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 PODOBNOSTÍ V PROGRAMOVÝCH KÓDECH
DETECTION OF SIMILARITY IN PROGRAM CODES

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **Kristýna Maťašová**
AUTHOR

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

BRNO 2019

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní obor **Biomedicínská technika a bioinformatika**
Ústav biomedicínského inženýrství

Studentka: Kristýna Maťašová ***ID:*** 195192
Ročník: 3 ***Akademický rok:*** 2018/19

NÁZEV TÉMATU:

Detekce podobností v programových kódech

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 GUI . 3) Zvolte vhodné příznaky pro detekci podobnosti a v prostředí Matlab 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: 4.2.2019 ***Termín odevzdání:*** 24.5.2019

Vedoucí práce: Ing. Jakub Kašpar
Konzultant:

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

ABSTRAKT

Bakalářská práce seznamuje s pojmem plagiátorství a jeho možnými druhy. Zaměřuje se na problematiku detekce podobnosti zdrojových kódů, zejména u grafických rozhraní v prostředí *MATLAB*. Dále představuje již existující detektory. Praktická část práce se věnuje nalezení vhodných příznaků k detekci podobnosti ve zdrojových kódech *GUI* a seznamuje s navrženou metrikou detekovaných příznaků. Popisuje interní logiku vytvořeného detektoru podobností a diskutuje dosažené výsledky ověření funkčnosti detektoru.

KLÍČOVÁ SLOVA

detekce, GUI, GUIDE, MATLAB, plagiátorství, podonost, příznak, zdrojový kód

ABSTRACT

The Bachelor introduces the concept of plagiarism and possible kinds of plagiarism. It focuses on the problem of detecting the similarity of source codes, especially with graphical interfaces in the *MATLAB* environment. It also describes already existing detectors. The practical part of thesis is focused on finding appropriate flags for detection of similarity in source codes and introduces the metric of detected flags. It also describes the internal logic of created detector of similarity and discusses the results of its testing.

KEYWORDS

detection, GUI, GUIDE, MATLAB, plagiarism, similiarity, flag, source code

MAŤAŠOVÁ, Kristýna. *Detekce podobností v programových kódech*. Brno, 2019, 64 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 podobností v programových kódech“ 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 za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

Brno ....

podpis autorky

Obsah

Seznam obrázků

Seznam tabulek

Seznam výpisů

Úvod

V dnešní době snadné dostupnosti veškerých informací prostřednictvím internetu se zvyšuje i množství případů plagiátorství. S plagiátorstvím je možné se zejména setkat na akademické půdě, kdy se předmětem plagiátorství často stávají závěrečné studentské práce a další publikace. Neméně častým případem plagiátorství jsou plagiáty zdrojových kódů studentských projektů. Bakalářská práce se věnuje právě plagiátorství programových kódů, konkrétně podobnostmi v programových kódech grafických uživatelských rozhraní.

Hlavním cílem bakalářské práce je navrhnout a vytvořit funkční detektor podobností zdrojových kódů *GUI*, který by díky své automatizaci usnadnil porovnávaní zdrojových kódů a uživateli přehledně zobrazil dosažené výsledky detekce.

Zadáním bakalářské práce bylo nastudovat problematiku plagiátorství z pohledu podobnosti zdrojových kódů, zvolit vhodné příznaky pro detekci podobností a navrhnout metriku, na základě které bude realizovaný detektor podobností vyhodnocovat podobnosti kódů.

První kapitola bakalářské práce se věnuje obecné definici plagiátorství, popisuje existující formy plagiátorství a seznamuje se související právní problematikou spojenou s autorskými právy. Součástí kapitoly je i zmínka o podobě plagiátorství ve zdrojových kódech.

Navazující kapitola se zabývá problematikou detekce podobností v programových kódech. Popisuje obecný princip nalezení plagiátu zdrojového kódu a představuje známé softwary pro odhalování plagiátorství textových souborů i zdrojových kódů.

Třetí kapitola teoretické části přibližuje pojem grafické uživatelské rozhraní, uvádí tvorbu *GUI* v prostředí *MATLAB* a specifikuje odlišnou strukturu zdrojového kódu *GUI*.

Následuje kapitola zaměřená na praktickou realizaci detektoru podobností. V této kapitole jsou zmíněny realizované kroky pro předzpracování zdrojového kódu, uvedeny navržené příznaky pro detekci podobností a je přiblížen proces detekce definovaných příznaků. Dále se kapitola zabývá použitou metrikou detekovaných příznaků a principem určení výsledné podobnosti zdrojových kódů. Představuje grafické uživatelské rozhraní detektoru a popisuje jeho vlastnosti.

V poslední kapitole jsou diskutovány dasažené výsledky detektoru podobností na testovací databázi studentských prací. Je zhodnocena celková úspěšnost vytvořeného detektoru a jsou zmíněny nalezené podobnosti.

1 Plagiátorství

Plagiátorství [1] lze dle definice volně vyložit jako veškeré převzetí textu či myšlenky cizí osoby za vlastní či nepřesná citace původního autora nebo její úplné vynechání. Citace je nutná nejen u textů, ale i u veškerého doprovodného materiálu, který do vlastní práce přejímáme. Obecně, vydáváme-li za své něco, co jsme však sami nevytvořili, stává se to tzv. plagiátem. Norma ČSN ISO 5127-2003 [2] definuje plagiát jako „*představení duševního díla jiného autora půjčeného nebo napodobeného v celku nebo z části, jako svého vlastního*“.

Definice plagiátorství dle Wikipedie: „*Plagiátorství je neoprávněné přivlastnění, krádež a publikování cizích myšlenek, nápadů nebo výrazů a jejich reprezentace jako vlastních. Plagiátorství je považováno za akademickou nepoctivost a porušování novinářské etiky. Podléhá sankcím a dokonce i vyloučení ze školy nebo práce. Moderní pojetí plagiátorství jako nemorálního a originality jako ideálu se objevilo v Evropě v 18. století, zejména s romantickým hnutím. Plagiátorství není samo o sobě trestným činem, ale může představovat porušení autorských práv. V akademické a průmyslové sféře se jedná o závažný etický trestný čin*“ [3].

V dnešní době, která nám dává možnost dostupnosti mnoha elektronických zdrojů, knih a publikací, je téma plagiátorství velmi aktuální. Předmětem plagiátorství se často stávají vědecké práce, eseje, odborné texty a v neposlední řadě zdrojové kódy školních projektů.

1.1 Autorská práva

Plagiátorství není jen nemorální, jde také o porušení autorského práva [4]. Příčinou úmyslného či neúmyslného plagiátu je většinou tvorba závěrečných prací, jedná se však o závažný přestupek akademické etiky. Studenta může postihnout napomenutí nebo až podmíněné či úplné ukončení studia. Proto se především vysoké školy zaměřují na metody, které plagiátorství odhalují. Veškerá vytvořená díla jsou předmětem autorskoprávní ochrany a jejich autor je odpovědný za to, že se při jejich tvorbě nedopustí plagiátorství.

Obecně je platné, že fakta chráněna autorskými právy nejsou, tudíž je autor ve své práci smí použít. Slova, která fakta vyjadřují, však chráněna autorskými právy být mohou, zvláště jedná-li se o originální formulaci. Informace z různých materiálů je možné využívat, ale vždy je nutné je formulovat vlastními slovy.

Otázce ochrany autorských práv se věnuje zákon č. 121/2000 Sb. „*Zákon o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon)*“ [5]. Zmíněný zákon definuje předmětem autorského práva autorské dílo, jehož autorem je fyzická osoba, která dílo vytvořila. Autorským dílem může

být dílo literární, umělecké či vědecké, jakožto výsledek činnosti autora. Dílo má podobu písemnou, hudební, fotografickou, výtvarnou atp. Za dílo je považován také počítačový program. Autorské právo se vztahuje na dílo dokončené i jeho jednotlivé fáze a části.

Dle §31 Autorského zákona musíme vždy při použití cizí myšlenky uvést její zdroj. Především je nutné uvést veškeré použité zdroje, aby bylo možné následné dohledání převzatých částí v původním textu. Plagiátorství a porušování autorských práv se do značné míry překrývají, ale nejsou to obdobné pojmy a mnoho druhů plagiátorství nepředstavuje porušení autorských práv. Plagiátorství není definováno ani potrestáno zákonem, ale institucemi (včetně profesních sdružení, vzdělávacích institucí a obchodních subjektů, jako jsou vydavatelské společnosti) [3].

Autorským zákonem je pro ochranu programů vyžadováno současné splnění dvou podmínek:

- **původnost** (originalita) – lze vytvořit vlastní program s podobnými funkcionalitami a podobným zdrojovým kódem, nelze však zkopírovat zdrojový kód,
- **objektivní vnímatelnost** – vyjádření zdrojovým kódem.

Software je chráněn jako celek, ale jsou chráněny i jeho části, například data uspořádaná do podoby databáze, vzhled, název. Osobnostní autorská práva jsou fixována na osobnost autora softwaru, který se jich nemůže vzdát, ani je nemůže převést na jinou osobu. Mezi osobní autorská práva patří právo na zveřejnění, uvedení softwaru na trh formou licence, být označen jako autor, právo na udělení souhlasu se zásahem do zdrojového kódu [6].

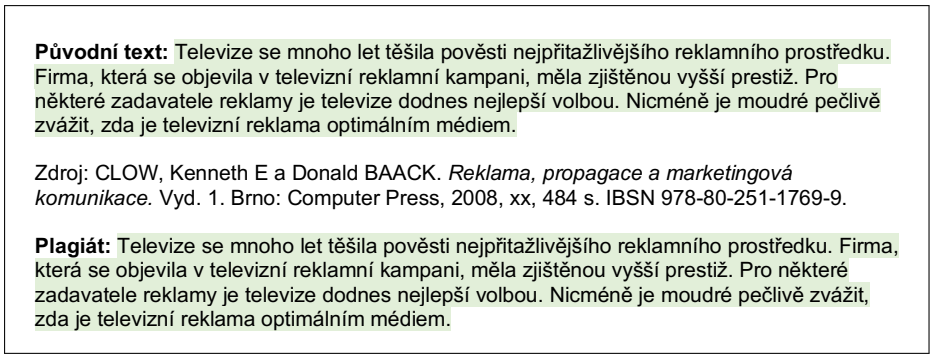
1.2 Formy plagiátorství

Plagiátorství můžeme nejjednodušeji rozdělit na dvě základní skupiny, na plagiátorství úmyslné a neúmyslné. Neúmyslného plagiátorství se obvykle autor dopustí tím, že nesprávně či nepřesně uvede citaci nebo považuje zmíněnou myšlenku za vlastní. Je tedy důležité uvést co nejpřesněji odkud převzaté texty pochází. Samotný systém citací je definován tzv. **citační normou**. Jednotlivé formy citací můžeme definovat jako citát, kdy se jedná o doslovné převzetí myšlenky z cizího textu, např. při použití přesných definicí, parafrázi, což je vyjádření cizí myšlenky vlastními slovy. Bezprostředně za citátem či parafrází uvádíme odkaz na zdroj v seznamu použité literatury. Samotná bibliografická citace je určitý záznam v soupisu použité literatury, uvádí autora, název, vydavatelství a další konkrétní údaje.

Úmyslné neboli vědomé plagiátorství je obecně považováno za podvod. Zdroj je v tomto případě neuveden zcela záměrně. Cizí práce může být i zakoupena a vydávaná za vlastní, což je však v naprostém rozporu s etikou.

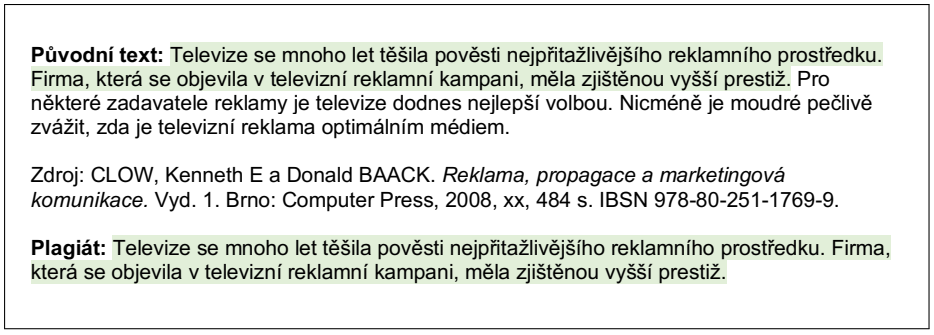
Podrobné rozdělení forem plagiátorství a jejich příčiny [7]:

- **Klonování** (obr. 1.1) – doslovné "okopírování" cizího dokumentu a jeho vydávání za vlastní je nejzávažnější formou plagiátorství. S touto formou plagiátorství je možné se setkat u žáků základních či středních škol, kdy žáci kopírují celé texty z dostupných internetových zdrojů (např. Wikipedia), u studentů vysokých škol je klonování omezené z důvodu využívání antiplagiátorských systémů. Existují také služby umožňující přímo zakoupení vypracovaného dokumentu, což lze považovat za podvod s možnými právními dopady.



Obr. 1.1: Příklad formy plagiátorství – klonování.

- **Kopírování** (obr. 1.2) – jedná se o podobnou formu plagiátorství jakou je zmíněné klonování, kdy doslovně přebíráme pasáže z textu bez uvádění citace, ovšem často se kopírování vyskytuje spolu s další formou plagiátorství (spojování, drobné úpravy). Opět se jedná o závažný druh plagiátorství.



Obr. 1.2: Příklad formy plagiátorství – kopírování.

- **Drobné úpravy** (obr. 1.3) – drobnými úpravami je myšlena například změna slovosledu převzatého textu, záměna některých slov jejich synonymy, vynechání zbytečných slov, atd. Pokud není uveden původní zdroj, lze vytvořený text rovněž považovat za plagiát, protože se jedná o plagiátorství samotné myšlenky, které však antiplagiátorské systémy mnohdy neodhalí.

Původní text: Televize se mnoho let těšila pověsti nejpřitažlivějšího reklamního prostředku. Firma, která se objevila v televizní reklamní kampani, měla zjištěnou vyšší prestiž. Pro některé zadavatele reklamy je televize dodnes nejlepší volbou. Nicméně je moudré pečlivě zvážit, zda je televizní reklama optimálním médiem.

Zdroj: CLOW, Kenneth E a Donald BAACK. *Reklama, propagace a marketingová komunikace*. Vyd. 1. Brno: Computer Press, 2008, xx, 484 s. IBSN 978-80-251-1769-9.

Plagiát: Televize se spoustu let těšila reputaci nejpřitažlivějšího reklamního nástroje. Firma, která se objevila v televizní reklamní kampani, měla zjištěnou vyšší prestiž. Pro některé zadavatele reklamy je televize doposud nejideálnější volbou. Nicméně je rozumné pečlivě zvážit, zda je televizní reklama vhodným médiem.

Obr. 1.3: Příklad formy plagiátorství – drobné úpravy.

- **Jeden zdroj** (obr. 1.4) – cílem odborné práce je prostudování více zdrojů literatury a výsledkem pak nový pohled na danou problematiku. Nedostatek času potřebného pro nastudování dostatečného množství zdrojů je obvykle příčinou toho, že autor textu čerpá pouze z jednoho jediného zdroje, čímž je jeho obsah poněkud omezený. V případě omezení se pouze na jeden zdroj obsahuje práce především kopírované části původního textu doplněné o vlastní komentáře. Citace je obvykle neuvedena nebo je neúplná.

Původní text: Televize se mnoho let těšila pověsti nejpřitažlivějšího reklamního prostředku. Firma, která se objevila v televizní reklamní kampani, měla zjištěnou vyšší prestiž. Pro některé zadavatele reklamy je televize dodnes nejlepší volbou. Nicméně je moudré pečlivě zvážit, zda je televizní reklama optimálním médiem.

Zdroj: CLOW, Kenneth E a Donald BAACK. *Reklama, propagace a marketingová komunikace*. Vyd. 1. Brno: Computer Press, 2008, xx, 484 s. IBSN 978-80-251-1769-9.

Plagiát: Televize se mnoho let těšila pověsti nejpřitažlivějšího reklamního prostředku. Díky ní mohly společnosti lépe působit na emoce spotřebitelů formou příběhů. Firma, která se objevila v televizní reklamní kampani, měla zjištěnou vyšší prestiž. Pro některé zadavatele reklamy je televize dodnes nejlepší volbou. Nicméně je moudré pečlivě zvážit, zda je televizní reklama optimálním médiem.

Obr. 1.4: Příklad formy plagiátorství – jeden zdroj.

- **Spojování textů** (obr. 1.5) – části textů vybraných zdrojů jsou autorem kopírovány a spojovány dohromady, chybí tedy vlastní přínos autora. Nový text vznikl pouze spojením určitých částí původních textů. Většinou je tato forma plagiátorství spojena s drobnými úpravami zmíněnými výše a některé pasáže nebývají citovány [8].

Původní text č.1: Televize se mnoho let těšila pověsti nejpřitažlivějšího reklamního prostředku. Firma, která se objevila v televizní reklamní kampani, měla zjištěnou vyšší prestiž. Pro některé zadavatele reklamy je televize dodnes nejlepší volbou. Nicméně je moudré pečlivě zvážit, zda je televizní reklama optimálním médiem.

Zdroj: CLOW, Kenneth E a Donald BAACK. *Reklama, propagace a marketingová komunikace.* Vyd. 1. Brno: Computer Press, 2008, xx, 484 s. IBSN 978-80-251-1769-9.

Původní text č.2.: Od roku 1997 je i v České republice realizován projekt „elektronické měření sledovanosti televize metodou peoplemetrů", jehož cílem poskytovat údaje sledovanosti televize pro televizní stanice, reklamní agentury a zadavatele reklamy.

Zdroj: VYSEKALOVÁ, Jitka a Donald BAACK. *Psychologie reklamy.* 4., rozš. a aktualiz. vyd. Praha: Grada, 2012, 324 s. Expert (Grada). IBSN 978-80-247-4005-8.

Původní text č.3.: Televizní reklamu lze rozdělit do následujících kategorií z hlediska přenosových možností: mezinárodní síť, národní televize, lokální, kabelová a internetová.

Zdroj: PŘIKRYLOVÁ, Jana a Hana JAHODOVÁ. *Moderní marketingová komunikace.* 1. vyd. Praha: Grada, 2010, 303 s., [16] s. obr. příl. Expert (Grada). ISBN 978-80-247-3622-8.

Plagiát: Televize se mnoho let těšila pověsti nejpřitažlivějšího reklamního prostředku. Firma, která se objevila v televizní reklamní kampani, měla zjištěnou vyšší prestiž. Pro některé zadavatele reklamy je televize dodnes nejlepší volbou.

Televizní reklamu lze rozdělit do následujících kategorií z hlediska přenosových možností: mezinárodní síť, národní televize, lokální, kabelová a internetová.

Od roku 1997 je i v České republice realizován projekt „elektronické měření sledovanosti televize metodou peoplemetrů", jehož cílem poskytovat údaje sledovanosti televize pro televizní stanice, reklamní agentury a zadavatele reklamy.

Obr. 1.5: Příklad formy plagiátorství – spojování textů.

- **Nedohledatelný zdroj** (obr.1.6) – v tomto případě se jedná o odkazování se na zdroj, který však neexistuje. Zdroj si buď autor úplně vymyslí, nebo vznikne neexistující zdroj překlepem v jeho názvu, uvedení špatného roku vydání apod.

Správná citace:
CLOW, Kenneth E a Donald BAACK. *Reklama, propagace a marketingová komunikace*. Vyd. 1. Brno: Computer Press, 2008, xx, 484 s. ISBN 978-80-251-1769-9.

Neexistující zdroj:
CLOW, Kenneth E. *Reklama a marketingová komunikace*. Vyd. 1. Brno: Computer Press, 2008.

Obr. 1.6: Příklad formy plagiátorství – nedohledatelný zdroj.

- **Vylepšování literatury** (obr. 1.7) – za formu plagiátorství je také považováno zmínění zdrojů, které jsme reálně při vytváření práce nepoužili. Autor tak poukazuje na malé množství použitých zdrojů nebo jejich nedostatečnou kvalitu.

Původní seznam literatury:
CLOW, Kenneth E a Donald BAACK. *Reklama, propagace a marketingová komunikace*. Vyd. 1. Brno: Computer Press, 2008, xx, 484 s. ISBN 978-80-251-1769-9.

Vylepšený seznam literatury:
CLOW, Kenneth E a Donald BAACK. *Reklama, propagace a marketingová komunikace*. Vyd. 1. Brno: Computer Press, 2008, xx, 484 s. ISBN 978-80-251-1769-9.

VYSEKALOVÁ, Jitka a Donald BAACK. *Psychologie reklamy.* 4., rozš. a aktualiz. vyd. Praha: Grada, 2012, 324 s. Expert (Grada). IBSN 978-80-247-4005-8.

PŘIKRYLOVÁ, Jana a Hana JAHODOVÁ. *Moderní marketingová komunikace.* 1. vyd. Praha: Grada, 2010, 303 s., [16] s. obr. příl. Expert (Grada). ISBN 978-80-247-3622-8.

Obr. 1.7: Příklad formy plagiátorství – vylepšování literatury.

- **Necitování v textu** (obr. 1.8) – častým prohřeškem autorů odborných prací bývá nezmínění citací přímo v textu. Autor odborné práce pouze předkládá seznam použité literatury, avšak konkrétní převzaté myšlenky jsou v textu nedohledatelné. Je proto nutné uvádět citace přímo v textu.

Původní text: Televize se mnoho let těšila pověsti nejpřitažlivějšího reklamního prostředku. Firma, která se objevila v televizní reklamní kampani, měla zjištěnou vyšší prestiž. Pro některé zadavatele reklamy je televize dodnes nejlepší volbou. Nicméně je moudré pečlivě zvážit, zda je televizní reklama optimálním médiem.
Nicméně je moudré pečlivě zvážit, zda je televizní reklama optimálním médiem. [1]

[1] CLOW, Kenneth E a Donald BAACK. *Reklama, propagace a marketingová komunikace.* Vyd. 1. Brno: Computer Press, 2008, xx, 484 s. IBSN 978-80-251-1769-9.

Plagiát: Televize se mnoho let těšila pověsti nejpřitažlivějšího reklamního prostředku. Firma, která se objevila v televizní reklamní kampani, měla zjištěnou vyšší prestiž. Pro některé zadavatele reklamy je televize dodnes nejlepší volbou. Nicméně je moudré pečlivě zvážit, zda je televizní reklama optimálním médiem.

CLOW, Kenneth E a Donald BAACK. *Reklama, propagace a marketingová komunikace*. Vyd. 1. Brno: Computer Press, 2008, xx, 484 s. IBSN 978-80-251-1769-9.

Obr. 1.8: Příklad formy plagiátorství – necitování v textu.

- **Autoplagiátorství** – poněkud odlišnou formou plagiátorství je tzv. autoplagiátorství neboli duplicitní publikace vlastního díla, kdy autor používá své vlastní myšlenky z jeho dřívější práce a neuvádí řádnou citaci. V tomto případě nejde o porušení autorských práv, ale uvedení citace je i v tomto případě nutné.

1.3 Problematika plagiátorství zdrojových kódů

Zdrojový kód je posloupnost příkazů pro funkci počítačového programu. Jedná se o zápis počítačového programu prostřednictvím textového editoru. Jednotlivé příkazy mohou být doplněny patřičnými komentáři objasňující jejich funkci. Tvorbu zdrojového kódu usnadňuje využití vývojového prostředí.

S plagiátorstvím se setkáváme i v oblasti programových kódů. Opět se může jednat o převzetí cizí myšlenky, v tomto případě celkové funkčnosti programu, či doslovné zkopírování zdrojového kódu nebo jeho částí.

Dvojice Parker a Hamblen ve své publikaci [9] definují plagiátorství v programovém kódu jako „*program, který byl vytvořen z jiného programu s nízkým počtem běžných transformací*“. Pánové Faidhi a Robinson ve své práci [10] rozřazují možné transformace na úrovně od 1. do 6. - žádné změny v kódu odpovídají první úrovni, nejsložitější změny pak šesté úrovni.

Změny v kódu programu můžeme obecně rozdělit do dvou kategorií [11]:

- **Lexikální změny** – úpravy, které lze provádět bez znalosti programovacího jazyka. Tyto změny mohou zahrnovat přidání nebo odstranění komentářů, změnu názvů proměnných (přibližně úroveň 1 – 3 dle Faidhi a Robinsona).
- **Strukturální změny** – aby bylo možné změnit strukturu programu tak, aby se nezměnila jeho funkčnost, je již zapotřebí určitá znalost programovacího jazyka. Změny mohou zahrnovat např. nahrazení ekvivalentních iteračních struktur nebo uspořádání operátorů (přibližně úroveň 4 – 6 dle Faidhi a Robinsona).

Autor plagiátorského programu většinou volí úpravu cizího zdrojového kódu, například záměnu názvů proměnných a funkcí, záměnu pořadí řádků kódu, odmazání či naopak připsání komentářů kódu apod. Zmíněné úpravy jsou relativně jednoduché, student tak může snadno a rychle změnit podobu kódu, aniž by změnil jeho funkčnost. Ze složitějších úprav může být zmíněna např. záměna cyklů `while` za `for`.

Obecně se k tomuto jednání často ubírají studenti z důvodu časové tísně, vlastní negramotnosti v programování či neschopnosti vymyslet vlastní logiku daného programu. Při stejném či podobném zadání úlohy však může vzniknout podobnost i zcela náhodně, studenti mohou být během výuky vedeni k jednotnému stylu vytváření kódu, používaným funkcím, názvů proměnných, atd. Rozdíl mezi úmyslným plagiátem a náhodně podobným kódem může být tedy v tomto případě velmi malý.

2 Detekce podobností v programových kódech

Odhalování plagiátorství v programových kódech je aktuálním tématem nejen vysokých škol. Porovnáváním dvou zdrojových kódů jsme schopni určit jejich vzájemnou podobnost.

Nejdříve je nutné si uvědomit skutečnost, jak může k vytváření podobností ve zdrojových kódech docházet a následně toho využít pro jejich účinné odhalování. Porovnávání zdrojových kódů může být z hlediska menšího počtu a specifičnosti slov snazší, než u textových dokumentů, protože každé klíčové slovo programového kódu má svůj přesně daný význam, což slova textového dokumentu často nesplňují.

Obecně je jednou z možností porovnávání zdrojových kódů jako celků, což je nejen časově náročné, ale v mnoha případech, kdy dojde například k záměně pořadí řádků, poměrně nespolehlivé. Před samotnou detekcí plagiátu je nutné definovat jednotlivé příznaky, které je následně možné vzájemně porovnat a každému z nich přiřadit určitou váhu pro celkové vyhodnocení plagiátu. Na základě zvolených příznaků je poté možné přesněji určit, zda-li se skutečně jedná o plagiát.

2.1 Příznaky pro detekci

Jedná se o charakteristické parametry zdrojového kódu, které i po jeho možných úpravách mohou přetrvávat nebo se alespoň podobat. K takovým příznakům řadíme například:

- celkový počet řádků kódu,
- počet znaků,
- délky řádků,
- počet proměnných
- a počet funkcí.

Velmi konkrétními příznaky pro zdrojové kódy jsou tzv. **klíčová slova**, jejichž vybraný přehled (obr. 2.1) lze zobrazit přímo ve vývojovém prostředí *MATLAB* prostřednictvím existujícího příkazu `iskeyword`. Jedná se o funkci, která na výstupu vrací logickou hodnotu *1* v případě, kdy je vstupní proměnná klíčovým slovem *MATLAB*, v opačném případě je výstupní hodnota funkce rovna *0*. Při vyvolání zmíněné funkce bez vstupní proměnné získáme buňkové pole klíčových slov definovaných v prostředí *MATLAB*.

Klíčová slova jsou vždy typická pro použitý programovací jazyk. Pro detekci podobnosti kódů je významný zejména jejich celkový počet v kódu.

```
20×1 cell array

  {'break'     }
  {'case'      }
  {'catch'     }
  {'classdef'  }
  {'continue'  }
  {'else'      }
  {'elseif'    }
  {'end'       }
  {'for'       }
  {'function'  }
  {'global'    }
  {'if'        }
  {'otherwise' }
  {'parfor'    }
  {'persistent'}
  {'return'    }
  {'spmd'      }
  {'switch'    }
  {'try'       }
  {'while'     }
```

Obr. 2.1: Přehled klíčových slov *MATLAB*.

Jelikož jsou klíčová slova většinou zásadní pro správnou funkčnost programu, málokdy se jejich celkový počet po úpravách liší. S rostoucím počtem hledaných klíčových slov a jejich odlišností roste samozřejmě i úspěšnost následné detekce plagiátu. Pro přesnější určení podobnosti je vhodné brát v úvahu také pozici klíčových slov v celém kódu nebo délku řádku, na kterém se vyskytují.

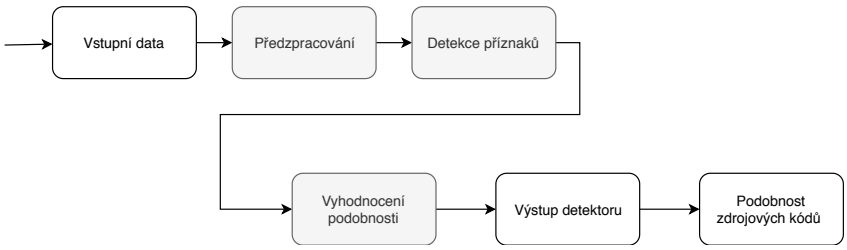
Pro detekci plagiátorství lze vymyslet celou řadu příznaků, pomocí kterých bude při jejich vhodné kombinaci úspěšně odhalována vzájemná podobnost zdrojových kódů. Mezi zvolené příznaky můžeme řadit a porovnávat tak například i názvy funkcí, názvy proměnných, jejich pozice v kódu i jejich celkový počet apod.

2.2 Princip detekce

Detekci podobnosti zdrojových kódu lze rozdělit celkově do tří dílčích částí:

- předzpracování,
- detekce příznaků,
- vyhodnocení plagiátorství.

Jednotlivé části detekce jsou znázorněny na blokovém schématu 2.2.



Obr. 2.2: Blokové schéma principu detekce plagiátu.

Předzpracování

Před samotným porovnáváním programových kódů je nutné jejich předzpracování. Bez předzpracování by byla detekce podobnosti z důvodu rozdílného formátování velmi obtížná. Pro dosažení stejné podoby kódů, co se týče formátování, jsou nutné následující kroky:

- odstranění komentářů,
- odstranění prázdných řádků,
- odstranění mezer mezi slovy, před slovy a za slovy,
- změna velkých písmen na malá.

Pro zabránění následné falešně pozitivní detekce plagiátu mohou být v rámci předzpracování zdrojových kódů odstraněny i automaticky generované části kódu, což je konkrétně významné u zdrojových kódu *GUI*.

```
1  function [BMI] = vypocet_BMI(hmotnost,vyska)
2  %Funcke pro vypocet BMI
3  %   BMI - Index telesne hmotnosti, cislo pouzivane jako
       indikator podvahy, normalni telesne hmotnosti, nadvahy
        a obezity. Umoznuje stasticky porovnat telesne
       hmotnosti lidi s ruznou vyskou.Vysledna hodnota musi
       byt interpretovana v souvislosti s vekem a pohlavim.
4  %   vystupni hodnota - BMI (index telesne hmotnosti)
5  %   vstupni promenne: hmotnost (kg), vyska (cm)
6
7  if vyska==0 %Pokud je jedna ze vstupnich hodnot nulova
8      BMI=0; %Vysledek je take nulovy
9  elseif hmotnost==0 %Pokud je jedna ze vstupnich hodnot
       nulova
10     BMI=0; %Vysledek je take nulovy
11 else
12     vyska=vyska/100; %Prevedeni hodnoty vysky na metry
13     BMI = hmotnost/(vyska*vyska); %Vzorec pro vypocet BMI
14 %   vysledna hodnota se pohybuje v rozmezi přibližně
       16-40 kg/m^2
15 end
16 end
```

Výpis 2.1: Ukázka 1. kódu před předzpracováním.

```
function [Body_Mass_Index] = vypocet_BMI(weight,high)
%funkce pro vypocet Body Mass Index

%   vystupni promenna - Body_Mass_Index,

%   vstupni promenne: weight, high

if high==0

   Body_Mass_Index=0;

elseif weight==0

   Body_Mass_Index=0;

else
   %prevod vysky na m

   high=high/100;

   %vysledek BMI

   Body_Mass_Index = weight/(high*high);

end

end
```

Výpis 2.2: Ukázka 2. kódu před předzpracováním.

První zdrojový kód je z pohledu počtu komentářů značně rozsáhlejší. Ve druhém kódu se navíc vyskytují přebytečné mezery. Po procesu předzpracování (obr. 2.3 a obr. 2.4) je však na první pohled zřejmá jejich vzájemná podobnost.

Ve výsledku představují oba zdrojové kódy 10 významných řádků, jejich struktura je tedy identická. Odlišností jsou pouze použité názvy proměnných.

```
10×1 cell array

  {'function[BMI]=vypocet_BMI(hmotnost,vyska)'}
  {'ifvyska==0'                                }
  {'BMI=0;'                                    }
  {'elseifhmotnost==0'                         }
  {'BMI=0;'                                    }
  {'else'                                      }
  {'vyska=vyska/100;'                          }
  {'BMI=hmotnost/(vyska*vyska);'               }
  {'end'                                       }
  {'end'                                       }
```

Obr. 2.3: První vzorový kód po předzpracování.

```
10×1 cell array

  {'function[Body_Mass_Index]=vypocet_BMI(weight,high)'}
  {'ifhigh==0'                                         }
  {'Body_Mass_Index=0;'                                }
  {'elseifweight==0'                                   }
  {'Body_Mass_Index=0;'                                }
  {'else'                                              }
  {'high=high/100;'                                    }
  {'Body_Mass_Index=weight/(high*high);'               }
  {'end'                                               }
  {'end'                                               }
```

Obr. 2.4: Druhý vzorový kód po předzpracování.

Detekce příznaků

Po předzpracování porovnávaných zdrojových kódů lze přistoupit k nalezení všech zvolených příznaků. Pro každý kód získáme výčet hodnot, jako je například počet výskytů klíčových slov a pozice řádků v daném kódu, na kterých se nachází, celkový počet znaků, počet použitých funkcí i počet proměnných a jejich názvy.

Vyhodnocení plagiátorství

Pro závěrečné vyhodnocení plagiátorství je zvolena kombinace vhodných příznaků a je využito dosažených výsledků detekce těchto příznaků v porovnávaných kódech. Zvolením vhodné metriky je vyhodnocena míra podobnosti posuzovaných zdrojových kódů.

2.3 Existující detektory

S rostoucím výskytem případů plagiátorství roste i množství a efektivita dostupných detektorů. Moderní programy pro detekci plagiátorství zdrojového kódu jsou stále více sofistikované. Jedná se o detektory plagiátorství v textových souborech i o detektory pro odhalování plagiátorství ve zdrojových kódech. Využití nalézají zejména na vysokých školách, například pro vyhodnocení podobnosti závěrečných prací.

Metody, které detektory využívají k vyhledávání podobností ve zdrojových kódech, lze rozdělit do několika typů [12], např.:

- měření strukturní, algoritmické a metrické podobnosti,
- využití stylu psaní kódu autora,
- využití tokenizace
- a normalizace.

Nejvíce používanými jsou typy metod, které měří strukturní, algoritmické a metrické podobnosti v kódu. Tyto metody vykazují vysokou míru přesnosti, nicméně zjištěná podobnost však nemusí být vždy jednoznačně plagiátorstvím.

V důsledku toho jsou navrhovány metody, které pro odhalování plagiátorství využívají i kódovací styl autora (způsob, jakým zapisuje zdrojový kód) a umožňují tak snížit falešně pozitivní detekce plagiátorství.

Další typ metod využívá „tokenizaci“ a normalizaci kódu. Token je popisován jako nejzákladnější jednotka jazyka, jedná se o jeden znak reprezentující každý programově významný kus kódu viz tab. 2.1. Tím se zkracuje délka programu, zatímco struktura programu zůstává zachována.

Významný kus kódu	**Token**	**Významný kus kódu**	**Token**
`for`	R	(	A
`main`	N	)	B
`int`	S	<	J
`return`	g	{	j
=	K	}	l
+	D	.	E
`ALPHANUM`	N	`STRING`	5

Tab. 2.1: Příklad tokenizace zdrojového kódu pro programovací jazyk *C* [13].

Výsledný řetězec tokenů dle tab. 2.1 může pro krátký zdrojový kód v jazyce *C* vypadat například následovně: `SNABjSNRANKNJNNDDBjNA5ENBlgNl`. V tabulce jsou rovněž uvedeny specifické zástupné funkce pro dané části kódu:

- `STRING` – reprezentuje sekvenci znaků ohraničenou uvozovkami,

- `ALPHANUM` – reprezentuje název funkce, název a hodnotu proměnné.

Normalizace neboli předzpracování je série úkonů, při nichž jsou z kódu odstraněny komentáře, nadbytečné mezery apod.

MOSS

MOSS (*Measure Of Software Similarity*) [14] je jedním z nejznámějších a nejpoužívanějších nástrojů pro odhalování podobnosti. Byl vyvinutý na Kalifornské univerzitě v Berkeley Alexem Aikenem v roce 1994. Jedná se jednoduchou internetovou službu. Uživatel poskytne soubory, které chce porovnat a samotné porovnání proběhne na straně serveru *MOSS*. Služba je dostupná zcela zdarma, nicméně pouze pro nekomerční použití. Hlavní nevýhodou detektoru *MOSS* je absence grafického uživatelského prostředí a nahrávání souborů na server prostřednictvím skriptu. Další nevýhodou nástroje *MOSS* je nemožnost porovnat více zdrojových kódů navzájem.

MOSS podporuje analýzu kódů řady programovacích jazyků, jako je například jazyk *C*, *C++*, *Java*, *C#*, *Javascript*, *MATLAB*, *Verilog*, *Perl* a mnoho dalších.

Detekce probíhá ve dvou fázích, v první fázi je analyzovaný zdrojový kód změněn na řadu tokenů. Jeden token může například reprezentovat název proměnné, student může použít jiný název pro odkazování se na tuto proměnnou. *MOSS* pouze potřebuje vědět, že v daném bodě existuje proměnná. Vlastnosti, jako je řádkování, odsazení a komentáře, se nezohledňují. Ve druhé fázi jsou porovnány tokenizované verze všech porovnávaných zdrojových kódů.

Moss Results

Sun Mar 14 15:24:02 PST 1999

Options -l c -m 10

[Text Report | How to Read the Results | Tips | FAQ | Contact Moss | Submission Scripts | Credits]

File 1	File 2	Tokens Matched	Lines Matched
mike_wolf.c (79%)	mike_fox.c (80%)	463	139
bill_smyth.c (86%)	bill_smith.c (88%)	456	133
jane_white.c (59%)	jane_blanco.c (68%)	354	111
john_doe.c (100%)	john_deer.c (100%)	220	49

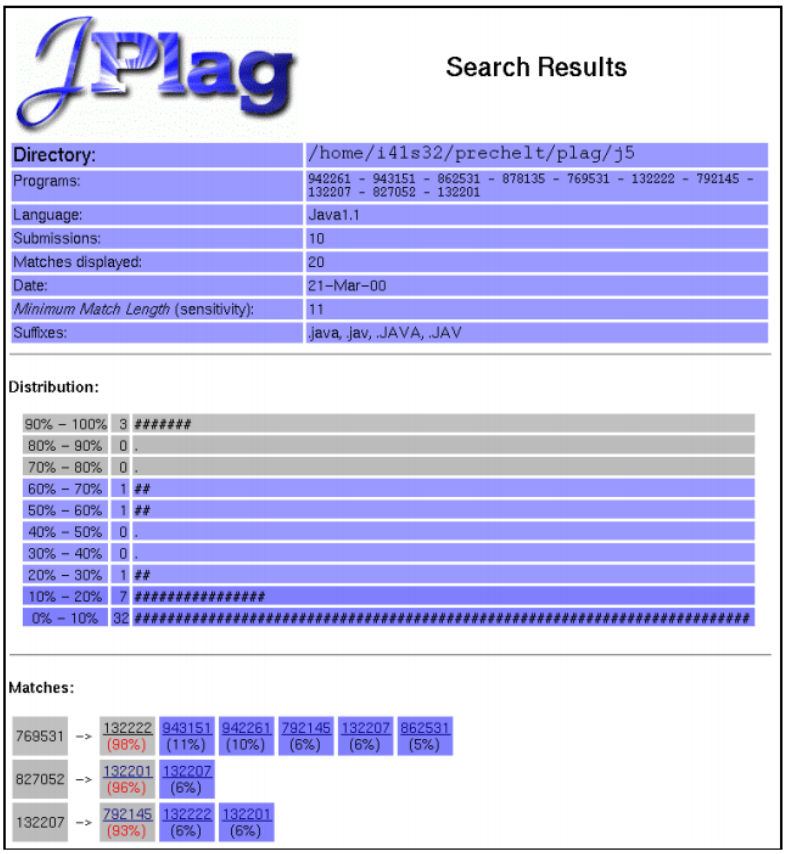
Any errors encountered during this query are listed below.

Obr. 2.5: Typický výstup serveru *MOSS* [15].

JPlag

JPlag [16] začal svůj vývoj v roce 1996. Program napsali Lutz Prechelt a Guido Malpohl z univerzity v Karlsruhe a Michael Philippsen z univerzity Erlangen-Norimberk. Jedná se o program implementovaný v jazyku Java, který podporuje programovací jazyky *C*, *C++*, *C#*, *Java* a *Scheme*, může sloužit i pro porovnání podobnosti obyčejných textových souborů.

Pro usnadnění používání detektoru je dostupná i webová služba. Program je svým algoritmem podobný detektorům *YAP3* a *MOSS*. Výhodou detektoru *JPlag* je grafické uživatelské prostředí, detektor se díky němu stává uživatelsky přívětivější než např. *MOSS*, také jeho výsledky jsou snadnější pro pochopení.



Obr. 2.6: Typický výstup detektoru *JPlag* [17].

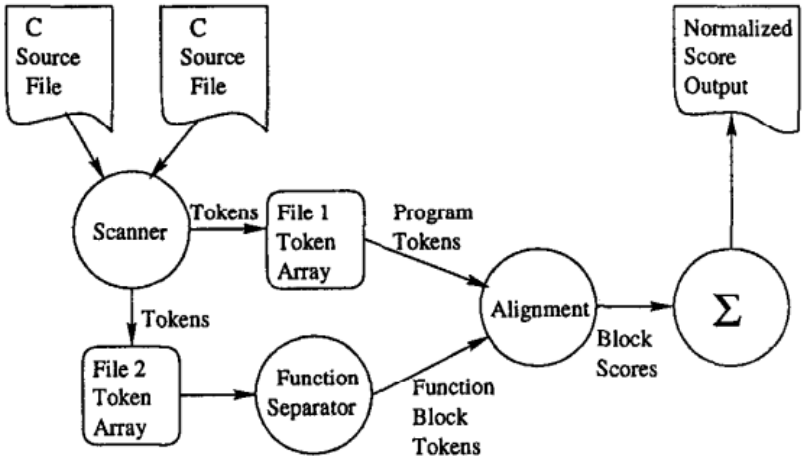
YAP3

Jedná se o nejnovější verzi programu *YAP* [18] pro detekci plagiátů, který byl vyvinutý Michaelem Wiseem na univerzitě v Austrálii. Program detekuje plagiátorství ve dvou fázích. Před zahájením detekce se text pročistí – komentáře jsou odebrány a znaky jsou přeměněny na malá písmena.

Následuje první fáze detekce, ve které jsou ze zdrojového kódu generovány sekvence tokenů. Funkce jsou převedeny na jejich základní ekvivalenty (například funkce `strncmp` na `strcmp`), poté je text ztokenizován a v druhé fázi algoritmu porovnán a vyhodnocen.

SIM

SIM [18] byl vyvinut Dickem Grunem z univerzity Vrije v Nizozemsku. Tento detekční systém plagiátů je veřejně dostupný. *SIM* detekuje podobnost v textových souborech i v programových kódech jazyků *C*, *C++*, *Java*, *Pascal*, *Modula-2*, *Miranda*, *Lisp* a *8086 assembler*. Jedná se o komplexní detektor, který dokáže detekovat i změny názvů metod, přeuspořádání pořadí příkazů a funkcí v kódu.

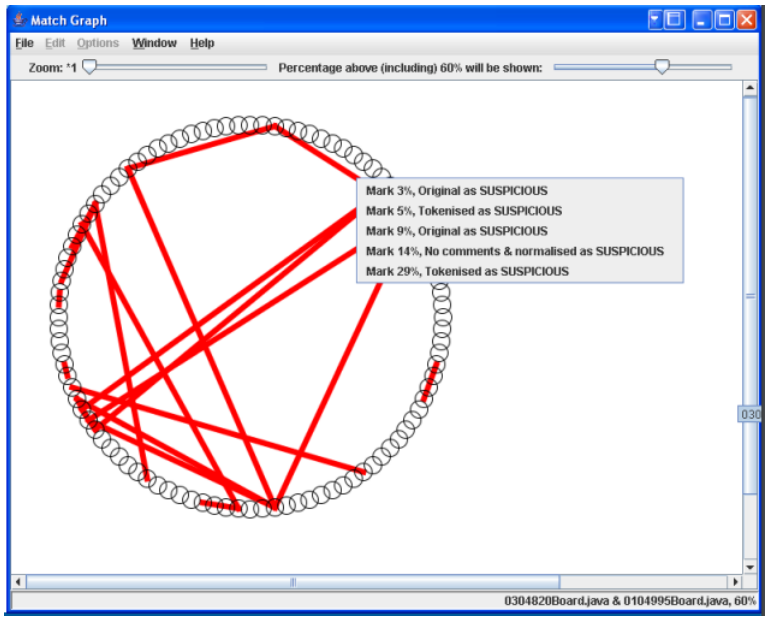


Obr. 2.7: Blokový diagram funkčnosti detektoru *SIM* [19].

Sherlock

Sherlock [20] byl vyvinutý na katedře informatiky na univerzitě Warwick pro detekci plagiátorství zdrojových kódů i textových dokumentů. Jedná se o další systém z řady detektorů založených porovnávání struktury kódu (jako jsou *JPlag*, *MOSS*, *YAP3*).

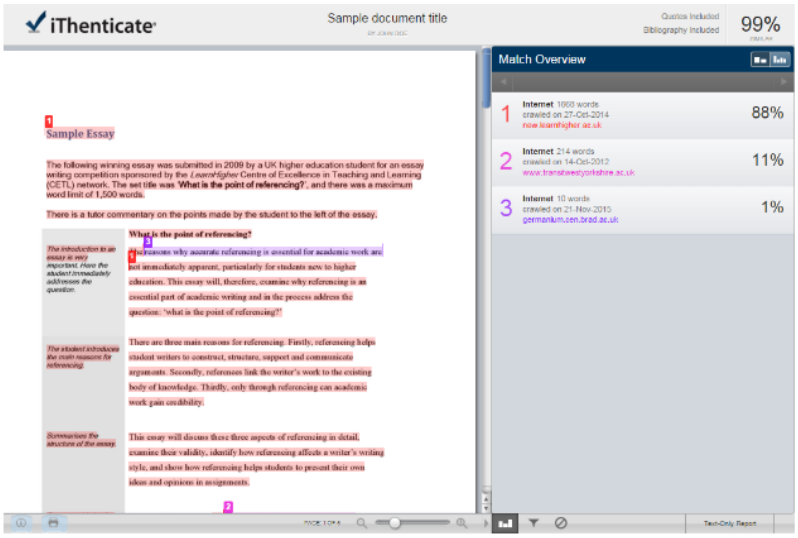
Princip detektoru je založený na tokenizaci a následném porovnání vytvořených tokenů a vyhodnocení. *Sherlock* podporuje velké množství programovacích jazyků, specificky optimalizovaný je pro programovací jazyk *Java*. Výhodou je, že výsledky lze vizualizovat i pomocí grafů (obr. 2.8).



Obr. 2.8: Vizualizace výsledků detektoru *Sherlock* [21].

iThenticate

iThenticate [22] je detekční placený program sloužící k vyhledávání plagiátů textových souborů. Jedná se o poskytovatele profesionální detekce plagiátů používaným na celém světě vědeckými vydavateli a výzkumnými institucemi.



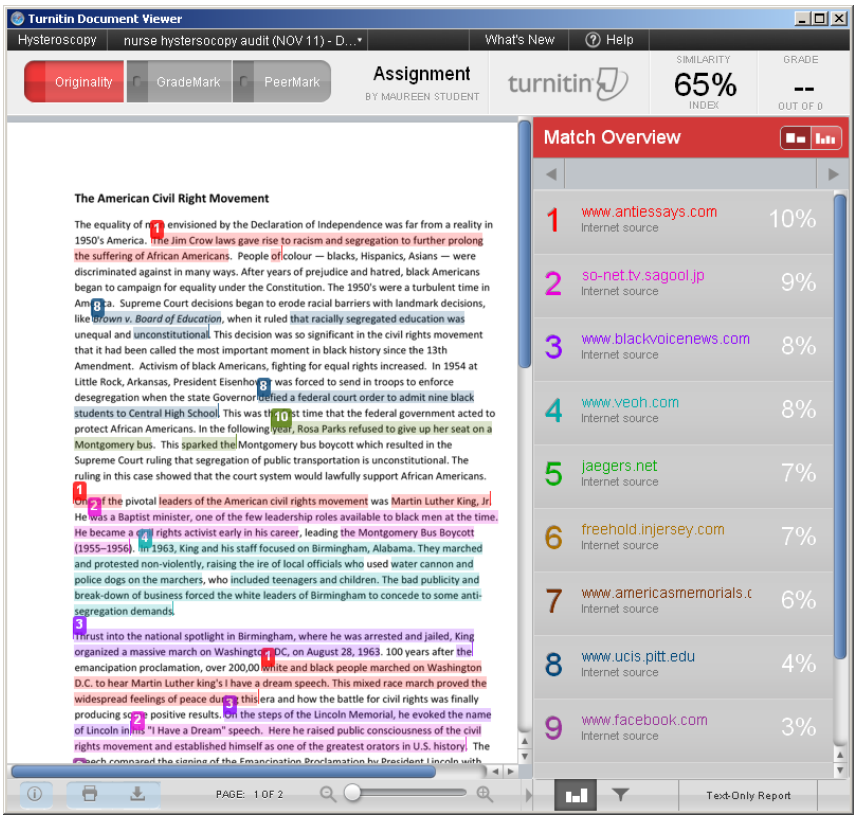
Obr. 2.9: Typický výstup detektoru *iThenticate* [23].

Tento nástroj umožňuje tedy zejména vydavatelům, společnostem a firmám okamžité ověření originality dokumentů i rukopisů.

Dále umožňuje zjistit, zda-li dokumenty jejich duševního vlastnictví nejsou neoprávněně použity v prostředí internetu. Technologie *iThenticate* byla vyvinuta společností *Turnitin*, která je vedoucí v oblasti plagiátorství a ověřování originality v rámci vzdělávacích institucí po celém světě.

Turnitin

Turnitin [24] je placeným programem využívaným zejména vyučujícími pro odhalení plagiátorství studentských prací. Pro studenty je program nedostupný, předchází se tak potenciálnímu akademickému pochybení. Práce vložená do systému je porovnána se všemi dokumenty v databázi, což umožňuje vyučujícím rychle rozpoznat množství podobností s ostatními pracemi, webovými stránkami i literaturou. Mezi porovnávané dokumenty mohou patřit stránky aktivních i archivovaných informací z internetu, úložiště již dřív odevzdaných prací do tohoto nástroje, úložiště časopisů či dalších publikací. Program usnadňuje i známkování a poskytování zpětné vazby pro studenty.



Obr. 2.10: Typický výstup detektoru *Turnitin* [25].

3 Detekce podobností grafických uživatelských rozhraní

Doposud se práce věnovala obecně detekci podobnosti v programových kódech. Praktická realizace je však zaměřena konkrétně na zdrojové kódy grafických uživatelských rozhraní pro vývojové prostředí *MATLAB*, proto se další odstavce věnují právě ***GUI*** (***Graphical User Interface***). Zejména výčet použitých příznaků se může v závislosti od specifické charakteristiky zdrojového kódu *GUI* lišit.

3.1 Vývojové prostředí MATLAB

Název *MATLAB* [27] vznikl zkrácením slov ***MAT****rix* ***LAB****oratory*, jedná se o interaktivní numerické výpočetní prostředí, jehož základem je programovací jazyk *MATLAB* s uzavřeným kódem. Původně byl jazyk určen pro matematické účely, časem byl však upraven, byly přidány nové funkce a rozšíření a dnes je využitelný v široké oblasti aplikací.

MATLAB vyvinutý firmou *MathWorks* umožňuje maticové manipulace, realizace algoritmů, vytvoření uživatelských rozhraní i propojení s programy napsanými v jiných jazycích. Milióny inženýrů a vědců již používají *MATLAB* pro analýzu dat, vývoj algoritmů či vytváření modelů a aplikací. *MATLAB* lze využít pro celou řadu aplikací včetně strojového učení, pro testování, měření, zpracování signálů, obrazů či videa.

3.2 Grafické uživatelské rozhraní MATLAB

GUI nebo-li grafické uživatelské rozhraní umožňuje interaktivní ovládání programu uživatelem bez nutnosti znalosti programovacího jazyka či psaní příkazů pro fungování aplikace. Uživatelské rozhraní by obecně mělo být jednoduché, určené i pro běžné uživatele, ne jen pro odborníky. Celkově by mělo zjednodušit práci se systémem.

Grafické uživatelské rozhraní je založeno na paradigmatu ***WIMP***, neboli ***Windows*** (okna – překonání omezenosti pracovní plochy), ***Icons*** (ikony – grafické komponenty reprezentující funkce, soubory apod.), ***Menus*** (menu - strukturovaně uspořádané grafické prvky), ***Pointing device*** (polohovací zařízení – myš, touchpad, které je reprezentováno ukazatelem).

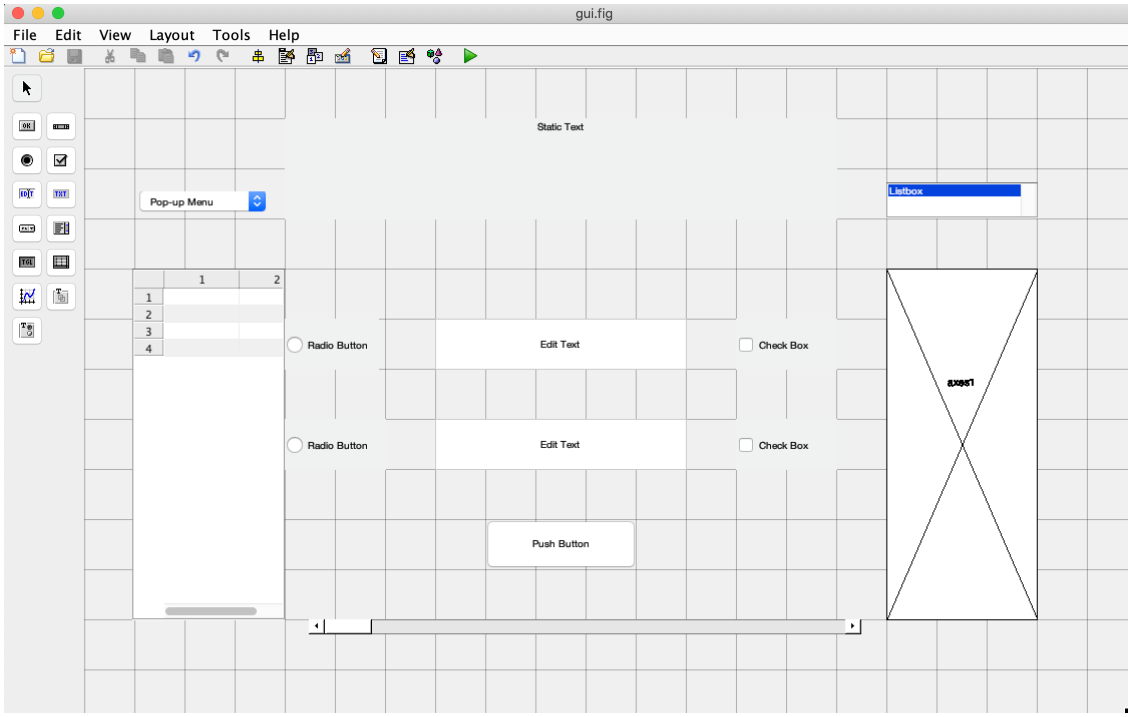
MATLAB nám v rámci tvorby *GUI* umožňuje vytvoření ovládacích prvků, jako jsou tlačítka, editační a zaškrtávací pole, posuvníky apod. Pro celé grafické uživatelské rozhraní i pro jeho dílčí komponenty můžeme nastavit vlastnosti (barva pozadí, pozice jednotlivých komponent, jejich viditelnost, aktivnost apod).

Vytvoření MATLAB GUI pomocí GUIDE

Graphical User Interface Development Environment [26] (zkráceno na *GUI Development Environment* nebo *GUIDE* – vývojové prostředí *GUI*) poskytuje nástroje pro návrh uživatelských rozhraní pro vlastní aplikace. Pomocí editoru *Layout GUIDE* je možné graficky navrhnout uživatelské rozhraní. Spuštění průvodce pro tvorbu *GUI* aplikací je možné více způsoby.

Jedním z nich je využití základního menu `File > New > GUI` nebo zápisem a potvrzením příkazu `guide`. Otevře se prázdné okno, které představuje formulář, do kterého se vkládají grafické objekty. U každého grafického objektu lze definovat vlastnosti (vizuální podobu a další parametry) pomocí *Property Inspector* (vyvolá se klikem pravého tlačítka myši na objekt).

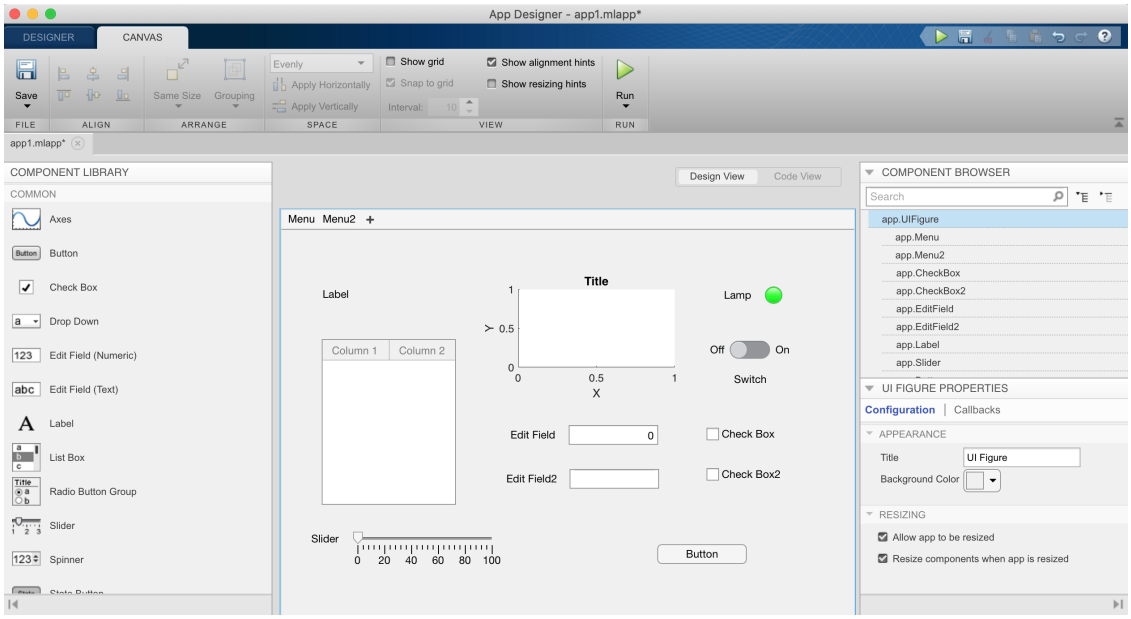
Parametr `Tag` reprezentuje jméno objektu. Formulář okna se ukládá jako soubor `nazev_okna.fig`, současně se také ukládá soubor stejného jména jako m-file `nazev_okna.m`, který obsahuje logiku rozhraní.



Obr. 3.1: Ovládací prvky *GUI* vývojového prostředí *MATLAB*.

Vytvoření aplikace MATLAB pomocí App Designer

V novější verzi *MATLAB* se setkáváme s nahrazením vývojového prostředí *GUIDE* za ***App Designer*** [26]. *App Designer* je také prostředím pro vytváření aplikací *MATLAB*. Slouží k integraci dvou primárních úloh budování aplikací, vytváření vizuálních komponent a programování chování aplikací. Umožňuje rychle přecházet mezi vizuálním designem na plátně a vývojem kódu v integrované verzi editoru *MATLAB*.



Obr. 3.2: *App Designer* vývojového prostředí *MATLAB*.

3.3 Specifické prvky zdrojového kódu GUI

Zdrojový kód grafického uživatelského rozhraní se liší v porovnání s klasickým zdrojovým kódem v několika prvcích. Po sestavení *GUI* je automaticky generován kód *MATLAB*, který je přidán do editoru *MATLAB* a ve kterém jsou automaticky připraveny funkce:

- **`jmenoObjektu_CreateFcn`** – volána při vytváření objektu,
- **`jmenoObjektu_Callback`** – volána při události na objektu (vždy jen jedna typická událost, např. klik u tlačítek, posun posuvníkem u lišty apod.).

```
1  function edit2_Callback(hObject, eventdata, handles)
2  % hObject    handle to edit2 (see GCBO)
3  % eventdata  reserved - to be defined in a future version
       of MATLAB
4  % handles    structure with handles and user data (see
       GUIDATA)
5  % Hints: get(hObject,'String') returns contents of edit2
       as text
6  %        str2double(get(hObject,'String')) returns
       contents of edit2 as a double
7  % --- Executes during object creation, after setting all
       properties.
8  function edit2_CreateFcn(hObject, eventdata, handles)
9  % hObject    handle to edit2 (see GCBO)
10 % eventdata  reserved - to be defined in a future version
       of MATLAB
11 % handles    empty - handles not created until after all
       CreateFcns called
12 % Hint: edit controls usually have a white background on
       Windows.
13 %       See ISPC and COMPUTER.
14 if ispc && isequal(get(hObject,'BackgroundColor'), get(0,
       'defaultUicontrolBackgroundColor'))
15     set(hObject,'BackgroundColor','white');
16 end
```

Výpis 3.1: Ukázka zdrojového kódu *GUI* vývojového prostředí *MATLAB* (automaticky vytvořené funkce).

Do vygenerovaných funkcí lze vkládat „matlabovský“ kód a definovat tak chování aplikace. Pro nastavení parametru objektu prostřednictvím zdrojového kódu se využívá funkce **`set(objekt, parametr, hodnota)`** viz výpis 3.2. Parametr `objekt` zde může být reprezentován:

- **`hObject`** – aktuálně fokusovaný objekt (např. ve funkci `Callback` pro daný objekt bude fokusovaný právě tento objekt),
- **`handles.jmenoObjektu`** – objekt, který není fokusován (všechny vytvořené objekty ve formuláři jsou položkami struktury `handles`).

```
1 set(handles.My_object,'Visible','on') %Nastaví
    viditelnost objektu My_object
2 x=get(handles.My_object,'String') %Uloží do proměnné x
    String objektu My_object
```

Výpis 3.2: Ukázka nastavení parametru objektu prostřednictvím zdrojového kódu.

Pro získání parametru objektu prostřednictvím zdrojového kódu (výpis 3.2) se využívá funkce **`get(objekt, parametr)`**. Parametr `objekt` může být reprezentován stejně jako v případě funkce `set`. Mezi další automaticky vytvořené funkce patří:

- **`jmenoProjektu_OpeningFcn`** – volána při otevírání formuláře a může být použita pro inicializační definice,
- **`varargout=jmenoProjektu_OutputFcn`** – volána při zavírání formuláře a může být použita pro předávání parametrů (výsledků).

```
1  % hObject    handle to figure
2  % eventdata  reserved - to be defined in a future version
     of MATLAB
3  % handles    structure with handles and user data (see
     GUIDATA)
4  function gui_OpeningFcn(hObject, eventdata, handles,
     varargin)
5  % This function has no output args, see OutputFcn.
6  % varargin   command line arguments to gui (see VARARGIN)
7  % Choose default command line output for gui
8  handles.output = hObject;
9  % Update handles structure
10 guidata(hObject, handles);
11 % --- Outputs from this function are returned to the
     command line.
12 function varargout = gui_OutputFcn(hObject, eventdata,
     handles)
13 % varargout  cell array for returning output args (see
     VARARGOUT);
14 % Get default command line output from handles structure
15 varargout{1} = handles.output;
```

Výpis 3.3: Ukázka zdrojového kódu *GUI* vývojového prostředí *MATLAB* (úvodní automaticky vytvořené funkce).

Kód je rozdělený do několika funkcí podle počtu vytvořených komponent. Pořadí jednotlivých funkcí v kódu nemá na samotnou funkčnost grafického uživatelského prostředí vliv. V kódu se objevují funkce **`get`** a **`set`** pro získání či nastavení určité vlastnosti daného objektu. K objektu se přistupuje prostřednictvím **`hObject`**, pokud jsou ovlivňovány parametry ve funkci příslušného objektu nebo pomocí **`handles.jmenoObjektu`**, nezávislé na části kódu, kde se uživatel právě nachází.

Jelikož se ve zdrojovém kódu *GUI* nachází zmíněné specifické prvky, naskýtá se tak možnost jejich zahrnutí pro detekci podobnosti těchto kódů. Názvy používaných funkcí lze přidat do výčtu hledaných klíčových slov, které se detekují a následně se jejich vlastnosti jako je počet či pozice porovnávají.

4 Vytvořený detektor podobností zdrojových kódů GUI

V rámci praktické části bakalářské práce byl vytvořen detektor podobností zdrojových kódů grafických uživatelských rozhraní. Detektor umožňuje porovnání zdrojových kódů za účelem odhalení plagiátorství. Je opatřen grafickým uživatelským rozhraním a uživateli poskytuje možnost vzájemného porovnání všech zdrojových kódů nacházejících se v jednom adresáři nebo možnost porovnání všech zdrojových kódů mezi sebou nacházejících se ve dvou adresářích. Je navíc opatřen funkcí pro export dosažených výsledků do formátu *CSV*.

Řešení bakalářské práce bylo realizováno programovacím jazykem *MATLAB* ve vývojovém prostředí *MATLAB 2018b*.

```
DETEKTOR
├── funkce ..... pomocné funkce detektoru.
├── predzprac_kody ..... adresář kódů po předzpracování (soubory .txt).
├── predzprac_funkce ..... adresář funkcí po předzpracování (soubory .txt).
├── UVODNI.mlapp ..... úvodní okno GUI detektoru.
├── guicko_main.mlapp ..... okno GUI při výběru jednoho adresáře.
├── cesty_dve.mlapp ..... okno GUI při výběru dvou adresářů.
├── vysledek_detekce.mlapp ... okno GUI pro zobrazení výsledků podobnosti.
├── podrobnosti.mlapp ..... okno GUI pro zobrazení podrobností.
└── app_properties.mat ..... data aplikace.
```

Obr. 4.1: Detektor podobností zdrojových kódů *GUI*.

4.1 Postup detekce podobnosti

Práce se zdrojovým kódem *GUI* je rozdělena do čtyř dílčích částí:

1. nahrání kódu,
2. předzpracování,
3. detekce příznaků
4. a formátovaný výstup nalezených podobností.

Každý z porovnávaných zdrojových kódů *GUI* je nejprve načten a následně se provede předzpracování. Po procesu předzpracování jsou detekovány jednotlivé předem definované příznaky v celém kódu, ale i samostatně ve všech funkcích kódu.

Dalším krokem detekce podobnosti je metrika nalezených příznaků, která je vypočtena pro příznaky všech jednotlivých funkcí porovnávaných kódů. Jsou vypočteny váhované poměry příznaků. Váhy jsou určeny na základě rozdílů pozic daných příznaků v kódech. Dále jsou porovnány nalezené názvy proměnných, tagů a funkcí.

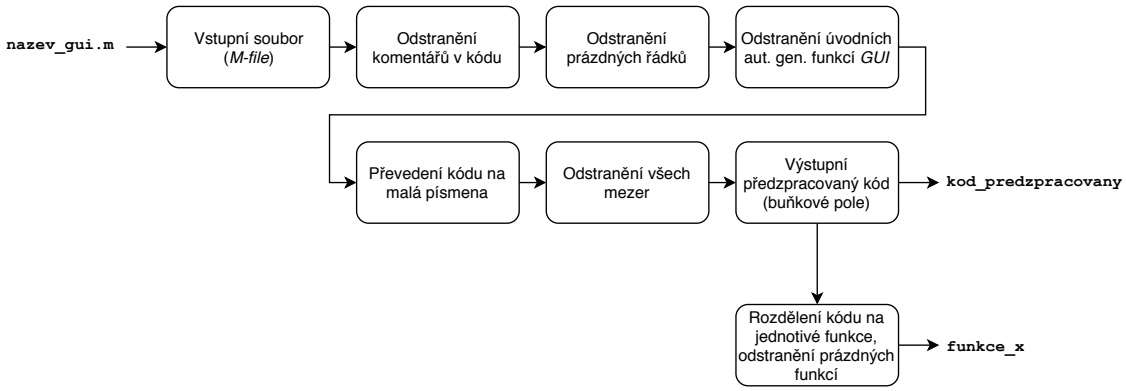
V posledním kroku jsou k sobě přiřazeny nejvíce podobné funkce porovnávaných zdrojových kódů. Na základě podobnosti přiřazených funkcí je vyhodnocena celková podobnost zdrojových kódů a poměr funkcí, které obsahují.

4.2 Předzpracování souborů

Před samotnou detekcí příznaků je nutná fáze předzpracování programových kódů za účelem sjednocení podoby jejich formátování. Jedná se o odstranění nepodstatných částí kódu pro detekci podobnosti.

V rámci procesu předzpracování (obr. 4.2) byly konkrétně odstraněny:

- všechny komentáře kódu,
- veškeré mezery v kódu
- a prázdné řádky.



Obr. 4.2: Blokové schéma kroků předzpracování zdrojového kódu.

Z celého zdrojového kódu byly tak vybrány pouze stěžejní řádky programu. Součástí předzpracování bylo dále převedení všech písmen obsažených v kódu na malá písmena. Z důvodu odlišnosti zdrojového kódu *GUI* (automaticky generované úvodní funkce) byly tyto funkce pro předcházení falešně pozitivní detekce podobnosti odstraněny.

Výsledný předzpracovaný kód je následně pro jednoduchost detekce příznaků a jejich snazší lokalizaci uložen do buňkového pole. Každý řádek předzpracovaného kódu odpovídá jednomu řádku ve vytvořeném buňkovém poli viz obr. 4.3. Celý předzpracovaný kód je rovněž převeden do podoby jednoho dlouhého řádku, což je přínosné například pro vyhodnocení celkového počtu znaků.

Poslední fází procesu předzpracování bylo rozdělení celého předzpracovaného zdrojového kódu na jednotlivé funkce (obr. 4.4), protože při následném vyhodnocení podobnosti je pracováno zejména s funkcemi.

```
functioncislo_x_callback(hobject,eventdata,handles)
functioncislo_x_createfcn(hobject,eventdata,handles)
c=0;
f=1;
b=3;
d=c+b;
ifispc&&isequal(get(hobject,'backgroundcolor'),get(0,'defaultuicontrolbackgroundcolor'))
set(hobject,'backgroundcolor','white');
end
functioncislo_y_callback(hobject,eventdata,handles)
functioncislo_y_createfcn(hobject,eventdata,handles)
ifispc&&isequal(get(hobject,'backgroundcolor'),get(0,'defaultuicontrolbackgroundcolor'))
set(hobject,'backgroundcolor','white');
end
functionspocitej_callback(hobject,eventdata,handles)
x=str2num(get(handles.cislo_x,'string'));
y=str2num(get(handles.cislo_y,'string'));
z=((x/100)*y);
set(handles.vysledek,'string',z);
```

Obr. 4.3: Předzpracovaný zdrojový kód `procenta.m`.

Pro rozdělení zdrojového kódu na funkce bylo využito nalezení pozic klíčového slova `function`. Prázdné funkce *GUI* byly následně odstraněny a dále se s nimi při vyhodnocení podobnosti již nepracuje.

```
functioncislo_x_createfcn(hobject,eventdata,handles)
c=0;
f=1;
b=3;
d=c+b;
ifispc&&isequal(get(hobject,'backgroundcolor'),get(0,'defaultuicontrolbackgroundcolor'))
set(hobject,'backgroundcolor','white');
end
```

```
functioncislo_y_createfcn(hobject,eventdata,handles)
ifispc&&isequal(get(hobject,'backgroundcolor'),get(0,'defaultuicontrolbackgroundcolor'))
set(hobject,'backgroundcolor','white');
end
```

```
functionspocitej_callback(hobject,eventdata,handles)
x=str2num(get(handles.cislo_x,'string'));
y=str2num(get(handles.cislo_y,'string'));
z=((x/100)*y);
set(handles.vysledek,'string',z);
```

Obr. 4.4: Výsledek rozdělení zdrojového kódu `procenta.m`.

4.3 Příznaky pro detekci podobnosti

Základem pro vytvoření funkčního detektoru podobností je definice a následné nalezení příznaků, se kterými bude detektor pracovat. Porovnáním detekovaných příznaků lze posléze určit podobnost srovnávaných souborů.

V rámci bakalářské práce byl vytvořen výčet příznaků, které jsou ve zdrojových kódech detekovány. Veškeré příznaky jsou detekovány nejen obecně v celém předzpracovaném kódu *GUI*, ale také samostatně v jednotlivých funkcích daného kódu.

Navržené příznaky

Detekované příznaky lze rozdělit do skupin:

- obecné vlastnosti kódu,
- informace o funkcích,
- informace o proměnných,
- informace o využívaných prvcích grafického uživatelského rozhraní
- a klíčová slova.

V rámci obecných vlastností kódu byl detekován a uložen počet řádků kódu, počet řádků komentářů, počet znaků a číslic. Některé z uvedených příznaků jsou detekovány i před samotným procesem předzpracování. Patří k nim zejména počet řádků komentářů a celkový počet řádků kódu. Počet řádků kódu je následně detekován i po procesu předzpracování.

V kódu je také detekována informace o počtu a pozici funkcí, následně se detekují i názvy jednotlivých funkcí.

Další skupinou hledaných příznaků jsou proměnné, konkrétně jejich celkový počet, pozice řádků, na kterých se v kódu nachází a jejich názvy.

Mezi detekované příznaky je zařazen také počet využívaných prvků grafického uživatelského rozhraní a jejich názvy – tzv. ***tagy***.

Při detekci klíčových slov ve zdrojovém kódu je prvním krokem jejich definice (tab. 4.1). Celkem bylo zvoleno 37 hledaných klíčových slov.

Seznam klíčových slov				
`if`	`else`	`elseif`	`end`	`fft`
`for`	`while`	`switch`	`case`	`ifft`
`return`	`break`	`size`	`length`	`fft2`
`plot`	`figure`	`title`	`isempty`	`ifft2`
`str2num`	`num2str`	`find`	`min`	`set`
`sum`	`mean`	`max`	`str2double`	`get`
`disp`	`cell2mat`	`strcmp`	`strfind`	`imshow`
`fastaread`	`genbankread`			

Tab. 4.1: Seznam detekovaných klíčových slov.

Hledaná klíčová slova můžeme rozdělit na

- základní příkazy (`while`, `for`, `if`, `switch` apod.),
- příkazy pro ovládání a práci s *GUI* (`set`, `get` apod.),
- elementární příkazy (`sum`, `max`, `min` apod.),
- signálové (`fft`, `fft2` apod.)
- a bioinformatické příkazy (`genbankread` , `fastaread` apod.).

Počet všech nadetekovaných klíčových slov a čísla řádků na kterých se v kódu nachází, jsou ukládány do vytvořeného buňkového pole (obr. 4.5).

	'Nazev'	'Pocet'	'Pozice'
1	'if'	1	2
2	'else'	0	[]
3	'elseif'	0	[]
4	'for'	0	[]
5	'switch'	0	[]
6	'while'	0	[]
7	'end'	1	4
8	'case'	0	[]
9	'return'	0	[]
10	'break'	0	[]
11	'length'	0	[]
12	'size'	0	[]
13	'plot'	0	[]
14	'figure'	0	[]
15	'find'	0	[]
16	'str2num'	0	[]
17	'num2str'	0	[]
18	'isempty'	0	[]
19	'set'	1	3
20	'get'	2	2
21	'sum'	0	[]
22	'max'	0	[]
23	'min'	0	[]
24	'str2double'	0	[]
25	'mean'	0	[]
26	'disp'	0	[]
27	'title'	0	[]
28	'imshow'	0	[]
29	'fft'	0	[]
30	'fft2'	0	[]
31	'cell2mat'	0	[]
32	'fastaread'	0	[]
33	'genbankre...	0	[]
34	'strcmp'	0	[]
35	'strfind'	0	[]
36	'ifft'	0	[]
37	'ifft2'	0	[]

Obr. 4.5: Výsledné buňkové pole klíčových slov s počtem a pozicemi řádků v první funkci kódu `procenta.m`.

Detekce příznaků

K detekci většiny příznaků jsou využity jednoduché funkce. Využívá se zejména „matlabovské“ funkce **`strfind`**, jejíž výstupem je pozice v řádku, na kterém se hledané klíčové slovo či znak nachází.

Při detekci klíčových slov je nutné kontrolovat několik prvků, např. pro detekci klíčového slova `end` je kontrolována jeho samostatnost na řádku. Pro správné nalezení klíčových slov je také ověřováno, zda nejsou součástí nějakého jiného slova a také, zda se nejedná pouze o část textového řetězce. Detekce jednotlivých příznaků je realizována prostřednictvím dílčích funkcí detektoru viz obr. 4.6. Výsledky detekce příznaků jsou ukládány do buňkového pole (obr. 4.7).

```
funkce
├── detekce_klic_slov.m ..... detekce klíčových slov.
├── detekce_promenne.m ..... detekce proměnných.
├── detekce_tagy.m ..... detekce tagů.
└── detektor.m ..... řídící funkce detekce příznaků.
```

Obr. 4.6: Funkce pro detekci příznaků zdrojových kódů *GUI*.

'pocet_radku_pred'	275
'pocet_radku_celkove'	93
'pocet_radku_komentaru'	110
'pocet_cislic_celkove'	46
'pocet_znaku_celkove'	3443
'pocet_funkci_pred'	16
'pozice_funkci_pred'	*1x16 double*
'nazvy_funkci'	*13x1 cell*
'pocet_funkci_po'	13
'pozice_funkci_po'	*1x13 double*
'nazvy_promenne'	*13x1 cell*
'pozice_promenne'	*13x1 cell*
'pocet_promennych'	13
'nazvy_tagu'	*8x1 cell*
'pocet_tagu'	8

Obr. 4.7: Detekované příznaky pro celý kód `paleta.m`.

Jelikož se v dalším kroku detekce podobnosti pracuje výhradně s jednotlivými funkcemi porovnávaných zdrojových kódů, jsou pro následné vyhodnocení důležité zejména nadetekované příznaky v jednotlivých funkcích kódů.

Do výsledného buňkového pole (obr. 4.8) se na pozici prvního sloupce ukládají indexy funkcí, v dalším sloupci jsou uloženy předzpracované kódy, třetí sloupec obsahuje kód funkce převedený do podoby jednoho řádku. V následujících sloupcích jsou již jednotlivé detekované příznaky, kterým odpovídá vždy název daného sloupce.

'index'	'kod'	'kod text'	'klicova slova'	'pocet radku'	'pocet znaku'	'pocet cislic'	'pocet tagu'	'nazvy tagu'	'pocet pro...	'nazvy promennych'	'pozice promennych'	'nazev funkce'	'pozice tagy'
1	*13x1 cell*	*1x239 char*	*38x3 cell*	13	239	5	2	*2x1 cell*	5	*5x1 cell*	*5x1 cell*	'red_callback'	*2x1 cell*
2	*5x1 cell*	*1x195 char*	*38x3 cell*	5	195	1	0	[]	0	[]	[]	'red_createfcn'	[]
3	*7x1 cell*	*1x197 char*	*38x3 cell*	7	197	1	2	*2x1 cell*	1	'input_g'	*1x1 cell*	'green_callba...	*2x1 cell*
4	*4x1 cell*	*1x184 char*	*38x3 cell*	4	184	1	0	[]	0	[]	[]	'green_creat...	[]
5	*7x1 cell*	*1x195 char*	*38x3 cell*	7	195	1	2	*2x1 cell*	1	'input_b'	*1x1 cell*	'blue_callback'	*2x1 cell*
6	*4x1 cell*	*1x183 char*	*38x3 cell*	4	183	1	0	[]	0	[]	[]	'blue_createf...	[]
7	*23x1 cell*	*1x615 char*	*38x3 cell*	23	615	24	5	*5x1 cell*	3	*3x1 cell*	*3x1 cell*	'michej_callb...	*4x1 cell*
8	*6x1 cell*	*1x362 char*	*38x3 cell*	6	362	0	7	*7x1 cell*	1	'hodnota_r'	*1x1 cell*	'red_posun_c...	*6x1 cell*
9	*4x1 cell*	*1x183 char*	*38x3 cell*	4	183	4	0	[]	0	[]	[]	'red_posun_c...	[]
10	*6x1 cell*	*1x360 char*	*38x3 cell*	6	360	0	7	*7x1 cell*	1	'hodnota_g'	*1x1 cell*	'green_posu...	*6x1 cell*
11	*4x1 cell*	*1x185 char*	*38x3 cell*	4	185	4	0	[]	0	[]	[]	'green_posu...	[]
12	*6x1 cell*	*1x361 char*	*38x3 cell*	6	361	0	7	*7x1 cell*	1	'hodnota_b'	*1x1 cell*	'blue_posun_...	*6x1 cell*
13	*4x1 cell*	*1x184 char*	*38x3 cell*	4	184	4	0	[]	0	[]	[]	'blue_posun_...	[]

Obr. 4.8: Výsledné buňkové pole detekovaných příznaků v jednotlivých funkcích zdrojového kódu `paleta.m`.

Jak již bylo zmíněno, v rámci procesu detekce příznaků byly detekovány některé příznaky i před procesem předzpracování. Tyto příznaky pro následné vyhodnocení podobnosti využity nebyly. Ve výše uvedeném buňkovém poli (obr. 4.8) jsou tedy pouze příznaky, se kterými se dále v procesu vyhodnocení pracuje.

Metrika příznaků

Pro vyhodnocení vzájemné podobnosti zdrojových kódu je nejdříve nutné stanovit metriku detekovaných příznaků. V rámci detekce příznaků byly získány počty příznaků v jednotlivých funkcích porovnávaných zdrojových kódů, také pozice řádků, na kterých se hledané příznaky nachází a použité názvy proměnných, tagů a funkcí. Veškerá metrika příznaků a následná logika výsledného vyhodnocení podobnosti je implementována ve funkci **`vyhodnoceni.m`**.

K určení hodnoty podobnosti zdrojových kódů jsou informace o získaném počtu, pozici a názvech porovnány. Jelikož jsou kódy rozděleny do jednotlivých funkcí, porovnávají se příznaky pro každou funkci jednoho zdrojového kódu s příznaky každé funkce druhého zdrojového kódů.

V případě pokusu o vytvoření plagiátu zdrojového kódu se může jednat pouze o přeskládání pořadí dílčích funkcí, které však mohou zůstat obsahově naprosto identické. Při porovnávání celého kódu, by tento plagiát nemusel být odhalen, jelikož by pozice nalezených příznaků byly příliš vzdálené. Při rozdělení kódů do jednotlivých funkcí je možné porovnat přímo pozice v daných funkcích, nalezení plagiátu vytvořeného změnou pořadí funkcí je v tomto případě mnohem pravděpodobnější.

Pro některé z příznaků je jejich porovnání provedeno pouze vzájemným poměrem, jelikož nejsou dále definovány jejich přesnými pozicemi. K těmto příznakům patří zejména obecné vlastnosti (počty řádků, počty znaků a počty číslic). Poměr příznaků je vždy vypočítán z rovnice 4.1, kde $\boldsymbol{P}$ je vypočtená hodnota poměru, $\boldsymbol{A}$ reprezentuje počet příznaků v jednom kódu a $\boldsymbol{B}$ počet příznaků v kódu druhém.

$$\mathbf{P} \quad = \quad \frac{\mathbf{min(A,B)}}{\mathbf{max(A,B)}} \tag{4.1}$$

Je nalezena menší z porovnávaných hodnot a ta je podělena hodnotou větší, aby byl výsledný poměr roven maximálně hodnotě 1. Výsledný poměr je vypočten ve vytvořené funkci `vypocet_pomer.m`.

U příznaků, pro které je získána hodnota pozic řádků, na kterých se nachází, je výsledný poměr váhovaný. Jedná se o nalezené příznaky počtu tagů, počtu proměnných a klíčová slova.

První fází získání váhy poměru, je výpočet rozdílu pozic příznaků, který je realizován ve funkci `vypocet_vzdalenosti.m`. Vektory pozic porovnávaných příznaků jsou vzájemně překrývány, aby bylo možné nalézt nejmenší hodnotu vzdálenosti pozic. Vzdálenost jednotlivých pozic ve vektoru je pro každý posun počítána jako absolutní hodnota jejich rozdílu. Pro určení výsledného rozdílu pozic jsou všechny nalezené vzdálenosti ve vektoru zprůměrovány. Následně je vybrána nejmenší nalezená hodnota. Pokud jsou pozice příznaků identické, je výsledný rozdíl pozic nulový.

Druhou fází je samotné určení váhy. Nejmenší nalezené vzdálenosti pozic je přiřazena hodnota odpovídající váhy z tab. 4.2. Výsledné určení váhy provádí funkce `vypocti_metriku.m`. Vstupem funkce je nejmenší vzdálenost pozic ($\boldsymbol{n}$) a výstupem je přiřazená hodnota váhy ($\boldsymbol{V}$).

Tabulka hodnot vah pro vzdálenosti pozic příznaků

Nejmenší vzdálenost pozic ($\boldsymbol{n}$)	Přiřazená váha ($\boldsymbol{V}$)
$n = 0$	1
$0 < n \leq 1$	0,9
$1 < n \leq 2$	0,8
$2 < n \leq 3$	0,7
$3 > n \leq 4$	0,6
$4 > n \leq 5$	0,5
$5 > n \leq 10$	0,4
$n > 10$	0,2

Tab. 4.2: Hodnoty vah pro vzdálenosti pozic příznaků.

V případě nalezení identických pozic příznaků je určena maximální váha s hodnotou 1. S postupně větším rozdílem pozic je váha snižována. Pro všechny vzdálenosti pozic větší než 10 je nastavena pevná váha s hodnotou 0,2. Příslušnou váhou $\boldsymbol{V}$ je vynásoben vypočtený poměr $\boldsymbol{P}$ a tím tak získána výsledná hodnota podobnosti příznaků $\boldsymbol{P}_V$ (rovnice 4.2).

$$\mathbf{P}_V \quad = \quad \mathbf{V} \cdot \mathbf{P} \tag{4.2}$$

Hodnota podobnosti klíčových slov je vypočtena součinem vektorů vah a poměrů. Výsledná hodnota podobnosti na základě klíčových slov je průměr vypočtených hodnot.

Mezi nadetekované příznaky patří i samotné názvy proměnných, tagů a funkcí. Nalezené názvy proměnných ve funkcích jsou vzájemně porovnány a je uložen jejich shodný počet. Názvy jednotlivých funkcí jsou rovněž porovnány, ukládá se logická hodnota *1* pro stejnost a logická hodnota *0* pro nestejnost názvu funkce. Porovnání detekovaných tagů je realizováno napříč celými zdrojovými kódy, jelikož se jedná o globálně používaný příznak.

Veškeré určené hodnoty podobnosti jsou ukládány do buňkového pole (obr. 4.9). První sloupec buňkového pole tvoří indexy funkcí kódů. Další sloupce obsahují hodnoty poměrů či váhovaných poměrů porovnávaných příznaků, počty shodných názvů proměnných, logickou hodnotu pro stejnost/nestejnost názvu funkce apod.

'indexy fun...	'vahy klicov...	'pomer klic...	'vysledek kli...	'pomer rad...	'pomer zna...	'pomer cisli...	'pomer tag...	'vysledek pr...	'nazev funk...	'nazev tagu'	'nazev pro...	'vahy prom...	'pomer pro...	'pomer pro...	'pomer tag...	'vahy tagy'	'vysledek ta...
[1 1]	[0.8000 1 0....	[0.5000 0.50...	0.3817	0.3333	0.1528	0.0833	0.2500	0.3333	0	[]	*1x30 double*	0.4000	0.8333	0.8333	0.1200	0.9000	0.1080
[1 2]	[0.7000 0.60...	[0.5000 0.50...	0.4213	0.1282	0.1247	0.0167	[]	[]	0	[]	[]	0	[]	[]	[]	0	[]
[1 3]	[0.8000 0.80...	[0.5000 0.50...	0.3833	0.1795	0.1260	0.0167	0.2500	0.1667	0	[]	[0 0 0 0 0 0]	1	0.1667	0.1667	0.1200	0.9000	0.1080
[1 4]	[0.7000 0.50...	[0.5000 0.50...	0.4088	0.1026	0.1176	0.0167	[]	[]	0	[]	[]	0	[]	[]	[]	0	[]
[1 5]	[0.8000 0.80...	[0.5000 0.50...	0.3833	0.1795	0.1247	0.0167	0.2500	0.1667	0	[]	[0 0 0 0 0 0]	1	0.1667	0.1667	0.1200	0.9000	0.1080
[1 6]	[0.7000 0.50...	[0.5000 0.50...	0.4088	0.1026	0.1170	0.0167	[]	[]	0	[]	[]	0	[]	[]	[]	0	[]
[1 7]	[1 0.9000 0....	[0.6667 0.60...	0.4173	0.5897	0.3932	0.4000	0.6250	0.4000	0	[]	*1x18 double*	0.4000	0.5000	1	0.5200	0.8000	0.4160
[1 8]	[1 0.9000]	[0.2000 0.40...	0.2800	0.1538	0.2315	[]	0.8750	0.1667	0	[]	[0 0 0 0 0 0]	1	0.1667	0.1667	0.3200	0.8000	0.2560
[1 9]	[0.7000 0.50...	[0.5000 0.50...	0.4088	0.1026	0.1170	0.0667	[]	[]	0	[]	[]	0	[]	[]	[]	0	[]
[1 10]	[1 1]	[0.2000 0.40...	0.3000	0.1538	0.2302	[]	0.8750	0.1667	0	[]	[0 0 0 0 0 0]	1	0.1667	0.1667	0.3200	0.8000	0.2560
[1 11]	[0.7000 0.50...	[0.5000 0.50...	0.4088	0.1026	0.1183	0.0667	[]	[]	0	[]	[]	0	[]	[]	[]	0	[]
[1 12]	[1 1]	[0.2000 0.40...	0.3000	0.1538	0.2308	[]	0.8750	0.1667	0	[]	[0 0 0 0 0 0]	1	0.1667	0.1667	0.3200	0.8000	0.2560
[1 13]	[0.7000 0.50...	[0.5000 0.50...	0.4088	0.1026	0.1176	0.0667	[]	[]	0	[]	[]	0	[]	[]	[]	0	[]

Obr. 4.9: Část výsledného buňkového pole porovnání příznaků pro první funkci prvního kódu se všemi funkcemi kódu druhého.

4.4 Vyhodnocení podobnosti

Pro výsledné vyhodnocení podobnosti je nutné přiřadit nejvíce podobné funkce zdrojových kódů. Pro kód s menším počtem funkcí jsou ke všem funkcím nalezeny nejpodobnější funkce druhého zdrojového kódu. Nejvyšší prioritu při hledání nejvíce podobných funkcí má hodnota podobnosti na základě klíčových slov, protože klíčová slova tvoří hlavní logiku daného kódu. Druhou nejvyšší prioritu má podobnost na

základě obecného počtu proměnných a tagů neboli podobnost celkového použitého počtu deklarovaných proměnných a tagů. Nejnižší prioritou je podobnost obecných vlastností, která shrnuje dílčí podobnosti počtu znaků, řádků a číslic.

Výsledky přiřazování funkcí jsou ukládány do buňkového pole, v prvním sloupci jsou indexy funkcí prvního z porovnávaných kódů, druhý sloupec tvoří nejvyšší nalezené hodnoty podobnosti na základě klíčových slov, následující sloupec je vyplněn indexy funkcí druhého kódu, pro které byla nalezena daná maximální podobnost klíčových slov. Do čtvrtého sloupce jsou postupně ukládány přiřazené indexy nejvíce podobných funkcí z druhého kódu. Další sloupce jsou vyplněny vypočtenými hodnotami podobnosti přiřazených funkcí.

V první iteraci (obr. 4.10) přiřazování funkcí jsou z buňkového pole dosažených výsledků podobností, pro všechny funkce, vybrány pouze funkce s nejvyšší hodnotou podobnosti klíčových slov. Vybrané hodnoty podobností jsou seřazeny sestupně.

'# fce kod_1'	'max podobnost klic.slov'	'# nejvic podobnych fci'	'# fce nejvic podobna kod_2'
3	1	[4;6;9;11;13]	[]
5	1	[4;6;9;11;13]	[]
7	0.7875	7	[]
6	0.6393	7	[]
2	0.5778	1	[]
4	0.5778	1	[]
1	0.4213	2	[]

Obr. 4.10: První iterace přiřazení podobných funkcí.

Ve druhé iteraci (obr. 4.11) jsou postupně procházeny přiřazené indexy funkcí a jsou odstraněny veškerá duplicitní přiřazení. Každá funkce může být přiřazena pouze jednou.

'# fce kod_1'	'max podobnost klic.slov'	'# nejvic podobnych fci'	'# fce nejvic podobna kod_2'
3	1	[4;6;9;11;13]	4
5	1	[6;9;11;13]	6
7	0.7875	7	7
6	0.6393	[]	[]
2	0.5778	1	1
4	0.5778	[]	[]
1	0.4213	2	[]

Obr. 4.11: Druhá iterace přiřazení podobných funkcí.

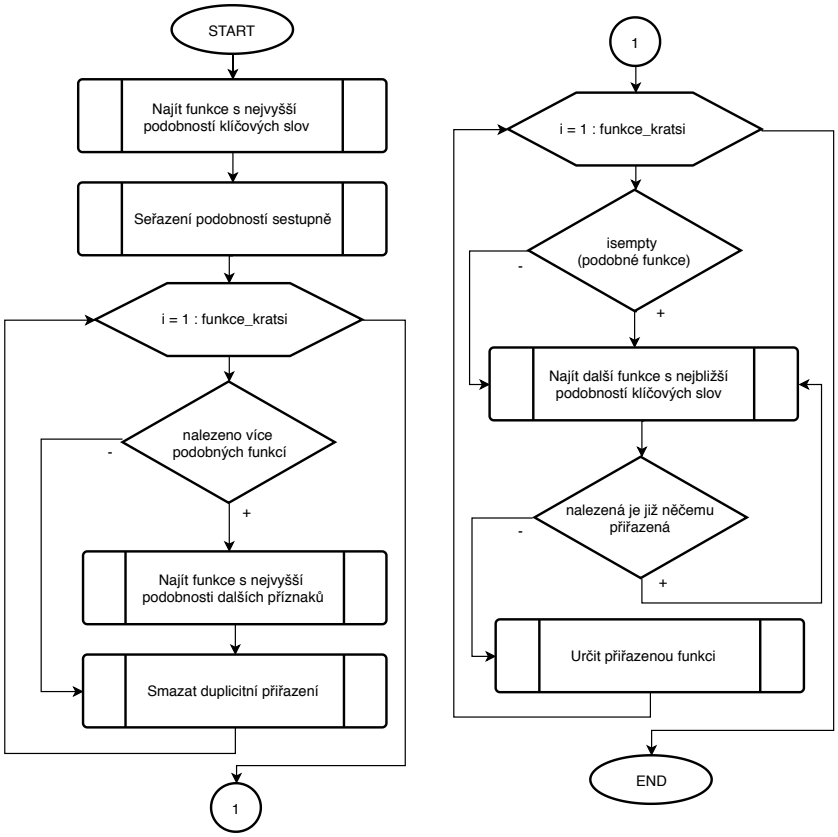
Pokud dojde k odstranění indexu přiřazené funkce, vybere se z buňkového pole podobností index funkce s další nejvyšší hodnotou podobnosti klíčových slov. Tento krok se opakuje, dokud není nalezen index dosud nepřiřazené funkce.

'# fce kod_1'	'max podobnost klic.slov'	'# nejvic podobnych fci'	'# fce nejvic podobna kod_2'
3	1	[4;6;9;11;13]	4
5	1	[6;9;11;13]	6
7	0.7875	7	7
6	0.5533	[3;5]	3
2	0.5778	1	1
4	0.5611	5	5
1	0.4213	2	2

Obr. 4.12: Výsledné přiřazení funkcí.

Pokud po první a druhé iteraci není každé funkci prvního kódu přiřazena pouze jedna funkce kódu druhého, určí se výsledné přiřazení na základě nejvyšší hodnoty další podobnosti prostřednictvím jedné z následujících podmínek v daném pořadí:

1. rozhodnutí na základě podobnosti obecného počtu tagů a proměnných,
2. rozhodnutí na základě podobnosti obecných vlastností.



Obr. 4.13: Vývojový diagram přiřazování funkcí.

Výsledné podobnosti jsou průměrnými hodnotami vypočtenými z dílčích podobností přiřazených funkcí. Míra výsledné podobnosti je ovlivněna počtem funkcí, na základě kterých byla podobnost vypočtena, a proto je nutné přihlížet k poměru

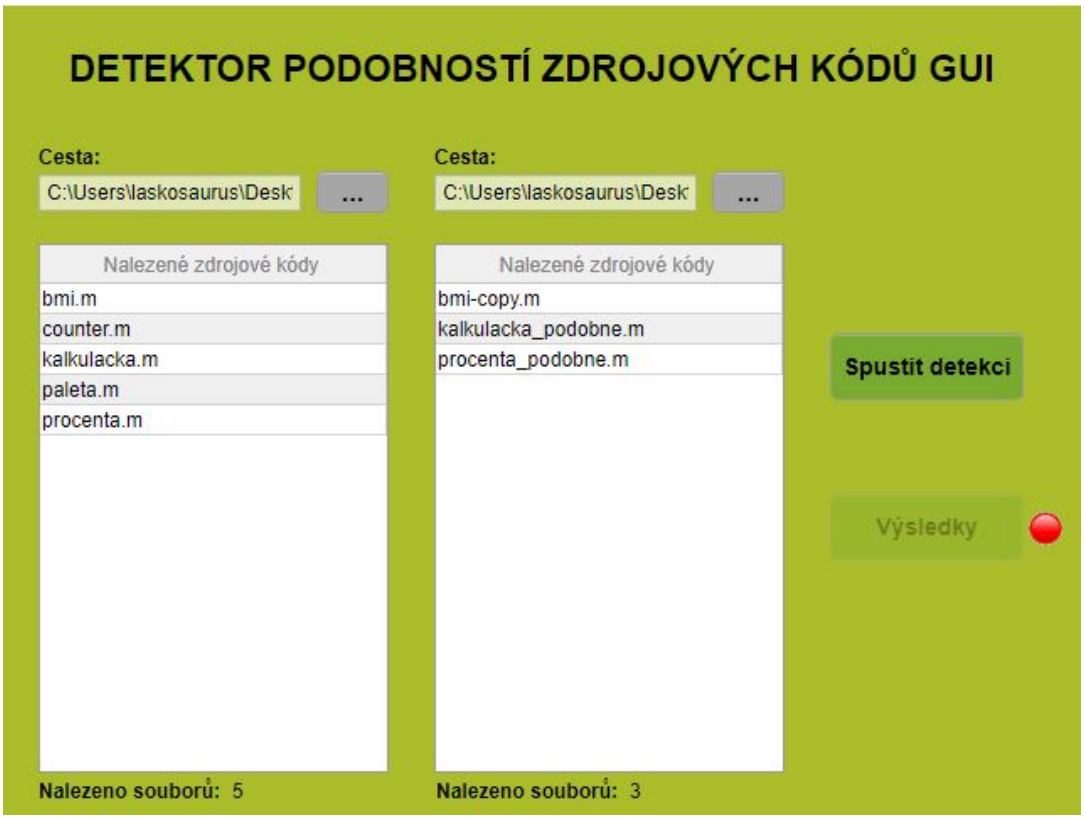
počtu funkcí zdrojových kódů. Výstupem detektoru podobností je skupina nalezených podobností ve zdrojových kódech *GUI*. Výsledky nebyly z důvodu zamezení vzájemného ovlivnění hodnot sjednoceny do podoby jednoho čísla.

4.5 Grafické uživatelské rozhraní detektoru

Vytvořenému detektoru podobností bylo pro snazší a přehlednější ovládání navrženo grafické uživatelské rozhraní. Uživateli umožňuje porovnání hned několika zdrojových kódů mezi sebou. Vzájemnou podobnost zdrojových kódů je možné stanovit pro kódy obsažené v jednom adresáři či ve dvou samostatných adresářích.

Detektor na vstupu automaticky vybere pouze zdrojové kódy grafických uživatelských rozhraní. Pro ostatní zdrojové kódy by bylo jejich porovnání rovněž možné, ale jelikož je detektor navržený výhradně pro zdrojové kódy *GUI*, dosažené výsledky by nemusely být přesné.

V úvodním okně uživatel vybere, jakým způsobem chce zdrojové kódy porovnat. Po kliknutí na příslušné tlačítko se mu zobrazí odpovídající okno pro výběr cesty k jednomu nebo ke dvěma adresářům (obr. 4.14), podle toho, jakou možnost uživatel v úvodním okně zvolil.

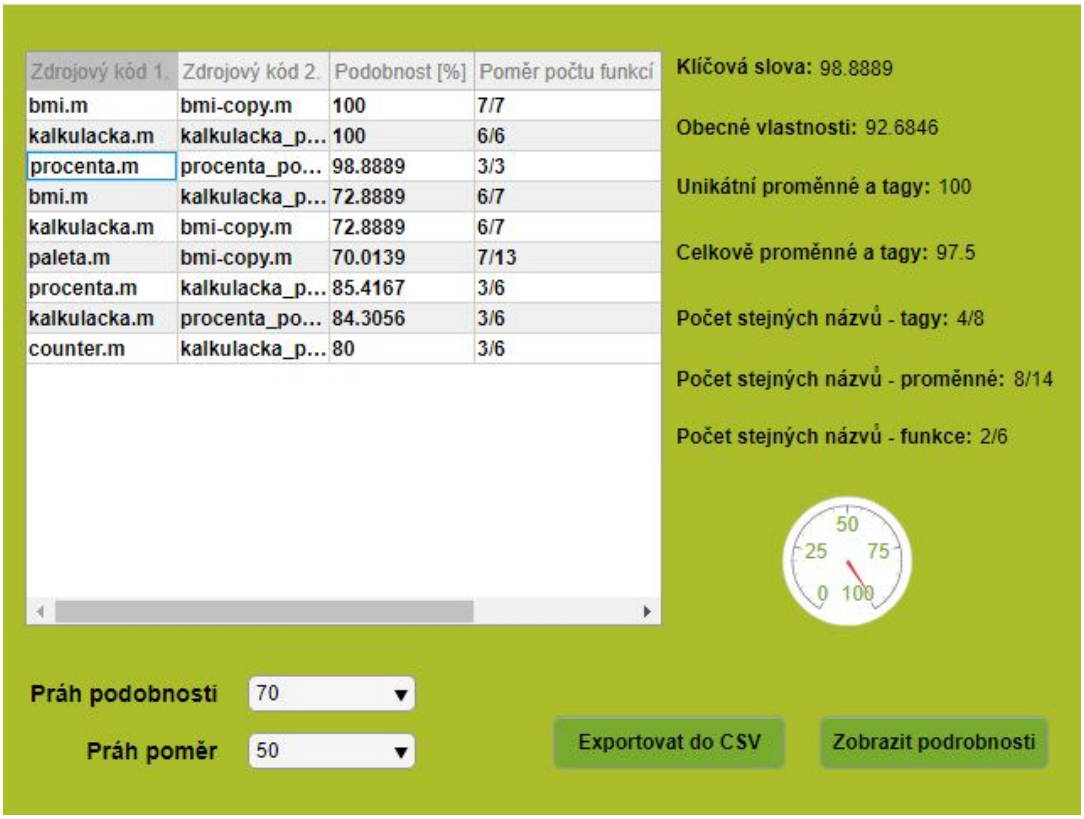


Obr. 4.14: Okno pro výběr cesty k adresářům.

Pro zvolení cesty uživatel klikne na tlačítko pro výběr cesty, všechna ostatní tlačítka jsou dosud neaktivní. Pokud uživatel zvolil možnost porovnat dva adresáře, vybere cesty k oběma těmto adresářům. Následně proběhne vyhledání všech zdrojových kódů *GUI* nacházejících se ve vybraném adresáři či adresářích.

Po dokončení nahrávání souborů se aktivuje tlačítko **`Spustit detekci`**, po kliknutí na toto tlačítko se všechny soubory předzpracují, nadetekují se v nich příznaky a vyhodnotí se jejich podobnost.

Po dokončení procesu detekce lze tlačítkem **`Výsledky`** zobrazit dosažené výsledky podobnosti. Výsledky se pro přehlednost zobrazí v dalším okně (obr. 4.15).



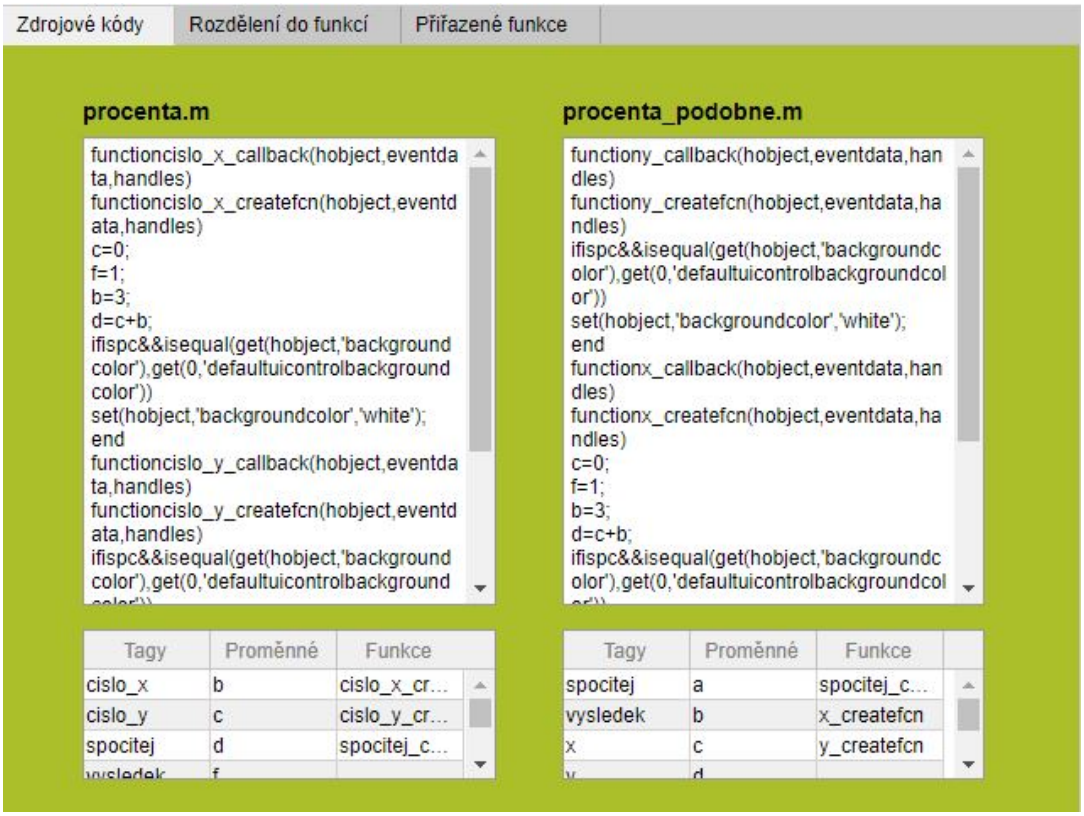
Obr. 4.15: Zobrazení výsledků s nastaveným filtrem prahu.

Grafické uživatelské rozhraní zobrazuje dosažené výsledky podobností pro všechny porovnávané kódy. V tabulce výsledků tvoří první dva sloupce názvy zdrojových kódů, které byly porovnány, v dalším sloupci je zobrazena nalezená podobnost kódů na základě klíčových slov, ve čtvrtém sloupci jsou hodnoty poměrů počtu funkcí v kódech a v posledním pátém sloupci se nachází vypočtená hodnota váhované podobnosti (z třetího sloupce) poměrem počtu funkcí (ze sloupce čtvrtého). Tabulka je seřazena sestupně od nejvyšší hodnoty podobnosti (třetí sloupec). Výběr výsledků, které chce uživatel zobrazit, lze definovat pomocí nastavitelných prahů (obr. 4.15) na základě podobnosti či na základě poměru počtu funkcí.

Hlavní tabulku výsledků lze tlačítkem **`Exportovat do CSV`** uložit do souboru s příponou `.csv`.

Vpravo vedle hlavní tabulky, jsou zobrazeny dílčí podobnosti (např. podobnost na základě klíčových slov, obecných vlastností apod.). Přehled odpovídá vždy informacím porovnávaných kódů uvedených na řádku v hlavní tabulce, který uživatel zvolí. Při výběru řádku v tabulce s výsledky se aktivuje tlačítko **`Zobrazit podrobnosti`**, které otevře další okno (obr. 4.16), ve kterém jsou zobrazeny informace týkající se pouze kódů vybraných z tabulky.

V první záložce okna podrobností se zobrazí oba předzpracované kódy. V dolní části okna se nachází výpisy všech použitých názvů tagů, proměnných a funkcí v daných kódech.



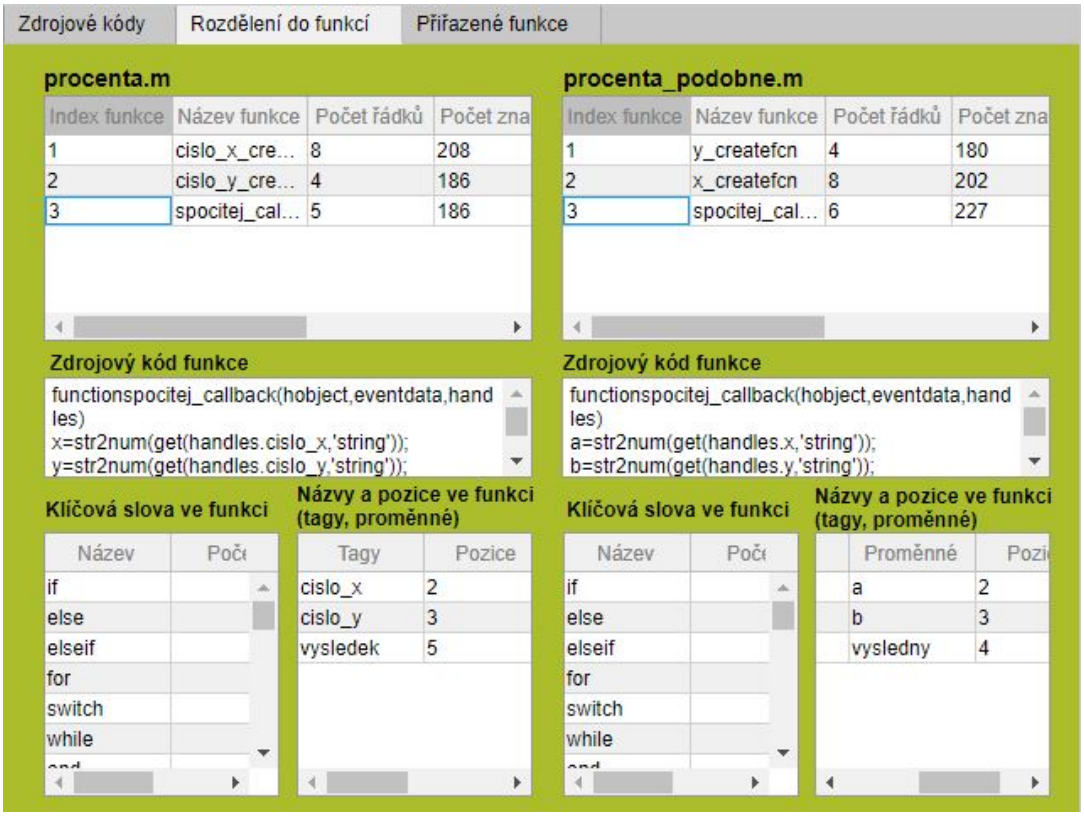
Obr. 4.16: Okno s podrobnostmi porovnávaných zdrojových kódů.

Další záložka (obr. 4.17) zobrazuje rozdělení kódů do jednotlivých funkcí. Celé okno je rozděleno na dvě poloviny, přičemž každá z nich odpovídá příslušnému zdrojovému kódu uvedeného v hlavičce. Všechny tabulky zobrazené pod tímto názvem se vztahují právě k danému kódu.

Hlavní tabulka zobrazuje v prvním sloupci indexy funkcí v kódu a v druhém sloupci názvy jednotlivých funkcí. Další sloupce tabulky jsou vyplněny nadetekovanými příznaky (počet řádků, znaků, číslic, tagů a proměnných).

Veškeré tabulky uvedené níže jsou vždy načteny pro jednu zvolenou funkci z hlavního přehledu. Výběr funkce uživatel provede kliknutím do odpovídajícího řádku tabulky příslušné funkce, pro kterou chce zobrazit podrobnosti. První ze zobrazených podrobností je předzpracovaný kód zvolené funkce.

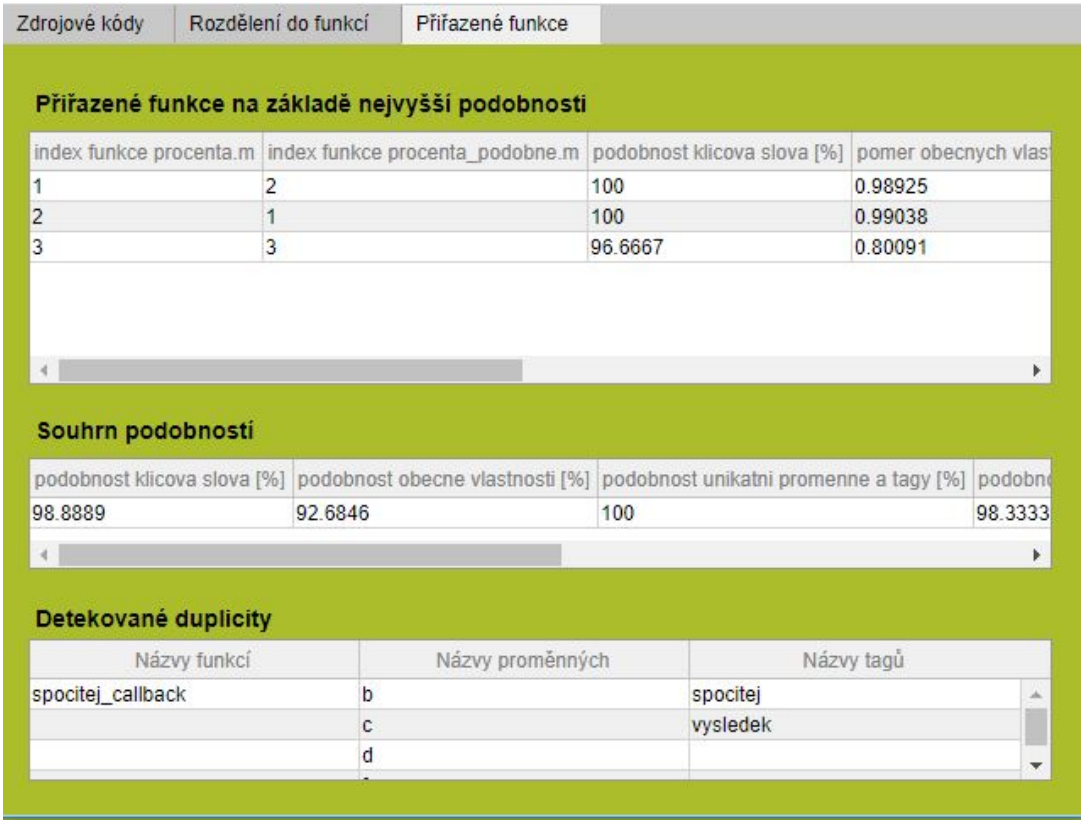
V dolních dvou tabulkách jsou přehledně zobrazeny další nalezené příznaky ve funkci. Tabulka ***Klíčová slova ve funkci*** zobrazuje výpis detekovaných klíčových slov, jejich nalezený počet a pozice řádků, na kterých se v příslušné funkci nachází. V tabulce ***Názvy a pozice ve funkci (tagy, proměnné)*** jsou uvedeny nalezené názvy tagů, proměnných a indexy řádků, na kterých se ve funkci nachází.



Obr. 4.17: Druhá záložka okna – *Rozdělení do funkcí*.

Poslední záložka (obr. 4.18) umožňuje zobrazit přehled přiřazených funkcí na základě jejich nejvyšší podobnosti. Horní tabulka ***Přiřazené funkce na základě nejvyšší podobnosti*** obsahuje indexy funkcí v příslušných zdrojových kódech a dále veškeré nalezené podobnosti těchto funkcí. Zobrazuje také počty shodujících se názvů proměnných a stejnost či nestejnost názvů funkcí.

Pod hlavní tabulkou se nachází celkový souhrn podobností a výčet nalezených duplicit názvů funkcí, tagů i proměnných.



Obr. 4.18: Záložka okna podrobností s přiřazenými funkcemi.

5 Ověření funkčnosti vytvořeného detektoru

Nedílnou součástí praktické části bakalářské práce bylo ověření funkčnosti vytvořeného detektoru podobností zdrojových kódů *GUI*. Detektor byl otestován na databázi zdrojových kódů existujících grafických uživatelských rozhraní. K testování byla využita databáze studentských projektů z předmětu *APRG*.

5.1 Testovací databáze

Využitá testovací databáze studentských prací je tvořena 78 zdrojovými kódy *GUI*. Databáze však původně neobsahuje žádné odhalené plagiáty. Podobnosti některých zdrojových kódů v databázi byly uměle vytvořeny. Pro tři zdrojové kódy *GUI* byly vytvořeny naprosté kopie.

Dále bylo vytvořeno osm rozdílně podobných zdrojových kódů *GUI*. K jejich realizaci bylo využito různých záměn ve zdrojových kódech reálně využívaných při tvorbě plagiátů. Jedná se zejména o změny v pořadí dílčích funkcí, změny názvů proměnných, připsání či vymazání části kódu, změny v klíčových slovech, přidání komentářů, mezer apod.

```
TESTOVACI_DATABAZE
├── projekty_aprg ..... testovací databáze studentským projektů.
├── kopie ..... adresář kopií zdrojových kódů GUI některých prací.
│   ├── hra_lode_kopie.m
│   ├── kodovani_kopie.m
│   └── clovece_nezlobse_kopie.m
├── bez_zmen_klic_slov .... adresář podobných kódů bez změn klíčových slov.
│   ├── candy_crash_podobne.m
│   ├── vyherni_automat_podobne.m
│   ├── prevodnik_podobne.m
│   ├── vrh_podobne.m
│   └── hanojske_veze_podobne.m
└── zmeny_klic_slov ..... adresář podobných kódů se změnami klíčových slov.
    ├── barvocit_podobne.m
    ├── sifry_podobne.m
    └── vlajky_aplikace_podobne.m
```

Obr. 5.1: Testovací databáze studentských prací s vytvořenými podobnostmi.

5.2 Vyhodnocení dosažených výsledku

Z dosažených výsledků detektoru na testovací databázi je vyhodnocena hodnota senzitivity a specificity. Tyto hodnoty charakterizují kvalitu navrženého vyhodnocení podobnosti vytvořeného detektoru. Senzitivita (5.1) udává procentuální úspěšnost nalezení podobných zdrojových kódů ze všech podobných zdrojových kódů. Vzorec pro specificitu je podílem úspěšně odhalených podobností *TP* a součtu úspěšně odhalených podobností *TP* s nenalezenými podobnostmi *FN*. Specificita (5.2) určuje kolik procent z porovnávaných kódů bylo správně vyhodnoceno jako „nepodobné“ zdrojové kódy. Vzorec popisující specificitu je podílem pravdivě označených nepodobných kódů *TN* vůči součtu pravdivě označených nepodobných kódů *TN* s falešně detekovanými podobnostmi *FP*.

$$\mathbf{SEN} = \frac{\mathbf{TP}}{\mathbf{TP + FN}} \cdot \mathbf{100} \tag{5.1}$$

$$\mathbf{SPE} = \frac{\mathbf{TN}}{\mathbf{TN + FP}} \cdot \mathbf{100} \tag{5.2}$$

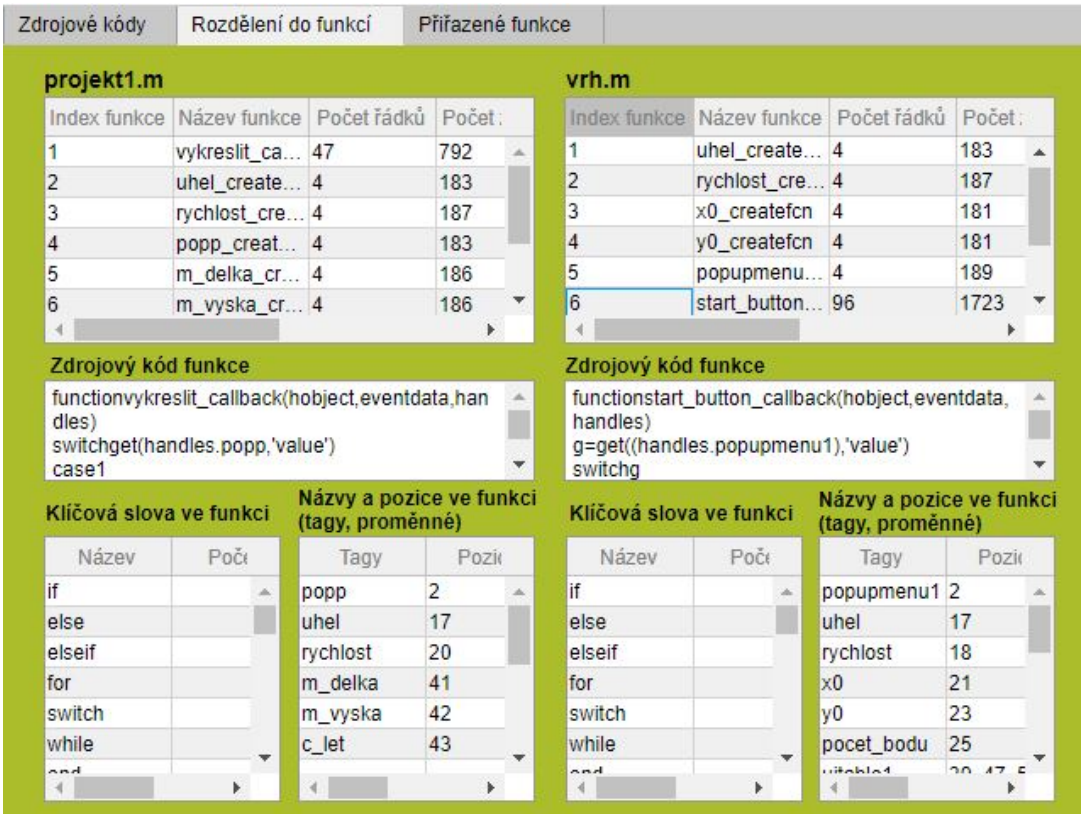
Výsledky testování jsou shrnuty v tab. 5.1. Hlavním parametrem nalezených podobností kódu je podobnost na základě klíčových slov. Pro stanovení těchto výsledků bylo počítáno s podobnostmi na základě klíčových slov váhovanými poměrem počtu funkcí, jelikož 100% podobnost může být například nalezena pouze v jedné z několika funkcí, tudíž se ve výsledku již nejedná o 100% podobnost.

Tabulka dosažených výsledků na testovací databázi		
Výsledné hodnoty	Podobnost klíč. slov	Všechny podobnosti
Zdrojové kódy *GUI*	78	78
Provedeno porovnání	3003	3003
Vytvořeno podobností	11	11
Detekováno podobností	36	11
Hodnota *TP*	12	12
Hodnota *FP*	24	0
Hodnota *TN*	2967	2991
Hodnota *FN*	0	0
Senzitivita (*SEN*)	100 %	100 %
Specificita (*SPE*)	99,1976 %	100 %

Tab. 5.1: Výsledky ověření funkčnosti detektoru.

Z dosažených výsledků lze usoudit, že detektor odhalil veškeré vytvořené podobnosti zdrojových kódů. Jednotlivé podobnosti kopií zdrojových kódů byly správně všechny označeny hodnotou 100 %, byly nalezeny veškeré identické použité názvy proměnných, tagů i funkcí. Dílčí podobnosti u vytvořených podobných kódů byly rovněž odhaleny. Doplněné komentáře či mezery v kódu byly správně procesem předzpracování odstraněny a na samotné porovnání kódů již neměly vliv. Vzhledem k tomu, že u pěti vytvořených podobných kódů nebyla změněna logika samotných kódů, detektor správně vyhodnotil maximální shodu na základě klíčových slov. Změna pořadí jednotlivých funkcí kódu, byla rovněž odhalena, vzájemně byly přiřazeny odpovídající funkce bez ohledu na jejich pořadí v kódu. Odlišnost v názvech funkcí, tagů a proměnných byla rovněž správně nalezena. U tří realizovaných podobných kódů bylo využito i změn na základě klíčových slov, zejména změny v jejich počtu a pořadí v jednotlivých funkcích. Dále byla např. odstraněna některá z dílčích funkcí kódu.

Nově byly odhaleny dva značně podobné zdrojové kódy (obr. 5.2), hodnota podobnosti na základě klíčových slov byla vyhodnocena – 82,5613 %. Jedná se o projekt identického zadání – výpočet trajektorie vrhu tělesa.



Obr. 5.2: Výsledky testování – nově nalezená podobnost zdrojových kódů.

Oba kódy obsahují identický počet funkcí, počty a pozice nalezených klíčových slov v jednotlivých funkcích jsou velice podobné. Pro některé funkce jsou použity stejné názvy, shody se vyskytují rovněž v názvech proměnných i některých tagů.

```
1  functionvykreslit_callback(hobject,eventdata,handles)
2  switchget(handles.popp,'value')
3  case1
4  g=9.81
5  case2
6  g=273.8
7  case3
8  g=23.12
9  case4
10 g=8.96
11 case5
12 g=3.7
13 case6
14 g=1.62
15 otherwise
16 end
17 uhel=get(handles.uhel,'string');
18 uhel=str2num(uhel);
19 uhel=degtorad(uhel);
20 rychlost=get(handles.rychlost,'string');
21 rychlost=str2num(rychlost);
22 [t]=cas_letu(uhel,rychlost,g);
```

```
1  functionstart_button_callback(hobject,eventdata,handles)
2  g=get((handles.popupmenu1),'value')
3  switchg
4  case1
5  g=9.81
6  case2
7  g=1.62
8  case3
9  g=3.7
10 case4
11 g=8.87
12 case5
13 g=3.7
14 case6
15 g=273.8
16 end
17 uhel=get((handles.uhel),'string')
18 v0=get((handles.rychlost),'string')
19 uhel=str2num(uhel)
20 v0=str2num(v0)
21 x0=get((handles.x0),'string')
22 x0=str2num(x0)
23 y0=get((handles.y0),'string')
24 y0=str2num(y0)
25 pocet_bodu=get((handles.pocet_bodu),'string')
26 pocet_bodu=str2num(pocet_bodu)
27 radian=(uhel*pi)/180
28 [td]=doba_letu(v0,radian,g,y0)
```

Obr. 5.3: Výsledky testování – část velmi podobné funkce nově nalezených podobných zdrojových kódů.

Pro výpočet hodnot senzitivity a specificity byla určena prahová hodnota váhované podobnosti na základě klíčových slov 80 %. Výsledná hodnota senzitivity je 100 %, což vypovídá o skutečnosti, že byly úspěšně nalezeny všechny vytvořené podobnosti zdrojových kódů a žádná nebyla opomenuta, navíc byl nově odhalen jeden případ podobnosti zdrojových kódů. Hodnota specificity je 99,1976 %, nalezené falešně pozitivní podobnosti byly detekovány zejména u zdrojových kódů vytvořených jedním autorem.

Jednalo se o funkčně zcela stejné zdrojové kódy, které autor využívá pro opakované použití v různých případech stejného chování (například `let_odlet_kveten.m` a `let_prilet_kveten.m` nebo `GUI_sudoku_plocha` a `GUI_sudoku_plocha_lehka`). Dále byla falešně pozitivní podobnost vyhodnocena v jednom případě zdrojových kódů tvořených téměř identickým počtem funkcí. V tomto případě byla nalezena určitá shoda v některých klíčových slovech a detektor tyto kódy vyhodnotil jako podobné. Jelikož je v tomto případě poměr funkcí roven skoro hodnotě jedna, nezmění se hodnota podobnosti ani po váhování. Nalezený počet falešně pozitivních detekcí podobnosti mohl být způsoben zejména tím, že pro vyhodnocení byla brána v úvahu pouze hodnota podobnosti na základě klíčových slov váhovaná poměrem

počtu funkcí. Pokud byly následně zahrnuty i další jednotlivé podobnosti, počet falešně pozitivních podobností klesl na hodnotu 0 a celková hodnota specificity nabyla hodnoty 100 %.

Většinu databáze tvoří zcela odlišné zdrojové kódy, čemuž odpovídá nalezená vysoká hodnota *TN*. Výsledné vyhodnocení podobnosti by bylo možné dále sofistikovaněji upravovat a vylepšovat, čímž by mohl detektor dosáhnout vyšších statistických hodnot.

Porovnání všech 78 zdrojových kódů *GUI* mezi sebou trvalo při testování 38 minut. Detektor provedl celkově 3003 operací. Delší doba vyhodnocení podobnosti je způsobena porovnáváním všech souborů vzájemně (mezi sebou) a zejména navrženou logikou přiřazování nejvíce podobných funkcí, kdy se porovnávají veškeré funkce vzájemně (např. pokud jeden kód obsahuje 63 funkcí a druhý kód 10 funkcí, provede se z důvodu nalezení a přiřazení nejvíce podobných dvojic celkově 630 vzájemných porovnání funkcí).

Zdrojovy_kod_1	Zdrojovy_kod_2	Podobnost	Pomer_poctu_funkci	Vysledna_podobnost
clovece_nezlobse_kopie.m	clovece_nezlobse.m	100	'13/13'	100
hra_lode_kopie.m	hra_lode.m	100	'5/5'	100
kodovani_kopie.m	kodovani.m	100	'8/8'	100
candy_crash_podobne.m	candy_crash.m	100	'3/3'	100
hanojske_veze_podobne.m	hanojske_veze.m	100	'13/13'	100
prevodnik_podobne.m	prevodnik.m	100	'11/11'	100
vrh_podobne.m	vrh.m	100	'9/9'	100
vyherni_automat_podobne.m	vyhedni_automat.m	100	'11/11'	100
vlajky_aplikace_podobne.m	vlajky_aplikace.m	99,1151	'63/63'	99,1151
sifry_podobne.m	sifry.m	97,4703	'12/12'	97,4703
barvocit_podobne.m	barvocit.m	100	'18/19'	94,7368

Obr. 5.4: Výsledky testování – správně nalezené vytvořené podobné zdrojové kódy.

6 Závěr

Cílem bakalářské práce bylo vytvoření funkčního detektoru podobností zdrojových kódů grafických uživatelských rozhraní.

V rámci analýzy potřeb k řešení zadání bakalářské práce bylo nejprve nutné seznámit se s pojmem plagiátorství, konkrétně plagiátorstvím ve zdrojových kódech se zaměřením na možné způsoby vytváření plagiátů. Dále bylo zapotřebí obeznámit se s tvorbou *GUI* ve vývojovém prostředí *MATLAB* a podobou zdrojového kódu grafického uživatelského rozhraní. Nastudované předpoklady pro řešení zadání bakalářské práce jsou shrnuty v teoretické části.

V úvodu praktické části je shrnuta celková funkčnost vytvořeného detektoru. Další části se věnují konkrétním krokům detekce podobnosti zdrojových kódů vytvořeného detektoru.

První část pojednává o předzpracování programových kódů, představuje použitý postup předzpracování a zmiňuje nezbytnost tohoto kroku pro následnou detekci příznaků. Navazující část se zaměřuje na stanovení příznaků a jejich detekci ve zdrojovém kódu. Dále je objasněna použitá metrika příznaků a výsledné vyhodnocení podobnosti zdrojových kódů *GUI*.

V rámci praktické části je představeno navržené grafické uživatelské rozhraní vytvořeného detektoru podobností, které umožňuje jednoduchou práci s detektorem. Přehledně zobrazuje dosažené výsledky podobností, umožňuje jejich export do formátu `.csv`, poskytuje možnost zobrazení předzpracované podoby porovnávaných kódů a tím i možnost vizuálního zhodnocení podobnosti kódů uživatelem.

Závěrem praktické části je zhodnoceno ověření funkčnosti vytvořeného detektoru podobnosti zdrojových kódů grafických uživatelských rozhraní na existující databázi studentských prací. Na testovací databázi bylo dosaženo kvalitních výsledků. Byla určena hodnota senzitivity 100 % a hodnota specificity 99,1976 % při posuzování pouze podobnosti na základě klíčových slov a následně hodnota specificity 100 % v případě posuzování všech dílčích nalezených podobností. Detektor byl schopen odhalit veškeré vytvořené podobnosti zdrojových kódů *GUI*. Vyhodnocení vzájemného porovnání 78 zdrojových kódů *GUI* detektor prováděl 38 minut. Delší doba vyhodnocení podobnosti je způsobená především komplexností navrženého programu. Dosažené výsledky vytvořeného detektoru nebyly porovnány s volně dostupnými softwary pro odhalování podobnosti, jelikož tyto softwary nebyly schopny detekovat plagiátorství zdojových kódů grafických uživatelských rozhraní. Porovnání se softwary nebylo realizováno rovněž z důvodu nemožnosti těchto volně dostupných softwarů porovnat několik zdrojových kódů mezi sebou současně.

Vytvořený detektor je plně automatizovaný program. Uživatelským vstupem detektoru je pouze cesta k adresářům obsahující zdrojové kódy. Detektorem jsou nalezeny veškeré kódy nacházející se ve všech podadresářích a automaticky vybrány pouze zdrojové kódy *GUI*. Využitím detektoru se usnadňuje proces nalezení vzájemných podobností zdrojových kódů. Je možné porovnávat zdrojové kódy mezi sebou v jednom adresáři, čímž je například umožněno nalézt podobnosti zdrojových kódů v rámci odevzdaných studentských programů s jednotným zadáním. Další možností je porovnání zdrojových kódů jednoho adresáře se všemi kódy jiného adresáře. Tímto způsobem může být např. porovnána databáze starších projektů s aktuálně odevzdanými projekty.

Všechny zadané cíle bakalářské práce byly úspěšně splněny, vytvořený detektor je plně funkční a je možné pracovat na jeho vývoji a dalším rozšiřování.

Literatura

[1] Plagiátorství. *Masarykova univerzita: Úřední deska* [online]. [cit. 2019-01-01]. Dostupné z: https://www.muni.cz/o-univerzite/uredni-deska/plagiatorstvi

[2] ČSN ISO 5127-2003. *Informace a dokumentace - Slovník: Akvizice, identifikace a analýza dokumentů a dat*. Praha: Český normalizační institut, 2003.

[3] Plagiarism. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001 [cit. 2019-01-01]. Dostupné z: https://en.wikipedia.org/wiki/Plagiarism

[4] *Problematika plagiátorství a ochrany autorských práv ve školních dílech*. Praha, 2009. Dostupné také z: https://www.vscht.cz/files/uzel/0001584/Problematika+plagi%C3%A1torstv%C3%AD+a+ochrany+autorsk%C3%BDch+pr%C3%A1v+ve.pdf?redirected

[5] ČESKÁ REPUBLIKA. Zákon o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon). In: *Zákon č. 121/2000 Sb.* 2000, ročník 20, 36/2000, číslo 121. Dostupné také z: https://www.zakonyprolidi.cz/cs/2000-121

[6] JANSA, Lukáš a Petr OTEVŘEL. *Softwarové právo: praktický průvodce právní problematikou v IT*. Brno: Computer Press, 2011, 340 s. : il., portréty. ISBN 978-80-251-3458-0.

[7] KRČÁL, Martin a Zuzana TEPLÍKOVÁ. *Naučte (se) citovat*. Blansko: Citace.com, 2014. ISBN 978-80-260-6074-1.

[8] KRČÁL, Martin, Karolína KRBCOVÁ a Jakub HORÁK. *Citujte jednoduše: Instrukce pro vyučující středních škol* [online]. Brno: Ucimcitace.cz, 2015 [cit. 2019-01-01]. Dostupné z: https://www.citace.com/download/Citujte-jednoduse.pdf

[9] PARKER, A. a J. O. HAMBLEN. Computer algorithms for plagiarism detection. *IEEE Transactions on Education* [online]. 1989, Květen 1989, 94-99 [cit. 2019-01-01]. DOI: 10.1109/13.28038. ISSN 1557-9638. Dostupné z: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=28038

[10] FAIDHI, J.A.W. a S.K. ROBINSON. An empirical approach for detecting program similarity and plagiarism within a university programming environment. *Computers & Education* [online]. Elsevier, 1987, 11-19 [cit. 2019-01-01]. DOI: 10.1016/0360-1315(87)90042-X. Dostupné z: `https://www.sciencedirect.com/science/article/pii/036013158790042X?via%3Dihub`

[11] CLOUGH, Pau. *Old and new challenges in automatic plagiarism detection* [online]. Velká Británie, 2003 [cit. 2019-01-01]. Dostupné z: `https://ir.shef.ac.uk/cloughie/papers/pas_plagiarism.pdf`. University of Sheffield.

[12] WHALE, G. Identification of Program Similarity in Large Populations. *The Computer Journal* [online]. 1. leden 1990, 140-146 [cit. 2019-01-01]. Dostupné z: `https://doi.org/10.1093/comjnl/33.2.140`

[13] BURROWS, Steven. *Efficient and effective plagiarism detection fot large code repositories* [online]. Melbourne, Australia, 2004 [cit. 2019-01-01]. Dostupné z: `https://www.researchgate.net/publication/230897316_Efficient_and_Effective_Plagiarism_Detection_for_Large_Code_Repositories`. Diplomová práce. School of Computer Science and Information Technology RMIT University. Vedoucí práce Seyed M. M. Tahaghoghi a Justin Zobel.

[14] AIKEN, Alex. MOSS: A System for Detecting Software Similarity. *Stanford Theory Group* [online]. 2010 [cit. 2019-01-01]. Dostupné z: `http://theory.stanford.edu/~aiken/moss/`

[15] BOWYER, Kevin W. a Lawrence O. HALL. *Experience Using -MOSS- to Detect Cheating On Programming Assignments* [online]. Florida [cit. 2019-01-01]. Dostupné z: `https://www3.nd.edu/~kwb/nsf-ufe/1110.pdf`. Department of Computer Science and Engineering University of South Florida.

[16] JPlag: Detecting Software Plagiarism. *KIT: Karlsruhe Institute of Technology* [online]. [cit. 2019-01-01]. Dostupné z: `https://jplag.ipd.kit.edu/`

[17] PRECHELT, Lutz, Guido MALPOHL a Michael PHLIPPSEN. *JPlag : Finding plagiarisms among a set of programs* [online]. Německo, 2000 [cit. 2019-01-01]. Dostupné z: `http://page.mi.fu-berlin.de/prechelt/Biblio/jplagTR.pdf`. Technická zpráva. Fakultat fur Informatik Universitat Karlsruhe.

[18] ARWIN, Christian a S. M. M. TAHOGHOGHI. *Plagiarism detection across Programming Languages* [online]. Melbourne, Australia [cit. 2019-01-01]. Dostupné z: https://pdfs.semanticscholar.org/3637/24feadea3ffcfc52d10f267f29ce7b68c728.pdf?_ga=2.130824716.207349206.1545759250-2120575924.1545759250. School of Computer Science and Information Technology RMIT University.

[19] GITCHELL, David a Nicholas TRAN. *Sim: A Utility For Detecting Similarity in Computer Programs* [online]. Kansas [cit. 2019-01-01]. Dostupné z: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.452.9879&rep=rep1&type=pdf. Department of Computer Science Wichita State University.

[20] JOY, Mike. *Detecting Source-Code Plagiarism* [online]. University of Warwick: Department of Computer Science [cit. 2019-01-01]. Dostupné z: http://cei.ust.hk/files/public/detecting_source-code_plagiarism.pdf

[21] Sherlock - Plagiarism Detection Software. *Warwick: The University of Warwick* [online]. [cit. 2019-01-01]. Dostupné z: https://warwick.ac.uk/fac/sci/dcs/research/ias/software/sherlock/

[22] *IThenticate* [online]. 1998-2018 [cit. 2019-01-01]. Dostupné z: http://www.ithenticate.com/

[23] IThenticate. In: *Lingnan University Hong Kong: Information for staff* [online]. [cit. 2019-01-01]. Dostupné z: https://www.ln.edu.hk/moodle/information-for-staff/ithenticate

[24] *Turnitin* [online]. 2019 [cit. 2019-01-01]. Dostupné z: https://www.turnitin.com/

[25] Turnitin. In: *University of Bradford: Viewing the Similarity Index / Originality Report* [online]. [cit. 2019-01-01]. Dostupné z: https://www.bradford.ac.uk/elearning/Plagiarism/Student-Guide-to-TurnitinUK/page_08.htm

[26] MATLAB GUI: Create Apps with Graphical User Interfaces in MATLAB. *MathWorks* [online]. [cit. 2019-01-01]. Dostupné z: https://www.mathworks.com/discovery/matlab-gui.html

[27] What is MATLAB?. *MathWorks* [online]. [cit. 2019-01-01]. Dostupné z: https://www.mathworks.com/discovery/what-is-matlab.html

Seznam symbolů, veličin a zkratek

FN	Počet falešně negativních detekcí
FP	Počet falešně pozitivních detekcí
GUI	Graphical User Interface
GUIDE	Graphical User Interface Development Environment
MATLAB	Matrix Laboratory
SEN	Senzitivita
SPE	Specificita
TN	Počet pravdivě negativních detekcí
TP	Počet pravdivě pozitivních detekcí
WIMP	Windows Icons Menus Pointing device

Seznam příloh

A Obsah přiloženého DVD

Přiložené *DVD* obsahuje elektronickou verzi bakalářské práce, zdrojové kódy a zkušební databázi zdrojových kódů *GUI* sloužící k ověření funkčnosti detektoru podobností. Dokument elektronické verze bakalářské práce **bp-xmatas02.pdf** je uložen v adresáři `dokumentace`. Zdrojové kódy vytvořené aplikace jsou přidány do archivu a vloženy do adresáře `zdojove_kody`. V posledním adresáři `overeni_funkcnosti` se nachází zkušební databáze zdrojových kódů *GUI* a návod pro práci s detektorem.

- `/` ..... kořenový adresář přiloženého *DVD*.
  - `dokumentace`
    - `bp-xmatas02.pdf` ..... elektronická verze bakalářské práce.
  - `zdrojove_kody`
    - `detektor_podobnosti.zip` ..... vytvořený detektor podobností.
  - `overeni_funkcnosti`
    - `zkusebni_databaze` ..... zkušební databáze zdrojových kódů *GUI*.
    - `navod.pdf` ..... kroky pro ověření funkčnosti detektoru podobností.