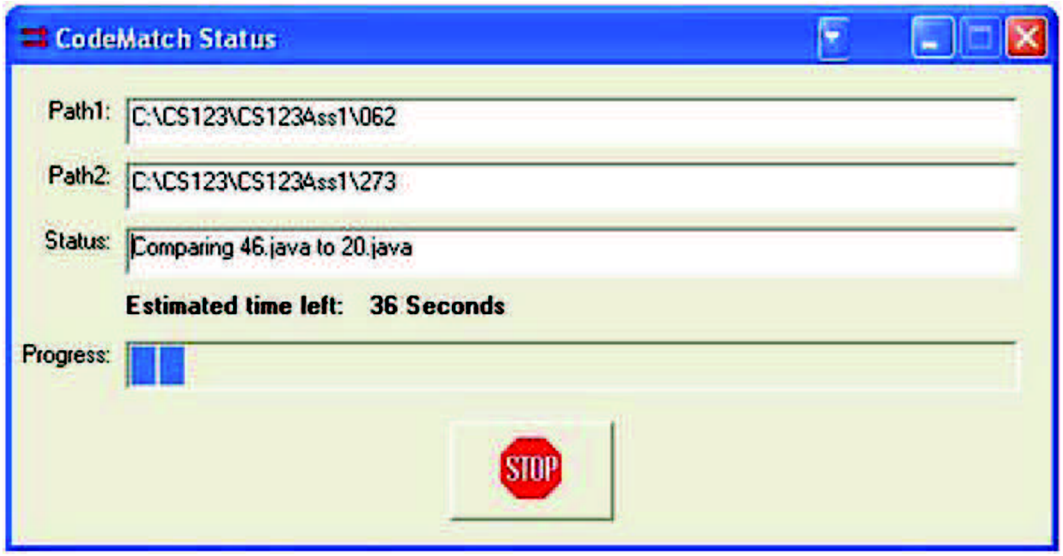
Po spuštění tohoto programu se nejprve vyberou složky, které obsahují

analyzované dokumenty. Případně se dvakrát zvolí stejná složka, pokud jsou v ní obsaženy všechny soubory pro porovnávání, jak je vidět i na obrázku 2. Dále se zvolí požadované nastavení (programový jazyk dokumentů, formát souborů, jaké algoritmy pro detekci použít a podobně).



Obrázek 2: Úvodní obrazovka programu CodeMatch [3]

Po stisknutí tlačítka pro porovnání *COMPARE*, začne srovnávání předložených souborů, viz obrázek 3.



Obrázek 3: CodeMatch - okno při průběhu porovnávání [3]

Výstupem programu jsou tabulky s výsledky. Pro každý porovnávaný soubor je jedna tabulka, v které jsou vypsány názvy dokumentů, které jsou s ním nejvíce shodné a

procentuální hodnota podobnosti. V uvedeném příkladu na obrázku 4 se zobrazila jména osmi nejvíce podobných souborů. Tato prahová hodnota pro počet zobrazených položek byla rovněž zvolena již v úvodní obrazovce.

37.java

Score	Compared To File
100	37.java
93	10.java
91	40.java
91	28.java
88	22.java
87	7.java
86	31.java
85	25.java

38.java

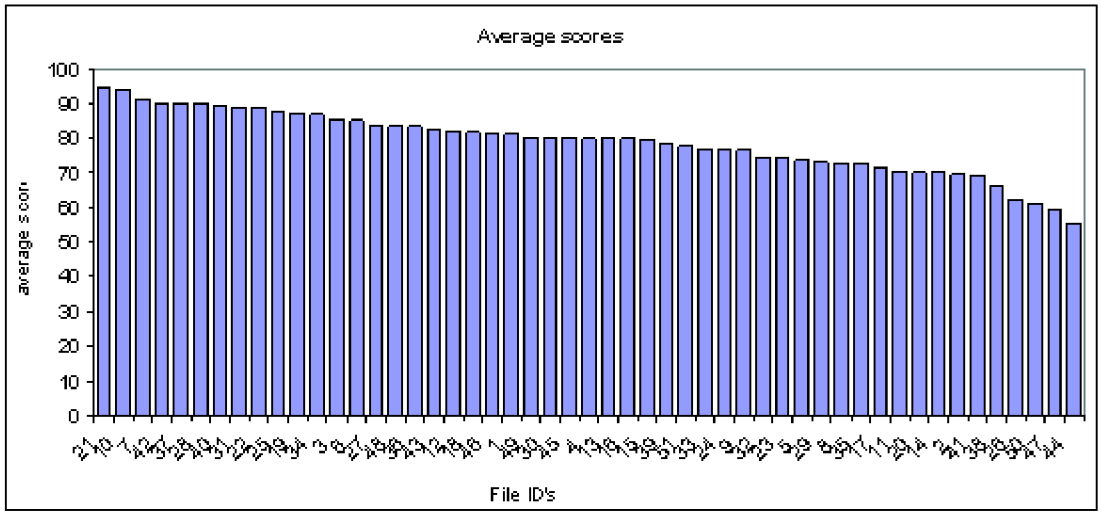
Score	Compared To File
100	38.java
69	23.java
68	29.java
67	35.java
66	32.java
55	20.java
53	41.java
52	17.java

39.java

Score	Compared To File
100	39.java
84	21.java
83	3.java
80	27.java
78	42.java
73	6.java
68	48.java
62	15.java

Obrázek 4: CodeMatch - příklady výsledů porovnávání

Za nevýhodu nástroje můžeme považovat vysoké procentuální hodnoty pro nepodobné soubory. Pro názornost jsou v grafu na obrázku 5 zobrazeny průměrná podobnostní vyhodnocení pro všech 51 dokumentů.



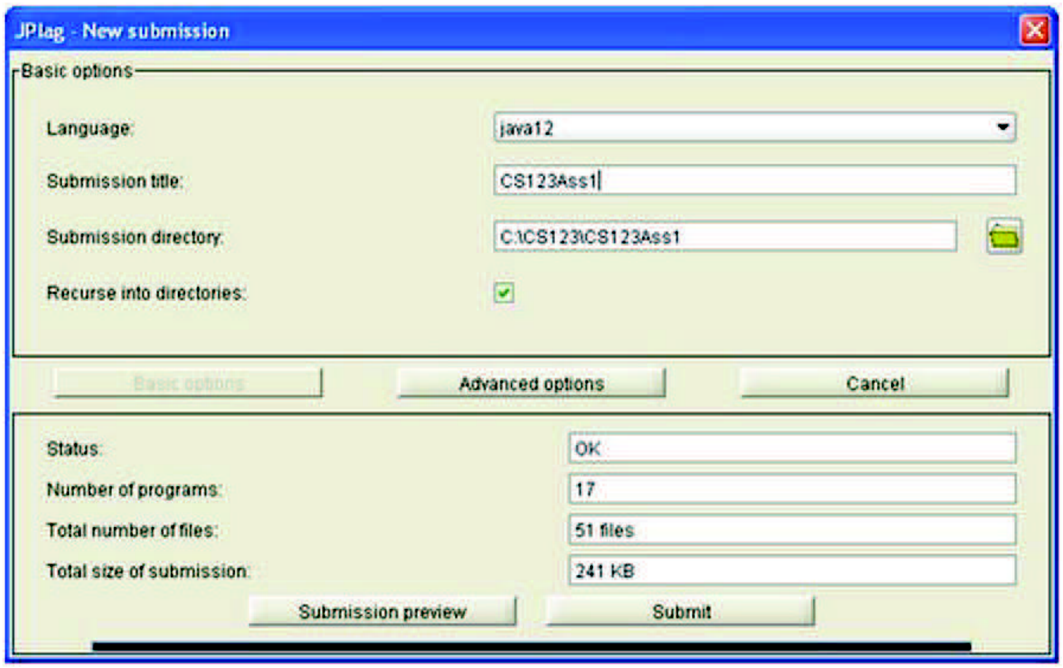
Obrázek 5: Průměrné výsledné procentuální skóre pro každý z 51 souborů [3]

2.7.2 JPlag

Jedním z nejznámějších nástrojů pro detekci plagiátorství ve zdrojových kódech je webová služba JPlag, která v současné době podporuje programovací jazyky Java, C #, C, C ++ a Scheme. Porovnávat je možné i text v přirozeném jazyce. JPlag je zdarma, ale uživatelé jsou povinni si vytvořit účet.

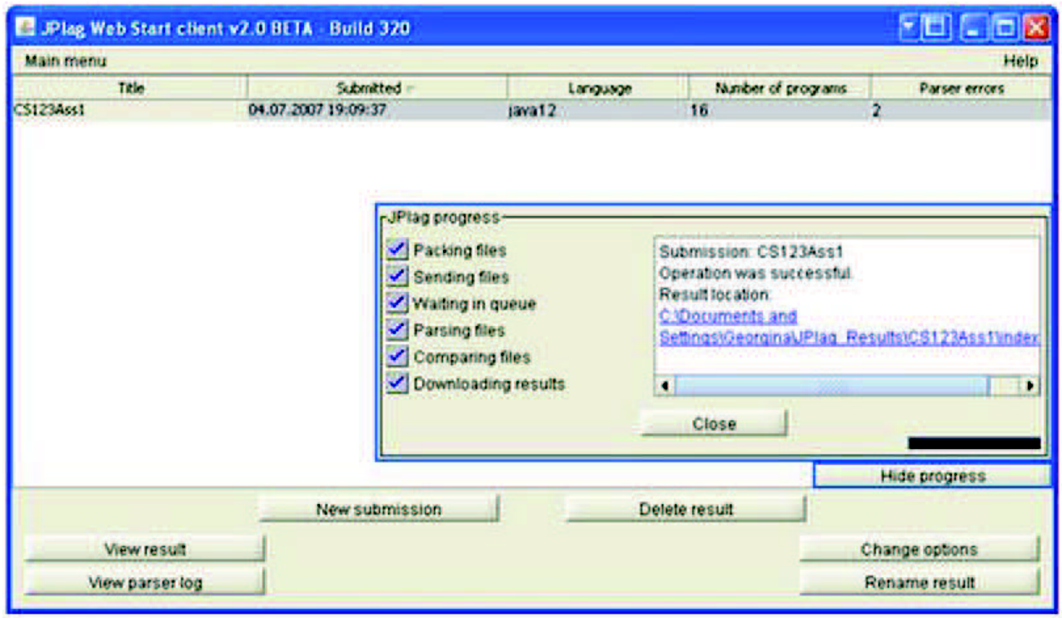
JPlag porovnává soubory se zdrojovým kódem předložené ve složkách nebo jako jednotlivé soubory. Pokud je práce každého studenta uložena v samostatné složce, pak JPlag vrátí skóre podobnosti mezi složkami, které obsahují podezřelé soubory.

Uživatel může začít srovnávací proces výběrem složky, která obsahuje studentskou práci, která bude ve srovnání, viz obrázek 6. Práce každého studenta byla uložena jako samostatná složka v podadresáři CS123Ass1.



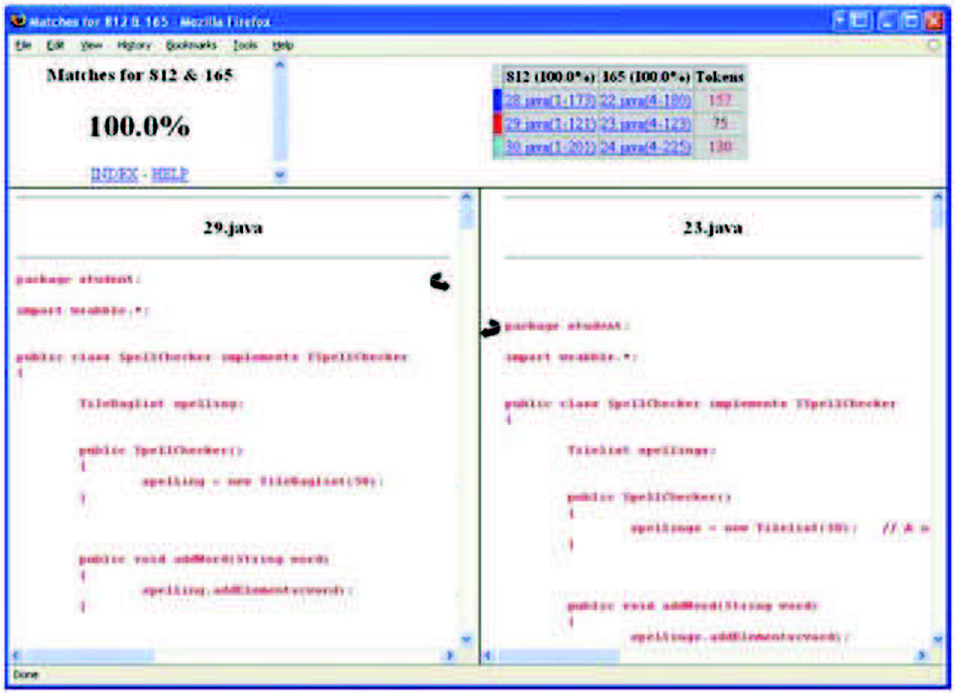
Obrázek 6: Úvodní obrazovka programu JPlag [3]

Po odeslání stiskem tlačítka *Submit*, se započne srovnávání souborů (viz obrázek 7).



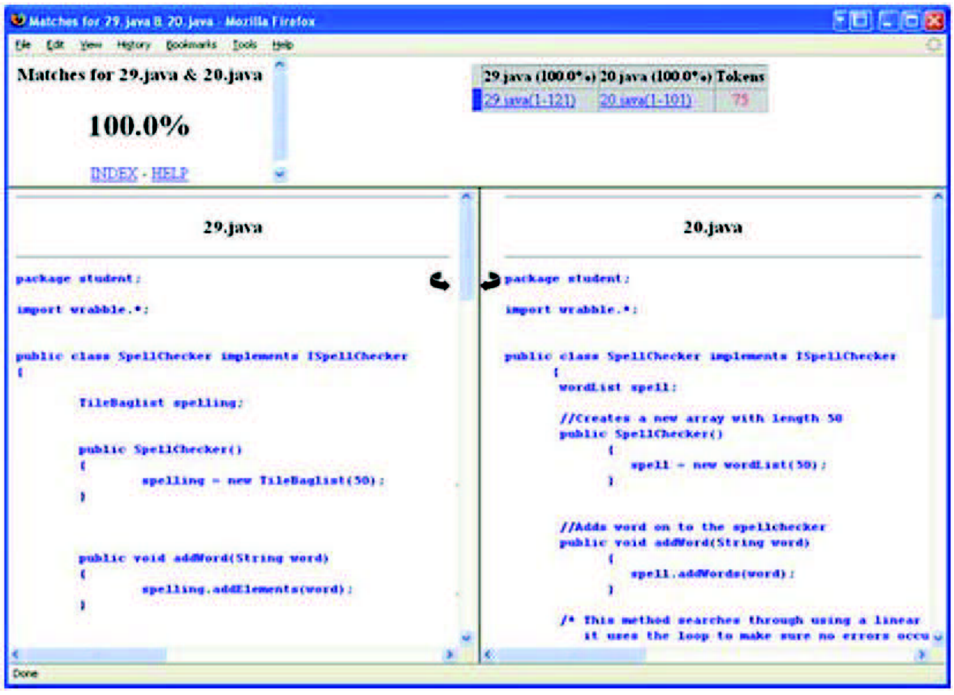
Obrázek 7: JPlag - proces detekce [3]

Výsledky se zobrazují ve formě histogramu. Pro kontrolu podobnosti mezi konkrétními složkami se po kliknutí na název souboru rychle zobrazí výsledky. Na obrázku 8 je příklad podobností pro složky *812* a *165* (čísla představují ID studentů).



Obrázek 8: JPlag - srovnání konkrétních podezřelých dokumentů [3]

Pokud je práce každého studenta uložena jako jeden soubor, pak je výstupem programu JPlag zjištěné skóre podobnosti mezi podezřelými dokumenty. Uživatel si může prohlížet podezřelé dvojice souborů pomocí histogramu a kliknutím na název souboru se otevře nové okno zobrazující jasně zvýrazněné podezřelé fragmenty zdrojového kódu. Na obrázku 9 je příklad výsledku pro soubory 29 a 20.



Obrázek 9: JPlag - srovnání podezřelých souborů [3]

Celkově je JPlag uživatelsky velmi jednoduchý, snadno lze prohlížet a porovnávat podezřelé soubory. Uživatelské rozhraní zobrazování výsledků jasně ukazuje podezřelé fragmenty kódu. Díky tomu se pak může rychle vynést verdikt o tom, zda se jedná či nejedná o plagiát.

3 PRAKTICKÁ ČÁST

V této kapitole je nejdříve rozebrán původní návrh programu a zkušební verze detektoru, která byla řešena v rámci semestrálního projektu a odzkoušena na fiktivní databázi programových kódů. Následně je popsána konečná verze programu, ke které se dospělo v rámci diplomové práce. Je zde popsáno grafické uživatelské rozhraní, kterým je program opatřen. V závěru je uvedeno statické zhodnocení aplikace programu na reálnou databázi studentských prací a diskuze získaných výsledků.

3.1 Návrh programu

Cílem praktické části této diplomové práce je vytvoření programu pro detekci plagiátu jednoduchých programových kódů, které studenti odevzdají v rámci předmětu jako samostatný projekt. Detektor bude vytvořen v prostředí Matlab a prioritně bude určen pro analýzu kódů v matlabovském programovacím jazyce, tedy pro soubory typu M (.m).

Program by měl porovnávat soubory v rámci předložené databáze, případně porovnávat všechny soubory jedné složky (projekty odevzdané v aktuálním roce) se soubory složky druhé (databáze odevzdaných projektů z loňských let).

Výstupem programu je možný soubor dvojic s podezřením na plagiát, které přesáhly uživatelem zvolenou prahovou hodnotu procentuální shody. Případně by každému souboru z první složky bylo přiřazeno například tři či více souborů ze složky druhé, které by se nejvíce s konkrétním kódem shodovaly. Uživatel by pak měl mít možnost nahlédnout na konkrétní výsledky, v čem si byly soubory podobné.

Program by měl sledovat několik atributů a porovnávat je vždy pro určitou vytvořenou dvojici. Mezi atributy by mohly patřit:

- velikost souboru v bytech,
- počet řádků a
- počet slov (příkazů), charakteristických pro určitý programátorský jazyk.

Jednotlivé hodnoty zkoumaných příznaků je nutné následně dosadit do vhodné metriky, která určí výslednou procentuální shodu analyzované dvojice programových kódů.

3.2 Zkušební verze programu

V této podkapitole je rozebrána první verze programu, která byla vytvořena v rámci semestrálního projektu a odzkoušena na malé, uměle vytvořené databázi programových kódů.

3.2.1 Zkušební databáze

Uměle vytvořená databáze studentských prací obsahuje pět krátkých zdrojových kódů s názvy 01.m až 05.m. Soubor 05.m obsahuje převzaté úseky kódu ze souboru 02.m a 04.m je přesným plagiátem souboru 03.m. Ostatní soubory by si neměly být nijak zvlášť podobné.

3.2.2 Popis programu

Program porovnává dva vstupní dokumenty zdrojových kódů typu soubor M (.m) pomocí metody počítání atributů. Po načtení souborů následuje předzpracování v podobě převedení celého obsahu na malá písmena.

Prvním atributem, který program sleduje je velikost souboru v bytech. Dále pro oba dokumenty zjistí jejich počet řádků a následuje hledání počtu konkrétních charakteristických slov. Hledanými slovy jsou *for*, *if*, *while*, *end*, *plot* a *figure*.

Jednotlivé atributy jsou zapsány do matice. Ke každému atributu je přidělena váha V, s jakou vstupuje do výsledné metriky. Použita byla jednoduchá metrika:

$$P = \frac{\alpha}{\beta} \cdot V , \tag{3.2.1}$$

kde α a β jsou hodnoty zkoumaných příznaků obou dokumentů, přičemž platí že $\alpha < \beta$, aby bylo zajištěno, že výsledný poměr bude z množiny:

$$P \in \langle 0;1 \rangle . \tag{3.2.2}$$

Tento poměr je vynásoben vahou V. Součet všech vah je roven jedné, aby bylo zaručeno snadné vyjádření v procentech.

Výstupem programu je procentuální shoda dvou analyzovaných dokumentů a pro detailnější náhled je zobrazena i matice, kde je v prvním sloupci zapsaná váha jednotlivých atributů, ve druhém sloupni výsledná hodnota jednotlivých příznaků prvního souboru a ve třetím sloupci to samé pro druhý soubor. Příklad výstupu je uveden v tabulce 1.

Tabulka 1: Příklad výstupu programu

Příznak	Váha	Soubor 1	Soubor 2
Velikost souboru v bytech	0,2	1007	737
Počet řádků	0,2	22	26
Počet slov plot	0,1	1	1
Počet slov while	0,1	0	1
Počet slov if	0,1	1	1
Počet slov for	0,1	1	1
Počet slov end	0,1	2	3
Počet slov figure	0,1	1	0
Výsledná shoda = 68,2273 %			

3.2.3 Zhodnocení výsledků

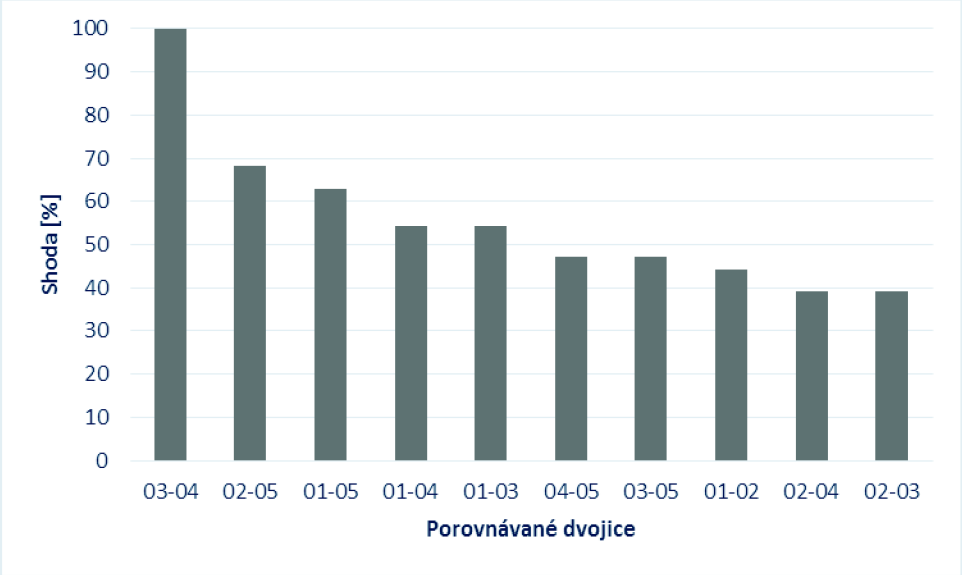
Program byl aplikován na deset spárovaných dvojic souborů. Výsledky procentuální shody pro jednotlivá porovnávání jsou zapsány do tabulky 2.

Tabulka 2: Výsledné procentuální shody porovnávaných souborů

1. soubor	2. soubor	Procentuální shoda
01.m	02.m	44,2639
01.m	03.m	54,3292
01.m	04.m	54,3440
01.m	05.m	62,8970
02.m	03.m	39,2053
02.m	04.m	39,2345
02.m	**05.m**	**68,2273**
03.m	*04.m*	*99,9658*
03.m	05.m	47,1865
04.m	05.m	47,2082

Z hodnot vyplývá, že i odlišné a nijak nesouvisící dokumenty jsou označeny přibližně 40% až 50% shodou. Pro dokumenty se shodnými částmi kódu (v tabulce vyznačeno tučně) vyšla shoda 68 %. Dokumenty shodné (v tabulce vyznačeny

kurzívou) program bezpečně detekoval. Pro názornost jsou v grafu obrázku 10 vyneseny sestupně všechny výsledky.



Obrázek 10: Graf sestupně seřazených výsledků

V následující tabulce 3 jsou rozepsány výsledky porovnávání souborů 02.m a 05.m, 02.m a 04.m a 03.m a 04.m separátně pro jednotlivé sledované příznaky spolu s výslednou procentuální shodou.

Tabulka 3: Výsledky porovnávání separátně pro jednotlivé příznaky

Příznak	02.m - 04.m	02.m - 05.m	03.m - 04.m
Velikost souboru	86,4	73,2	99,8
Počet řádků	44,0	84,6	100
Počet charakteristických slov	30,6	61,1	100
Výsledná shoda	**39,2**	**68,227**	**99,9**

Z tabulky vyplývá, že při použití pouze některého příznaku by procentuální shoda neměla takovou vypovídající hodnotu jako použití všech dohromady. Neboli v konkrétním případu dvojice 02.m - 04.m by se například atribut velikosti souboru v bytech přikláněl k tomu, že se jedná o plagiát, kdežto zkoumání charakteristických slov to téměř vylučuje.

3.3 Finální verze programu

V rámci diplomové práce byl zkušební program rozšířen, opatřen grafickým uživatelským rozhraním (GUI) a otestován na reálné databázi studentských prací.

3.3.1 Databáze

Použitá databáze obsahuje 30 studentských prací v matlabovském programovacím jazyce. Soubory jsou pojmenovány file01.m až file30.m z nichž bylo utvořeno 35 dvojic pro porovnávání. Pět dvojic obsahuje vzájemné plagiáty, 15 dvojic bylo skombinováno tak, aby oba soubory měly stejné zadání a 15 dvojic studentských projektů má různé zadání a kódy by si navzájem neměly být podobné.

3.3.2 Popis programu

Prvním problémem, který bylo nutno vyřešit při použití reálné databáze studentských programátorských prací, bylo samotné nahrávání analyzovaných dokumentů. Nakonec bylo zvoleno nahrání souboru pomocí matlabovské funkce *importdata*, která vytvořila buňkovou strukturu <cell>, kde jedna buňka obsahovala jeden řádek kódů. Při nahrávání větších souborů však docházelo k problému, že se kód nenahrál celý. To bylo ošetřeno nastavením parametrů *delimiterln* a *headerlinesln*, které určují znak pro přerušení nahrávání a maximální počet načtených řádků.

```
delimiterln = '';       % parametr pro přerušení nahrávání (žádný)
headerlinesln = 1000;   % parametr pro maximální počet načtených řádků
```

Následně prošel soubor úpravou, při které byl celý text převeden na malá písmena, byly odstraněny všechny komentáře a veškeré „white space“ jako prázdné řádky a mezery. Příklad úpravy konkrétního kódu je zobrazen na obrázku 11 a 12.

```
for ii=1:n % varianty/možnosti položek
    polozka(ii,:) = zeros(1,n+2);
    polozka(ii,ii) =1;
    polozka(ii,n+1) = cena(ii);
    polozka(ii,n+2) = vaha(ii);
end
reseni = polozka;

optimalnireseni = [];
for kz = 1:batoh %všechny velikosti od minima až po velikost batohu
    %vyberte kombinaci řešení jako nové řešení

    %dvojitá smyčka by mohla fungovat lépe
    for jj = 1:size(reseni,1) %vzit v úvahu všechny možnosti
        for kk = 1:size(reseni,1) %v párech
            if ~sum(reseni(jj,1:end-2).*reseni(kk,1:end-2))
                %ujistit se, ze váha položek nepřesahuje kapacitu batohu
                if (reseni(jj,end) + reseni(kk,end))<=kz
                    %navrnutí optimálního řešení
                    optimalnireseni(end+1,:) = reseni(jj,:) + reseni(kk,:);
                end
            end
        end
    end

    if ~isempty(optimalnireseni) %navrhované řešení
```

Obrázek 11: Ukázka programového kódu bez předzpracování

```
reseni=zeros(1,n+2);
forii=1:n
polozka(ii,:)=zeros(1,n+2);
polozka(ii,ii)=1;
polozka(ii,n+1)=cena(ii);
polozka(ii,n+2)=vaha(ii);
end
reseni=polozka;
optimalnireseni=[];
forkz=1:batoh
forjj=1:size(reseni,1)
forkk=1:size(reseni,1)
if~sum(reseni(jj,1:end-2).*reseni(kk,1:end-2))
if(reseni(jj,end)+reseni(kk,end))<=kz
optimalnireseni(end+1,:)=reseni(jj,:)+reseni(kk,:);
end
end
end
end
if~isempty(optimalnireseni)
temp=find(optimalnireseni(:,end-1)==max(optimalnireseni(:,end-1)));temp=temp(1);
reseni(end+1,:)=optimalnireseni(temp,:);
```

Obrázek 12: Ukázka programového kódu po předzpracování

U takto upraveného dokumentu byl zjištěn počet řádků, který byl prvním zkoumaným příznakem. Druhým zkoumaným příznakem byla velikost souboru v bytech, která se jednoduše zjistila pomocí příkazu *dir*.

Třetí zkoumaný parametr byl počet shodných řádků. Použita byla funkce *ismember*, která každý řádek z prvního zkoumaného dokumentu porovná s každým řádkem dokumentu druhého.

Jako čtvrtý příznak byl zjišťován počet charakteristických slov. Použito bylo 40 slov, která jsou zde přehledně vepsána do následující tabulky 4. Pomocí funkce *regexp* byl zjištěn počet jednotlivých charakteristických slov v prvním i druhém analyzovaném kódu.

Tabulka 4: Seznam porovnávaných charakteristických slov

Hledaná charakteristická slova							
plot	end	length	otherwise	fprintf	linspace	str2num	struct
while	figure	input	zeros	elseif	pause	num2str	mod
if	size	switch	sum	disp	imagesc	eval	abs
for	else	case	isempty	break	str2double	fliplr	strcmp
imshow	return	error	get	sort	ones	numel	uicontrol

Bylo tedy získáno několik číselných hodnot o prvním i druhém analyzovaném dokumentu, z kterých se následně vypočítala výsledná metrika.

3.3.3 Metrika

Pro výpočet podobnosti souborů byla použita párová metrika. Tzn. výstupem programu je míra podobnosti dvou prací.

Pro názornost uvažujme soubory A a B s počtem řádků r a s. Porovnávané příznaky jsou označeny $J(A)$ a $J(B)$ a jejich podobnost P.

Pro příznak velikosti souboru v bytech, počet řádků a příznak výskytu charakteristických slov byla zvolena symetrická metrika. Výsledná hodnota tedy nehodnotí jednotlivé soubory zvlášť, ale dvojici jako celek a platí

$$P(A,B) = P(B,A). \tag{3.3.1}$$

Jelikož pro jednotlivé příznaky vycházejí řádově různé hodnoty, byla použita

jednoduchá metrika, která byla získána poměrem menší získané hodnoty k větší získané hodnotě:

$$P(A,B) = \frac{\min(J(A),J(B))}{\max(J(A),J(B))}. \tag{3.3.2}$$

Výsledná hodnota se tedy bude pohybovat v rozmezí od nuly do jedné:

$$P(A,B) \in (0;1\rangle. \tag{3.3.3}$$

Např. pro příznak velikosti v bytech bude pro stejný počet bytů poměr roven jedné a při stále se zvětšujícím rozdílu velikosti se bude výsledná metrika blížit nule. Pro vyjádření v procentech stačí poměr vynásobit konstantou:

$$P(A,B) = \frac{\min(J(A),J(B))}{\max(J(A),J(B))} \cdot 100. \tag{3.3.4}$$

Pro příznak charakteristických slov se provede poměr nalezeného počtu každého slova zvlášť. Nejprve je však potřeba zjistit, zda pro oba analyzované kódy nejsou hodnoty nulové, aby se zabránilo dělení nulou. Jinak řečeno pokud ani jeden analyzovaný kód neobsahuje určité hledané charakteristické slovo, nebere se ve výsledném výpočtu v potaz. Z takto ošetřených hodnot se vypočítá výsledná procentuální shoda charakteristických slov aritmetickým průměrem.

Nechť P_1 až P_n jsou atributy podobnosti počtu n zkoumaných charakteristických slov, potom výsledný příznak počtu charakteristických slov je vypočítán podle vzorce

$$P_{char.slov} = \frac{\sum_{i=1}^{n} P_i}{n}. \tag{3.3.5}$$

Výpočet podobnosti řádků vychází z nesymetrické metriky. Program zjištěný počet shodných řádků vydělí počtem řádků menšího souboru podle vzorce

$$P(A,B) = \frac{J(A) \cap J(B)}{\min(r,s)}. \tag{3.3.6}$$

Pokud tedy budou všechny řádky jednoho souboru obsaženy v souboru druhém, výsledná metrika bude rovna jedné. Se zmenšujícím se počtem shodných řádků a se zvětšujícím se počtem řádků souboru se bude výsledná metrika blížit nule.

Konečná výsledná metrika kombinací všech jednotlivých příznaků se počítá aritmetickým průměrem, do kterého vstupuje každý příznak s určitou vahou V podle vzorce

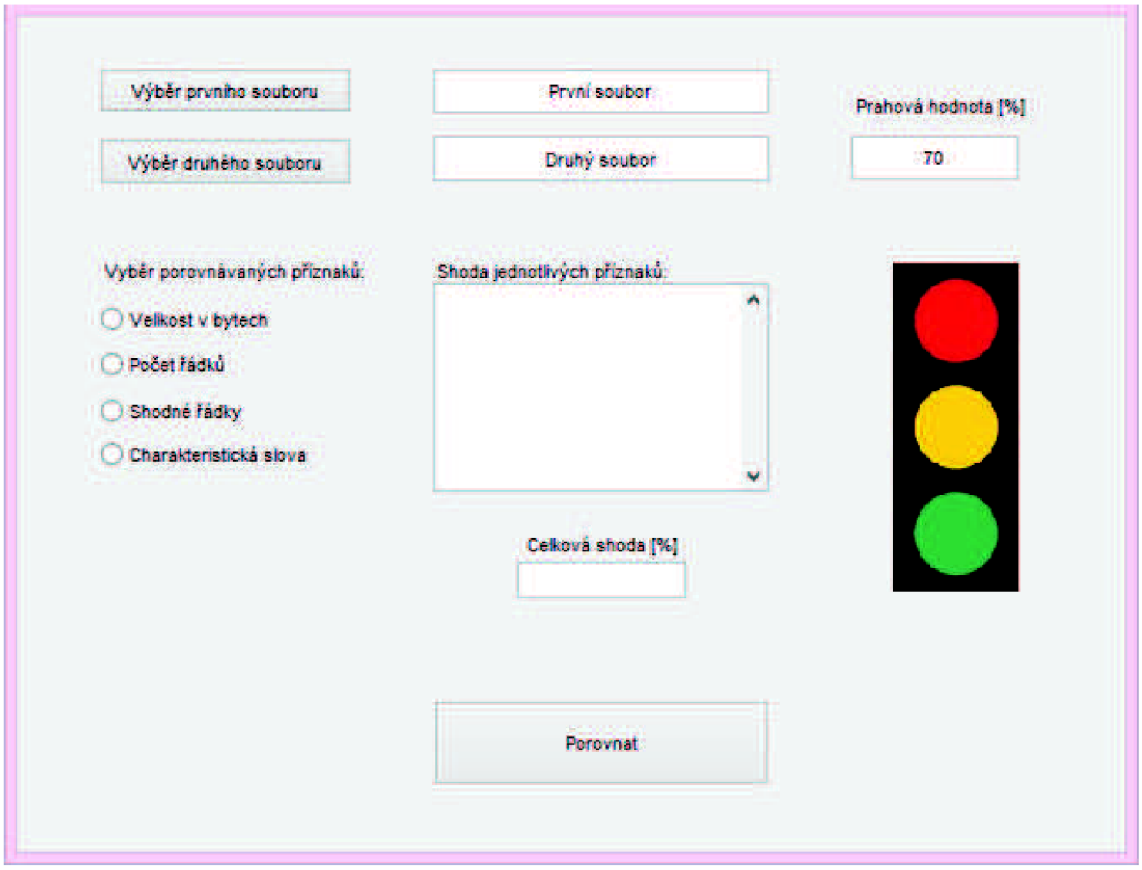
$$P = \frac{P_1 \cdot V_1 + P_2 \cdot V_2 + P_3 \cdot V_3 + P_4 \cdot V_4}{V_1 + V_2 + V_3 + V_4}. \qquad (3.3.7)$$

Tento vzorec je výhodný mimo jiné i pro případ, kdy se nehodnotí všechny příznaky, ale pouze pár vybraných. V tom případě se snadno programově nastaví váha příznaku, který nechceme zahrnout v porovnávání, jako nulová a výsledná podobnost stále vychází v procentech.

3.4 Grafické uživatelské rozhraní

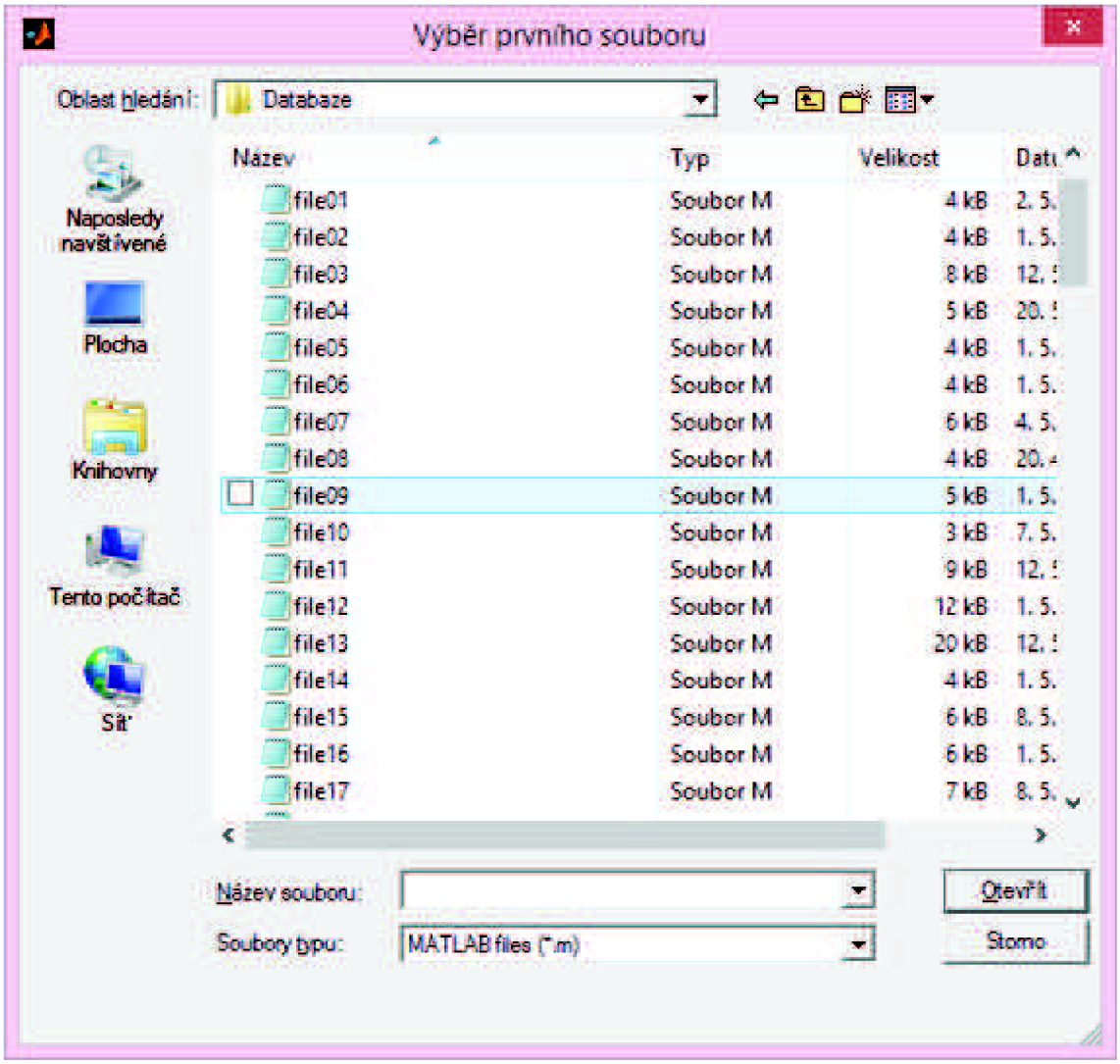
Program byl opatřen jednoduchým a přehledným grafickým rozhraním s několika tlačítky, možností volby konkrétních nastavení a grafickým prvkem semaforu.

Při spuštění programu se otevře úvodní panel grafického uživatelského rozhraní, viz obrázek 13.



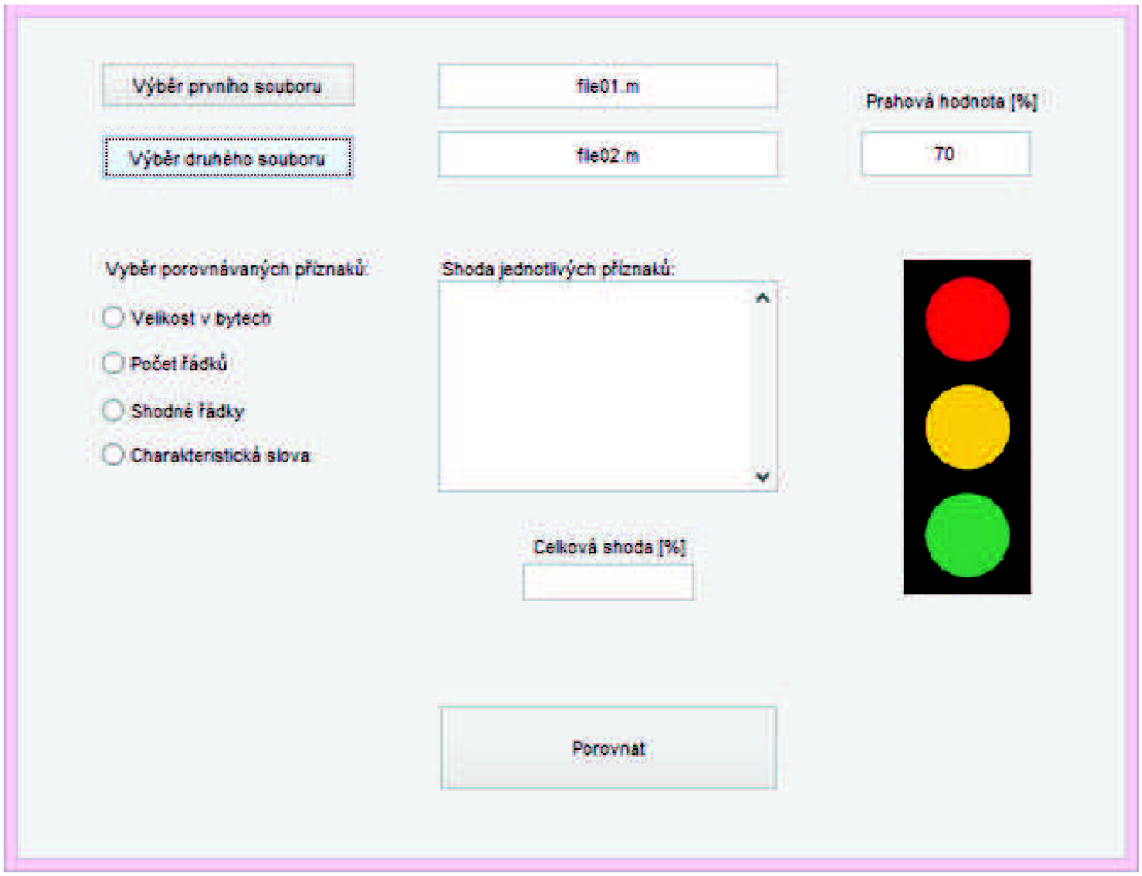
Obrázek 13: GUI - uvítací panel po spuštění programu

Po stisknutí tlačítka *Výběr prvního souboru* vyskočí dialogové okno, viz obrázek 14.



Obrázek 14: GUI - dialogové okno pro výběr souboru

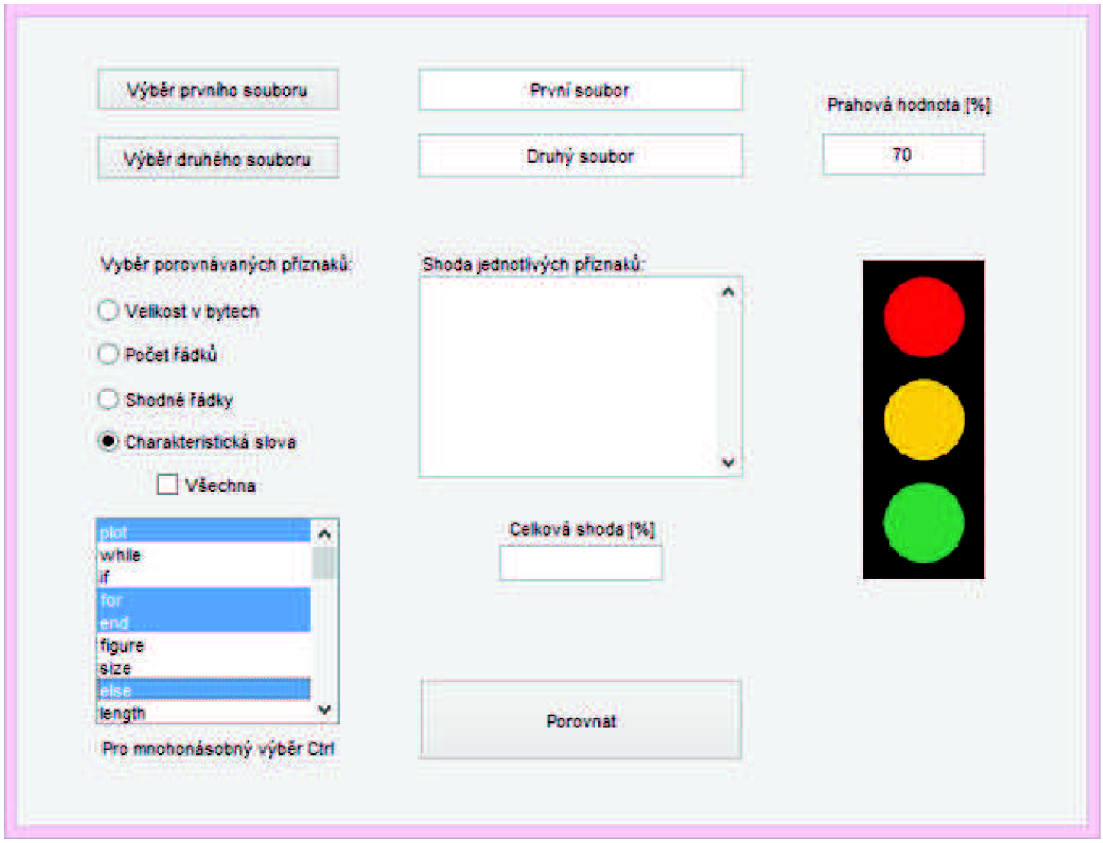
Při dvojkliku na požadovaný soubor či za použití tlačítka *Otevřít* se zapíše jméno vybraného souboru do kolonky *První soubor*. Totéž se provede pro výběr druhého souboru, viz obrázek 15.



Obrázek 15: GUI - zobrazení jmen vybraných souborů pro analýzu

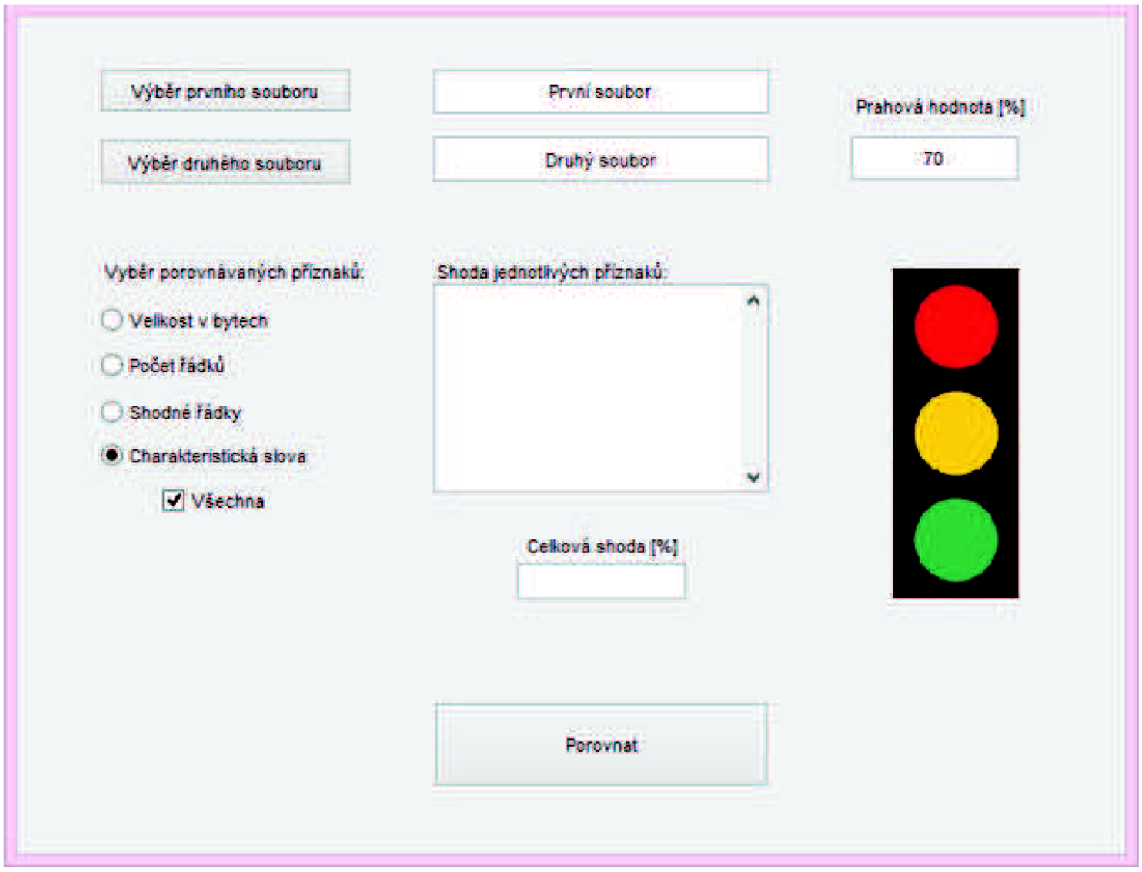
Uživatel si může nastavit prahovou hodnotu, která bude určovat mezní procentuální hranici pro označení analyzované dvojice za plagiáty. Defaultně je prahová hodnota nastavena na 70 %, ale uživatel ji může kdykoli přepsat, i během testování.

V levé části panelu si může uživatel navolit, které příznaky chce v analyzovaných kódech porovnávat. Po zvolení příznaku *Charakteristická slova*, se rozevře nabídka, z které si uživatel může pro analýzu vybrat jedno konkrétní slovo, nebo se stisknutím *Ctrl* několik požadovaných slov, viz obrázek 16.



Obrázek 16: GUI - zobrazení charakteristických slov

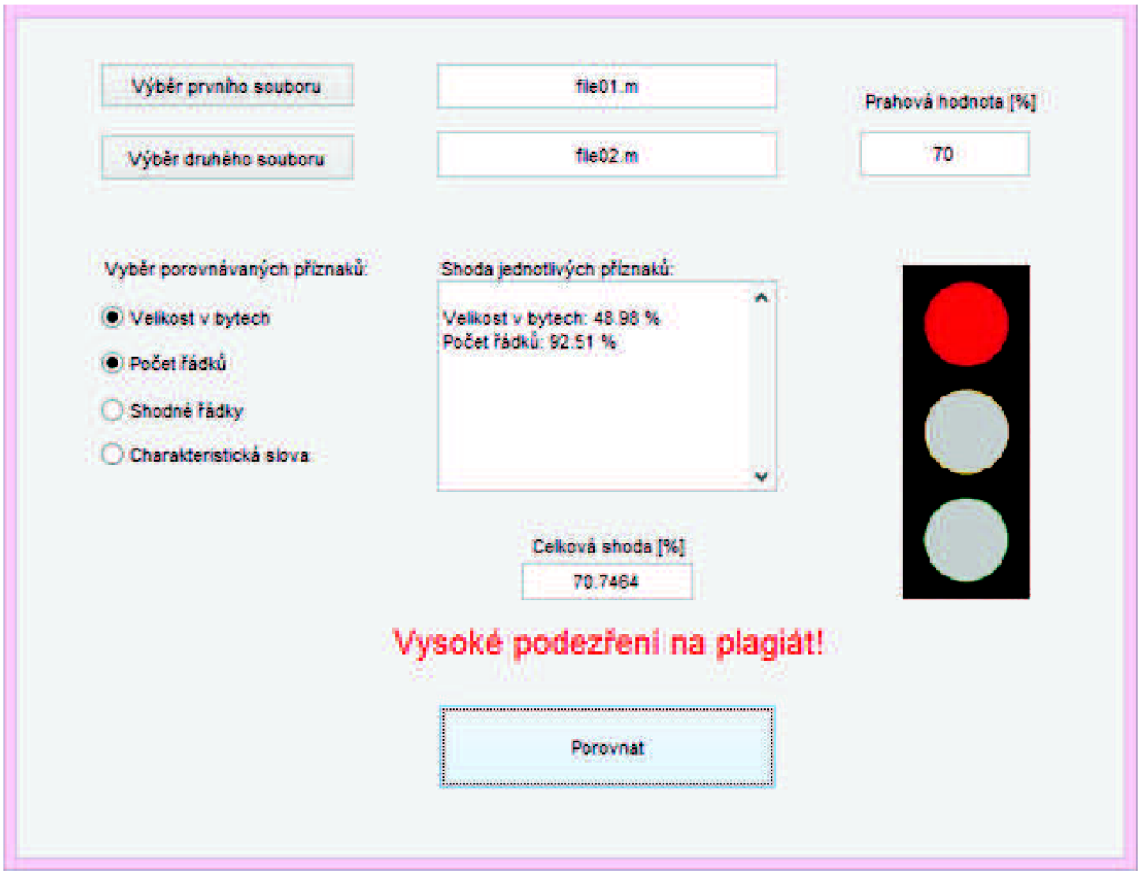
Pokud uživatel nechce zahrnout do porovnávání jen určitá slova, ale všechny, stačí zaškrtnout volbu s titulkem *Všechny*. Pro lepší přehlednost je při této volbě nabídka slov opět skryta, jak vidíme na obrázku 17.



Obrázek 17: GUI - výběr všech charakteristických slov pro analýzu

Po navolení všech nastavení se samotná analýza vybraných souborů spustí tlačítkem *Porovnat*. Do kolonky s titulkem *Shoda jednotlivých příznaků* se vepíší procentuální výsledky podobnosti konkrétních atributů, které uživatel požadoval zahrnout v porovnávání. V samostatné kolonce s titulem *Celková shoda [%]* se pak objeví výsledné procentuální zhodnocení podobnosti dvojice analyzovaných kódů.

Panel grafického uživatelského rozhraní je opatřen i grafickým prvkem v podobě semaforu, který názorně upozorňuje na překročení prahové hodnoty červeným světlem. Zároveň se v červeném písmu zobrazí hláška „Vysoké podezření na plagiát!“, jak je vidět na obrázku 18.

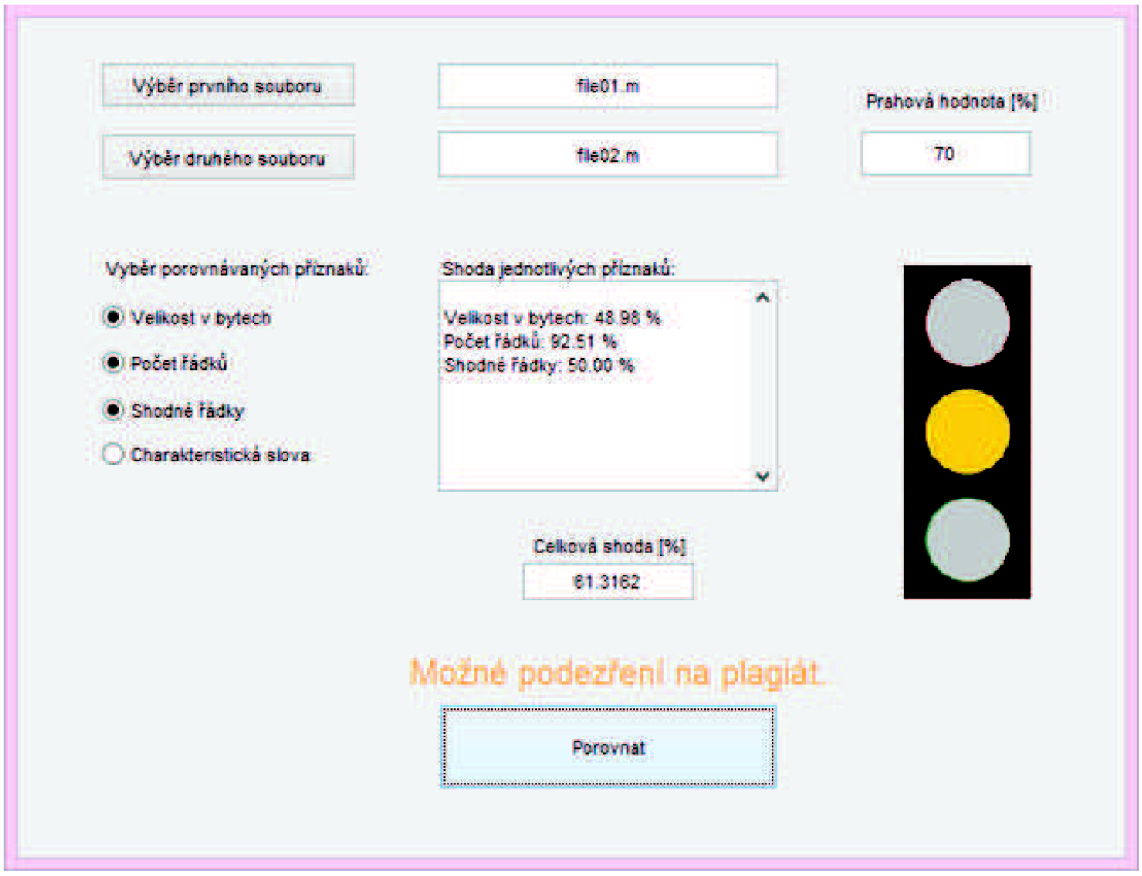


Obrázek 18: GUI - zobrazení informativní hlášky při překročení prahové hodnoty

Při nízké procentuální shodě, neboli při nastavení prahové hodnoty mnohem vyšší, něž je výsledek, se zobrazí zelená barva a zeleným písmem se vypíše hláška „Bez podezření na plagiát“. Příklad tohoto případu zobrazuje obrázek 19.

Obrázek 19: GUI - zobrazení informativní hlášky při nízké procentuální shodě

Třetí možný případ je přiblížení se výsledku k prahové hodnotě. Tehdy zůstane na semaforu svítit oranžová a uživatel dostane upozornění „Možné podezření na plagiát.“ Viz obrázek 20. Rozmezí pro vyhodnocení možného podezření bylo stanoveno na 15 %.



Obrázek 20: GUI - zobrazení informativní hlášky při přiblížení se k prahové hodnotě

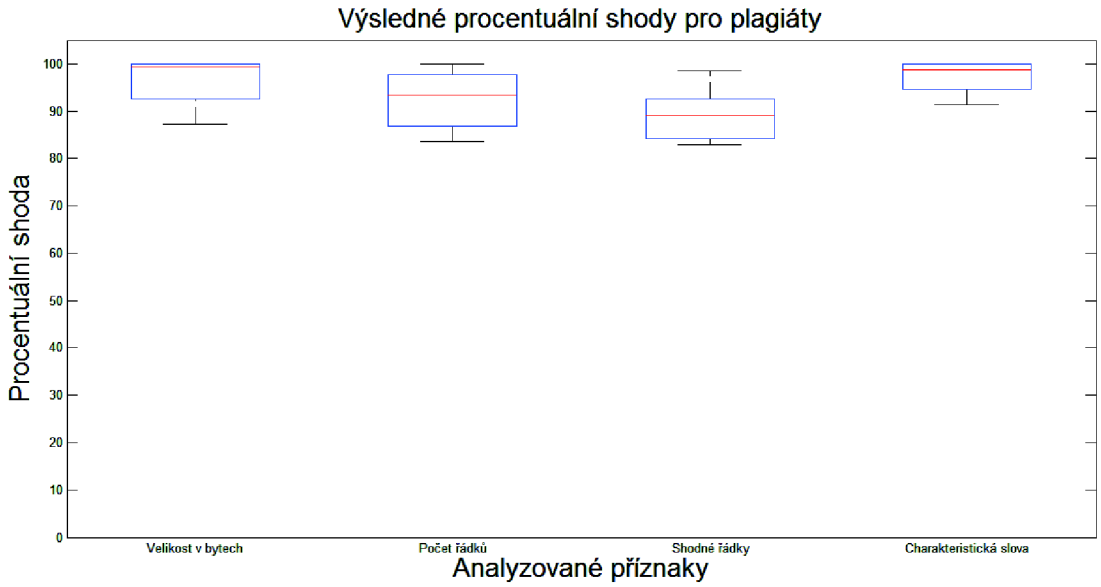
3.5 Testování

Vytvořený program byl otestován na databázi studentských projektů. Nejprve byly zkoumány výsledné hodnoty porovnávání jednotlivých atributů.

Pro přehlednost a názornost byly výsledky zavedeny do grafů zvlášť pro dvojice, o kterých bylo známo, že jsou navzájem plagiáty, zvlášť pro dvojice, které měli navzájem shodná zadání a zvlášť pro dvojice kódů s rozdílným zadáním, které by si neměly být navzájem podobné.

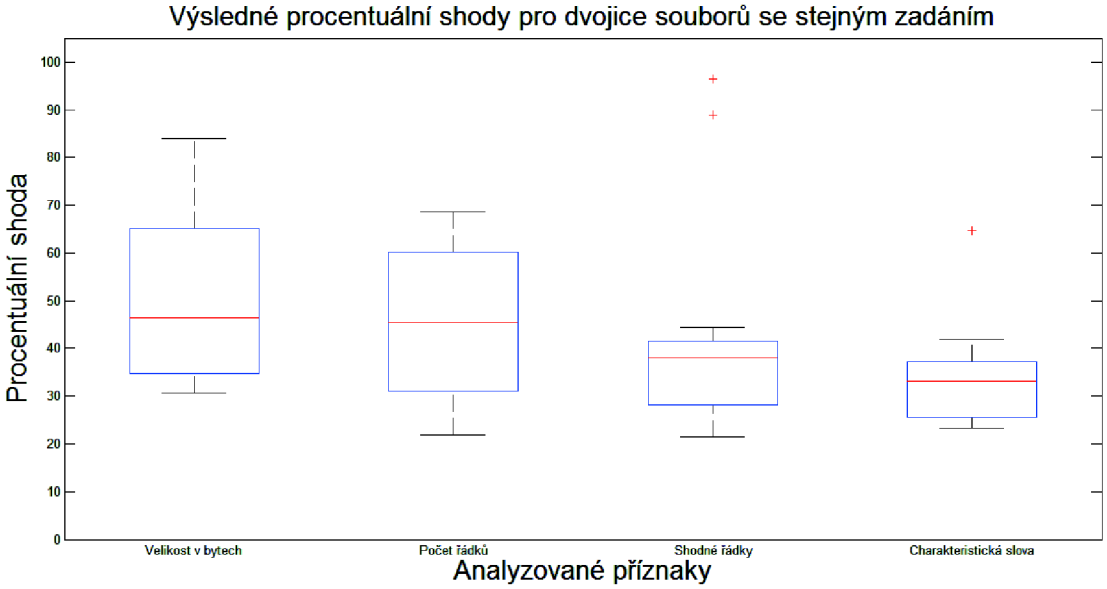
Výsledky byly zaneseny do krabicového grafu pomocí matlabovského příkazu *boxplot*, který pro získané hodnoty zobrazuje pomocí černých značek jejich celkové rozpětí. Modrou barvou je ohraničeno pole výsledků od 25. percentilu po 75. percentil. Červenou čarou je pak v grafu vyznačen medián.

Grafické zobrazení získaných výsledků pro dvojice plagiátů na obrázku 21 ukazuje spolehlivé ohodnocení vysokou procentuální shodou pro všechny zkoumané atributy.



Obrázek 21: Graf procentuálních shod sledovaných příznaků pro soubory označené za plagiáty

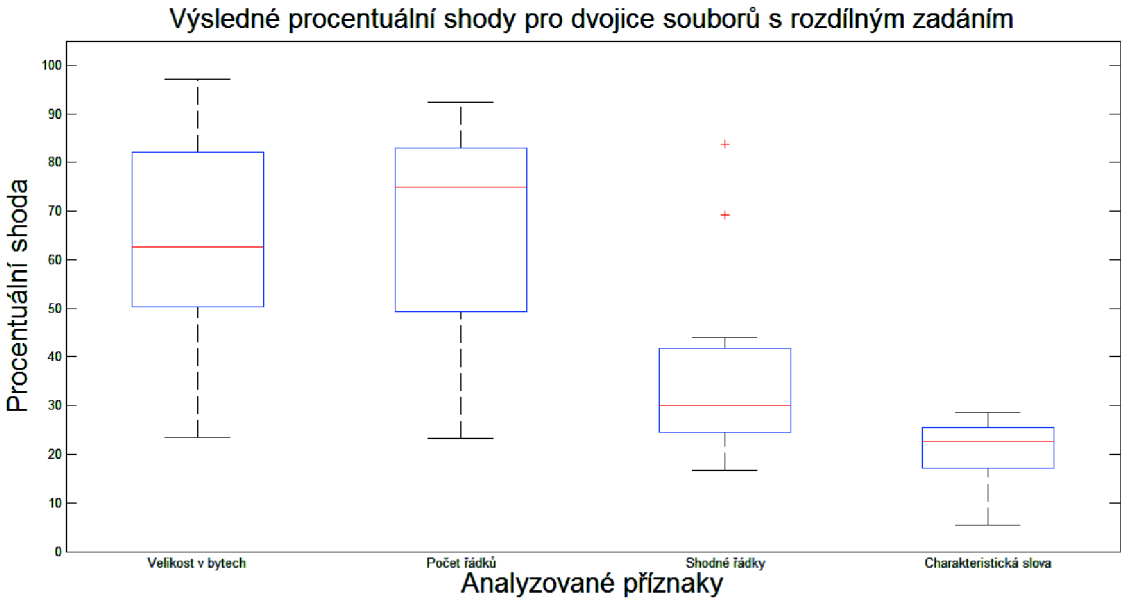
Z grafu na obrázku 22 je zřejmé, že pro soubory, které nejsou plagiáty, ale mají stejné zadání, je atribut velikosti souboru v bytech a atribut počtu řádků ve větším rozpětí a obsahuje vyšší procentuální shody, i když se o plagiáty nejedná. Naopak příznak shodných řádků a příznak počtu charakteristických slov zůstal nízký, tedy je můžeme označit za věrohodnější příznaky.



Obrázek 22: Graf procentuálních shod sledovaných příznaků pro soubory se stejným zadáním

Třetí graf (viz obrázek 23) shrnuje výsledky pro dvojice kódů, které nejsou navzájem podobné. Potvrdil se závěr z předešlého grafu a to, že nejvíce vypovídající je

atribut sledující počet charakteristických slov a naopak pro tuto databázi je nejméně vypovídající příznak velikosti souboru v bytech a počet řádků. To může být odůvodněno tím, že se jedná o studentské projekty, jejichž zadání, ač jsou různá, by si měla být podobná náročností, a tedy i potřebnou délkou kódu.



Obrázek 23: Graf procentuálních shod sledovaných příznaků pro nepodobné soubory

Jak již bylo řečeno, graf zobrazuje modrou barvou rozpětí mezi 25. a 75. percentilem. Tyto hodnoty byly použity i pro stanovení, s jakou vahou budou vstupovat jednotlivé příznaky do výsledné metriky.

Nechť $Q1_{25/100}$ až $Q4_{25/100}$ je 25. percentil a $Q1_{75/100}$ až $Q4_{75/100}$ je 75. percentil čtyř zkoumaných příznaků, potom se váhy $V1$ až $V2$ získají výpočtem podle následujících vzorců. Pro první příznak je uveden příklad výpočtu:

$$V1 = \frac{(Q1_{75/100} - Q1_{25/100})}{(Q1_{75/100} - Q1_{25/100}) + (Q2_{75/100} - Q2_{25/100}) + (Q3_{75/100} - Q3_{25/100}) + (Q4_{75/100} - Q4_{25/100})} \quad (3.5.1)$$

$$V1 = \frac{(82.11 - 50.44)}{(82.11 - 50.44) + (82.86 - 49.35) + (41.68 - 24.41) + (25.57 - 17.02)}$$

$$V1 = 2{,}8728$$

$$V1 \cong 3$$

$$V2 = \frac{(Q2_{75/100} - Q2_{25/100})}{(Q1_{75/100} - Q1_{25/100}) + (Q2_{75/100} - Q2_{25/100}) + (Q3_{75/100} - Q3_{25/100}) + (Q4_{75/100} - Q4_{25/100})} \quad (3.5.2)$$

$$V2 = 2{,}7155$$

$$V2 \cong 3$$

$$V3 = \frac{(Q3_{75/100} - Q3_{25/100})}{(Q1_{75/100} - Q1_{25/100}) + (Q2_{75/100} - Q2_{25/100}) + (Q3_{75/100} - Q3_{25/100}) + (Q4_{75/100} - Q4_{25/100})} \quad (3.5.3)$$

$$V3 = 5{,}2691$$

$$V3 \cong 5$$

$$V4 = \frac{(Q4_{75/100} - Q4_{25/100})}{(Q1_{75/100} - Q1_{25/100}) + (Q2_{75/100} - Q2_{25/100}) + (Q3_{75/100} - Q3_{25/100}) + (Q4_{75/100} - Q4_{25/100})} \quad (3.5.4)$$

$$V4 = 10{,}6523$$

$$V4 \cong 11$$

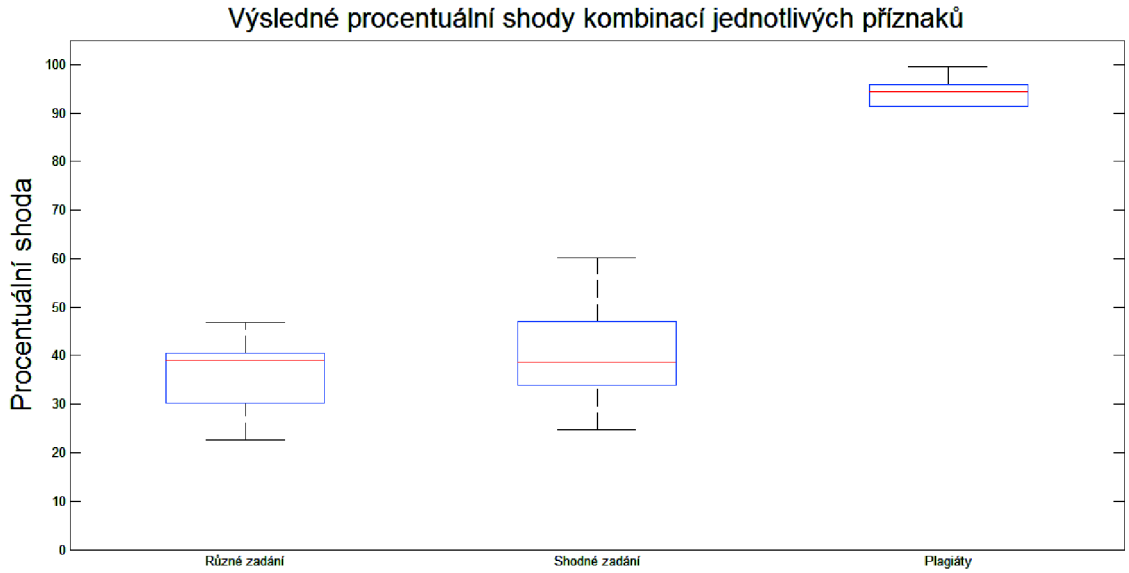
Volně řečeno vzorce přiřazují nejmenší váhu tomu sledovanému atributu, který nabýval hodnoty v největším rozmezí. A naopak atributu, jehož výsledky nabývaly podobných hodnot pro celou databázi, respektive pro charakteristickou část databáze (plagiáty, rozdílné projekty nebo projekty se stejným zadáním), přiřadil vzorec váhu velkou.

Vypočítané váhy jednotlivých hledaných atributů jsou vepsány do přehledné tabulky:

Tabulka 5: Výsledné váhy použité pro výslednou metriku

Příznak	Váha
Velikost v bytech	3
Počet řádků	3
Počet shodných řádků	5
Počet charakteristických slov	11

Tyto získané váhy byly dosazeny do výsledné metriky a program opětovně otestován. Výsledné procentuální shody jsou zaneseny do krabicového grafu separátně pro dvojice plagiátů, dvojice se shodným zadáním a pro dvojice s nepodobnými kódy, viz obrázek 24.



Obrázek 24: Graf výsledných procentuálních shod získaných kombinací všech příznaků

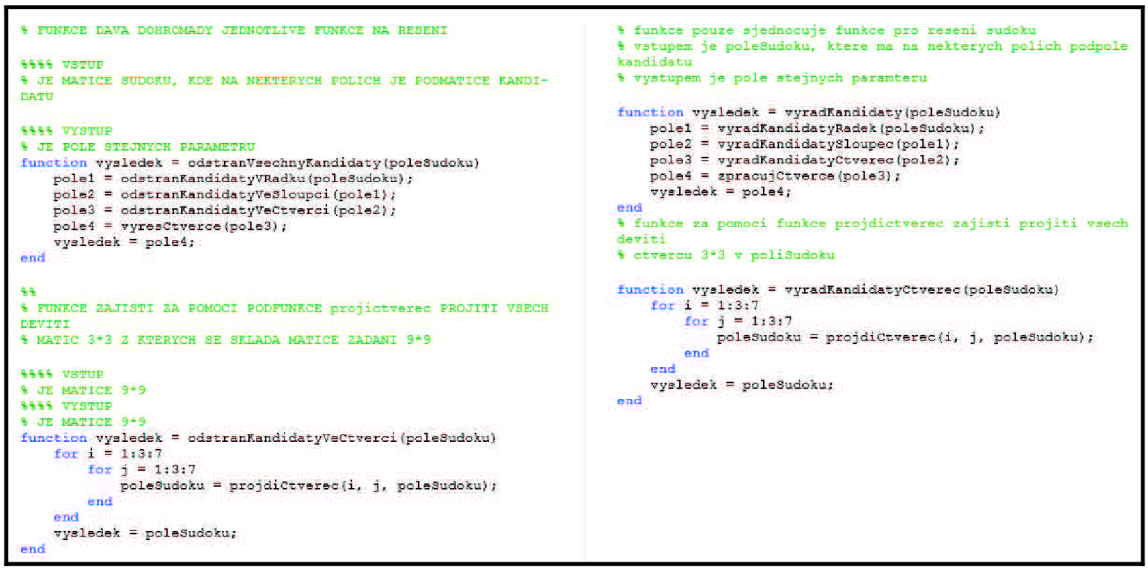
3.6 Diskuze

Původním záměrem bylo, aby program navzájem porovnával dvě předložené složky dokumentů. Od toho se však upustilo, protože byl zvolen způsob nahrávání pomocí příkazu *importdata*. Ten vyžaduje, aby nahrávaný dokument byl ve stejné složce, jako program, ve kterém je tento příkaz volán. Výhodou však je nahrávání dokumentu po řádcích, což je pro práci s programovými kódy obzvlášť výhodné.

Z grafů v předešlé kapitole lze vidět, že program pracuje s vysokou spolehlivostí. Nemálo k ní přispívá předzpracování analyzovaného kódu. Převedení celého dokumentu na malá písmena a odstranění mezer zajistilo vyšší úspěšnost atributu porovnávání řádků. Ten by šel jinak snadno obejít například pouhým připsáním mezer mezi matematické znaky v rovnici nebo opatřením názvů proměnných velkým počátečním písmenem. Odstranění prázdných řádků pak zvýšilo úspěšnost příznaku počtu řádků.

Nejvíce však zřejmě přispělo ke správné detekci odstranění veškerých komentářů. Do atributu počtu řádků nepřispívají řádkové komentáře, mezi charakteristická slova se nepočítají slova obsažená v komentářích a příznak shodných řádků sleduje pouze řádky čistého kódu.

Na obrázku 25 a 26 je příklad konkrétní dvojice kódu, která je na první pohled rozdílná, po předzpracování však jsou téměř totožné a plagiátorství bylo bezpečně odhaleno.



```
% FUNKCE DAVA DOHROMADY JEDNOTLIVE FUNKCE NA RESENI

%%%% VSTUP
% JE MATICE SUDOKU, KDE NA NEKTERYCH POLICH JE PODMATICE KANDI-
DATU

%%%% VYSTUP
% JE POLE STEJNYCH PARAMETRU
function vysledek = odstranVsechnyKandidaty(poleSudoku)
    pole1 = odstranKandidatyVRadku(poleSudoku);
    pole2 = odstranKandidatyVeSloupci(pole1);
    pole3 = odstranKandidatyVeCtverci(pole2);
    pole4 = vyresCtverce(pole3);
    vysledek = pole4;
end

%%
% FUNKCE ZAJISTI ZA POMOCI PODFUNKCE projictverec PROJITI VSECH
DEVITI
% MATIC 3*3 Z KTERYCH SE SKLADA MATICE ZADANI 9*9

%%%% VSTUP
% JE MATICE 9*9
%%%% VYSTUP
% JE MATICE 9*9
function vysledek = odstranKandidatyVeCtverci(poleSudoku)
    for i = 1:3:7
        for j = 1:3:7
            poleSudoku = projdiCtverec(i, j, poleSudoku);
        end
    end
    vysledek = poleSudoku;
end
```

```
% funkce pouze sjednocuje funkce pro reseni sudoku
% vstupem je poleSudoku, ktere ma na nekterych polich podpole
kandidatu
% vystupem je pole stejnych paramteru

function vysledek = vyradKandidaty(poleSudoku)
    pole1 = vyradKandidatyRadek(poleSudoku);
    pole2 = vyradKandidatySloupec(pole1);
    pole3 = vyradKandidatyCtverec(pole2);
    pole4 = zpracujCtverce(pole3);
    vysledek = pole4;
end
% funkce za pomoci funkce projdictverec zajisti projiti vsech
deviti
% ctvercu 3*3 v poliSudoku

function vysledek = vyradKandidatyCtverec(poleSudoku)
    for i = 1:3:7
        for j = 1:3:7
            poleSudoku = projdiCtverec(i, j, poleSudoku);
        end
    end
    vysledek = poleSudoku;
end
```

Obrázek 25: Příklad dvojice porovnávaných kódů

functionvysledek=vyradkandidaty(polesudoku)	functionvysledek=odstranvsechnykandidaty(polesudoku)
pole1=vyradkandidatyradek(polesudoku);	pole1=odstrankandidatyvradku(polesudoku);
pole2=vyradkandidatysloupec(pole1);	pole2=odstrankandidatyvesloupci(pole1);
pole3=vyradkandidatyctverec(pole2);	pole3=odstrankandidatyvectverci(pole2);
pole4=zpracujctverce(pole3);	pole4=vyresctverce(pole3);
vysledek=pole4;	vysledek=pole4;
end	end
functionvysledek=vyradkandidatyctverec(polesudoku)	functionvysledek=odstrankandidatyvectverci(polesudoku)
fori=1:3:7	fori=1:3:7
forj=1:3:7	forj=1:3:7
polesudoku=projdictverec(i,j,polesudoku);	polesudoku=projdictverec(i,j,polesudoku);
end	end
end	end
vysledek=polesudoku;	vysledek=polesudoku;
end	end

Obrázek 26: Příklad dvojice porovnávaných kódů po úpravě

Rozdílnost výsledků pro porovnávání neupravených souborů nebo čistě kódů je patrná z tabulky 6. Uvedeny jsou příklady výsledků procentuálních shod pro dvě dvojice souborů, které měly stejná zadání a pro jednu dvojici s rozdílným zadáním.

Tabulka 6: Příklad rozdílnosti výsledků porovnávání kódů za použití předzpracování a bez něj

		file15 - file25	file05 - file17	file29 - file23
Velikost v bytech	S úpravou	31,34	30,88	78,08
	Bez úpravy	31,34	30,88	78,08
Počet řádků	S úpravou	42,45	30,82	82,68
	Bez úpravy	35,23	24,12	58,38
Shodné řádky	S úpravou	35,56	44,44	30,48
	Bez úpravy	64,52	98,41	24,35
Charakteristická slova	S úpravou	35,56	23,36	26,19
	Bez úpravy	64,52	21,60	25,66
Výsledná shoda	S úpravou	35,91	30,19	41,94
	Bez úpravy	40,42	40,67	36,96

Hodnoty jasně ukazují, že předzpracování může ovlivnit výsledek až o deset procent a to oběma směry. Detektor se tím spíše může přiklonit k podezření z plagiátorství, ale i naopak se může vypočítaná shoda souborů snížit. Pochopitelně pouze příznak podobnosti velikosti souboru v bytech předzpracování neovlivní.

Nutno však dodat, že soubory v poskytnuté databázi, o kterých bylo známo, že jsou plagiáty, se řadí k těm jasným případům, které vznikly zkopírováním celého kódu a následným poupravením.

V poskytnuté databázi byla i dvojice, o které bylo známo, že první soubor je částečný plagiát druhého, neboli oba dokumenty obsahovaly stejný úsek kódu, který byl ale v poměru s velikostí celého kódu poměrně malý. Vytvořený detektor dvojici ohodnotil výslednou shodou 56,27 %, což je pro tuto databázi běžná hodnota pro projekty se stejným zadáním.

Náhled kódů zobrazuje obrázek 27. Pro přehlednost je zobrazen kód s již odstraněnými komentáři.

```
function zprava=Trans_sifra(y,klic)

n=length(y);
radky=n/klic;
A=ceil(radky);
delka=1;
matice=ones(A,klic);

for i=1:1:A ;
    for p=1:klic;
        if delka>n;
            matice(i,p)=y;
        else
            matice(i,p)=y(delka);
            delka=delka+1;
        end
    end
end

ozi=matice;
dop=ozi;
delka=1;
C=klic*A;
sifr=ones(1,C);


for i=1:1:klic;
    for p=1:A;
        sifr(delka)=dop(i,p);
        delka=delka+1;
    end
end
zprava=char(sifr);
disp(Zašifrovaná zpráva)
```

```
function text=trans(x,klic,smer)
if smer == 1;
n=length(x);
radky = n/klic;
B = ceil(radky);
delka=1;
matice=ones(B,klic);

for i=1:1:B
    for r=1:klic
        if delka>n;
            matice(i,r)=x;
        else
            matice(i,r)=x(delka);
            delka=delka+1;
        end
    end
end
ot=matice;
ascii = ot;
delka=1;
D=klic*B;
sifr=ones(1,D);


for i=1:1:klic;
    for r=1:B;
        sifr(delka)=ascii(i,r);
        delka=delka+1;
    end
end
text=char(sifr);

disp(Váš zašifrovaný)
```

Obrázek 27: Ukázka částí kódů, které jsou plagiáty

Je vidět, že jsou použity stejné cykly a příkazy, pouze s přepsáním proměnných. Pokud by se porovnávaly pouze tyto zobrazené části, byly by odhaleny jako jasný plagiát. Avšak první projekt je složen z 247 řádků a druhý má 133 řádků programového kódu, takže je tato část zanedbatelná a plagiátorství nebylo odhaleno.

Pro další konkrétní dvojici ze skupiny se stejným zadáním detektor přiřadil procentuální shodu 60,17 % a na panelu GUI se objevila hláška „Možné podezření na plagiát.“ O této dvojici nebylo předem známo, že by se jednalo o plagiát. Po náhledu do obou prací však lze považovat upozornění za správné.

Porovnávané kódy jsou poskytnuty k náhledu v příloze. Pro lepší přehlednost jsou opět uvedeny již jako ošetřené od komentářů.

Za jednu z předností vytvořeného detektoru lze považovat rychlost prováděné analýzy, a to především proto, že program vyhodnocuje podobnost na základě

kombinace několika jednoduchých příznaků, které nejsou náročné na výpočet. Výsledkem této práce je rychlý a zároveň spolehlivý detektor plagiátů programových kódů.

ZÁVĚR

V práci bylo rozebráno téma plagiátorství z hlediska procesu detekce a rozdělení detekčních nástrojů. Předně se práce zaměřila na detekci plagiátů programových kódů, v rámci čehož je řešena i spolehlivost detekce a prevence plagiátorství. Přehledně uvádí do problematiky z technického hlediska detekčních nástrojů a zároveň byl čtenář seznámen s obsluhou dostupných nástrojů JPlag a CodeMatch.

V prostředí Matlab byl realizován vlastní program pro detekci plagiátů programových kódů. V rámci semestrálního projektu byl vytvořen program, který pracuje na základě počítání atributů, a posléze byl aplikován na uměle vytvořenou databázi krátkých programových kódů. Nástroj jednoznačně detekoval dva téměř stejné dokumenty a pro dva dokumenty, které obsahovaly stejné části kódu, vyhodnotil na první pohled větší procentuální shodu, než pro ostatní zdrojové kódy, které plagiátem nebyly.

V rámci diplomové práce byl původní program rozšířen a aplikován na reálnou databázi studentských programátorských projektů. O dokumentech v databázi bylo známo, které dvojice kódů jsou plagiáty a které měly shodná zadání, ale o plagiátorství se nejednalo. Porovnávanými atributy byly: velikost souboru v bytech, počet řádků, počet charakteristických slov a počet shodných řádků.

Pro plagiátorské dvojice, pro dvojice se shodným zadáním a dvojice s různým zadáním byly separátně otestovány jednotlivé hledané příznaky. Výsledné hodnoty vyšly velice dobře. Pro skupinu plagiátů se výsledky všech atributů pohybovali nad 80% hranicí. Pro další dvě skupiny se pohybovaly hodnoty atributů velikosti v bytech a počet řádků v rozsahu od 20 % až do téměř 100 %. Z toho bylo odvozeno, že tyto dva příznaky budou do výsledné metriky podobnosti vstupovat s menší vahou. A naopak atribut charakteristických slov, se pro všechny tři testované skupiny pohyboval v nejmenším rozsahu, a tak mu byla přiřazena největší váha.

S takto naváženými příznaky se program znovu aplikoval na jednotlivé skupiny dvojic a výsledky byly zaneseny do grafu. Z něho je jasně patrné, že pro plagiátorské dvojice jsou výsledné procentuální shody mnohem vyšší, než pro zbytek databáze.

Navíc byla odhalena i jedna dvojice kódů, o které předem nebylo známo, že se jedná o plagiát, ale detektor na tuto dvojici upozornil přiřazením vyšší procentuální shody.

Program byl opatřen přehledným uživatelským rozhraním, pomocí kterého si může

uživatel zvolit, které příznaky chce zahrnout do porovnávání. Může porovnávat jeden či více příznaků. U příznaku charakteristických slov si navíc může zvolit buď všechny, nebo jen konkrétní slova z nabídky. O výsledku informují nejen číselné hodnoty celkové procentuální shody a jednotlivých příznaků, ale zároveň i grafický prvek semaforu. Ten se po překročení stanoveného prahu rozsvítí červeně, při přiblížení se k prahové hodnotě se rozsvítí oranžově, jinak zeleně. Procentuální hranici pro označení dvojice za plagiát si může uživatel měnit kdykoli během testování. Detektor tedy bezpečně odhalí plagiát a může být považován za úspěšně zpracovaný.

Všechny body diplomové práce byly úspěšně splněny. Byl vytvořen funkční detektor, který spolehlivě vyhodnocuje výsledky jak na uměle vytvořené databázi krátkých zdrojových kódů, tak na reálné databázi studentských projektů. Dále může být použit při výuce předmětů o programování a zvýšit tak kvalitu výuky.

LITERATURA

[1] ALZAHRANI, Salha. M. *Understanding Plagiarism Linguistic Patterns, Textual Features, and Detection Methods* [online]. 2012 [cit. 2014-5-5]. Dostupné z: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?tp=&arnumber=5766764&queryText%3Dplagiarism+detection

[2] BLAŠKA, Jan, KRUMPHOLC, Michal, SEDLÁČEK, Miloš. Využití grafického uživatelského rozhraní Matlabu ve výzkumu a výuce měření [online]. [cit. 2015-5-14]. Dostupné z: http://dsp.vscht.cz/konference_matlab/matlab03/blaska.pdf

[3] COSMA, Georgina, JOY, Mike, WHITE, Daniel, YAU, Jane. *Plagiarism Prevention and Detection* [online]. 2007 [cit. 2014-12-20]. Dostupné z: http://www.ics.heacademy.ac.uk/resources/assessment/plagiarism/detect_plagiarism.html

[4] ČAPEK, Miloslav. Uživatelské aplikace - návrh GUI části programu v Matlabu [online]. 2010 [cit. 2015-5-14]. Dostupné z: http://old.elmag.org/lib/exe/fetch.php/wiki:user:capek:maa_pr12_gui.pdf

[5] ČERNOHLÁVKOVÁ, Kateřina. *Plagiátorství na vysokých školách*. Brno: Masarykova univerzita, Filozofická fakulta, Kabinet knihovnictví, 2004. 108 s. Vedoucí diplomové práce Mgr. Petra Šedinová. Dostupné z: http://is.muni.cz/th/109574/ff_m/diplomova_prace.pdf

[6] FLÉGL, Jan. *Sofistikované metody pro kontrolu elektronických textů*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2009. 40 s. Vedoucí bakalářské práce Ing. Václav Pfeifer. Dostupné z: https://www.vutbr.cz/www_base/zav_prace_soubor_verejne.php?file_id=18302

[7] Grafické uživatelské rozhraní v prostředí Matlab (Přednáška předmětu Algoritmizace a programování) [online]. 2015 [cit. 2015-5-14]. Dostupné z: https://www.vutbr.cz/elearning/file.php/148619/prednasky_2015/09_APRG_graficke_uzivatelske_rozhrani.pdf

[8] HAUZÍREK, Michal. *Možnosti automatické detekce plagiátů* [online]. 2007 Vedoucí diplomové práce Ing. Luboš Pavlíček. Vysoká škola ekonomická v Praze. Fakulta informatiky a statistiky. Katedra informačních technologií. Dostupné z: http://www.trochu.kvalitne.cz/diplomka/DP_xhaum07_v4_se_zadanim.pdf

[9] CHÝLA, Roman. *Detekce plagiátorství* [online]. 2009 [cit. 2014-12-20]. Dostupné z: http://ikaros.cz/detekce-plagiatorstvi

[10] Infogram.cz. *Definice plagiátu* [online]. 2014 [cit. 2014-12-20]. Dostupné z: http://www.infogram.cz/findInSection.do?sectionId=1115&categoryId=1168

[11] Knihovna.cvut.cz. *Co vše je plagiátorství* [online]. 2010 [cit. 2014-12-20]. Dostupné z: http://knihovna.cvut.cz/studium/jak-psat-vskp/doporuceni/plagiatorstvi/co-je-plagiatorstvi.html

[12] MASKERI, G. *Version history based source code plagiarism detection in proprietary systems* [online]. 2012 [cit. 2014-4-20]. Dostupné z: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?tp=&arnumber=6405334&queryText%3Dplagiarism+detection+programing+code

[13] MathWorks. http://www.mathworks.com/

[14] Modelování a identifikace [online]. 20014 [cit. 2014-11-15]. Dostupné z: https://sites.google.com/site/modelovaniaidentifikace/Home

[15] Opisování.cz. *Kontrola plagiátorství* [online]. 20014 [cit. 2014-12-20]. Dostupné z: http://www.opisovani.cz/kontrola-plagiatorstvi/

[16] Physics.csbsju.edu *Box Plot* [online]. 2007 [cit. 2015-5-14]. Dostupné z: http://www.physics.csbsju.edu/stats/box2.html

[17] SI, A., H.V. LEONG a R.W.H. LAU. CHECK: *A Document Plagiarism Detection System. In Proceedings of ACM Symposium for Applied Computing*. 1997

[18] Matematika.cz. *Percentil* [online]. 2014 [cit. 2015-5-16]. Dostupné z: http://www.matematika.cz/rozptyl

[19] Wikipedia.org. *Plagiát* [online]. 2014 [cit. 2014-12-20]. Dostupné z: http://cs.wikipedia.org/wiki/Plagi%C3%A1t

[20] ZOUHAR, Petr. *Systémy pro kontrolu elektronických textů*. Brno: Vysoké učení technické v Brně. Fakulta elektrotechniky a komunikačních technologií. Ústav telekomunikací, 2008. 49 s. Bakalářská práce. Vedoucí práce Ing. Václav Pfeifer. Dostupné z: https://www.vutbr.cz/www_base/zav_prace_soubor_verejne.php?file_id=9264

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

J(X)	množina základních jednotek dokumentu X
res(A,B)	podobnost (resemplance) souborů A a B (symetrická metrika)
con(A,B)	obsah (containment) souborů A a B (nesymetrická metrika)
P(A,B)	podobnost souborů A a B
r	počet řádků souboru A
s	počet řádků souboru B
n	počet hledaných charakteristických slov
V	váha atributu
$Q1_{25/100}$	25. percentil výsledků testovaných atributů
$Q1_{75/100}$	75. percentil výsledků testovaných atributů
GUI	grafické uživatelské rozhraní (Graphical User Interface)

SEZNAM PŘÍLOH

Alfons.m	hlavní funkce, v rámci které se zjišťuje atribut velikosti souboru v bytech a jsou volány funkce *odtrankoment.m*, *charslova.m* a *metrika.m*
odtrankoment.m	funkce pro předzpracování kódu a zjištění atributu počtu řádků
charslova.m	funkce pro vyhodnocení atributu počtu charakteristických slov
metrika.m	funkce pro výpočet výsledné metriky kombinací jednotlivých atributů
untitled.m	funkce pro spuštění GUI, která si volá hlavní funkci *Alfons.m*
untitled.fig	uživatelský panel
semafor.JPG	grafický prvek využívaný v rámci GUI
semaforGreen.JPG	grafický prvek využívaný v rámci GUI
semaforOrange.JPG	grafický prvek využívaný v rámci GUI
semaforRed.JPG	grafický prvek využívaný v rámci GUI

UKÁZKA POROVNÁVANÝCH KÓDŮ

Dvojice kódů se shodným zadáním s výslednou procentuální shodou 60 %

```
function [ nejvetsi_mozny_zisk pouzite_predmety ] = batoh( vaha, cena, nosnost )
[m,n] = size(vaha);
for i=1:n
    v=vaha(:,i);
   if v<0
    error(Váha předmětu nemůže nabývat záporných hodnot.)
   end
end
if nosnost<0
    error(Kapacita batohu nemůže nabývat záporných hodnot.);
end
[p,q] = size(cena);
for j=1:q
    z=cena(:,j);
   if z<0
    error(Cena předmětu nemůže nabývat záporných hodnot.)
   end
end
if numel(vaha) ~= numel(cena)
       error(Velikost vektoru vaha musí být stejná jako velikost vektoru cena.);
end
if numel(nosnost) > 1
       error(Je nutné zadat jen jednu maximální kapacitu batohu.);
end
       [jakost] = pocitej_jakost(vaha, cena);
       [ D ] = reseni_problemu( vaha, cena, nosnost, jakost );
nejvetsi_mozny_zisk=D(1,end)

pouzite_predmety=D(1,1:end-1)
end

function [ jakost ] = pocitej_jakost( vaha, cena )
for k = 1:length(vaha);
```

```
function [najlepsie optimalne] = ruksak(hmotnost, cena, nosnost)
[x,y]= size(hmotnost);
for i=1:y
    h=hmotnost(:,i);
    if h<0;
        error (Zadal si zápornú hmotnosť.);
    end
end
if nosnost<0;
    error (Nosnosť ruksaku nemôže byť záporná);
end
[c,d]=size(cena);
for j=1:d
    p=cena(:,i);
    if p<0;
        error (Cena nemôže byť záporná)
    end
end
if numel(hmotnost)~=numel(cena);
    error(Vektor hmotnosti musí mať rovnaký rozmer ako vektor ceny. );
end
if numel(nosnost)>1
    error(Nosnosť je len jedno číslo.);
end
A=zeros(length(hmotnost)+1,nosnost+1);
for i=1:length(hmotnost)
    for j=1:nosnost
        if hmotnost(i)>j
            A(i+1,j+1)=A(i,j+1);
        else
            A(i+1,j+1)=max(A(i,j+1), cena(i)+A(i,j-hmotnost(i)+1));
        end
    end
```

```
    jakost (k) = cena(k)/vaha(k);
end
end
function [ D ] = reseni_problemu( vaha, cena, nosnost, jakost )
 A=zeros(3, length(vaha));
 A(1,:)=jakost(:,:);
 A(2,:)=vaha(:,:);
 A(3,:)=cena(:,:);
 [iA,iA]= sort(A(1,:),descend);
 B=A(:,iA);
 C=zeros(length(vaha): length(vaha)+1);
for e=1:length(vaha);
    soucet=0;
    for f=e:length(vaha);
       if soucet+B(2,f) <= nosnost;
        soucet=soucet+B(2,f);
        C(e,f)=1;
       else
        soucet=soucet;
        C(e,f)=0;
       end
    end
    C(e,end)=soucet;
 end

    for d=1:length(vaha);
        zisk=0;
        for g=1:length(vaha);
          if C(d,g)==0;
          zisk=zisk;
          else C(d,g)==1;
           zisk=zisk + B(3,g);
          end
        C(d,end)=zisk;
        end
    end
 [iC,iC]= sort(C(:,end),descend);
 D=C(iC,:);
end
```

```
end
najlepsie=A(end, end);
optimalne = zeros(length(hmotnost),1);
q = najlepsie;
 j = length(hmotnost);
 while q > 0
      while A(j+1,nosnost+1) == q
      j = j - 1;
      end
      j = j + 1;
      optimalne(j) = 1;
      nosnost = nosnost - hmotnost(j);
      j = j - 1;
      q= A(j+1,nosnost+1);
 end
 optimalne = reshape(optimalne,x,y)
end
```