VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

BAKALÁŘSKÁ PRÁCE

Brno, 2020 Barbora Bláhová

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV BIOMEDICÍNSKÉHO INŽENÝRSTVÍ
DEPARTMENT OF BIOMEDICAL ENGINEERING

DETEKCE PODOBNOSTI ZDROJOVÝCH KÓDŮ
DETECTION OF SOURCE CODE PLAGIARISM

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **Barbora Bláhová**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. Jakub Kašpar**
SUPERVISOR

BRNO 2020

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní obor **Biomedicínská technika a bioinformatika**

Ústav biomedicínského inženýrství

Studentka: Barbora Bláhová ***ID:*** 191532

Ročník: 3 ***Akademický rok:*** 2019/20

NÁZEV TÉMATU:

Detekce podobnosti zdrojových kódů

POKYNY PRO VYPRACOVÁNÍ:

1) Nastudujte problematiku plagiátorství z pohledu podobnosti zdrojových kódů. Vypracujte literární rešerší v této oblasti. 2) Vytvořte vlastní databázi vhodnou pro testování detekce podobnosti v programových kódech. 3) Zvolte vhodné příznaky pro detekci podobnosti a v prostředí Python otestujte jejich vhodnost na vytvořené databázi. Dosažené výsledky diskutujte. 4) Vytvořte detektor podobností založený na kombinování zvolených příznaků. 5) Navržený detektor otestujte. Dosažené výsledky diskutujte a statisticky vyhodnoťte.

DOPORUČENÁ LITERATURA:

[1] 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.

[2] CHÝLA, R. Detekce plagiátorství. Ikaros [online]. 2009, roč. 13, č. 2. Dostupné z: http://ikaros.cz/node/5253.

Termín zadání: 3.2.2020 ***Termín odevzdání:*** 5.6.2020

Vedoucí práce: Ing. Jakub Kašpar

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

ABSTRAKT

Cílem práce je seznámit se s problematikou plagiátorství zdrojových kódů a navrhnout postupy, které budou využity při tvorbě vlastního detektoru plagiátů. Teoretická část uvádí definice a nejčastější způsoby plagiátorství, jak obecně, tak ve zdrojových kódech. Dále představuje již existující typy detektorů a uvádí principy, na kterých lze detekci provádět. Praktická část se zabývá realizací detektoru a jeho testováním.

KLÍČOVÁ SLOVA

Detekce, plagiát, plagiátorství, příznaky, zdrojový kód

ABSTRACT

The purpose of this thesis is to introduce the matters of plagiarism of source codes and to suggest methods, that will be used to create the original plagiarism detector. Theoretical part of the thesis states the definitions and the most common methods of plagiarism, both in general and in source codes. Furthermore it presents already existing types of the detectors and states principals, according to which the detection can be performed. Practical part deals with implementation of detecton and it´s testing.

KEYWORDS

Detection, plagiarism, flags, source code

BLÁHOVÁ, Barbora. *Detekce podobnosti zdrojových kódů*. Brno, 2020, 42 s. Bakalářská práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav biomedicínského inženýrství. Vedoucí práce: Ing. Jakub Kašpar

PROHLÁŠENÍ

Prohlašuji, že svou bakalářskou práci na téma „Detekce podobnosti zdrojových kódů“ jsem vypracovala samostatně pod vedením vedoucího bakalářské 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 autorka uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské 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 autorského 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.

Brno ....

podpis autorky

PODĚKOVÁNÍ

Ráda bych poděkovala vedoucímu bakalářské práce panu Ing. Jakubu Kašparovi, Ph.D. za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.
Děkuji také kolegovi Janu Vykydalovi, za cenné rady zkušeného programátora.

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Plagiátorství je závažným prohřeškem proti akademické etice. V posledních letech také často rezonuje společností, zvláště pokud se prokáže, že autorem plagiátu byla některá z veřejně známých osobností.

Mohlo by se zdát, že případů plagiátorství přibývá. Otázkou však zůstává, zda se zvyšuje četnost plagiátorství, nebo četnost jeho odhalení. Dnešní doba totiž, kromě velkého množství dostupných informací, nabízí také množství technologií, díky kterým se plagiáty snáze odhalují.

Tato práce ze zabývá softwarovým řešením, které má ušetřit práci a čas akademickým pracovníkům a lektorům při odhalování plagiátů. Cílem práce bylo nastudovat problematiku plagiátorství z pohledu zdrojových kódů a navrhnout vhodný software pro hledání podobností ve zdrojových kódech tvořených pro rozhraní Matlab.

První část práce obsahuje rešerši, zabývá se jak plagiátorstvím obecně z pohledu legislativy, tak plagiátorstvím zdrojových kódů a metodami jeho detekce. Druhá část práce pak popisuje praktickou realizaci detektoru podobností, od předzpracování, přes detekce příznaků až po volbu srovnávací metriky a vyhodnocování.

1 Plagiátorství

Úvodem je důležité plagiátorství definovat. V českých zákonech není tento termín přímo uveden. Je tedy nutné použít definice z jiných dostupných zdrojů.

Databáze Národní knihovny České Republiky plagiátorství definuje jako *„vydávání cizího literárního nebo jiného uměleckého nebo vědeckého díla za vlastní, popřípadě převzetí části cizí práce, bez uvedení použitých zdrojů“* [2].

Norma ČSN ISO 5127-2003 uvádí jako definici plagiátorství *„představení duševního díla jiného autora půjčeného nebo napodobeného v celku nebo z části, jako svého vlastního“* [14]

1.1 Legislativa

V České republice se problematickou plagiátorství zabývá zákon sbírky č. 121/2000, tedy Autorský zákon, který upravuje právo autora k jeho dílu. Tento zákon upravuje autorská práva a povinnosti pro mnoho způsobů vyjádření díla. Dle § 2 odstavce 2: *„Za dílo se považuje též počítačový program“* jej můžeme aplikovat také na zdrojové kódy [9]. Dle tohoto zákona má autor právo libovolně se svým dílem nakládat. Toto právo trvá po dobu autorova života a 70 let po jeho smrti. V případě spolupráce autorů pak trvá 70 let po smrti nejdéle žijícího autora.

V Autorském zákonu se v § 11 dále uvádí, že autorských práv se autor nemůže vzdát, tato práva jsou nepřevoditelná. V § 61 je uvedeno, že je-li dílo vytvořeno na objednávku, autor poskytuje objednateli licenci za účelem, který je sjednán ve smlouvě [9]. Jedná se ovšem pouze o licenci, zadavatel si tedy nemůže nárokovat autorství a práci publikovat svým jménem.

Dle § 31 do práva autorského nezasahuje ten, kdo v odůvodněné míře použije dílo nebo část díla jiného autora, vždy je však nutno uvést, je-li to možné, jméno autora, název díla a pramen [9].

1.2 Druhy plagiátorství

Plagiátorství je možné rozdělit podle různých kritérií. Například u rozdělení dle motivace je rozlišováno vědomé a nevědomé plagiátorství [13].

Jak vyplývá z názvu, v případě vědomého plagiátorství se autor tohoto přestupku dopouští zcela vědomě, ve snaze usnadnit si práci. Nevědomé plagiátorství je pak takové, které si autor sám neuvědomuje a může mít různé důvody, například autorovy malé znalosti a zkušenosti s uváděním zdrojů. Častým důvodem je také špatné určení všeobecně známých faktů, které není nutno citovat. V tomto případě

je obecným doporučením citovat v každém případě, kdy si nejsme zcela jisti, zda jde o známý fakt.

Dále je možné plagiátorství rozdělit dle způsobu provedení. Oxfordská univerzita uvádí tyto:

(a) Doslovný opis bez uvedení citace
V případě použití přesného znění úryvku z cizího díla, je nutné tento úryvek označit uvozovkami a opatřit správnou citací dle citační normy. Pokud citace nesplňuje všechny náležitosti, můžeme tento úryvek označit jako plagiát [10].

(b) Kopírování z internetu bez uvedení citace
Také v případě užití úryvků nebo myšlenek z internetu je potřeba řádně uvést původní zdroj. Při čerpání z webu by mělo být také samozřejmostí informace zvážit a ověřit jejich pravdivost, například zjištěním důvěryhodnosti autora nebo ověřením z jiných zdrojů [10].

(c) Parafráze
Parafráze je přepis vlastními slovy. I v tomto případě je ale nutno správně uvést zdroj parafrázované myšlenky. Plagiátem se tedy stává úryvek, který je nesprávně odcitován, nebo není odcitován vůbec. Další chybou v případě parafráze může být nepřesná interpretace původní myšlenky, vytržení z kontextu, nebo jinak zavádějící vyznění původního textu [10].

(d) Zatajená spolupráce
Obzvláště v případě akademických děl se předpokládá, že autor vypracuje práci sám a samostatně. Závěrečné práce pak navíc obsahují čestná prohlášení o samostatné práci. I přesto se však vyskytují případy, kdy se na práci podílí další osoby. Problémem tohoto typu plagiátorství je velmi malá šance na odhalení, protože není využito dohledatelné dílo, které bylo plagiováno [10]. Odhalitelná je tedy pouze v případě, že údajný autor není schopen prokázat dostatečně schopnosti k vytvoření daného díla.

(e) Komerční plagiátorství
Jako komerční plagiátorství jsou označeny případy, kdy je dílo vypracováno jinou osobou než uvedeným autorem (dále objednavatelem) [7]. Osoba (dále poskytovatel) dílo vypracuje dle zadání objednavatele a vzdá se v objednavatelův prospěch svých autorských práv. Toto jednání je však v rozporu s autorským zákonem. Poskytovatelé proto dané služby neinzerují jako díla na zakázku, ale jako podklady pro díla. Na českém trhu funguje několik poskytovatelů podobných služeb. Odměnou pro poskytovatele je finanční obnos, lišící se dle typu díla. Nejlevnější jsou zpravidla referáty a eseje, s odbornou náročností práce pak ceny stoupají. Například web napisemezavas.cz nabízí bakalářskou práci za 299 Kč na jednu normostranu, diplomovou práci za 339 Kč na normostranu a dokonce habilitační práci za 449 Kč na normostranu. Stejně jako u zatajené

spolupráce, i tato forma plagiátorství je téměř neodhalitelná, poskytovatel díla objednavatele záměrně chrání a odevzdané dílo je originál. V tomto případě tedy plagiátorství neodhalí detektory. Jediný způsob odhalení je při konzultacích s vedoucím práce, případně při její obhajobě, a to pouze pokud obhajující objednavatel neprokáže znalosti nutné k vypracování díla dané kvality.

(f) Autoplagiátorství

I pokud autor užije svá starší díla pro tvorbu nového, je povinen se na starší dílo odkázat. Není možné odevzdat jednu práci vícekrát [10].

2 Plagiátorství zdrojových kódů

Typy plagiátorství uvedené v předchozí kapitole, obzvláště poslední tři body, se dají aplikovat také na plagiátorství zdrojových kódů. Protože ale jde o specifický typ díla, mohou také nastat případy podobnosti mezi dvěma kódy, které za plagiátorství nepovažujeme.

Bob Zeidman, odborník na odhalování plagiátů ve zdrojových kódech, uvádí 6 případů podobností, které mohou nastat.

(a) Zdrojový kód třetí strany
Pokud porovnáváme kódy, ve kterých byl použit shodný open source kód, najdeme podobnost. Přesto však tato podobnost není plagiátorstvím [12].

(b) Automaticky generované kódy
Při porovnávání kódů, při jejichž tvorbě byly využity stejné nástroje k automatickému generování kódu, může dojít k podobnosti [12]. Typickým příkladem je generátor kódu pro GUI (grafické uživatelské rozhraní) v prostředí MATLAB. Při porovnávání více kódů s použitím GUI, je velká pravděpodobnost, že dojde k false-positive detekci. Pokud tedy víme, že budeme detektor používat také k detekci kódů s GUI, je vhodné to ve fázi předzpracování zohlednit a tyto automaticky generované části kódu odstranit.

(c) Běžně užívané názvy proměnných
Řada běžných názvů je vyučována ve školách i užívaná v praxi. Typických příkladem jsou názvy „temp“ nebo „result“. I v programech, které spolu nemají žádnou souvislost, mohou být detekovány jako podobnost [12]. Z tohoto důvodu je vhodné použít takový detektor, který nebere názvy proměnných v potaz.

(d) Běžně užívané algoritmy
Algoritmus je posloupnost úkonů, jimiž lze vyřešit nějaký problém. Pro některé úlohy jsou známé konkrétní a ověřené algoritmy a pro programátora by bylo zatěžující a zbytečné, vymýšlet již vymyšlené. Proto se i v nesouvisejících programech může vyskytnout stejný algoritmus, na jehož základě může být detekována podobnost [12]. Příkladem může být Erastothenovo síto, které slouží k vyhledání všech prvočísel, která jsou menší než zadaná horní mez.

(e) Stejný autor
Každý programátor má osobitý styl psaní kódů, což může být důvodem pro detekovanou podobnost u dvou rozdílných kódů, psaných stejným člověkem. Při správně nastavených detekčních parametrech by však k falešně pozitivní detekci dojít nemělo [12].

(f) Zkopírovaný kód
Pokud jsou zkopírovány části, nebo celý kód od jiného autora, každý detekční

program by tyto pasáže měl odhalit. O plagiátorství pak rozhoduje rozsah zkopírovaných částí, důvodnost kopírování a také, zda má programátor svolení od autora původního kódu [12].

2.1 Metody detekce

2.1.1 Metoda počítání atributů

Metodu počítání atributů využívají například systémy Accuse či Faidhi-Robinsonův. Tento způsob detekce podobností je založen na porovnávání několika parametrů kódu. Systém Accuse detekuje sedm parametrů. Faidhi-Robinsonův systém detekuje dvacet tři parametrů, od snadno měnitelných (počet řádků s komentáři) až po složité (celkový počet modulů). Autoři Faidhi-Robinsonova systému uvádějí vyšší specificitu systému, než systém Accuse [11].

Systém Accuse komentáře a prázdná místa kódu zanedbává, detekuje tyto parametry:

- počet unikátních operátorů a operandů
- celkový počet operátorů a operandů
- počet řádků
- počet deklarovaných a užitých proměnných
- celkový počet příkazů

Nevýhodou metody počítání atributů je fakt, že zanedbává strukturu kódu, což může vést k častějším false-positive detekcím.

2.1.2 Metoda porovnávání struktury

Porovnávání řetězců

Porovnávání řetězců je nejjednodušší metodou porovnávání. Předzpracováním je odstaněno formátování a následně jsou porovnávány podřetězce textu. Při dosažení prahu podobnosti jsou určeny za podobné. Tato metoda je však spíše nevhodná, protože ji lze velice snadno obejít změnou názvů proměnných nebo manipulací s počtem řádků [4].

Porovnávání tokenů

Při předzpracování jsou nejčastěji odstraněny jak formátování, tak komentáře. Celý kód je následně tokenizován – příkazy, nebo bloky příkazů jsou zaměněny za unikátní zástupné znaky (tokeny). Toto předzpracování snadno odhalí zamaskovaný plagiát. Při detekci jsou pak porovnávány posloupnosti tokenů [4].

2.2 Existující detektory podobností

MOSS - Measure Of Software Similarity

Automatický systém pro určení podobnosti programů, vyvinutý Stanfordskou univerzitou v roce 1994. Původně sloužil k ulehčení práce učitelům univerzity, později byl částečně zveřejněn lektorům programovacích kurzů. Nyní je pro nekomerční užití volně přístupný online. Před použitím je však nutno založení uživatelského účtu.
Kolem nástroje Moss existuje široká komunita lidí, přispívající k jeho aktualizacím. V současnosti je program schopen detekce pro 25 programovacích jazyků, kontrétně jsou to tyto:
C, C++, Java, C#, Python, Visual Basic, Javascript, FORTRAN, ML, Haskell, Lisp, Scheme, Pascal, Modula2, Ada, Perl, TCL, Matlab, VHDL, Verilog, Spice, MIPS assembly, a8086 assembly, a8086 assembly, HCL2 [1].

Codequiry

Nástroj Codequiry je komerční detektor podobností, který vznikl jako reakce na slabé stránky nástroje Moss. Nabízí srovnání zkoumaného kódu také s kódy volně přístupnými na internetu.
Nalezené podobnosti jsou jím zvýrazněny. Rozbor podobností je velmi detailní, nevýhodou oproti nástroji Moss je ovšem cena. Měsíc používání Codequiry stojí 29 amerických dolarů.
V současnosti detekuje přes 20 programovacích jazyků, na webové stránce jsou uvedeny:
C, C++, Python, C#, Go, PHP, Perl, SQL, Assembly, Ruby, XML, VB, Lisp, Javascript, Typescript, HTML, Haskell, Pascal, Matlab, prostý text a další [3].

Jplag

JPlag je primárně určen lektorům a učitelům ke vzájemnému porovnávání studentských prací. Je však schopen zpracovat také rozsáhlá data. Webová stránka uvádí, že nástroj Jplag byl v minulosti několikrát využit soudními znalci v případě sporů o duševní vlastnictví.
Tento nástroj je zcela zdarma. Výhodou oproti nástroji MOSS je jeho grafické uživatelské rozhraní, výsledky detekce jsou díky přehlednější a uživatelsky přívětivější. V současnosti JPlag pracuje s pěti programovacími jazyky (Java, C#, C, C++, Scheme) a prostým textem [5].

SidPlag

Volně přístupný program SidPlag nabízí řadu volitelných možností. Uživatel si může zvolit mezi srovnáváním podřetězců nebo tokenů. Dále je možno určit zda mají být porovnávány také komentáře. U výsledků si pak uživatel může nechat zobrazit pouze nejpodobnější kódy, nebo všechny.
V současnosti SidPlag podporuje programovací jazyky C++ a Java [4].

Všechny zmíněné detektory jsou uvedeny v následující srovnávací tabulce:

	Moss	**JPlag**	**Codequiry**	**SigPlag**
Počet jazyků	26	5	20+	2
Matlab	Ano	Ne	Ano	Ne
Cena	Zdarma	Zdarma	$29/měsíc	Zdarma
Srovnání s online zdroji	Ne	Ne	Ano	Ne
GUI	Ne	Ano	Ano	Ano
Ke stažení / online	Online	Online	Online	Ke stažení
Metoda detekce	Porovnávání podřetězců	Tokenizace	Tokenizace	Volitelně
Open source	Ne	Ne	Ne	Ne

Tab. 2.1: Srovnání existujících detektorů

3 Implementace detektoru

3.1 Předzpracování

Ještě před přistoupením k samotné detekci podobností je vhodné programové kódy předzpracovat. I plagiátor s minimální znalostí jazyka je schopen plagiát zamaskovat změnami, které mají velký vliv na vzhled kódu, ale nulový vliv na funkčnost.

Typickým příkladem je změna komentářů, nebo změna názvů proměnných. Dále je možné přidat prázdné řádky, nebo počet řádků naopak zredukovat použitím více příkazů na jednom řádku, oddělených středníkem. Stejně jako řádkování, i množství mezer se může u dvou kódů se stejnou funkcí lišit, je tedy vhodné veškeré formátování odstranit tak, aby zbyl jen čistý, tzv. užitečný kód.
V rámci předzpracování je tedy vhodné:

- Odstranění komentářů
- Redukce mezer a odsazení
- Rozdělení příkazů na samostatné řádky, jsou-li zapsány na jednom
- Odstranění prázdných řádků
- Převedení velkých písmen na malé

3.1.1 Možné problémy s předzpracováním

Komplikací při předzpracování může být méně obvyklá syntaxe. Je nutné uvážit všechny možné formy zápisu funcí i dalších částí kódu.

Komentáře

Komentáře mohou být označeny tak, že je před každý komentář umístěn symbol %. Pro větší bloky komentáře však může být použita také následující syntaxe:

```
1 %{
2 Tento blokový komentář může
3 zmást detektor stejným způsobem,
4 jako zmátl balíček pro sazbu zdrojových kódů.
5 Ve skutečnosti je komentářem celý tento kód
6 a měl by být vysázen cursivou.
7 %}
```

Výpis 3.1: Ukázka blokového komentáře

Je nutné tuto možnost zohlednit při programování softwaru pro předzpracování. Detekce komentářů založená pouze na počátečním symbolu % by komentář neodstranila. Ponechání velkého bloku komentářů mezi užitečným kódem může rapidně snížit kvalitu detekce podobnosti.

Kromě dvou různých zápisů může být problém také samotný symbol %, který nemusí být použit jako značka komentáře, může se jednat o část kódu. Pokud by byl odstraněn jakýkoli text za tímto symbolem bez ošetření podmínek, mohlo by docházet ke ztrátě užitečné části kódu.

Bílé znaky

Bílé znaky je možné v kódu ponechat, nebo je v rámci předzpracování částečně či úplně odstranit. Pokud je jako jeden z příznaků pro detekci zvolen počet znaků, mohou vícenásobné bílé znaky přispět k zavádějícím hodnotám a snížit tak přesnost detekce. Odstranění veškerých bílých znaků v kódu je nejjednodušší, avšak může při detekci zakrýt například některé proměnné.

3.1.2 Realizace předzpracování

Z důvodů uvedených v předchozí podkapitole bylo odstraněny bílé znaky na začátcích a koncích řádků a všechny vícenásobné mezery byly nahrazeny právě jednou mezerou. Díky tomuto postupu nedochází ke zkreslení výsledků při detekci proměnných, funkcí ani počtu znaků.

V realizovaném programu byly také ošetřeny všechny body uvedené v textu Předzpracování (viz kapitolu 3.1). Pro ukázku je použit krátký kód, ke kterému byl vytvořen stejný kód s jiným formátováním.

Z ukázky je jasně patrné, že změnou komentářů a řádkování lze velkou mírou ovlivnit vzhled kódu. Na první pohled tak nemusí být zřejmé, že se jedná o totožný kód. Po předzpracování obou kódu je to však nezpochybnitelné, protože výsledky předzpracování jsou pro oba kódy naprosto totožné.

```
1  function [Zaporne,Kladne,Nuly] = rozdelovac(vektor)
2    %{funkce rozdelovac rozdeli vektor podle ...%}
3    Zaporne = []; Kladne = [];    %deklarace promennych
4    for i = 1:length(vektor);     %pro cely vektor
5      if vektor(1, i) < 0         %pokud je mensi nez 0
6        Zaporne=[Zaporne, vektor(1, i)]
7      elseif vektor(1, i) > 0     %pokud je vetsi nez 0
8        Kladne=[Kladne, vektor(1, i)]; end
9    end
10 end
```

Výpis 3.2: První kód před předzpracováním

```
1  function [zaporne,kladne,nuly] = rozdelovac(vektor)
2    %funkce rozdeli vektor podle hodnot
3    % in: vektor
4    % out: tri vektory
5
6    zaporne = [];
7    kladne = [];
8    for i = 1:length(vektor)
9      if vektor (1, i) < 0
10       zaporne=[zaporne, vektor(1, i)]
11     elseif vektor(1, i) > 0
12       kladne=[kladne, vektor(1, i)]
13     end
14   end
15 end
```

Výpis 3.3: Druhý kód před předzpracováním

```
1  function [zaporne,kladne,nuly] = rozdelovac(vektor)
2  zaporne = []
3  kladne = []
4  for i = 1:length(vektor)
5  if vektor(1, i) < 0
6  zaporne=[zaporne, vektor(1, i)]
7  elseif vektor(1, i) > 0
8  kladne=[kladne, vektor(1, i)]
9  end
10 end
11 end
```

Výpis 3.4: Výsledek předzpracování pro první i druhý kód

3.2 Detekce příznaků

Pro tvorbu detektoru bylo využito principu porovnávání pětice příznaků.
Konkrétně jsou to:

- Klíčová slova
- Proměnné
- Počet řádků kódu
- Průměrná délka řádku
- Počet čísel v kódu

3.2.1 Klíčová slova

Klíčová slova jsou hlavním příznakem detektoru.
Bylo zvoleno celkem 58 klíčových slov. Část z nich tvoří 20 příkazů, které definuje jako klíčová slova sám Matlab a lze je získat voláním příkazu **iskeyword** bez vstupního argumentu. Tato slova jsou v tabulce klíčových slov zvýrazněna tučným písmem. Zbylá klíčová slova jsou kombinace příkazů často využívaných při výuce informatiky a číslicových signálů, případně elementární matematické nebo logické příkazy.

Klíčová slova			
abs	fft2	isempty	size
break	figure	mat2gray	sort
case	find	max	**spmd**
catch	**for**	mean	stem
cell2mat	fprintf	min	str2double
classdef	**function**	mod	str2num
continue	genbankread	num2str	strcmp
diff	**global**	ones	strfind
disp	**if**	**otherwise**	sum
else	ifft	**parfor**	switch
elseif	ifft2	**persistent**	**try**
end	ifftshift	plot	**while**
fastaread	imshow	print	zeros
fft	length	**return**	
fftshift	mat2cell	rgb2gray	

Tab. 3.1: Abecedně řazený seznam detekovaných klíčových slov

Specifika klíčových slov

Při detekci klíčových slov je důležité vzít v potaz jejich syntaxi. Různé příkazy se obvykle vyskytují na různých místech kódu. Aby detekce proběhla s co nejmenším počtem chyb, je důležité s těmito rozdílnostmi pracovat.
Podle umístění rozlišujeme příkazy následovně:

(a) Příkazy stojící vždy na začátku řádku
Mezi takové příkazy patří například *for, while, if, else* apod. V předchozí verzi programu, která odstraňovala veškeré bílé znaky, by mohlo docházet k falešně pozitivní detekci v případě, že by řádek začínal názvem proměnné, obsahující některý z těchto příkazů. Například výskyt proměnné *forest* na začátku řádku by vedl k falešné detekci klíčového slova *for*. Tento problém byl odstraněn úpravou procesu předzpracování. Současná verze programu tedy pracuje i s mezerami a v případě klíčových slov na začátku řádku vyhledá pouze klíčové slovo, po kterém bezprostředně následuje mezera.

(b) Příkazy stojící na samostatném řádku
Tímto specifickým typem klíčových slov jsou příkazy *end, break* a *continue*. Jejich syntaxe dělá detekci velmi jednoduchou. Problém by mohl způsobit pouze zápis na řádku s jinými příkazy, s použitím středníku. Díky předzpracování jsou ale příkazy oddělené středníkem rozděleny do více řádků a detekce funguje bez problémů.

(c) Příkazy stojící na libovolné pozici
Příkazy, které nespadají do předchozích dvou kategorií, mohou stát na libovolném místě kódu. Jejich detekce tak není založena na pozici, ale na nutnosti užití závorek pro argument.
Mezi takovéto příkazy patří například *strfind(), length(), size()* apod.
U některých příkazů je nutné ošetřit podmínky tak, aby nebyly vícekrát detekovány překrývající se názvy příkazů. Například příkaz *strfind()* by mohl být započítán dvakrát, podruhé jako *find()*. Této chybě detektor předchází kontrolou pozice bezprostředně předcházející danému klíčovému slovu. Slovo se detekuje pouze v případě, že se na této pozici nenachází žádný znak abecedy.

3.2.2 Proměnné

Dalším detekovaným příznakem jsou proměnné, pro které jsou detekovány tři hodnoty.

Nejprve program nalezne každý řádek, na kterém je proměnné přiřazena hodnota, tzn. řádek, na kterém se proměnná nachází před znakem =. Tuto proměnnou uloží do seznamu proměnných, který je následně pročištěn od duplicit. V případě, že je deklarováno více proměnných na jednom řádku, program tyto proměnné rozdělí a

přidá do seznamu každou zvlášť. Délka seznamu pak udává první důležitou hodnotu, a totiž **počet deklarovaných proměnných**.
Proměnné z tohoto seznamu jsou dále vyhledány v celém kódu. Podle polohy vůči rovnítku se proměnná pričte buď k počtu výskytů proměnných **před rovnítkem** nebo k počtu výskytů proměnných **za rovnítkem**.

3.3 Vyhodnocení podobnosti

Každý kód je po zpracování reprezentován pěticí příznaků ve formě čísel. Každá jedna reprezentace je porovnána se všemi ostatními.

Klíčová slova

Nejvyšší váhu při výpočtu podobnosti mají klíčová slova, reprezentovaná vektorem. Každé klíčové slovo má ve vektoru danou pevnou pozici, na kterou je uložena četnost výskytu slova v kódu.
Pro názornost uveďme jako příklad následující kód:

```
1 for c = 1:ncols
2     for r = 1:nrows
3         if r == c
4             A(r,c) = 2;
5         end
6     end
7 end
```

Výpis 3.5: Kód pro ukázku reprezentace klíčových slov

Výsledkem pro uvedený kód je níže vypsaný vektor. Protože se v kódů nacházejí právě tři různá klíčová slova, vektor obsahuje tři nenulové pozice. Každý vektor má právě 58 pozic, protože je hledáno 58 různých klíčových slov. Pro potřeby textu bylo z uvedeného příkladu posledních 46 nulových hodnot vypuštěno.

$$[\,0,\quad 0,\quad 0,\quad \underset{\uparrow\texttt{end}}{3},\quad 0,\quad 0,\quad 0,\quad \underset{\uparrow\texttt{for}}{2},\quad 0,\quad 0,\quad 0,\quad \underset{\uparrow\texttt{if}}{1},\quad \ldots\,]$$

Porovnání dvojic vektorů je provedeno Bray–Curtisovou metrikou (3.1). Jde v podstatě o city-block metriku, rozšířenou o normalizaci, díky které výsledná vzdálenost nabývá hodnot od 0 do 1 [6]. Pro získání hodnoty podobnosti, vhodné pro finální výpočet, je třeba hodnotu vzdálenosti převrátit dle vztahu 3.2.

Ve finálním výpočtu je hodnota podobnosti vektorů váhována – s klesající podobností ze zvyšuje její váha (3.3). Tedy pokud je podobnost vektorů nízká, je váha

podobnosti vysoká. I v případě velkých podobností pro jiné příznaky tak zůstane celková podobnost nižší, protože podobnost ostatních příznaků může být snáze dílem náhody.

Naopak v případě, že bude podobnost vysoká, její váha bude srovnatelná s vahou ostatních příznaků. Cílem proměnlivé váhy je eliminace false-positive detekcí.

$$d_{ij} = \frac{\sum_{k=1}^{n} |x_{ik} - x_{jk}|}{\sum_{k=1}^{n} (x_{ik} + x_{jk})} \tag{3.1}$$

$$p_{ij} = 1 - d_{ij} \tag{3.2}$$

$$v(p_{ij}) = \begin{cases} 3 & \text{pro } p_{ij} < 0,5 \\ 2 & \text{pro } 0,5 \leq p_{ij} \leq 0,7 \\ 1 & \text{pro } p_{ij} > 0,7 \end{cases} \tag{3.3}$$

Ostatní příznaky

Pro určování podobnosti všech ostatních příznaků je využito jednoduchého poměru (3.4). V tomto poměru je vždy menší z dvojice porovnávaných hodnot v čitateli a větší ve jmenovateli. Výsledek nabývá, stejně jako výsledek pro klíčová slova, hodnot od 0 do 1.

$$p_{ij} = \frac{\min(i, j)}{\max(i, j)} \tag{3.4}$$

Podle významu pro detekci jsou podobnosti příznaků váhovány. Počet řádků a průměrná délka řádku mají přidělenou váhu 1, počet čísel má pro svou nejmenší významnost váhu 0,5. Nejvyšší váhu z příznaků porovnávaných poměrem mají proměnné.

Podobnost proměnných je určena aritmetickým průměrem poměrů pro všechny tři hodnoty (počet deklarovaných proměnných a počty proměnných před a za rovnítkem).

Finální výpočet podobnosti (3.5) je pak sumou dílčích podobností, násobených vlastními vahami, děleným sumou všech vah. Dělení vahami zajišťuje výsledek v rozmezí 0 a 1.

$$P = \frac{\sum_{k=1}^{5} v_k p_k}{\sum_{k=1}^{5} v_k} \tag{3.5}$$

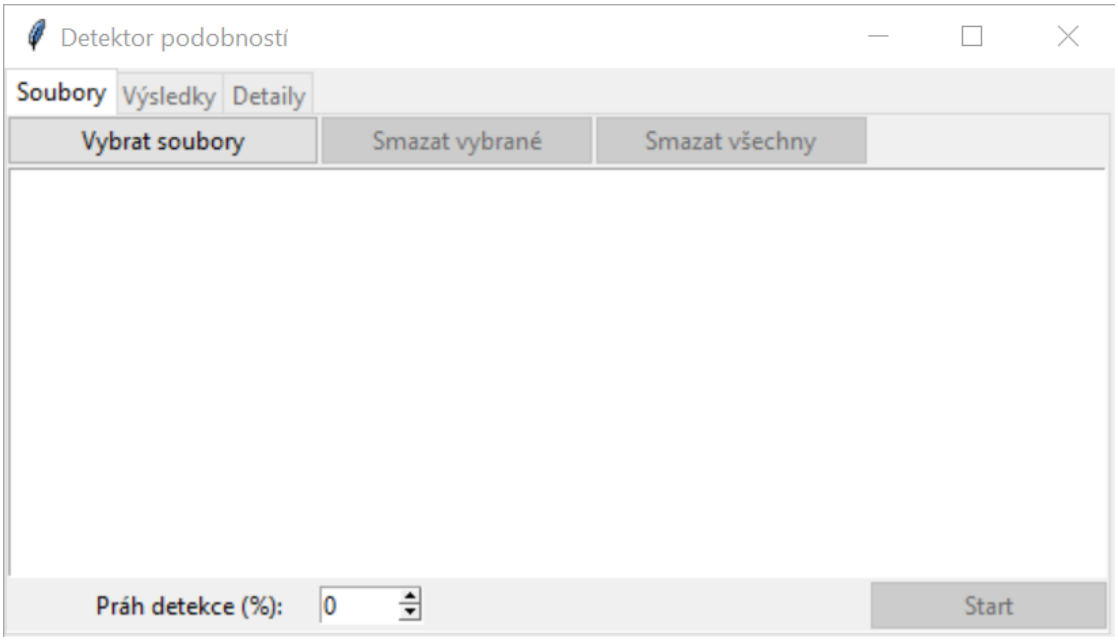
4 Grafické uživatelské rozhraní a testování detektoru

4.1 Grafické uživatelské rozhraní

Pro snazší a pohodlnější použití byl detektor vybaven grafickým uživatelským rozhraním (GUI), realizovaným pomocí vestavěného pythonovského modulu Tkinter. GUI disponuje třemi okny.

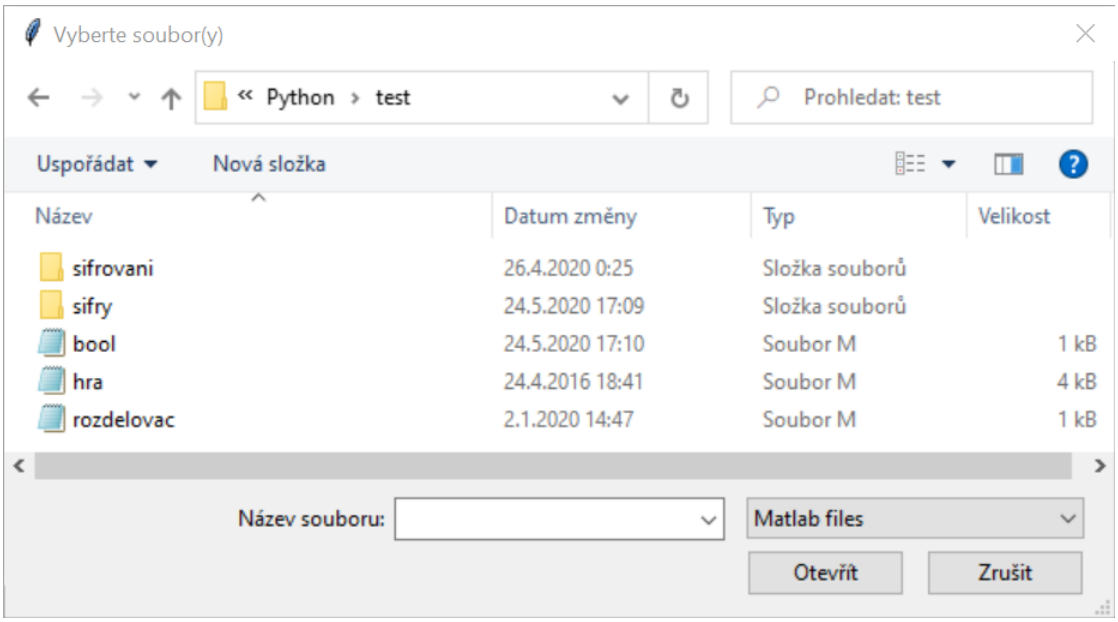
Okno pro výběr souborů

V prvním okně (Obr. 4.1) uživatel vybírá soubory ke zpracování.



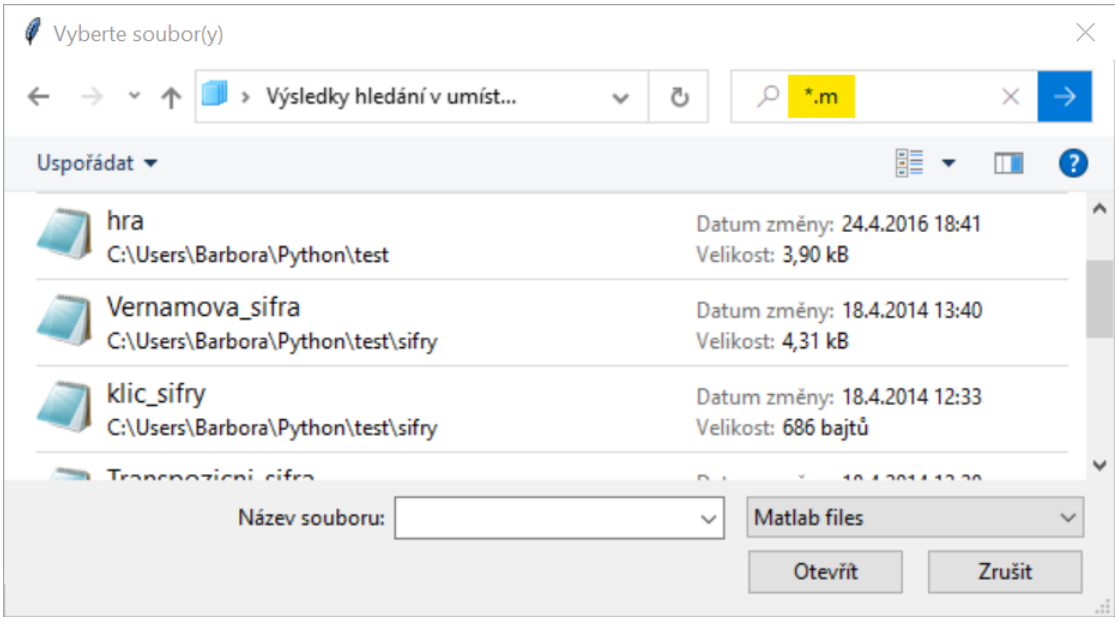
Obr. 4.1: První okno detektoru, slouží k manipulaci se soubory

Po stisknutí tlačítka *Vybrat soubory* se zobrazí systémové dialogové okno pro výběr souborů ve formátu *.m* (Obr. 4.2). V tomto okně je možné vybrat více souborů, a to tažením kurzoru, případně zakliknutím jednoho souboru s následnou kombinací kláves Shift + ↑ nebo Shift + ↓. Všechny soubory najednou ve vybraném adresáři je možné označit klávesovou zkratkou Ctrl + A.



Obr. 4.2: Systémové okno pro výběr souborů

Pro otevření souborů ze všech podadresářů lze využít vyhledávání (pro systém Windows v pravém horním rohu okna). Do vyhledávacího pole je nutno vypsat **.m*, což je zadání pro vyhledání všech souborů v matlab formátu. Po vyhledání se všechny soubory ze všech podadresářů zobrazí v okně (Obr. 4.3), kde je možné je jednoduše vybrat některým ze způsobů uvedených v předchozím odstavci.



Obr. 4.3: Ukázka výsledku vyhledávání v dialogovém okně. Vyhledávací pole je označeno žlutě.

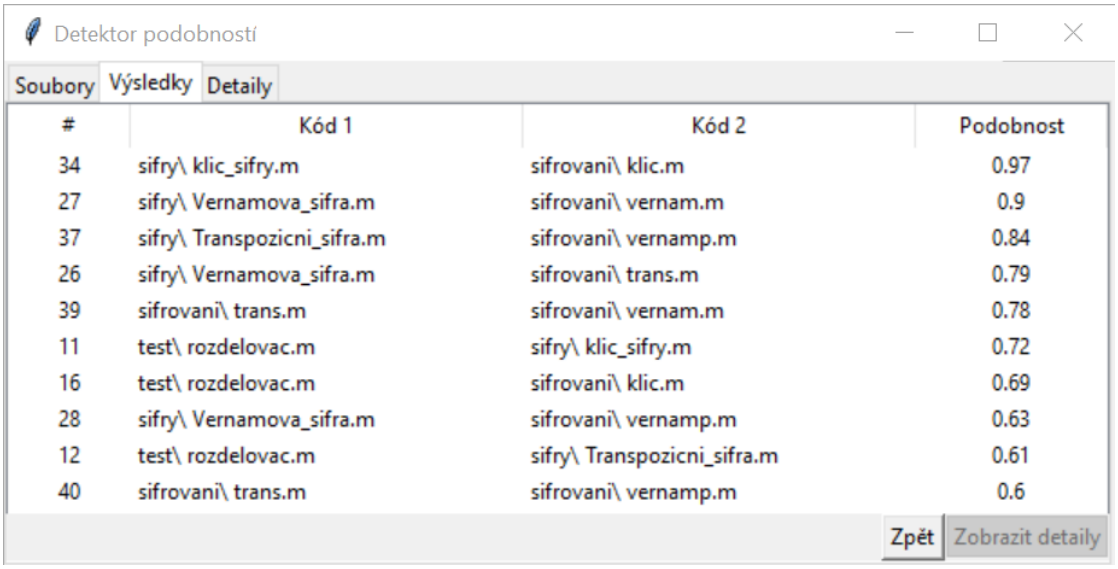
Po vybrání souborů se seznam jejich cest zobrazí v tabulce. V té je možné navigovat jak šipkami, tak kurzorem. Pro smazání více vybraných souborů je možné soubory vybrat kombinací klávesy Shift a šipek, případně klikáním za držení klávesy AltGr.

Ve spodní části stránky se nachází pole pro nastavení prahu detekce. Ve výsledkové tabulce se zobrazí pouze dvojice kódů s hodnotou podobnosti větší, než je hodnota prahu. Použití prahu je vhodné v případě zpracovávání velkého objemu dat – urychlí celý proces.

Pokud je uživatel spokojen s vybranými soubory, klikne na tlačítko *Start*. Program spustí všechny výpočty a přepne grafické rozhraní na druhé okno.

Okno pro zobrazení výsledků

Druhé okno (Obr. 4.4) je složeno z tabulky výsledků. Každý řádek obsahuje 4 informace – číslo porovnání (spíše pro zajímavost), názvy obou srovnávaných kódů a jejich vypočtenou podobnost. Názvy kódů jsou vypsány i s názvem adresáře, ve kterém se nacházejí, pro přehlednost v případě, že je srovnáváno více různých souborů se stejným názvem. Tabulku lze seřadit podle podobnosti, vzestupně i sestupně, kliknutím na hlavičku jejího sloupce.



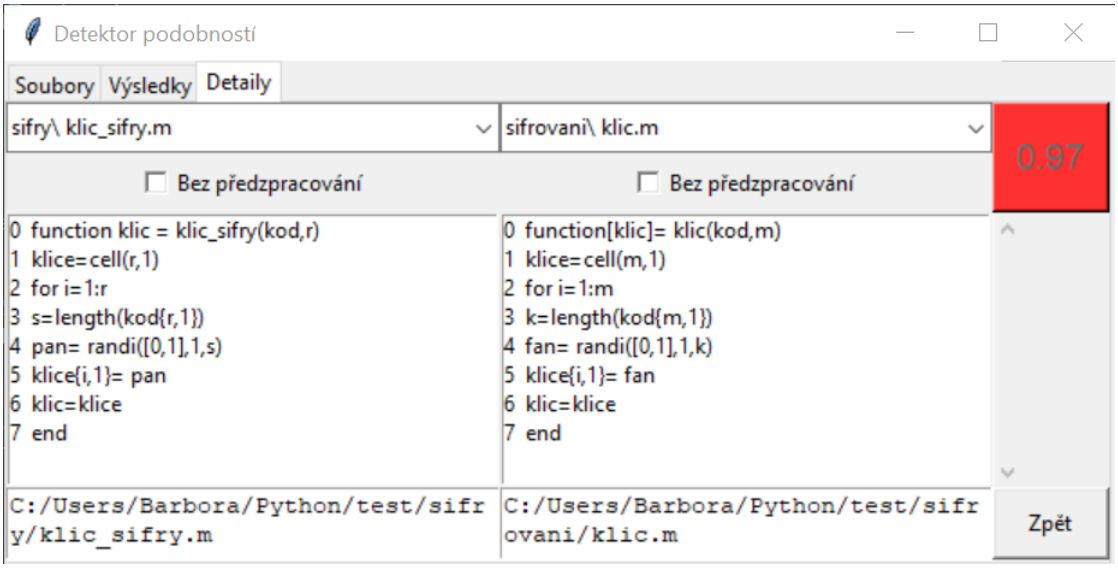
Obr. 4.4: Druhé okno detektoru se zobrazením výsledků

Stiskem tlačítka *Zpět* se uživatel může vrátit do prvního okna. Pokud si uživatel přeje podrobně prohlédnout libovolnou dvojici kódů, kliknutím označí příslušný řádek tabulky a stiskne *Zobrazit detaily*. Tímto krokem se přesune do třetího okna a zobrazí v něm dvojici vybraných kódů.

Okno pro zobrazení detailů

Třetí okno programu (Obr. 4.5) se skládá ze dvou totožných bloků. Je v nich možno vybrat si jakýkoli z vložených kódů a prohlédnout si jak jeho předzpracovanou, tak i původní verzi. Ve spodní části se pak zobrazuje celá cesta k souboru.

Po pravé straně se zobrazuje barevné okénko s hodnotou podobnosti pro vybranou dvojici a tlačítko zpět, které uživatele vrací do druhého okna. Mezi těmito prvky se nachází posuvník, který posunuje zároveň oba kódy, což je praktické v případě dvou podobných, dlouhých kódů.



Obr. 4.5: Třetí okno detektoru pro zobrazení detailů

4.2 Hodnocení výsledků detekce

4.2.1 Databáze

Pro testování detektoru podobností je využita databáze studentských projektů. Projekty byly vytvořeny studenty předmětu APRG ve vývojovém prostředí MATLAB. V rámci ochrany autorských práv a ochrany studentů před nařčením z plagiátorství jsou veškeré ukázky z těchto projektů anonymizovány. Před použitím detektoru byla databáze vyčištěna od souborů nekompatibilních se systémem Windows a od skriptů pro grafické uživatelské rozhraní, na které není detektor konfigurován. Celkem databáze čítá 1976 souborů, rozdělených dle tématu a dále ještě dle autora.

4.2.2 Testování

Pro testování byly zvoleny nejprve dva tematické okruhy a následně celá databáze (celkem 26 okruhů). V případě testování všech dat najednou byl zvolen vyšší práh detekované podobnosti pro usnadnění kontroly výsledků. I přes 95% práh bylo nalezeno 315 podobných dvojic kódů. Všechny důležité hodnoty jsou pro přehlednost uvedeny v tabulce 4.1

	Integrály	**Batoh**	**Všechny okruhy**
Počet kódů celkem	202	24	1 976
Počet známých plagiátů	14	3	76
Počet true-positive detekcí	21	3	157
Počet false-positive detekcí	22	0	158
Počet false-negative detekcí	0	0	0
Počet porovnání	20 301	276	1 951 300
Zvolený práh podobnosti [%]	89	89	95
Senzitivita [%]	100	100	100
Specificita [%]	88	100	91

Tab. 4.1: Informace o testovacích souborech a statistickém vyhodnocení

Hodnota senzitivity, která udává schopnost detektoru správně rozeznat skutečné plagiáty, je vypočtena ze vztahu 4.1. Zde *TP* značí true-positive, tedy správně detekované plagiáty, a *FN* false-negative, tedy skutečné plagiáty, které detektor neodhalil.

$$senzitivita = \frac{TP}{TP + FN} \tag{4.1}$$

Hodnota specificity, udávající schopnost detektoru správně určit všechny kódy, které nejsou plagiáty, je dána vztahem 4.2. *TN* zde značí true-negativ, tedy kódy které detektor správně neoznačil jako plagiáty. *FP* značí false-positive, tedy hodnoty, které byly detektorem nesprávně označeny jako plagiáty.

$$specificita = \frac{TN}{TN + FP} \tag{4.2}$$

Velmi vysoká míra false-positive detekcí je způsobena množstvím kódů od stejných autorů. V databázi je velmi častý výskyt kódů, psaných stejným člověkem, které jsou téměř totožné, pouze s drobnými změnami. Jeden autor dokonce do databáze přispěl dvěma verzemi svého projektu, jednou s komentáři a druhou bez. Tyto detektor samozřejmě vyhodnotil jako 100% plagiáty.

Velký podíl false-positive také způsobily projekty, jejichž soubory si byly navzájem podobné a navíc byly plagiovány. Například pokud se v jednom projektu nachází tři velmi podobné soubory, detektor určí tři false-positive dvojice. Pokud však bude vytvořen plagiát celého projektu, počet false-positive stoupne na dvanáct, přičemž skutečně plagiované soubory jsou pouze tři. Tyto výpočty, stejně jako výpočty počtů porovnání v tabulce 4.1 vycházejí ze vztahu 4.3. N značí celkový počet porovnání a n počet porovnávaných souborů.

$$N = \frac{n(n-1)}{2} \tag{4.3}$$

V případě, že je projekt jednoho autora rozdělen do více souborů, je velmi vhodné tyto soubory umístit do samostatného adresáře, jako je tomu v testovací databázi. Po zpracování detektorem je v tabulce podobností zahrnut kromě názvu souboru také název adresáře. Uživatel detektoru v takovém případě na první pohled pozná, zda jde o dílo jednoho autora.

Skutečné vyhodnocení false-positive, kdy detektor označil dva odlišné kódy za podobné, nastalo pouze ve dvou případech. V obou případech šlo o kódy kratší než čtyři řádky. Pokud dva takto krátké kódy obsahují definici funkce, shodují se hned ve dvou klíčových slovech (*function, end*), což vede k jejich vysoké podobnosti i v případě, že obsahy funkcí podobné nejsou.

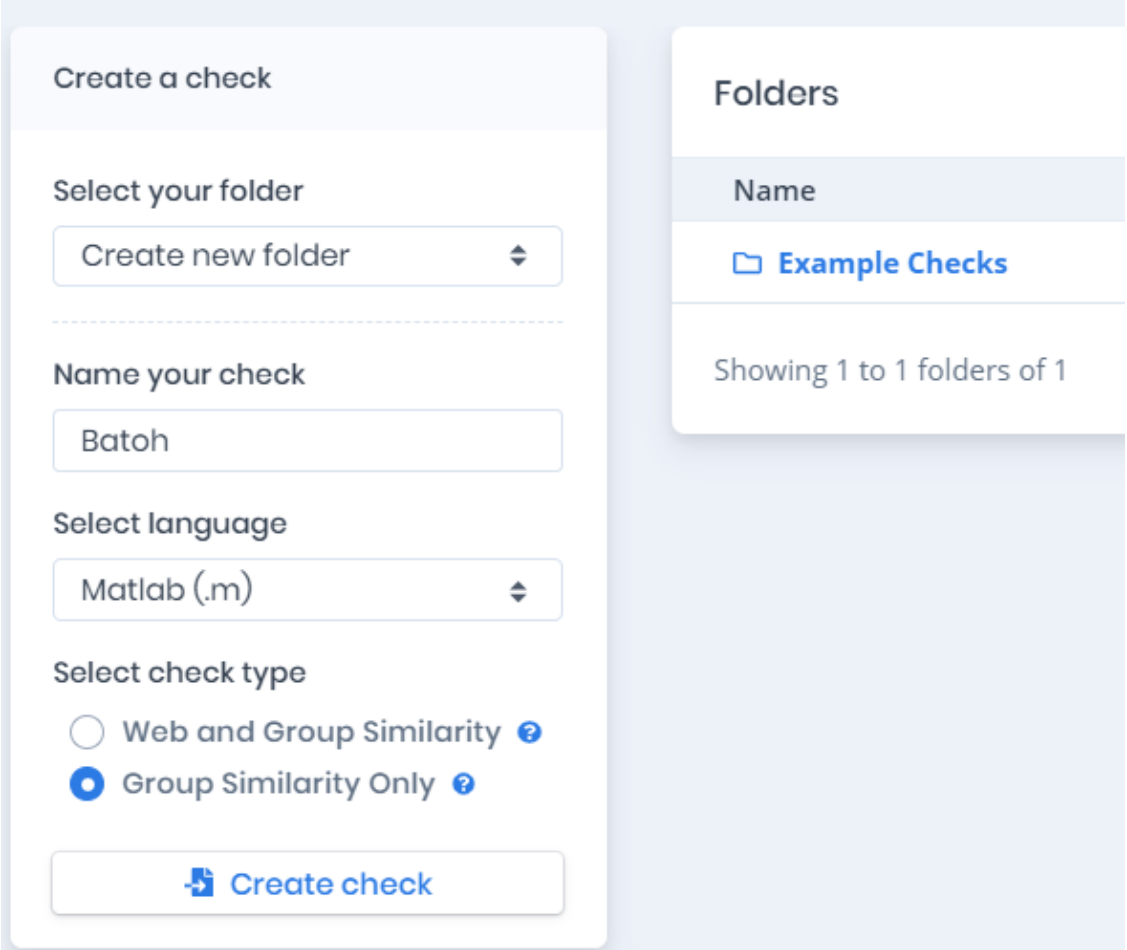
4.3 Srovnání s Codequiry

V rámci hodnocení funkčnosti proběhlo srovnání vlastního detektoru s komerčním detektorem plagiátů Codequiry.

Codequiry očekává na vstupu projekty o více souborech, proto je nutné nahrávat každý projekt zvlášť, archivovaný ve formátu zip. Výhodou tohoto přístupu je zamezení false-positive detekcím v rámci kódů od stejného autora. Nevýhodou se tento přístup stává v případě, kdy chce uživatel otestovat mnoho dat, které ale v zip archivu nejsou (tak jako v případě testovací databáze). Program neumí pracovat s adresáři, takže v případě vložení jednoho archivu, obsahujícího několik adresářů, nenabídne žádné výsledky. V případě snahy o kontrolu celé databáze studentských prací by tedy bylo nutno všech 1976 souborů rozdělit do zip archivů dle autora.

Pro ukázku byly zazipovány a použity projekty z tematického okruhu *Batoh*. Codequiry prakticky rozděluje testované soubory do složek. Před každým novým spuštěním se uživatele zeptá, zda chce vytvořit novou složku, nebo detekci zahrnout do některé již existující (viz obrázek 4.6). Dále musí uživatel nastavit programovací jazyk vkládaných kódů a určit, zda má porovnávání proběhnout jen v rámci vlastních souborů, nebo i se soubory online. Pro potřeby srovnání obou detektorů bylo

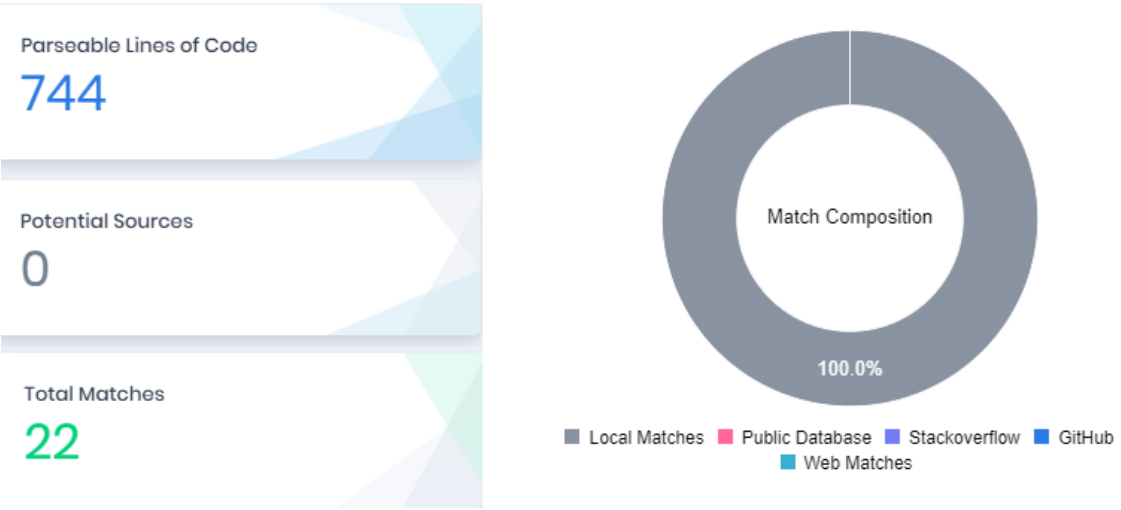
vyhledávání podobností s online zdroji vypnuto.



Obr. 4.6: Vkládání souborů a volba parametrů pro detekci v Codequiry

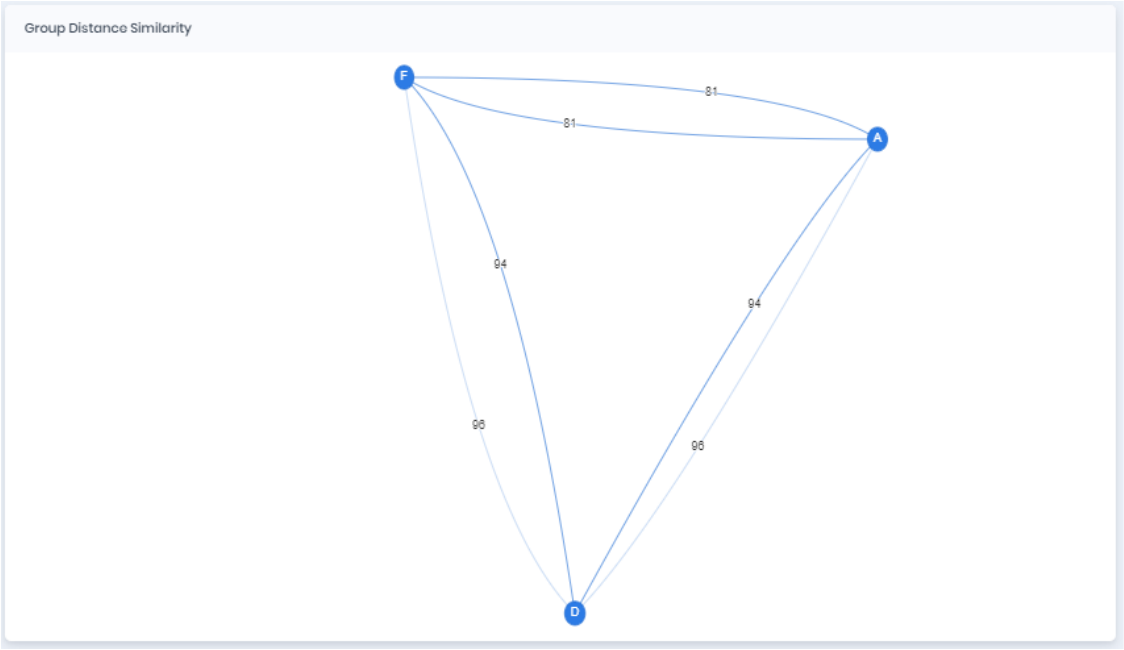
První ze způsobů zobrazení výsledků dá uživateli informace o počtu analyzovaných řádků, počtu potenciálních online zdrojů a počtu nalezených shod. Na výsečovém grafu zobrazí podíl zdrojů podobných souborů (zda se kódy shodují s ostatními ve skupině, nebo s některým z online zdrojů)(obrázek 4.7). Dále určí míru pravděpodobnosti plagiátorství ve skupině souborů Poslední informací na této stránce je výpis všech analyzovaných projektů, přičemž každý řádek obsahuje název projektu a procentuální shodu s jiným zdrojem (obrázek 4.8).

Druhé zobrazení nabízí mapu podobností (obrázek 4.9), dále pak seznam vzájemně podobných dvojic s procentuální podobností a sloupcových graf podobností (obrázek 4.10)

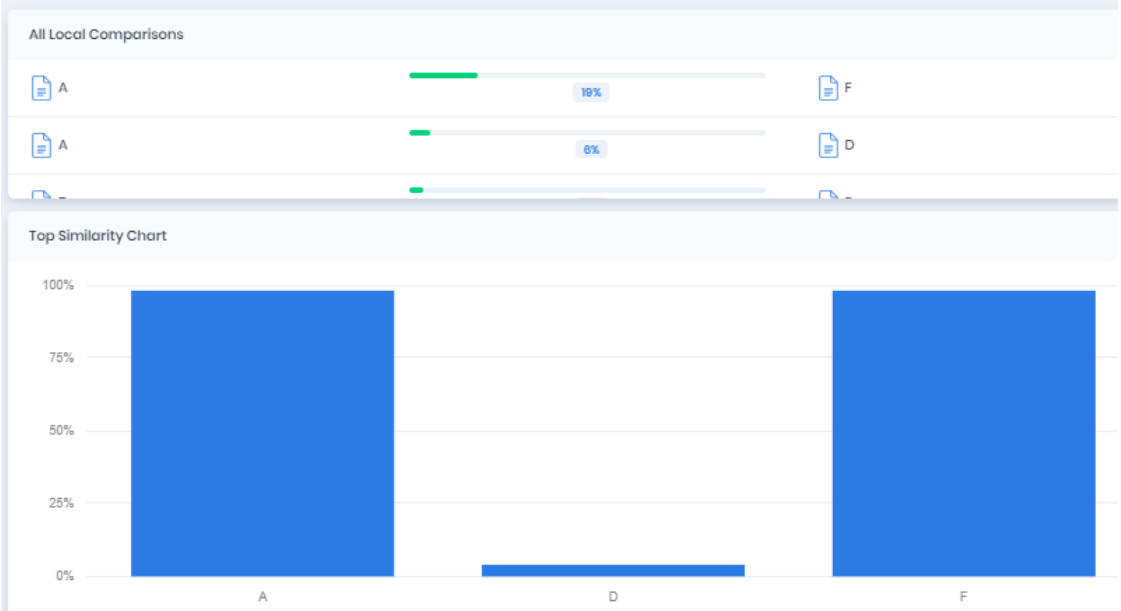


Obr. 4.7: Codequiry: Počet analyzovaných řádků, počet shod, podíl zdrojů

Obr. 4.8: Codequiry: seznam dvojic kódů s procentuální shodou a podobnostní skóre



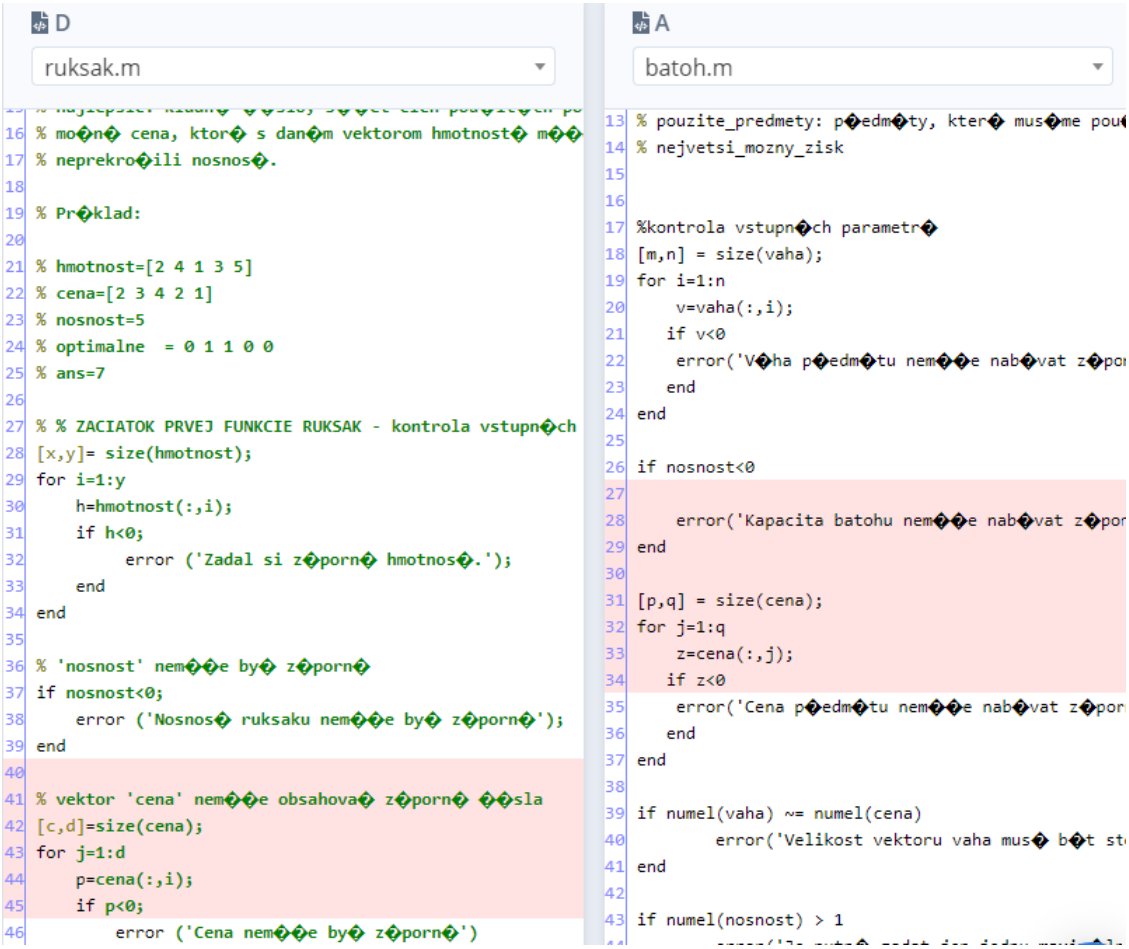
Obr. 4.9: Mapa podobností porovnávaných projektů v Codequiry



Obr. 4.10: Srovnání podobnosti dvojic a sloupcový graf podobností v Codequiry

Dále Codequiry nabízí možnost zobrazení dvojice kódů vedle sebe. Velkou výhodou je zvýraznění podobných částí kódu. Za nevýhodu je možné považovat kódování, které není kompatibilní s češtinou, proto stránka špatně zobrazuje části kódu s diakritikou. Další nevýhodu je možné spatřit ve způsobu, jakým uživatel kódy ke

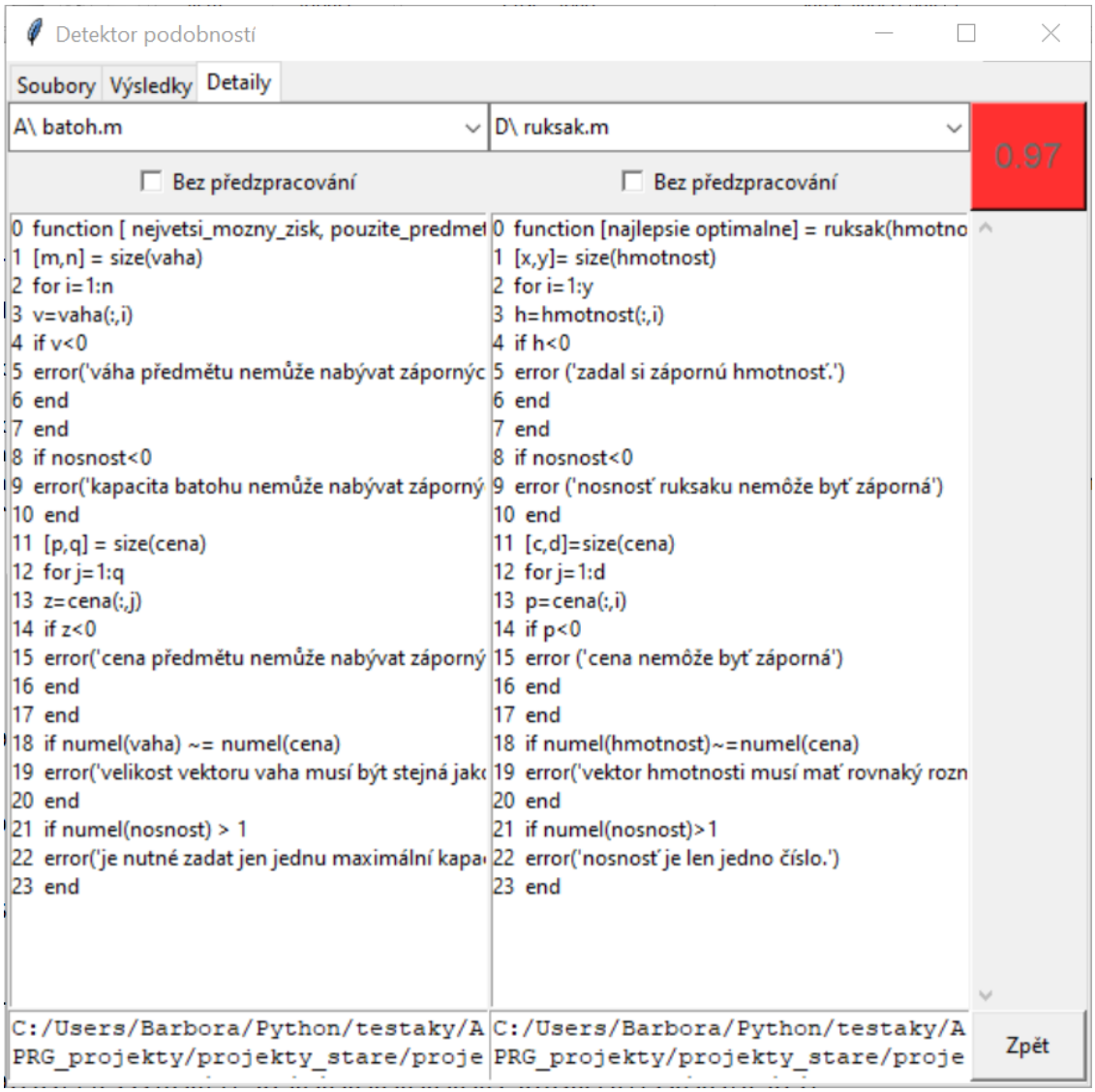
zobrazení vybírá. V rolovacím okně uživatel vybere projekt, ke kterému chce zobrazit shody. Výsledkem je výpis dvojic kódů o určité podobnosti, ze kterého však není zřejmé, ze kterého projektu pochází druhý kód z dvojice. V detektoru implemenovaném v rámci této práce je adresář, tedy nositel informace o projektu, uveden na všech místech, kde je zobrazen název kódu. Také je schopen pracovat s diakritikou a uživatel si může zobrazit jakékoli dva kódy, nejen ty, u kterých byla detekována podobnost.



Obr. 4.11: Srovnání dvojic kódů v Codequiry. Otazníky v kódech jsou způsobeny použitím jiného kódování, než bylo použito při psaní kódů.

Z obrázku 4.11 je patrné, že detekce probíhá bez předchozího ostranění komentářů. V případě, že se plagiátor neobtěžuje plagiát alespoň minimálně zamaskovat, je to jistě výhodné. Ovšem v momentě, kdy dojde ke změnám v komentářích či názvech proměnných, výsledky detekce neodpovídají skutečné míře podobnosti. Tento fakt je z obrázku také zřejmý – ač jde o dva totožné kódy, ve kterých byly změněny pouze drobnosti bez skutečného vlivu na funkčnost, detektor Codequiry nebyl schopen rozpoznat plagiát a určil vzájemnou podobnost dvojice na pouhých 5 %.

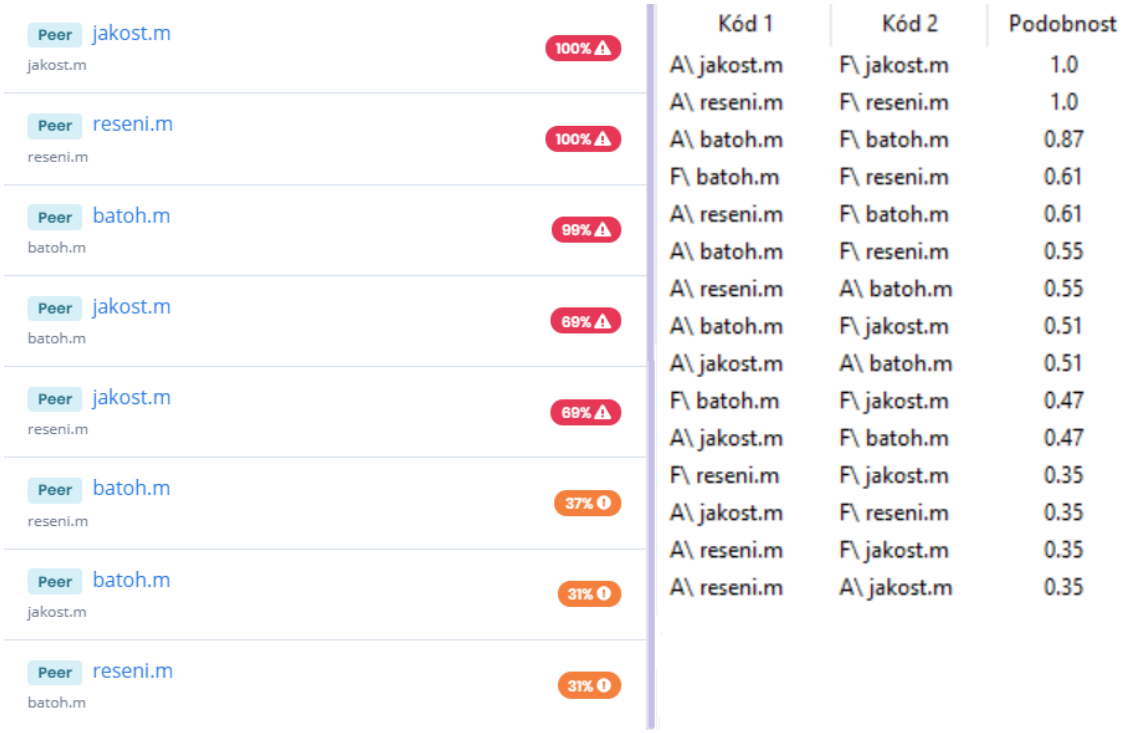
Vytvořený detektor podobností byl v tomto případě, díky kvalitnímu předzpracování, mnohem úspěšnější. Jak je patrno z obrázku 4.12, vzájemnou podobnost správně určil na 97 %. Tři procenta rozdílnosti v této dvojici způsobily různé průměrné délky řádku, neboť plagiátor změnil i text v příkazu *disp*, který při předzpracování není odstraněn.



Obr. 4.12: Zobrazení dvojice kódů ve vytvořeném detektoru

Obrázek 4.13 obsahuje výstup obou detektorů při stejných vstupních datech. Dvě počáteční dvojice kódů oba detektory správně vyhodnotily jako totožné. Ve třetí dvojici jde o totožné kódy, jeden z nich je ovšem o několik řádků delší. Codequiry vyhodnotil vyšší podobnost než vytvořený detektor na základě totožných komentářů. U čtvrté srovnávné dvojice určil detektor Codequiry podobnost 69 % a vytvořený detektor pouze 47 %. Zatímco Codequiry v tomto případě považuje tyto kódy za

možné plagiáty, hodnota podobnosti 47 % je ve vytvořeném detektoru považována spíše za náhodnou podobnost. Tento rozdíl ve výsledných hodnotách vznikl opět kvůli totožným komentářům, které Codequiry bere v potaz. Stejný princip pak platí i pro všechny ostatní výsledky.



Kód 1	Kód 2	Podobnost
A\ jakost.m	F\ jakost.m	1.0
A\ reseni.m	F\ reseni.m	1.0
A\ batoh.m	F\ batoh.m	0.87
F\ batoh.m	F\ reseni.m	0.61
A\ reseni.m	F\ batoh.m	0.61
A\ batoh.m	F\ reseni.m	0.55
A\ reseni.m	A\ batoh.m	0.55
A\ batoh.m	F\ jakost.m	0.51
A\ jakost.m	A\ batoh.m	0.51
F\ batoh.m	F\ jakost.m	0.47
A\ jakost.m	F\ batoh.m	0.47
F\ reseni.m	F\ jakost.m	0.35
A\ jakost.m	F\ reseni.m	0.35
A\ reseni.m	F\ jakost.m	0.35
A\ reseni.m	A\ jakost.m	0.35

Obr. 4.13: Srovnání výsledků obou detektorů při spuštění na stejné části databáze

Závěrem je možné konstatovat, že ač Codequiry skýtá mnoho výhod, jako například porovnávání s online dostupnými kódy, nebo detekci i pro jiné programovací jazyky než Matlab, jeho porovnávací algoritmus není ideální a detekce nedosahuje 100% senzitivity. Mezi další výhody můžeme zařadit velmi zdařilé grafické rozhraní, které se však z počátku může zdát neintuitivní. Významnou výhodou je také automatické ukládání všech výsledků detekce k pozdějšímu nahlédnutí, což vytvořený detektor neumožnuje. I toto však s sebou nese úskalí, protože Codequiry umožňuje zpracování pouze omezeného počtu dat. Omezení platí také pro vstupní soubory, kdy každý vstupní archiv musí mít velikost menší než 2 MB. Jako nevýhodu je třeba zmínit také finanční stránku, kdy měsíc využívání Codequiry stojí v přepočtu cca 700 Kč. K výhodám vytvořeného detektoru patří důkladné předzpracování, větší senzitivita, možnost zpracování libovolného množství dat a lepší přehled uživatele o datech díky zobrazení názvu adresáře u každého zpracovaného souboru.

Závěr

Cílem této bakalářské práce bylo vytvořit detektor podobností zdrojových kódů, založený na kombinování příznaků. Implementaci detektoru předcházelo nastudování problematiky a vytvoření teoretické rešerše, která tvoří první část práce.

První kapitola pojednává o plagiátorství obecně, zahrnuje definici, jeho druhy a pohled českého zákona na tuto problematiku. Ve druhé kapitole jsou nastíněny různé metody detekce. Tato část je velmi stručná z důvodu nedostatku volně přístupných dokumentací existujících detektorů. Dále kapitola popisuje některé významné existující detektory a nabízí jejich srovnání na základě několika vybraných parametrů.

Druhou část práce tvoří implementace vlastního detektoru v jazyce Python. Ve třetí kapitole je podrobně popsaná problematika předzpracování, které je pro zvolený způsob detekce stěžejní. Bez kvalitního předzpracování analyzovaných kódů by nebylo možno dosáhnout uspokojivých výsledků při detekci. Dále je zde popsán princip detekce příznaků i způsob, jakým se z detekovaných parametrů počítá celková podobnost dvojice kódů. Poslední kapitola podrobně popisuje grafické uživatelské rozhraní vytvořeného detektoru a funguje zároveň jako návod k použití. Druhá část kapitoly pak obsahuje statistické vyhodnocení výsledků detektoru.

Testování detektoru probíhalo na databázi studentských projektů. Detektor korektně odhalil všechny již známé plagiáty. Hodnota senzitivity tedy činí 100 %, a to jak při testování projektů rozdělených dle témat, tak i při testování na všech 1972 souborech najednou. Hodnota specificity byla nižší a lišila se pro různé okruhy testování. Velká míra falešně pozitivních detekcí však nebyla způsobena nedostatečností detektoru, nýbrž velkým množstvím velice podobných kódů, psaných stejným autorem. Pouze dvě falešně pozitivní detekce nastaly z jiného důvodu, a to kvůli velmi krátkým kódům. Kódy kratší než 4 řádky zpravidla po převedení na počty příznaků neobsahují dostatek informací ke spolehlivé detekci.

Kromě statistického testování bylo pro lepší představu o výhodách i nedostatcích detektoru provedeno srovnání s komerčním detektorem plagiátů Codequiry. Tento online nástroj předčil vytvořený detektor v grafickém zpracování, detailnosti rozboru i možnosti hledání podobných kódů v internetových databázích. Výhoda vytvořeného detektoru oproti Codequiry pak spočívá v precizním předzpracování kódů a vhodnějším algoritmu pro vyhledávání podobností – zatímco Codequiry využívá srovnávání řetězců, vytvořený detektor funguje na principu srovnávání příznaků, díky čemuž dokáže odhalit i zamaskované plagiáty. Tímto také dosahuje větší senzitivity.

Literatura

[1] Alex Aiken: MOSS.
URL https://theory.stanford.edu/~aiken/moss/

[2] Brandejsová, J.: Národní registr VŠKP a systém na odhalování plagiátů. *Čtenář*, ročník 61, č. 1, 2009: s. 3–7, ISSN 0011-2321.
URL https://svkkl.cz/ctenar/archiv/2009

[3] Codequiry, Inc: Codequiry pushes the boundaries of traditional code plagiarism checking. 2019.
URL https://codequiry.com/moss/measure-of-software-similarity

[4] Havlas, J.: *Předzpracování zdrojových kódů pro účely detekce plagiátů*. Bakalářská práce, Technická univerzita Ostrava, Fakulta elektrotechniky a informatiky, Katedra informatiky, Ostrava, 2015.

[5] Institute for Program Structures and Data Organization: JPlag.
URL http://jplag.ipd.kit.edu/

[6] Karásek, J.: *Citlivost metod pro měření podobnosti kvantitativních proměnných*. Diplomová práce, Vysoké učení technické v Brně, Brno, 2012.

[7] Packová, M.: *Komerční plagiátorství*. Diplomová práce, Masarykova univerzita, Filozofická fakulta, Ústav české literatury a knihovnictví, 2013.

[8] The MathWorks, Inc.: Add Comments to Programs.
URL
https://ch.mathworks.com/help/matlab/matlab_prog/comments.html

[9] Tiskárna ministerstva vnitra: Zákon o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon). 2000.

[10] University of Oxford: Plagiarism.
URL https://www.ox.ac.uk/students/academic/guidance/skills/plagiarism?wssl=1#

[11] Verco, K. L.; Wise, M. J.: Software for detecting suspected plagiarism. *Proceedings of the first Australasian conference on Computer science education - ACSE '96*, ročník 1, č. 1, 1996: s. 81–88, doi:10.1145/369585.369598.
URL http://portal.acm.org/citation.cfm?doid=369585.369598

[12] Zeidman, B.: What, Exactly, Is Software Plagiarism? *Intellectual Property Today*, ročník 13, č. 2, 2007: str. 17.
URL `https://www.zeidmanconsulting.com/documents/What,%20Exactly,%20Is%20Software%20Plagiarism.pdf`

[13] Černohlávková, K.: *Plagiátorství na vysokých školách*. Diplomová práce, Masarykova univerzita, Filozofická fakulta, Ústav české literatury a knihovnictví, Brno, 2008.

[14] Český normalizační institut: *Informace a dokumentace - Slovník*. Praha, druhé vydání, 2003.

Seznam symbolů, veličin a zkratek

FN	Falešně negativní výsledek detekce
FP	Falešně pozitivní výsledek detekce
GUI	Grafické uživatelské rozhraní (z angl. graphical user interface)
TN	Správně vyhodnocený negativní výsledek detekce
TP	Správně vyhodnocený pozitivní výsledek detekce

Seznam příloh

A Obsah elektronické přílohy

Elektronická příloha bakalářské práce obsahuje zdrojové kódy vytvořeného detektoru, soubory k ověření funkčnosti programu a elektronickou verzi bakalářské práce, která slouží zároveň jako dokumentace.

Ověřovací soubory jsou rozděleny do adresářů označených abecedně. Jde o výňatek z databáze studentských prací a každý adresář obsahuje jeden studentský projekt. Abecední značení bylo zvoleno za účelem anonymizace a zároveň zachování rozdělení do adresářů dle autorů.

Pro spuštění vytvořeného programu je třeba spustit skript s názvem *similarity_detection*.

```
/ ..... kořenový adresář elektronické přílohy
├── dokumentace
│   └── Barbora_Blahova_BP.pdf ..... elektronická verze bakalářské práce
├── ověřovací soubory ..... výňatek z testovací databáze k ověření funkčnosti
│   ├── A
│   ├── B
│   ├── C
│   ├── D
│   ├── E
│   └── F
└── zdrojové kódy ..... zdrojové kódy vytvořeného detektoru podobností
    ├── flags_detection.py
    ├── preprocessing.py
    ├── processing.py
    └── similarity_detection.py ..... skript pro spuštění programu
```