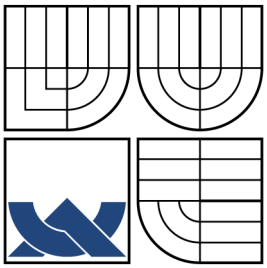
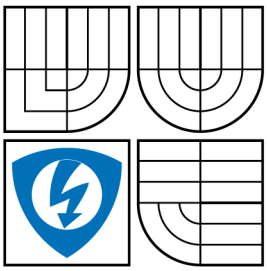


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY
A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND
COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

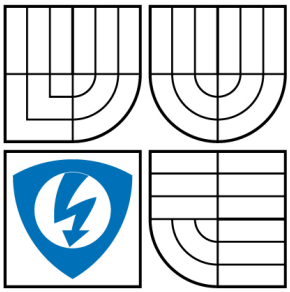
SYSTÉMY PRO KONTROLU ELEKTRONICKÝCH TEXTŮ
SYSTEMS FOR CHECKING ELECTRONIC TEXTS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — PETR ZOUHAR
AUTHOR

VEDOUCÍ PRÁCE — ING. VÁCLAV PFEIFER
SUPERVISOR

BRNO 2008



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Bakalářská práce

bakalářský studijní obor
Teleinformatika

Student:	Zouhar Petr	***ID:***	78323
Ročník:	3	***Akademický rok:***	2007/2008

NÁZEV TÉMATU:

Systémy pro kontrolu elektronických textů

POKYNY PRO VYPRACOVÁNÍ:

Kontrola autenticity dat je v dnešní době závažným problémem. Zjistěte současný stav problematiky a proveďte jednoduché měření, které srovná současně vytovořené systémy. Měření proveďte na různých souborech dat (Zdrojový text, dokumenty apod.). Co je to syntaktická a sémantická analýza a jaký má vzah ke kontrole elektronických textů ? Na základě teoretických znalostí navrhněte jednoduchou koncepci systému vlastního, který se bude aplikovat při kontrole zdrojovýxh textů v rámci programovacích předmětů na ústavu Telekomunikací. Dále navrhněte vhodný jazyk pro samotnou realizaci kontrolního systému. Proveďte další měření, která budou definovat úspešnost Vámi zvoleného systému.

DOPORUČENÁ LITERATURA:

[1] PFEIFER, V.; BALÍK, M.; ŠKORPIL, V. Current techniques assuring data authenticity. In International Conference on Computer Science and Information Technologies. 2007. s. 106-108.
[2] Spell, B.: JAVA Programujeme profesionálně. Nakladatelství CPRESS 2002. ISBN: 80-7226-667-5.

Termín zadání:	11.2.2008	***Termín odevzdání:***	4.6.2008

Vedoucí práce: Ing. Václav Pfeifer

prof. Ing. Kamil Vrba, CSc.
předseda oborové rady

UPOZORNĚNÍ:

LICENČNÍ SMLOUVA
POSKYTOVANÁ K VÝKONU PRÁVA UŽÍT ŠKOLNÍ DÍLO

uzavřená mezi smluvními stranami:

1. Pan/paní

Jméno a příjmení: Petr Zouhar

Bytem:

Narozen/a (datum a místo): 10.4.1986, Vyškov

(dále jen "autor")

a

2. Vysoké učení technické v Brně

Fakulta elektrotechniky a komunikačních technologií

se sídlem Údolní 244/53, 60200 Brno 2

jejímž jménem jedná na základě písemného pověření děkanem fakulty:

prof. Ing. Kamil Vrba, CSc.

(dále jen "nabyvatel")

Článek 1
Specifikace školního díla

1. Předmětem této smlouvy je vysokoškolská kvalifikační práce (VŠKP):
   - ☐ disertační práce
   - ☐ diplomová práce
   - ☑ bakalářská práce

   jiná práce, jejíž druh je specifikován jako .....

(dále jen VŠKP nebo dílo)

Název VŠKP: Systémy pro kontrolu elektronických textů

Vedoucí/školitel VŠKP: Ing. Václav Pfeifer

Ústav: Ústav telekomunikací

Datum obhajoby VŠKP: .....

VŠKP odevzdal autor nabyvateli v:

- ☑ tištěné formě - počet exemplářů 1
- ☑ elektronické formě - počet exemplářů 1

2. Autor prohlašuje, že vytvořil samostatnou vlastní tvůrčí činností dílo shora popsané a specifikované. Autor dále prohlašuje, že při zpracovávání díla se sám nedostal do rozporu s autorským zákonem a předpisy souvisejícími a že je dílo dílem původním.
3. Dílo je chráněno jako dílo dle autorského zákona v platném znění.
4. Autor potvrzuje, že listinná a elektronická verze díla je identická.

Článek 2

Udělení licenčního oprávnění

1. Autor touto smlouvou poskytuje nabyvateli oprávnění (licenci) k výkonu práva uvedené dílo nevýdělečně užít, archivovat a zpřístupnit ke studijním, výukovým a výzkumným účelům včetně pořizovaní výpisů, opisů a rozmnoženin.
2. Licence je poskytována celosvětově, pro celou dobu trvání autorských a majetkových práv k dílu.
3. Autor souhlasí se zveřejněním díla v databázi přístupné v mezinárodní síti
   - ☑ ihned po uzavření této smlouvy
   - ☐ 1 rok po uzavření této smlouvy
   - ☐ 3 roky po uzavření této smlouvy
   - ☐ 5 let po uzavření této smlouvy
   - ☐ 10 let po uzavření této smlouvy

   (z důvodu utajení v něm obsažených informací)
4. Nevýdělečné zveřejňování díla nabyvatelem v souladu s ustanovením § 47b zákona č. 111/1998 Sb., v platném znění, nevyžaduje licenci a nabyvatel je k němu povinen a oprávněn ze zákona.

Článek 3

Závěrečná ustanovení

1. Smlouva je sepsána ve třech vyhotoveních s platností originálu, přičemž po jednom vyhotovení obdrží autor a nabyvatel, další vyhotovení je vloženo do VŠKP.
2. Vztahy mezi smluvními stranami vzniklé a neupravené touto smlouvou se řídí autorským zákonem, občanským zákoníkem, vysokoškolským zákonem, zákonem o archivnictví, v platném znění a popř. dalšími právními předpisy.
3. Licenční smlouva byla uzavřena na základě svobodné a pravé vůle smluvních stran, s plným porozuměním jejímu textu i důsledkům, nikoliv v tísni a za nápadně nevýhodných podmínek.
4. Licenční smlouva nabývá platnosti a účinnosti dnem jejího podpisu oběma smluvními stranami.

V Brně dne: .....

....

Nabyvatel Autor

ABSTRAKT

Práce se zabývá možností kontroly elektronických textů. Ať už se jedná o zdrojové kódy či běžné textové dokumenty. První kapitola je věnována stručnému vysvětlení plagiátu a jeho znaků. V dalších částech textu popisujeme metody a metriky využívané k odhalování plagiátorů. Poté věnujeme pozornost detekování plagiátů ve volném textu a zdrojových kódech. U volného textu popisujeme způsob předzpracování souboru a výběr základních jednotek, které následně dokument zastupují při porovnávání. Zdrojové kódy mají svou přesně danou syntaxi, a proto se v kapitole popisující přístupy k jejich kontrole věnujeme syntaktické a sémantické analýze. Druhá polovina práce je zaměřena na praktickou část, zejména na programy určené ke kontrole zdrojových kódů. Programy rozdělíme na volně dostupné a komerční. Poté následuje jejich stručný popis a v případě, že umožňují bezplatné vyzkoušení, jsou u nich uvedeny výsledky porovnávání. K tomu účelu jsme vytvořili korpus zdrojových kódů. V závěru práce se věnujeme návrhu programu, který porovnává dva zdrojové kódy na základě statistické podobnosti.

KLÍČOVÁ SLOVA

Autorská práva, detekce textu, korpus, plagiát, zdrojový kód

ABSTRACT

The work deals with the possibility of control of electronic texts. Whether it is a source codes or standard text documents. The first chapter is devoted to a brief explanation of the term plagiarism and its characters. Sequentially we describe the methods and metrics used to detect plagiarist. Then we pay attention to detect plagiarism in the free text and source codes. We describe the way of preprocessing of a file and choice of basic units, which represent the document in the comparing. Source codes have a exact syntax. Therefore we attend to the syntax and semantic analysis in the chapter, which describes the check of source codes. The second half of the work is focused on the practical part, particularly on programs intended to control the source codes. The programs are divided to the freely available and the commercial. This is followed by their brief description and if it is a free trial possible we mention the results from this comparing. So we created a corpus of source codes. At the end of the work we focus on design of a program, which compares two source codes on the basis of statistical similarities.

KEYWORDS

Copyright, text detection, corpus, plagiarism, source code

ZOUHAR, P. *Systémy pro kontrolu elektronických textů*. Brno: Vysoké učení technické v Brně. Fakulta elektrotechniky a komunikačních technologií. Ústav telekomunikací, 2008. 49 s. Bakalářská práce. Vedoucí práce Ing. Václav Pfeifer.

PROHLÁŠENÍ

Prohlašuji, že svou bakalářskou práci na téma „Systémy pro kontrolu elektronických textů“ jsem vypracoval 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 autor uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 152 trestního zákona č. 140/1961 Sb.

V Brně dne ....

(podpis autora)

PODĚKOVÁNÍ

Děkuji vedoucímu bakalářské práce Ing. Václavu Pfeiferovi za cenné rady a pomoc při psaní bakalářské práce.

OBSAH

SEZNAM TABULEK

ÚVOD

Na informace o plagiátech a porušování autorských práv dnes narážíme stále častěji. Ať jde o elektronické texty, kterými se budeme zabývat (především tedy zdrojové soubory jazyka C), oblast multimedií či průmyslové výrobky, jejichž padělaním přichází renomované firmy o obrovské sumy peněz. Takto vzniklé ztráty nutí firmy bránit se proti poškozování své značky (např. snaha chránit CD a DVD nosiče proti kopírování).

S kontrolou elektronických textů se budeme setkávat stále častěji a jistě se jí bude věnovat více osob a institucí. Rozmáhání plagiátorství napomáhají texty dostupné v elektronické podobě, jejichž části nebo celé obsahy lze snadněji kopírovat. Dostupnost těchto zdrojů umožnilo zejména masivní rozšíření internetu v posledních letech, kdy počet domácností s přístupem k internetu vzrostl dle údajů Českého statistického úřadu [9] z 15 % (čtvrté čtvrtletí 2003) na 32 % (druhé čtvrtletí 2007).[1]

Na boj s plagiátory se zaměřují i školy, které začínají od studentů požadovat práce v elektronické podobě, což umožňuje rychleji porovnávat dokumenty a efektivněji odhalovat plagiáty. V případě tištěné podoby by se asi málo komu chtělo kontrolovat autentičnost odevzdané práce. Použití programů nám však většinou nedokáže dát stoprocentní odpověď na otázku, kdo je autorem a kdo plagiátorem. Problém automatické kontroly plagiátů nastává při volbě způsobu zpracování dokumentů, neboť existuje mnoho metod a různých programů, které jsou pro odhalování plagiátů určeny, ale jejich procentuální úspěšnost často není příliš vysoká. Problematické také někdy bývá stanovit, která práce je originálem a která plagiátem. Ohlédnutím za vývojem a popisem současného stavu programů a využívaných metod by se měla zaobírat tato práce. Dále je její součástí i návrh vlastního programu pro kontolu autentičnosti zdrojových souborů jazyka C.

Tato bakalářská práce navazuje na stejnojmenný semestrální projekt [8], který dále rozšiřuje zejména o praktickou část.

[1] Firma Dema a. s. [4] uvádí, že počet domácností s přístupem na internet se zvýšil dokonce ze 2 % (1997) na 57 % (konec roku 2007), přičemž pro rok 2003 udává, že k internetu bylo připojeno 32 % domácností, tedy hodnota, kterou udává ČSÚ pro polovinu roku 2007.

1 PLAGIÁT

Na úvod je vhodné definovat si pojem **plagiát**. Co v sobě toto slovo vlastně skrývá a co si pod ním představovat? Použijeme popis uvedený v internetové encyklopedii Wikipedia [12], který nám připadá stručný, ale zato výstižný: „*Plagiát je napodobenina uměleckého, literárního nebo vědeckého díla, u níž je jako původce díla (autor) uveden napodobitel, tzv. plagiátor. Při tvorbě plagiátu je zcela nebo z části použito dílo jiného autora. Tento původní autor je úmyslně nebo neúmyslně zatajen. Plagiát nemusí nutně znamenat porušení autorského zákona.*“ Lze tedy říci, že ne vždy se jedná o autorův úmysl stát se plagiátorem. Někdy si prostě jen neuvědomí, že zapomněl označit nějaký zdroj. Ovšem neznalost zákona neomlouvá.

1.1 Rozpoznání plagiátu

Rozpoznat, zda nějaká práce je plagiátem, lze podle několika podezřelých znaků. Základním znakem je samozřejmě shoda dvou dokumentů, popřípadě částí textu. Následující seznam podezřelých znaků, který je inspirován Hauzírkem [5], je impulsem pro zpozornění, a je třeba tento náznak možného plagiátorství pečlivě přezkoumat, protože ne vždy to znamená objevení plagiátu.

Mezi podezřelé znaky patří:

- Střídání různého formátování textu
- Shodné pravopisné chyby
- Shodné členění textu
- Střídání stylu, popř. i jazyka[2]
- Používání pojmů, u kterých nejsme přesvědčeni, že autor zná jejich význam.

1.2 Boj s plagiátory

První zmínky o automatické detekci se objevují v 70. letech 20. století, kdy bylo plagiátorství problémem především na akademické půdě. Vývoji programů se věnovali zejména vyučující programování a od toho se odvíjelo i zaměření prvních programů, tedy na kontrolu zdrojových kódů. Většinou však s odchodem vyučujícího či studenta, který se vývojem programu zabýval, bývá spojeno ukončení práce na tomto programu.

[2]Vlastní zkušenost z laboratoří předmětu Fyzika 1 (BFY1), kdy jeden ze studentů odevzdal protokol, ve kterém se střídala česká a slovenská část.

Kromě odhalování plagiátorů je nutné věnovat čas a úsilí také k výchově studentů, a předcházet tak tvorbě plagiátů. Ideální by bylo zařadit problematiku autorských práv do některého předmětu již na základní škole, aby se žákům dostala do podvědomí, a poté se k ní vrátit i na školách středních. Vhodným předmětem by mohla být výuka informatiky, popřípadě občanské nauky, neboť plagiátorství nesouvisí jen s výpočetní technikou. [5]

2 VYUŽÍVANÉ METODY A POSTUPY

V této kapitole se zaměříme na postupy, se kterými se u programů pro odhalování plagiátů můžeme setkat. Nástroje se liší především přístupem ke zdrojům, se kterými porovnáváme náš zkoumaný dokument. Při porovnávání zdrojových kódů ve většině případů vycházíme ze znalosti jejich syntaxe a zpracováváme je specifickým způsobem.

Pro skupinu dokumentů[3], se kterými daný program pracuje, budeme používat označení **korpus** [5]. Toto označování se objevuje i v jiných zdrojích a je vhodné, neboť „*Jazykový korpus je (většinou rozsáhlý) soubor textů...*“, jak lze nalézt např. v encyklopedii Wikipedia [10].

Nástroje pro odhalování plagiátů můžeme rozdělit na nástroje pracující s obsahem dokumentu a ty ostatní (tj. nástroje fungující jiným způsobem). Jednotlivým postupům a používaným metrikám se budeme věnovat v následujících podkapitolách, přičemž budeme vycházet z Hauzírka [5].

2.1 Nástroje pracující s obsahem dokumentu

U této metody se využívá porovnávání zkoumaného dokumentu s určitým korpusem. Nástroje se liší množstvím dokumentů (velikostí korpusu), se kterými probíhá porovnávání, a umístěním zdrojů (PC, školní síť, internet). Lze tedy očekávat, že i získané výsledky se budou lišit. Nástroje můžeme rozdělit do několika skupin, přičemž základem jsou nástroje extrakorpální a intrakorpální.

2.1.1 Nástroje extrakorpální

Extrakorpální nástroje pro porovnávání dokumentů využívají externích zdrojů (např. internet, školní databázi), neprovádí se však porovnávání dokumentů v rámci daného korpusu.

Výhodou metody je rozsáhlejší počet zdrojů, tudíž i větší pravděpodobnost nálezu případného plagiátu. Druhou výraznou výhodou extrakorpálního nástroje je porovnávání se zdroji z internetu, čímž je zajištěna aktualizace zdrojů a lze o tomto nástroji říci, že je dynamický.

Mezi nevýhody těchto nástrojů patří především delší časový úsek nutný k prohledání zdrojů a následnému porovnání se zkoumaným dokumentem. Dalším minusem

[3]Možná by bylo lepší hovořit o souborech, a ne o dokumentech, aby se zdůraznilo, že neporovnáváme pouze textové dokumenty, jako jsou PDF a DOC, ale také zdrojové kódy, které se získávají editací souborů příslušného programu.

metody je pak nutnost připojení k internetu, což také ovlivňuje rychlost porovnávání. K urychlení extrakorpálních nástrojů se využívá rychlejšího vyhledávání, které ale může snížit úspěšnost odhalování plagiátů. Provádí se proto zmenšení počtu dokumentů, se kterými bude náš zkoumaný dokument porovnáván.

2.1.2 Nástroje intrakorpální

Intrakorpální nástroje hledají možný plagiát pouze v daném korpusu. Znamená to tedy menší počet zdrojů určených pro porovnání, a tudíž i menší pravděpodobnost nalezení plagiátu.

Výhodou metody je vhledem k menšímu množství dokumentů rychlejší porovnávání souborů a možnost provést detailnější zkoumání případných shod.

Nevýhodou je již zmiňovaný omezený počet souborů, mezi nimiž dochází k vyhledávání plagiátů. Tuto nevýhodu můžeme částečně odstranit použitím většího korpusu.

2.1.3 Nástroje kombinované

Dalším nástrojem pracujícím s obsahem dokumentu je kombinace intrakorpálního a extrakorpálního nástroje. Takový nástroj spojuje oba typy přístupu k souborům a umožňuje hledání podobností dokumentů v rámci korpusu a současně i ze zdrojů externích.

2.1.4 Nástroje vnitřní

U tohoto nástroje spočívá způsob odhalování plagiátu v analýze obsahu dokumentu, kdy je zkoumáno, zda se nějaký úsek dokumentu liší svou formou od zbylého textu. V případě nálezu takové části lze proto předpokládat, že se jedná o zkopírovaný úsek, jehož autorem je někdo jiný.

Výhodou této metody je její rychlost, neboť se neprovádí složité porovnávání objemného korpusu.

Problémy mohou nastat v případě, kdy velkou část dokumentu tvoří zkopírovaný text od jednoho autora, protože nám tak klesá šance na správné vyhodnocení. Tento postup se často používá v kombinaci s některou jinou metodou, kdy vnitřní nástroj slouží k vyhledání podezřelých částí dokumentu a následně je použita metoda, která se snaží odhalit zdroj.

2.2 Nástroje založené na jiném principu

Tyto nástroje nevyužívají porovnávání textu, ale převážně pracují s daty, která jsou do souboru záměrně vkládána a umožňují následnou detekci plagiátů. Takové metody při dobrém navržení dosahují vysoké úspěšnosti, což je jejich nespornou výhodou. I tento nástroj však má své nevýhody. Ty tkví v přidávání metadat do dokumentů.

2.2.1 Neviditelné značkování

Touto problematikou se zabývaly dva týmy a praktické odzkoušení proběhlo na akademické půdě (konkrétně na univerzitách v Irsku a Jihoafrické republice). Autoři (Charlie Daly a Jane Horgan) se nechali inspirovat principem vodoznaků, který se používá při zajišťování ochrany autorských práv. Originál je označen pro případné vytvoření nelegální kopie. To se provádí přiřazením jedinečného vodoznaku konkrétnímu uživateli (potažmo dílu). Identifikace autora se vkládá do souboru, kde je zakódována pomocí drobných změn v textu (např. šířka mezer mezi řádky či odstavci, rozložení textu).

Ve výše zmíněném případě, který uvádí Hauzírek v [5], se identifikace autora vkládala do souboru při jeho odevzdávání elektronickou formou a současně se vkládala také do souboru uloženého na disku autora. Kromě identifikace studenta, která byla zajištěna nutností přihlášení, se do dokumentu ukládal také kontrolní součet. Veškeré informace se do souboru zapisovaly ve tvaru binárních čísel v podobě netisknutelných znaků (mezer a tabulátorů). Pro uložení této identifikace se použilo místo v souboru, kde byla malá pravděpodobnost smazání. Jestliže takto označený dokument použil jiný student, nevědomě se svou prací odevzdal i identifikaci práce původní.

Výhodu tato metoda má především v utajené kontrole autenticity odevzdávaných dokumentů. Oproti klasickým nástrojům používaným v boji proti plagiátorům má neviditelné značkování několik nesporných výhod, z nichž bychom na prvním místě měli jmenovat schopnost rozlišení autora a plagiátora, což často bývá velmi obtížné, jak již bylo zmíněno v úvodu. Dále tato metoda umožňuje odhalovat plagiáty i v krátkých textech a s velkou jistotou odhaluje skutečné plagiáty, nikoliv pouze podezřelé soubory.

Zaměříme-li se na nevýhody, nalezneme jich také hned několik. Především je to nutnost utajení prováděné kontroly souborů a použití cizích souborů až po jejich předchozím označení. Asi nejednoho čtenáře této práce by napadlo, že se také může stát, že student označení smaže. Dále je třeba, aby nebyly kopírovány jen menší části cizího díla, neboť by tak byla vysoká šance, že nově vytvořená práce nebude

obsahovat identifikaci původního souboru.

Metodu neviditelného značkování je vhodné použít jako utajený doplněk k programu pro odhalování plagiátů, jehož používání může být studentům známo.

2.2.2 Editor zabraňující plagiátorství

Tato metoda byla opět vytvořena pro účely univerzity. Je založena na psaní zdrojového kódu přímo ve speciálním editoru, čímž se zamezuje plagiátorství. Oproti předchozí metodě se však ukládají doplňující data ihned po začátku psaní a jsou viditelná. Mezi vkládanými daty je historie ukládání souboru. Pro případné snahy o změnu historie souboru jsou data šifrována a jsou vytvářeny kontrolní součty. Proti případným plagiátorským pokusům je navíc zakázáno vkládání textu z jiných souborů.

2.2.3 Metoda doplňování textu

Tato metoda připomíná starší způsob kontroly textů. Spočívá ve smazání určité části (popř. více částí) dokumentu a následně má její autor zpětným doplněním dokázat své autorství na tomto díle.[4]

Příkladem programu, který je na tomto principu založen, je komerční program Glatt Plagiarism Screening Program [21]. Tento program odstraní z textu každé páté slovo, a poté sleduje úspěšnost doplnění slov autorem. Po skončení doplňování je určena pravděpodobnost plagiátorství. Nevýhodou programu je nemožnost automatické kontroly, protože je nutná přítomnost autora. Jedná se o metodu, kterou by bylo vhodné použít jako doplněk k některému jinému programu.

2.3 Metrika nástrojů

Metriky se využívají pro porovnávání obsahu dokumentů a jsou využívány nástroji pracujícími s obsahem dokumentu. Metriky využívané při odhalování plagiátů je třeba rozdělit na dva typy. První typ slouží k vyhledávání plagiátů. Jako výsledek získáváme vyjádření míry shodnosti dokumentů. Druhý typ je určen k porovnávání jednotlivých nástrojů z hlediska jejich úspěšnosti použití. My se zaměříme se na první typ.

[4] S tímto způsobem ověřování autentičnosti jsem se setkal v předmětu Počítače a programování 2, kdy u obhajoby závěrečného projektu byla část programu smazána a úkolem autora bylo smazaný kód doplnit.

2.3.1 Metriky pro vyhledávání v dokumentech

Lancaster a Culwin definují dvě hlediska: rozměr a komplexita metriky [5]. Nás bude zajímat první hledisko. Rozměr metriky udává, kolik dokumentů současně metrika zpracovává. Na základě rozměru metriky rozlišujeme metriky singulární, párové, multidimenzionální a korpální.

Singulární metriky pracují s jediným dokumentem, a proto není jejich použití příliš časté.

Ve většině případů se používají párové metriky, protože nejčastěji provádíme postupné porovnávání námi zkoumaného dokumentu s jednotlivými dostupnými dokumenty (možnými zdroji pro plagiátory). Metrika vyhodnocuje míru podobnosti dvou dokumentů (např.délka nejdelšího společného řetězce, počet stejných slov). Jelikož je tato metrika nejfrekventovanější, zaměříme se na ni i v dalších částech textu.

Multidimenzionální metriky zkoumají současně několik dokumentů a používají se při hledání několika podobných dokumentů.

Korpální metrika je speciálním případem multidimenzionálních metrik, neboť pracuje s celým korpusem.

2.3.2 Typy používaných metrik

V praxi se používají různé postupy výpočtu podobnosti, a výsledky jsou tak samozřejmě odlišné. Z hlediska porovnávání se nejčastěji využívá vyhledávání shodných částí. Základní jednotkou pro porovnávání může být slovo, řádek nebo třeba celý odstavec.

Symetrická metrika

Označování a vzorce převezmeme z [5]. $M(a)$ značí množinu základních jednotek dokumentu a. Základní párová metrika k porovnání dvou dokumentů bývá značena res jako resemblance (podobnost). Vztah podobnosti dvou dokumentů

$$res(a,b) = \frac{|M(a) \cap M(b)|}{|M(a) \cup M(b)|}. \tag{2.1}$$

Symetričnost metriky dokazuje vzorec

$$res(a,b) = res(b,a). \tag{2.2}$$

Výsledkem porovnávání je číslo od nuly do jedné, přičemž nula značí, že v porovnávaných dokumentech nebyla nalezena shoda. Jednička potom znamená, že jeden dokument je kopií druhého.

Lze se setkat i s jinou variantou (vzorcem) pro výpočet podobnosti:

$$res(a,b) = \frac{|M(a) \cap M(b)|}{|M(a)| + |M(b)|}. \tag{2.3}$$

Výsledkem rovnice 2.3 je oproti 2.1 číslo v rozmezí $0-0,5$. To je způsobeno hodnotou ve jmenovateli, neboť v případě naprosté shody je zde dvojnásobný počet základních jednotek (v případě sjednocení v 2.1 je minimální hodnota jmenovatele rovna počtu jednotek jednoho dokumentu). Pro porovnání se vzorcem 2.1 je proto třeba výsledek vynásobit dvěmi.

Symetrickou metriku je třeba používat na stejně velké dokumenty (stejný počet základních jednotek), neboť v případě nestejně dlouhých dokumentů může být zkoumaný soubor chybně označen za dokument, který není plagiátem.

Asymetrická metrika

Tato metrika nemá nevýhodu uvedenou u symetrické metriky. Asymetrická metrika nezkoumá pouze podobnost souborů, ale také obsah jednoho souboru ve druhém. V anglickém textu se objevuje označení containment, proto budeme používat pro asymetrickou metriku značení *con* [5]. Výslednou shodu získáme ze vzorce

$$con(a,b) = \frac{|M(a) \cap M(b)|}{|M(a)|}. \tag{2.4}$$

Hodnota *con* je opět v rozmezí nuly a jedničky. Jedničky je dosaženo, jestliže dokument a je obsažen v dokumentu b.

Obsah dokumentu b v dokumentu a lze určit ze znalosti $con(a,b)$ a velikosti dokumentů a,b dle vzorce

$$con(b,a) = con(a,b) * \frac{|M(a)|}{|M(b)|}. \tag{2.5}$$

Výhodou této metriky je možnost jejího použití na dokumenty různých délek. Za drobnou nevýhodu bývá někdy považována dvojice odlišných výsledků při porovnání dvou souborů, a proto se objevují snahy vyjádřit míru shodnosti jediným číslem. K tomu se využívají symetrizované asymetrické metriky, kterých je několik. Jako příklad uvedeme způsob použitý D.Whitem a M.Joyem zmíněným u Hauzírka [5], který spočívá v jednoduchém výpočtu průměrné hodnoty. Použijeme označení *scon* (symetrizovaná *con*)

$$scon(a,b) = \frac{con(a,b) + con(b,a)}{2}. \tag{2.6}$$

3 DETEKCE VOLNÉHO TEXTU

Tato kapitola se zaměřuje na odhalování plagiátů ve volném textu. Programy pro odhalování plagiátů ve volném textu pracují s přirozeným textem, a je tedy nejdříve třeba převést formátovaný text (DOC, PDF, atd.) do požadované podoby. Postupy, které si ukážeme, jsou získány z otevřených zdrojů (především z univerzitních systémů), neboť firmy většinou tají své „know how“. Je však vysoce pravděpodobné, že metody, které firmy využívají, jsou založeny na stejném principu. Na druhou stranu je třeba si uvědomit, že čím více údajů je zveřejněno, tím větší je šance, že někdo najde způsob, jak program obejít. Setkat se můžeme s porovnáváním obsahů či přímým porovnáváním souborů [5].

3.1 Porovnávání zástupců obsahu souborů

U tohoto postupu se využívá převodu obsahu dokumentu, případně jeho části, na reprezentaci, která je poté porovnávána. Výhodou použití části dokumentu k reprezentaci je urychlení procesu porovnávání.

Nejdříve je dokument převeden do podoby čistého textu, poté následuje jeho úprava (např. odstranění často se opakujících slov). V další části se dokument převede na základní jednotky, které se využívají při porovnávání, a ty jsou poté nahrazeny číselnou prezentací. K převodu základních jednotek na čísla se využívá hashovací funkce. Ze získaných čísel jsou pak vybrána ta čísla, která budou při porovnávání zastupovat daný dokument. V následujících kapitolách se zaměříme na jednotlivé kroky výše zmíněného postupu a budeme přitom vycházet z Hauzírka [5].

3.1.1 Předzpracování souboru

Předzpracování spočívá v úpravě dokumentu převedeného do podoby prostého textu, protože je důležité, aby případná shoda byla správně vyhodnocena.

Používá se například náhrada některých znaků (dvojtečka, uvozovky, středník, atd.) standardní mezerou.[5] Další možnou úpravou je odstranění interpunkce či převod textu na malá písmena. Někdy jsou odstraňovány i číselné údaje, neboť plagiát může spočívat ve zkopírování práce a přepsání číselných hodnot. Z převedeného textu se také někdy odstraňují často se opakující slova či slovní spojení.

[5] V dalších částech textu uvidíme, že u zdrojových kódů naopak často odstraňujeme mezery (tzv. „whitespaces“).

3.1.2 Převod na základní jednotky

Základní jednotky jsou po převodu na číslo reprezentanty zastupujícími dokument při porovnávání. Vyvstává nám proto otázka, co zvolit jako základní jednotku. Celý dokument je příliš velký a případné odhalení shody je téměř vyloučeno (muselo by se jednat o přesnou kopii daného dokumentu). Druhým extrémem je zvolení písmena jako základní jednotky. V tomto případě by naopak jako plagiát byly označeny snad všechny dokumenty. Správná volba základního jednotky je poměrně náročná, a je proto otázkou kompromisu. Často tak záleží na charakteru prací, které chceme porovnávat.

Obvykle se můžeme setkat s volbou věty či skupiny slov jako základní jednotkou. V případě, že za základní jednotku zvolíme například sedm po sobě jdoucích slov, jedno slovo označíme jako prvek základní jednotky, jejíž délka je tedy rovna sedmi.

Pokud bychom porovnávali základní jednotky například o délce tři (složené ze tří slov), může se stát, že v případě plagiátu se vytvoří trojice, která bude obsahovat jen jedno respektive dvě shodná slova, a dokument tak nebude vyhodnocen jako plagiát. Z důvodu větší možnosti odhalení plagiátu se nejčastěji používá překrývání základních jednotek, čímž se odstraní předchozí nedostatek, ovšem za cenu větší prostorové náročnosti.

Problematikou překrývání základních jednotek a jejich výběrem se zabývají na katedře informatiky a výpočetní techniky Západočeské univerzity v Plzni [3]. Využívají zde n-gramy (tj. porovnávají n-tice slov) pro odhalování plagiátů.

3.1.3 Hashovací funkce

Hashovací funkce se používá pro převod prvků dokumentu na číselné vyjádření, což má za následek možnost rychlejšího porovnávání dokumentů. Při používání hashovací funkce však může docházet ke kolizím, protože výsledkem hashování různých slov může být stejná číselná reprezentace. Může tak dojít k situacím, kdy porovnáváme různá slova, která jsou však po hashování vyjádřena stejným číslem, a je nám hlášena shoda. Například Finkel, uvedený u Hauzírka [5], pro hashování využívá funkci MD5, čímž získá 128bitové číslo v podobě třicetidvou hexadecimálních znaků. Z důvodu úspor místa a větší rychlosti však využívá jen prvních deset hexadecimálních znaků, protože zůstává poměrně malá pravděpodobnost kolize.

Využitím hashovací funkce lze při převodu na základní jednotky vytvářet jednotky s rozlišnou délkou, kdy jednotlivým slovům je hashováním přiřazena číselná hodnota a konec základních jednotek je dán číslem, které je beze zbytku dělitelné zadanou konstantou. Použití tohoto způsobu znamená, že nejmenší jednotka obsahuje jedno slovo, ovšem nejdelší jednotka není nijak omezena, a může se tak v nejhorším

případě stát, že celý dokument bude tvořit jednu základní jednotku.[6] Výhodou této metody je úspora místa a větší zabezpečení programu, neboť případný útočník by musel znát hashovací funkci a dělící konstantu.

3.1.4 Výběr čísel a porovnávání

Z hashováním získaných číselných vyjádření slov se vybírají ta, které budou daný dokument zastupovat při porovnávání. Vypuštěním některých číselných reprezentací se urychlí porovnávání, ovšem vzniká riziko, že můžeme vypustit právě části, které by nás dovedly k odhalení plagiátu. Otázkou je, kolik zástupců pro jeden dokument je třeba zvolit.

Lze vybrat pevně daný počet jednotek, čímž uspoříme místo, ovšem při porovnávání nelze pracovat s délkou dokumentu. Nebo můžeme vybírat určitou poměrnou část těchto jednotek. V případě konstantní délky je převážně doporučována volba 100 až 200 jednotek. Finkel, kterého zmiňuje Hauzírek [5], je zastáncem volby poměrné části a doporučuje volit odmocninu všech základnách jednotek.

Kromě otázky, kolik jednotek vybrat, je třeba stanovit, které jednotky se mají pro porovnání vybírat. Na tuto volbu máme několik požadavků. Je nutné, aby zvolené jednotky obsáhly celý dokument a byly rovnoměrně rozloženy. Proti možným útokům by se měly jednotky vybírat náhodně. Pro porovnání je potom vhodné z dokumentů vybírat podobné základní jednotky. Porovnávání může být postupné, nebo program může zpracovávat všechny soubory současně (záleží na množství dokumentů).

3.2 Přímé porovnávání obsahů dokumentů

Tento způsob odstraňuje nevýhody porovnávání obsahu pomocí číselných reprezentací (například kolize hashovací funkce). Tato metoda také umožňuje zobrazování shod v textu. Při přímém porovnávání jsou porovnávána jednotlivá slova (řetězce slov), nikoliv jejich nějakým způsobem vybraní zástupci.

Většina programů používá posouvání vzoru zkoumaného dokumentu po textu porovnávaného dokumentu. První algoritmy provádí posun o jeden prvek, efektivnější algoritmy vychází ze znalosti vzoru a části dokumentu, kterou už porovnal. Tím dochází k urychlení prohledávání.

Dalším používaným postupem je vyhledávání nejdelšího řetězce nebo sekvence společné oběma porovnávaným dokumentům. U sekvence nemusí být hledané prvky bezprostředně za sebou.

[6]Možnost, že by tato situace nastala, je však velmi nepravděpodobná. Muselo by se jednat o velmi krátký dokument

I u přímého porovnávání dokumentů se snažíme celý proces urychlit. Využívá se k tomu snižování počtu porovnávání například tím, že do dokumentů přidáváme metadata charakterizující povahu dané práce. Před porovnáváním poté vybereme skupinu (oblast), v níž se mají možné plagiáty hledat.

4 DETEKCE ZDROJOVÝCH KÓDŮ

Tato kapitola nám přiblíží problematiku vyhledávání plagiátů ve zdrojových kódech. Potíže způsobuje odlišnost jednotlivých programovacích jazyků. Asi nejčastěji se setkáme s programy určenými pro Javu, C, C#, C++. Oproti volnému textu je zdrojový kód vázán mnoha pravidly (má přesně danou syntaxi), která je nutné dodržet pro správnou funkčnost programu.

Pro rozlišení míry plagiátorství ve zdrojových kódech převezmeme obrázek uvedený Hauzírkem v [5] a sestavíme z něj tabulku 4.1.

Tab. 4.1: Míra plagiátorství ve zdrojových kódech dle Jonese (převzato z [5])

1	Přesná kopie
2	Změna komentářů
3	Změna mezer, konců řádků (whitespaces) a formátování
4	Přejmenování identifikátorů
5	Změna pořadí bloků kódu
6	Změna pořadí příkazů v rámci bloků kódu
7	Změna pořadí operandů/operátorů ve výrazech
8	Změna datových typů
9	Přidání nadbytečných přikazů nebo proměnných
10	Náhrada řídících struktur jejich ekvivalenty

Tabulka obsahuje 10 bodů, z nichž je vidět, že s rostoucím počtem úprav je třeba použít důmyslnější a náročnější systémy pro odhalování plagiátů.

4.1 Postup k odhalování plagiátů ve zdrojových kódech

Programy obvyle vychází ze znalosti syntaxe použitého jazyka. Rozlišují se tři způsoby práce se zdrojovými kódy, jak udává Hauzírek [5].

První možností je hledání plagiátů, kdy je podezřelé, že dva programy od různých autorů pracují podobným, ne-li stejným způsobem. Takový způsob však není vhodný pro školní případy, kdy bývá v hodinách programování zadáván určitý příklad velkému množství studentů a pravděpodobnost shody (nemyslíme teď plagiátu) je tedy vysoká. Nevýhodou toho způsobu porovnávání je dále nutnost, aby zdrojové kódy neobsahovaly chyby (především syntaktické) bránící překladu.

Druhou možností je brát zdrojový kód jako běžný text a také s ním tak pracovat. Nevýhoda však tkví v nutnosti porovnávat celé soubory, nejen jejich prezentace,

což však vzhledem k převážně menší délce zdrojových kódů nevadí. Se zdrojovým kódem lze jako s běžným textem pracovat za předpokladu, že jsou na základě znalosti programovacího jazyka provedeny některé úpravy a zdroj je vhodně převeden na základní prvky. Výhodná je potom aplikace porovnání na komentáře a označení proměnných. Jak však zmiňuje Zeidman [16], většina programů odstraňuje komentáře, jména identifikátorů a prázdná místa (whitespace). Tím jsou však odstraněny užitečné informace.

Obě výše zmíněné metody poté spojuje třetí postup. Ten vychází ze znalosti programovacího jazyka a aplikuje na něj některý z postupů zpracování textu. Při předzpracování jsou ze zdrojových kódů odstraněny komentáře, identifikátory a další méně důležité informace. Nevýhodou výše zmiňovaných postupů je nutnost znalosti konkrétního programovacího jazyka.

4.2 Univerzální systémy pro detekci zdrojových kódů

Předchozí kapitola byla zaměřena na postupy pro odhalování plagiátů v předem známém prostředí (programovacím jazyce). Snahou však je vytvořit systém, který by byl, jak již zmiňuje název kapitoly, univerzální. V tom případě program pracuje s univerzálními prvky společnými pro většinu jazyků, a přicházíme o výhodu znalosti syntaxe programovacího jazyka. Tuto metodu, jak je zmíněno v literatuře [5], použil Ducasse s kolegy. Při předzpracování jsou odstraněny pouze komentáře a mezery. Jako základní jednotku pak berou řádek textu. Jednotlivé řádky souboru jsou poté porovnávány s řádky ostatních dokumentů. Pro zobrazení výsledků lze využít nejen obvyklé textové, ale i grafické podoby, kdy se jednotlivá porovnání zaznamenávají do matice. Řádky matice tvoří jednotlivé řádky jednoho dokumentu, sloupce jsou potom zástupcem řádků souboru druhého. V případě shody se na odpovídajícím průsečíku objeví značka. Souvislý úsek značek v diagonále značí větší blok shodných řádků, případný posun diagonály značí vložení či smazání určitého řádku.

4.3 Vliv syntaktické a sémantické analýzy na porovnávání zdrojových kódů

Syntaktická i sémantická analýza (respektive syntaktický a sémantický analyzátor) jsou součástí překladače (kompilátoru), od nějž očekáváme výstup v podobě překladu do strojového kódu. Jeho dalšími částmi je lexikální analyzátor, překladač do mezikódu, optimalizátor a generátor kódu. Celkem se tak překladač skládá z šesti

částí [13]. Z nich se podrobně zmíníme o syntaktickém a sémantickém analyzátoru, u ostatních jen nastíníme jejich úlohu.

4.3.1 Lexikální analyzátor

Jeho úkolem je hledat ve zdrojovém souboru tzv. lexémy (např. „begin“, „for“, tedy základní výrazy jazyka použitého zdroje). Tyto údaje předává lexikální analyzátor dalšímu článku kompilace, tj. syntaktickému analyzátoru. Kromě lexémů pak předává také jejich typ (operátor, identifikátor,...) a v případě identifikátorů udává odkaz na tabulku identifikátorů, což umožňuje rychlejší překlad. Veškeré informace, které jsou výstupem lexikálního analyzátoru, se označují jako **token**. Lexikální analyzátor ze zdroje odstraňuje komentáře a prázdná místa.[7] [7], [11], [13]

4.3.2 Syntaktický analyzátor

Činnost syntaktického analyzátoru (anglicky parser) se označuje jako syntaktická analýza. Syntaktický analyzátor zpracovává informace od analyzátoru lexikálního a tyto řetězce ASCII znaků převádí do datové struktury (syntaktického stromu), která představuje vstupní data pro sémantickou analýzu. Úkolem syntaktického analyzátoru je kontrolovat správnost zápisu programu, zda například nechybí ukončovací závorky. Pro zpracování běžného textu je realizace syntaktické analýzy poměrně složitá a především výpočetně náročná. Důležitou roli ale hraje ve zdrojových kódech, kterými se primárně zabýváme a pro které budeme navrhovat programu. Rozlišujeme dva základní druhy syntaktické analýzy: shora dolů a zdola nahoru, kdy každý typ má svou definovanou gramatiku. [2], [13], [15]

U zdrojových kódů je třeba, aby porovnávané programy (jejich zdrojové kódy) neobsahovaly chyby. Mohlo by to vést k nesprávnému vyhodnocení, nebo by program pro odhalování plagiátů daný kód ani nepřeložil (hlásil by syntaktickou chybu). Správnost syntaktické analýzy je tedy důležitá nejen pro přeložení a možnost spuštění programu, ale také pro korektní vyhodnocení při odhalování plagiátů. V případě zjišťování míry podobnosti souborů můžeme porovnávat jejich strukturu.

4.3.3 Sémantický analyzátor

Sémantický analyzátor přebírá syntaktický strom, v němž analyzuje význam jednotlivých operací a kontroluje jejich správnost. Lze říci, že sémantická anylýza ověřuje korektnost obsahu, kdežto syntaktická analýza dohlíží na správnou strukturu. Sémantický analyzátor zaznamenává jména a datové typy proměnných či funkcí, v

[7] O odstraňování tzv. whitespace jsme se zmiňovali již dříve.

případě potřeby provádí konverzi datových typů (přetypování). Všechny informace sémantické analýzy jsou soustředěny v tabulce symbolů [14], na kterou je kladen požadavek rychlého prohledávání. Sémantický analyzátor doplňuje vstupní syntaktický strom dalšími informacemi a provedenými konverzemi. [13], [14]

Stejně jako syntaktickou, i sémantickou analýzu lze využít k odhalování plagiátů. V prvním kroku se sémanticky analyzují zkoumané soubory. Zaznamenávájí se funkce (název a datový typ jejich vstupní proměnné a typ návratové hodnoty), proměnné (jména a datové typy) a dále například podmínky u příkazů if, for atd.

4.3.4 Překladač do mezikódu

Překladač do mezikódu není nezbytně nutnou součástí všech překladačů (některé provádí přímé přeložení do požadované podoby bez tohoto mezikroku). Jeho úkolem je převod ze syntaktického stromu (doplněného sémantickým analyzátorem) na lineární posloupnost instrukcí. [13]

4.3.5 Optimalizátor a generátor kódu

Optimalizátor se snaží zjednodušit vstupní mezikód a převést ho na rychlejší mezikód, který potom generátor kódu převádí na požadovaný výstup (cílový jazyk) překladače. Obvykle je jím strojový kód. [13]

5 PROGRAMY

Tato kapitola shrnuje dostupné programy sloužící k odhalování plagiátů. Oblast použití programů se liší dle typu dokumentu a formátu, pro nějž jsou určeny. Setkat se můžeme s programy pro volný text (např. DOC, PDF, HTML a RTF), zdrojové kódy (např. Java, C, C++, C#), popřípadě programy zvládající oba tyto typy dokumentů. Programy se také odlišují svou rychlostí a úspěšností zpracování zdroje. Dále je můžeme dělit na intrakorpální a extrakorpální, na programy běžící z webového rozhraní a ty, které se instalují na pevný disk. Další rozdíl může spočívat ve schopnosti programu pracovat s diakritikou.

My se zaměříme na rozdělení podle distribuce nástroje. Jednotlivé tabulky převezmeme z literatury [5] a upravíme podle aktuálnosti, popřípadě doplníme o další nalezené programy.

Tab. 5.1: Nástroje komerční

Název	**Charakteristika**
CatchItFirst	Extrakorpální nástroj pro volný text
CopyCatch	Intrakorpální a intrinsic nástroj pro volný text
Eve2	Extrakorpální nástroj pro volný text
Glatt Plagiarism Services	Intrinsic nástroj pro volný text
MyDropBox	Kombinovaný nástroj pro volný text
Plagiarism-Finder	Extrakorpální nástroj pro volný text
TurnItIn	Kombinovaný nástroj pro volný text

Tab. 5.2: Nástroje volně dostupné

Název	**Charakteristika**
CodeSuite	Intrakorpální nástroj pro zdrojové kódy
Ferret	Intrakorpální nástroj pro volný text a zdrojové kódy
JPlag	Intrakorpální nástroj pro volný text a zdrojové kódy
Pl@giarism	Intrakorpální nástroj pro volný text
Sherlock1	Intrakorpální nástroj pro volný text a zdrojové kódy
Sherlock2	Intrakorpální nástroj pro volný text a zdrojové kódy
WCopyFind	Intrakorpální nástroj pro volný text a zdrojové kódy

Jednotlivé programy si přiblížíme. Ty, u nichž bylo možné provést otestování, si popíšeme detailněji a samozřejmě uvedeme získané výsledky.

Pro test kontroly zdrojových kódu jsme vytvořili složku Balik[8]. Tato složka obsa-

[8]Složka Balik je naším korpusem a je součástí přiloženého CD.

huje sedm souborů, přičemž první je námi zkoumaný originál, na který se zaměříme:

- s1.cpp: originál (zdrojový kód závěrečného projektu z předmětu BPC2)
- s2.cpp: funkce „abecedne“ přesunuta o tři funkce výš a funkce „ukaz“ přejmenována na „zobraz“
- s3.cpp: ve funkci „hprvnipis“ cyklus „for (i=0;i<pocet;i++)...“ nahrazen „while(i<pocet)...“
- s4.cpp: jiné vypracování stejného zadání od xplchj00 [27]
- s5.cpp: přejmenování proměnných :
  - jmeno=prvni
  - prijmeni=druhy
  - cislo=telefon
  - ulice=treti
  - mesto=ctvrty
  - psc=vydelek
- s6.cpp: program z 10. cvičení BPC2 (aby se ve zdrojích nacházel i jeden zcela jiný program)
- s7.cpp: kombinace úprav $2, 3, 5$

Pro textové dokumenty jsme použili dva soubory o optických vláknech, jeden český [1] a druhý anglický [6]. Text ze souboru optika.pdf i OptoCoupler.pdf jsme překopírovali do souborů optika.doc, optika.txt, OptoCoupler.doc a OptoCoupler.txt. Poté jsme jeden odstavec z anglického textu vložili do textu českého a vytvořili soubor optikasang s příponami doc, pdf a txt.

5.1 Komerční nástroje

5.1.1 CatchItFirst

CatchItFirst je online služba porovnávající textové soubory. Text se vkládá do připraveného pole, které pojme maximálně 20 000 znaků, způsobem CTRL–C, CTRL–V. Zadaný text je zřejmě porovnáván pomocí internetového vyhledavače. Po porovnání přijdou výsledky na e-mail a po přihlášení je lze také najít na stránkách služby [32]. Po registraci jsme měli možnost program (pokud vůbec v tomto případě

lze hovořit o programu) vyzkoušet na třech vzorcích. Použili jsme výše zmíněné textové soubory. Ve všech případech byla výsledkem zpráva o 100% originálu. Cena za využívání této služby je 5 USD za pět souborů, nebo 8 USD za deset souborů. [5], [32]

5.1.2 CopyCatch

CopyCatch je sada programů, které pro kontrolu souborů upřednostňují použití celého dokumentu před klíčovými slovy a slovními spojeními. Tyto programy jsou navrženy tak, že jsou schopny najít podobnost i v případě změny slovosledu, vložení nebo smazání slova. Copycatch Gold již řadu let využívají vysokoškolští učitelé ke kontrole sudentských prací. Copycatch Investigator je sada algoritmů, určená pro široké spektrum uživatelů, a je pravidelně upravována pro specifické účely. Přístup ke službě probíhá pomocí klienta napsaného v Javě. Program CopyCatch pracuje s formáty HTML, RTF a DOC. [5], [20]

5.1.3 EVE2

EVE je zkratkou slov Essay Verification Engine. Program si poradí s prostým textem, dokumenty Microsoft Word a formátem Word Perfect. Je určen pro vyučující, kteří s jeho pomocí kontrolují autentičnost studentských prací. Kontrola probíhá tak, že vstupní soubor je porovnáván s informacemi dostupnými na internetu. Ve výsledku potom EVE červeně zvýrazňuje opsané části a zobrazuje internetové odkazy, ze kterých bylo čerpáno. Jedná se tedy o extrakorpální nástroj. Cena jedné licence je 29,99 USD a dle stránek programu [19] musí mít každý učitel svou vlastní licenci. Nelze provést žádné bezplatné vyzkoušení programu.

5.1.4 Glatt Plagiarism Services

Tato společnost nabízí tři programy zaměřené na boj s plagiátorstvím, a to Glatt Plagiarism Teaching Program (GPTeach), Glatt Plagiarism Screening Program (GPSP) a Glatt Plagiarism Self-Detection Program (GPSD). GPTeach je určen pro výchovu studentů, aby se nedopouštěli plagiátorství. GPSP slouží ke kontrole dokumentů a poslední z trojice (GPSD) odhaluje neúmyslné případy plagiátorství. Glatt Plagiarism Teaching Program i Glatt Plagiarism Screening Program jsou dostupné pouze na CD, a to za cenu 300 USD, při zakoupení obou je cena 500 USD. V programu Glatt Plagiarism Self-Detection si lze zdarma otestovat, zda dokážeme doplnit každé páté smazané slovo vlastního textu (podobně funguje GPSP). [5], [21]

5.1.5 MyDropBox

MyDropBox je sada online nástrojů určená pro školy. Program slouží nejen pro odhalování plagiátů, ale umožňuje například opravovat práce studentů. Detekci plagiátů provádí SafeAssignment, který porovnává práce odevzdávané elektronicky pomocí služby MyDropBox Course Management Toolset, Blackboard nebo WebCT s internetovými zdroji a školní databází. Využívá k tomu vlastní techologii, která umí rozpoznat podíl cizích zdrojů i v případě pozměněného textu. Výsledkem porovnání je hlášení obsahující odkazy na shodné zdroje. Program si poradí s formátem DOC, PDF, RTF a HTML.

Lze si koupit měsíční licenci za 24,95 USD, nebo licenci pro školu, jejíž cenu se nám však nepodařilo zjistit. Ta, jak uvádí Hauzírek [5], stojí ročně 0,60 USD na studenta, ale částka již může být jiná, neboť stejný autor udává cenu individuální licence 90 USD na rok. [5], [29]

5.1.6 Plagiarism-Finder

Plagiarism-Finder je extrakorpální nástroj, který zdroje možného plagiátora vyhledává na internetu. Zvládá formáty PDF, DOC, HTML, TXT a RTF. Lze využít třicetidenní zkušební verzi, která neumožňuje porovnávat více souborů najednou a porovnává pouze 5000 znaků. Licence programu stojí 125 USD (98 eur). [5], [26]

Testování našich souborů při výchozím nastavení vždy dopadlo s výsledkem 0% podobnosti. Při zmírnění kritérií jsme dostali spoustu odkazů, ale žádný nevedl na adresu zdroje [1], nebo [6].

5.1.7 TurnItIn

V případě programu TurnItIn se opět jedná o sadu nástrojů určenou nejen ke kontrole textů, ale také k výchově studentů a správě výuky. Je to smíšený nástroj, který požadovaný soubor porovnává se soubory z internetu, s vlastní databází dříve porovnávaných souborů programem TurnItIn a s databází článků z novin a časopisů. TurnItIn běží v prohlížeči a není třeba instalovat žádný software. Poradí si s přirozeným textem, formáty DOC, PDF, HTML, RTF, PS a WPD. Poskytováno je několik typů licencí, ovšem jejich cena není veřejně přístupná. [5], [23]

5.2 Volně přístupné nástroje

5.2.1 CodeSuite

Jedná se o program mající tři intrakorpální nástroje (BitMatch, CodeDiff a CodeMatch). Pro získání programu je třeba provést registraci. Volné porovnávání lze provádět jen se soubory, jejichž celková velikost je menší než 1MB.

BitMatch porovnává spustitelné binární soubory (např. exe, dll) s dalšími soubory stejného typu, popřípadě zdrojovými kódy. Program je užitečný především pro případ, kdy máme přístup k programu, ale ne ke zdrojovému kódu. Výsledky zobrazuje v html souboru, kde kliknutím na porovnávanou dvojici souborů lze zobrazit podrobné informace o shodě. BitMatch provádí hrubé vyhodnocení podobnosti. Pro stanovení definitivní jistoty shody zdrojových kódů je třeba použít CodeMatch. [28]

Programy CodeDiff a CodeMatch si popíšeme v následujících podkapitolách.

CodeDiff

CodeDiff porovnává zdrojové kódy a hledá shodné řádky dvou souborů. Je rychlejší a méně přesný oproti CodeMatch. CodeDiff zvládá snad všechny používané programovací jazyk. Výstup je stejně jako v případě BitMatch v souboru html s možností prohlédnutí shodných částí. Výsledky pro náš korpus jsou uvedeny v tabulce 5.3. [28]

Tab. 5.3: Výsledky porovnání programem CodeDiff

	s1.cpp	**s2.cpp**	**s3.cpp**	**s4.cpp**	**s5.cpp**	**s6.cpp**	**s7.cpp**
s1.cpp	X	98	100	23	89	19	87
s2.cpp	98	X	98	23	87	19	89
s3.cpp	100	98	X	23	89	19	87
s4.cpp	23	23	23	X	21	12	22
s5.cpp	89	87	89	21	X	19	98
s6.cpp	19	19	19	12	19	X	19
s7.cpp	87	89	87	22	98	19	X

CodeMatch

Třetí program z CodeSuite CodeMatch porovnává komentáře, příkazy, identifikátory a pořadí instrukcí. Zvládá podobně jako CodeDiff mnoho programovacích jazyků. Jazyky, ze kterých lze v programu vybírat, jsou zaznamenány v tabulce 5.4. Na stránkách programu [28] je navíc zmíněn ještě FoxPro a Visual Basic. Výstup je

opět v podobě html s možností zobrazení detailů. Výsledky našeho porovnávání jsou uvedeny v tabulce 5.5. [28]

Tab. 5.4: Programovací jazyky podporované programem CodeMatch

ActionScript	Basic	C#	C
C++	Delphi	Java	JavaScript
MASM	MATLAB	Pascal	Perl
PHP	PowerBuilder	REALBasic	Ruby
SQL	Verilog	VHDL	

Tab. 5.5: Výsledky porovnání programem CodeMatch

	s1.cpp	**s2.cpp**	**s3.cpp**	**s4.cpp**	**s5.cpp**	**s6.cpp**	**s7.cpp**
s1.cpp	X	87	91	25	93	31	79
s2.cpp	87	X	87	25	79	31	84
s3.cpp	91	87	X	25	84	31	79
s4.cpp	25	25	25	X	23	19	23
s5.cpp	93	79	84	23	X	31	87
s6.cpp	31	31	31	19	31	X	31
s7.cpp	79	84	79	23	87	31	X

5.2.2 Ferret

Jedná se o intrakorpální nástroj vhodný pro zdrojové kódy i textové soubory, který vznikl na univerzitě v Hertfordshire. Verze 4 automaticky převádí soubory typu PDF a DOC do podoby prostého textu (txt). Výsledky program prezentuje prostřednictvím tabulky, poklepáním na řádek v tabulce zobrazíme příslušnou porovnávanou dvojici souborů s vyznačenou shodou. Výsledky lze uložit do formátu PDF. Naprosté shodě odpovídá v tabulce hodnota 1. Pro vzájemé porovnávání výsledků s ostatními programy uvádíme výsledek v %, tj. jednotlivé hodnoty vynásobíme číslem 100. [25]

Výsledky porovnávání zdrojových kódů jsou v tabulce 5.6, pro textové soubory jsme sestavili tabulku 5.7, v níž jsou zaznamenány výsledky s nenulovými hodnotami. Můžeme pozorovat vliv vypuštění obrázků, které obsahují pouze originální soubory optika.pdf a OptoCoupler.pdf. Ostatní soubory vznikají kopírováním textu.

Tab. 5.6: Výsledky porovnání zdrojových kódů programem Ferret

	s1.cpp	**s2.cpp**	**s3.cpp**	**s4.cpp**	**s5.cpp**	**s6.cpp**	**s7.cpp**
s1.cpp	X	98,81	99,68	11,11	92,87	4,09	91,47
s2.cpp	98,81	X	98,5	11,11	91,77	4,09	92,57
s3.cpp	99,68	98,5	X	11,11	92,57	4,12	91,77
s4.cpp	11,11	11,11	11,11	X	10,47	3,57	10,47
s5.cpp	92,87	91,77	92,57	10,47	X	4,09	98,5
s6.cpp	4,09	4,09	4,12	3,57	4,09	X	4,12
s7.cpp	91,47	92,57	91,77	10,47	98,5	4,12	X

Tab. 5.7: Výsledky porovnání textových dokumentů programem Ferret

Dokument 1	**Dokument 2**	**Podobnost**
optikasang.doc	optika.doc	82,48
optikasang.txt	optika.txt	78,74
optikasang.pdf	optika.pdf	37,80
optikasang.doc	OptoCoupler.doc	10,66
optikasang.txt	OptoCoupler.txt	9,20
optikasang.pdf	OptoCoupler.pdf	8,59

5.2.3 JPlag

Program JPlag je produktem univerzity v Karlsruhe (autorem je Guido Malpohl). V roce 2005 ho Emeric Kwemou a Moritz Kroll převedli na webovou službu a umožnili uživatelům navrhovat si vlastní klienty.

Jedná se o intrakorpální nástroj, který při porovnávání využívá znalost syntaxe formátu kontrolovaného souboru. JPlag podporuje Javu, C#, C, C++, Scheme a přirozený text. Výsledky zobrazuje v html souboru, který je rozdělen na čtyři části. První obsahuje informace o parametrech porovnávání (porovnávaných souborech, zvoleném formátu, počtu zobrazených porovnání, minimální srovnávané délce), druhá nás informuje o rozložení výsledků v úsecích po 10 %. Zbylé dvě části zobrazují procentuální podobnost mezi jednotlivými soubory těch porovnání, která přesáhla nastavenou hranici. Jedno seřazení vyjadřuje průměrnou podobnost, druhé potom podobnost maximální. Kliknutím na výsledek si můžeme zobrazit detaily porovnávání, kde jsou barevně odlišené bloky, jež se shodují. Program JPlag nám tedy dává přehledné informace o výsledcích, a právě proto je v současnosti jedním z nejoblíbenějších (ne-li tím úplně nejpoužívanějším).

Pro používání programu je třeba získat účet. Při registraci je třeba uvést, za jakým účelem chceme daný program používat. Po schválení registrace nám již nic

nebrání ve využití klienta (start.jnlp) a v používání programu. [24]

Při testování složky Balik jsme ponechali výchozí nastavení předdefinované autorem. Soubor s6.cpp při všech porovnáních neprošel přes nastavená pravidla. Z výsledků (tab. 5.8) je patrné, že s drobnými změnami si program poradí bez problému. Přejmenování proměnných JPlag nijak neošálí. Do tabulky jsme zaznamenali hodnoty průměrné podobnosti.

Pro přirozený text jsme nastavili Minimum Match Length na hodnotu 4. Podobnost souborů optikasang.txt a OptoCoupler.txt je 17,9%, pro soubory optikasang.txt a optika.txt je shoda 85,4%.

Tab. 5.8: Výsledky porovnání zdrojových kódů programem JPlag

	s1.cpp	**s2.cpp**	**s3.cpp**	**s4.cpp**	**s5.cpp**	**s7.cpp**
s1.cpp	X	99,6	97,5	20,1	100	96,3
s2.cpp	99,6	X	96,3	20,1	99,6	97,5
s3.cpp	97,5	96,3	X	20,1	97,5	99,6
s4.cpp	20,1	20,1	20,1	X	20,1	20,1
s5.cpp	100	99,6	97,5	20,1	X	96,3
s7.cpp	96,3	97,5	99,6	20,1	96,3	X

5.2.4 Pl@giarism

Program Pl@giarism je jediný volně dostupný nástroj, který je zaměřen čistě na porovnávání textových dokumentů. Pochází opět z akademické půdy (univerzita v Maastricht) a stejně jako ostatní volně dostupné nástroje je intrakorpální. Pl@giarism porovnává dokumenty z nastavené složky, popřípadě lze nastavit cestu k internetovému vyhledávači pro porovnání frází ze zkoumaného dokumentu. Jak uvádí Hauzírek [5], program při porovnávání bere jako základní jednotku tři souvislá slova, přičemž využívá překrývání slov s posunem o jedno slovo. Výsledky jsou uváděny v tabulce, která obsahuje výsledek symetrické metriky a obou metrik asymetrických (viz. kap.2.3.2). Kliknutím na porovnávanou dvojici souborů lze zobrazit podrobné informace o shodných částech.

Program umožňuje v neregistrované verzi, označované jako demo, porovnávat současně jen dva projekty s maximálně dvaceti dokumenty. Při snaze provést otestování našich zdrojů se nám nedařilo spustit porovnávání. Dokumenty MS Word se nepodařilo porovnat, přestože podle informací uvedených na stránkách programu [30] byly problémy s různými verzemi MS Office ustáleny. Výsledkem naší snahy tak bylo pouze porovnání souborů v přirozeném textu (viz. tab. 5.9, v níž jsou hodnoty uvedeny v %).

Tab. 5.9: Výsledky porovnání programem Pl@giarism

Dokument a	**Dokument** b	$res(a,b)$	$con(a,b)$	$con(b,a)$
optikasang.txt	optika.txt	84,34	76,05	94,66
optikasang.txt	OptoCoupler.txt	18,74	22,27	16,17
optika.txt	OptoCoupler.txt	0,09	0,12	0,07

5.2.5 Sherlock1

Sherlock1[9] [22] je intrakorpální nástroj, který podobně jako JPlag běží z klienta (sherlock.jar). Pro jeho použití není třeba žádná registrace. V nastavení lze vybrat několik možností zpracování, většinou souvisejícími s komentáři (zda je použít, nebo odstranit) a bílými místy. Výsledky je možné prohlížet v kruhovém grafu, kde každý bod na obvodu reprezentuje jeden z porovnávaných souborů, tabulce i detailně s vyznačením shodných částí. [22]

Výsledky porovnání nejsou příliš vypovídající, neboť hodnoty jsou uvedeny v % a dosahují velikosi i 295 (viz. tab. 5.10). Lze předpokládat, že to vzniká součtem výsledků z vícero kroků. Pro možnost porovnání s ostatními programy jsme vyšli z předpokladu, že 100% shodě odpovídá hodnota 300, a tak jsme hodnoty podělili třemi a sestavili druhou tabulku (tab. 5.11), jejíž hodnoty budeme považovat za správný údaj v %.

Při porovnávání volného textu získáme při ponechání výchozího nastavení výsledek pouze pro dvojici optikasang.txt a optika.txt (77 %). Podobnost anglického textu se souborem, s nímž se shoduje v jednom odstavci, je pro Sherlock1 tak nepatrná, že nám ji ani nenahlásí.

Tab. 5.10: Výsledky porovnání programem Sherlock1

	s1.cpp	**s2.cpp**	**s3.cpp**	**s4.cpp**	**s5.cpp**	**s6.cpp**	**s7.cpp**
s1.cpp	X	273	295	13	277	1	241
s2.cpp	273	X	267	12	245	3	274
s3.cpp	295	267	X	9	270	-	246
s4.cpp	13	12	9	X	10	3	10
s5.cpp	277	245	270	10	X	1	266
s6.cpp	1	3	-	3	1	X	-
s7.cpp	241	274	246	10	266	-	X

[9] Z důvodu existence dvou stejnojmenných programů zaměřených na odhalování plagiátů jsme je odlišili číslem.

Tab. 5.11: Výsledky porovnání programem Sherlock1 v %

	s1.cpp	**s2.cpp**	**s3.cpp**	**s4.cpp**	**s5.cpp**	**s6.cpp**	**s7.cpp**
s1.cpp	X	91	98,3	4,3	92,3	0,33	80,3
s2.cpp	91	X	89	4	81,7	1	91,3
s3.cppž	98,3	89	X	3	90	-	82
s4.cpp	4,3	4	3	X	3,3	1	3,3
s5.cpp	92,3	81,7	90	3,3	X	0,33	88,7
s6.cpp	0,33	1	-	1	0,33	X	-
s7.cpp	80,3	91,3	82	3,3	88,7	-	X

5.2.6 Sherlock2

Druhý program s názvem Sherlock (označujeme Sherlock2) je dílem jakéhosi Roba Pike. Ten vytvořil dvojici programů: sig a comp. Sig generoval digitální značky a ukládal je do souboru. Comp pak porovnával tyto značky a určoval podobnost. Tyto dva programy potom Loki[10] spojil do podoby, kterou známe dnes, a program pojmenoval Sherlock. Jedná se o intrakorpální nástroj, který má výhodu v tom, že nevytváří žádné pomocné soubory. Zpracovává přirozený text i zdrojové kódy.

Program (sherlock.c) je třeba po stažení přeložit do spustitelného souboru (sherlock.exe). Spuštění programu probíhá z příkazové řádky. Zápisem „sherlock *.cpp> results.txt“ budou porovnány všechny soubory s příponou cpp a výsledek uložen do souboru results.txt. Při tomto zadání bude provedeno porovnání podle autorem definovaných pravidel [31], která lze nastavit pomocí parametrů:

- -t: nastavuje minimální procentální shodu nutnou k uvedení podobnosti dvou souborů v souboru s výsledky, výchozí hodnota je 20 %.
- -z: nastavuje důkladnost porovnávání, vyšší číslo znamená menší přesnost a větší rychlost. Může to být číslo mezi 0 a 31, standardně je to číslo 4.
- -n: tento parametr určuje, kolik slov tvoří digitální značku. Vyšší číslo zde znamená větší přesnost. Přednastavena je hodnota 3.
- -o: slouží k určení výstupního souboru.

Pro porovnání našeho korpusu jsme použili zápis „sherlock -t 0% -o resultas.txt *.cpp“, abychom získal hodnoty pro všechny kombinace. Výsledky jsou uvedeny v tabulce 5.12.

[10]Bližší informace o tom, zda se jedná o nějakou přezdívku nebo příjmení, se na stránkách programu ([31]) nevyskytují

Tab. 5.12: Výsledky porovnání programem Sherlock2

	s1.cpp	**s2.cpp**	**s3.cpp**	**s4.cpp**	**s5.cpp**	**s6.cpp**	**s7.cpp**
s1.cpp	X	97	100	2	82	0	80
s2.cpp	97	X	97	2	80	0	82
s3.cpp	100	97	X	2	82	0	80
s4.cpp	2	2	2	X	2	1	2
s5.cpp	82	80	82	2	X	0	97
s6.cpp	0	0	0	1	0	X	0
s7.cpp	80	82	80	2	97	0	X

Při porovnávání textových souborů stanovuje Sherlock2 oproti programu Sherlock1 podobnost textů optikasang.txt a OptoCoupler.txt (viz. tab 5.13).

Pro rychlé orientační porovnání je tento program vhodný, ale určitě najdeme důvěryhodnější programy.

Tab. 5.13: Výsledky porovnání programem Sherlock2 pro textové soubory

Dokument 1	**Dokument 2**	**Podobnost**[%]
optikasang.txt	optika.txt	75
optikasang.txt	OptoCoupler.txt	7
optika.txt	OptoCoupler.txt	0

5.2.7 WCopyfind

WCopyfind je další z řady intrakorpálních nástrojů, jenž vznikly na univerzitě (tentokrát ve Virginii). Je určen pro přirozený text, zdrojové kódy (zpracovává je jako text), html a měl by zvládat i dokumenty MS Word, což se však v praxi příliš nepotvrdilo, jak zmiňuje také Hauzírek [5]. Program umožňuje podobně jako Sherlock1 nastavit způsob porovnávání. Zde lze například ignorovat veškerou interpunkci, ignorovat čísla nebo nastavit nejkratší výraz k porovnání. Výsledky jsou zobrazeny v tabulce a kliknutím se nám zdrojové kódy porovnávaných souborů otevřou v internetovém prohlížeči s vyznačenými stejnými částmi.[18]

My jsme pro porovnávání nastavili Shortest Phrase to Match i Fewest Matches to Report na 5, abychom získali hodnoty pro všechna vzájemná porovnání zadaných zdrojových kódů. Toto nastavení využijeme i pro porovnání textových dokumentů. Program využívá asymetrickou metriku, takže získáváme dvojici výsledků. Do tabulek uvádíme aritmetický průměr těchto hodnot. Výsledky pro naši složku Balik

(tab. 5.14) jsou srovnatelné s jinými programy. Pro porovnávání souborů s s4.cpp a s6.cpp vykazuje program minimální shodu.

Při porovnávání textů provedl program špatný převod z MS Wordu, a z toho důvodu vykazuje vysokou podobnost souboru optikasang.doc a OptoCoupler.doc (viz. tab 5.15).

Tab. 5.14: Výsledky porovnání zdrojových kódů programem WCopyfind

	s1.cpp	**s2.cpp**	**s3.cpp**	**s4.cpp**	**s5.cpp**	**s6.cpp**	**s7.cpp**
s1.cpp	X	99	99	2,5	93	3	91
s2.cpp	99	X	99	2,5	92	3	92
s3.cpp	99	99	X	2,5	92	3	92
s4.cpp	2,5	2,5	2,5	X	1,5	1	1,5
s5.cpp	93	92	92	1,5	X	3	99
s6.cpp	3	3	3	1	3	X	3
s7.cpp	91	92	92	1,5	99	3	X

Tab. 5.15: Výsledky porovnání textů programem WCopyfind

Dokument 1	**Dokument 2**	**Podobnost**[%]
optikasang.txt	optika.txt	85
optikasang.doc	optika.doc	62,5
optikasang.doc	OptoCoupler.doc	45
optikasang.txt	OptoCoupler.txt	18,5

6 NÁVRH VLASTNÍHO PROGRAMU

Program HP byl vytvořen v programu Dev-C++ verze 4.9.9.2 [17]. V této části práce se budeme věnovat popisu námi navrženého programu. Program se jmenuje HP, jako zkratka pro slovní spojení Hledej Podobnost, a je určen pro kontrolu zdrojových kódů jazyka C.

6.1 Popis programu

Program HP provádí statické porovnání, porovnává dva zadané vstupní soubory a v případě úpravy, po které by mezi sebou vzájemně porovnal všechny soubory ze zadané složky, by jeho přístup ke zdrojům odpovídal programu Sherlock2. Program spouštíme z příkazové řádky v podobě „hp soubor1 soubor2“. Například pro naše testované soubory s1.cpp a s2.cpp ze složky Balik spouštíme porovnání příkazem „hp balik\s1.cpp balik\s2.cpp“.

HP provádí postupně 3 kroky, než dojdeme k výsledku. Nejprve se provede předzpracování obou vstupních souborů (funkce predzpracuj), které spočívá v odstranění komentářů a prázdných řádků (prázdných řádků původního zdrojového kódu i vzniklých odstraněním komentáře). Takto upravené zdrojové kódy se uloží do souboru zdroj1.tmp a zdroj2.tmp.

Ve druhém kroku zpracováváme soubory s příponou tmp, v nichž hledáme a zaznamenáváme počet funkcí, proměnných a počet bloků první, druhé a třetí úrovně. Tyto údaje jsou pak vstupem pro určení podobnosti. Pro zaznamenávání počtu funkcí a bloků využíváme jednorozměrné pole *blok*, pro hlídání aktuální úrovně potom proměnnou *uroven*. Jestliže při prohledávání zdrojového kódu narazíme na znak „{“, inkrementujeme příslušnou pozici pole *blok* (*blok*[*uroven*]) a proměnnou *uroven*, v případě znaku „}“ hodnotu *uroven* dekrementujeme. Do počtu proměnných zaznamenáváme proměnné typu int, double, float, char, ShortString a bool. Získané počty se ukládají na samostatné řádky do souboru obsah1.vys a obsah2.vys v pořadí počet funkcí, počet bloků první úrovně, bloků druhé úrovně, bloků třetí úrovně, počet proměnných.

Ve třetím kroku provádíme výpočet podobnosti. Jednotlivým počtům, zaznamenaným v souborech obsah1.vys a obsah2.vys, je přiřazena určitá váha. Součet vah je, pro možnost vyjádření v procentech, 100. Výpočet podobnosti obecně probíhá podle vztahu

$$shoda+ = \frac{mensi}{vetsi} * vahy[i], \tag{6.1}$$

kde do proměnné *shoda* postupně přičítáme dílčí podobnosti, *mensi* a *vetsi* představuje menší a větší hodnotu nacházející se na pozici i v souboru obsah1.vys a obsah2.vys. Proměnná $vahy[i]$ určuje váhu pro příslušné porovnání, i určuje příslušné porovnání. Postup (6.1) se opakuje 5krát (pro jednotlivé záznamy v souborech s příponou vys). Výsledná podobnost je poté zobrazena v konzolovém okně a zároveň uložena do souboru Výsledek.txt.

V případě porovnávání testovaného korpusu (viz. tab. 6.1) nám pro nepatrné úpravy vychází 100% shoda. Pro program, který realizuje stejné zadání (s4.cpp), vychází shoda 74,13%. Program provádí statické porovnání, takže v případě dvou zcela odlišných programů přibližně stejné délky by s největší pravděpodobností porovnání kódu přineslo výsledek mezi 90% a 100%. Vylepšením by tedy bylo zaznamenávání názvů a datových typů proměnných, názvů a typů návratových hodnot funkcí a podmínek u příkazů if, while, for. Důležitá je pro nás bezchybnost zdrojového kódu, protože pokud by kód například obsahoval syntaktickou chybu v podobě chybějící závorky „{“ nebo „}“, znamenalo by to zkreslení výsledků.

Tab. 6.1: Výsledky porovnání programem HP

	s1.cpp	**s2.cpp**	**s3.cpp**	**s4.cpp**	**s5.cpp**	**s6.cpp**	**s7.cpp**
s1.cpp	X	100	100	74,13	100	24,67	100
s2.cpp	100	X	100	74,13	100	24,67	100
s3.cpp	100	100	X	74,13	100	24,67	100
s4.cpp	74,13	74,13	74,13	X	74,13	21,76	74,13
s5.cpp	100	100	100	74,13	X	24,67	100
s6.cpp	24,67	24,67	24,67	21,76	24,67	X	24,67
s7.cpp	100	100	100	74,13	100	24,67	X

7 SHRNUTÍ VÝSLEDKŮ

V tabulce 7.1 jsou shrnuty výsledné podobnosti jednotlivých zdrojových kódů z korpusu vůči našemu originálu (s1.cpp). Pro soubor s2.cpp, kde jednu funkci přesuneme a jednu přejmenujeme, dosahují všechny programy kromě CodeMatch a Sherlock1 hodnotu blízkou 100 %. Podobně je tomu v případě souboru s3.cpp, kde je cyklus for nahrazen cyklem while. V tomto případě se výrazněji liší pouze CodeMatch. U souborů s4.cpp a s6.cpp programy rozeznají, že se jedná o jiné soubory. Zde se potvrzuje výše zmíněná myšlenka, že námi navržený program může nesprávně vyhodnotit jiný program stejné délky.

Celkově se nejlépe jeví program JPlag, který tak potvrzuje svou vysokou oblibu. Za něj bychom zařadili programy Ferret a WCopyfind.

Tab. 7.1: Porovnání výsledků jednotlivých programů k originálu (s1.cpp)

	s2.cpp	**s3.cpp**	**s4.cpp**	**s5.cpp**	**s6.cpp**	**s7.cpp**
CodeMatch	87	91	25	93	31	79
CodeDiff	98	100	23	89	19	87
Ferret	98,81	99,68	11,11	92,87	4,09	91,47
HP	100	100	74,13	100	24,67	100
JPlag	99,6	97,5	20,1	100	-	96,3
Sherlock1	91	98,3	4,3	92,3	0,33	80,3
Sherlock2	97	100	2	82	0	80
WCopyfind	99	99	2,5	93	3	91

Podíváme-li se na porovnávání textových souborů, můžeme říci, že porovnání souborů optika a optikasang odpovídá podobnost v rozmezí 75 % a 85 %. Pro dokumenty optikasang a OptoCoupler pak hodnoty mezi 10 % a 20 %. Nástroje extrakorpální k hledání podobného dokumentu obvykle využívají internetové vyhledávače či určité servery, a často se tak stává, že jejich výsledky nejsou úplně korektní. Další problém, s nímž jsme se setkali, byl špatný převod z formátovaného textu do prosté textové podoby.

8 ZÁVĚR

Bakalářská práce byla zaměřena na problematiku boje s plagiátorstvím, tedy na používané metody porovnávání dokumentů a dostupné programy. Setkali jsme se však s několika programy, které nebyly určeny pouze pro odhalování plagiátů, ale byly zaměřeny i na předcházení jejich vzniku tím, že se věnovaly vzdělávání studentů.

V první části práce, jsme si popsali možné způsoby porovnávání souborů, využívané metriky a přístup k textovým souborům a zdrojovým kódům.

Pro porovnávání textových souborů se využívá přirozený text a některé programy mají problémy s konverzí formátů. Při kontrole se využívá převod na základní jednotky. Přesné určení počtu slov, z nichž se má skládat základní jednotka, neexistuje. Za vhodnou volbu se považuje hodnota mezi třemi a deseti slovy, to ale záleží na charakteru dané práce. Ke zdrojovému kódu lze přistupovat stejným způsobem, obvykle se však využívá znalosti syntaxe daného jazyka.

Druhá polovina práce byla věnována popisu dostupných programů a návrhu programu vlastního. Pro popis programů jsme použili rozdělení podle jejich dostupnosti.

Komerční programy se zaměřují na odhalování plagiátů v textových souborech. Obvykle se jedná o extrakorpální programy, které jako zdroj pro porovnávání soborů využívají internet. Nejčastěji jsou určeny pro formáty DOC, PDF, HTML a RTF. Vlastníci většinou neumožňují bezplatné vyzkoušení programu.

Programy pro kontrolu zdrojových kódů vznikají převážně na akademické půdě a jsou volně dostupné. Jedná se intrakorpální nástroje porovnávající soubory v zadaném korpusu. Největší oblibě se u uživatelů, zřejmě po zásluze, těší program JPlag. Ten správně vyhodnotil porovnání zdrojových kódů i přirozeného textu.

Námi navržený program je určen pro porovnávání souborů jazyka C. Využívá k tomu znalost syntaxe. Porovnání probíhá na základě statistického vyhodnocení počtu funkcí, proměnných a bloků první až třetí úrovně. Potíže se správným vyhodnocením nastávají u programu v případě, že porovnáváme dva zcela odlišné programy s přibližně stejnou délkou zdrojového kódu.

LITERATURA A POUŽITÉ ZDROJE

[1] BARTOŇOVÁ, Jiřina. *Optické kabely* [online]. 2008 [cit. 2008-05-27]. Dostupný z WWW: <http://spoje.sous.cz/soubory/esf2/download/optika.pdf>.

[2] ČERMÁK, Martin. *Syntax analysis based on combination of several methods* [online]. 2006 [cit. 2007-11-10]. 3 s. Bakalářský projekt. Dostupný z WWW: <http://www.feec.vutbr.cz/EEICT/2006/sbornik/01-Bakalarske_projekty/09-Pocitacove_systemy/01-xcerma16.pdf>.

[3] ČEŠKA, Zdeněk. *Využití n-gramů pro odhalování plagiátů* [online]. 2007 [cit. 2008-04-24]. Dostupný z WWW: <http://textmining.zcu.cz/publications/NGramyProPlagiaty-ITAT2007-Czech.pdf>.

[4] DEMA. *Vybavenost domácností občanů ČR technikou VIII* [online]. 16.4.2008 [cit. 2008-05-07]. Dostupný z WWW: <http://www.dema-praha.cz/z_t/tz0801.doc>.

[5] HAUZÍREK, Michal. *Možnosti automatické detekce plagiátů* [online]. 2007 [cit. 2008-05-07]. 133 s. Vedoucí diplomové práce Ing. Luboš Pavlíček. Vysoká škola ekonomická v Praze. Fakulta informatiky a statistiky. Katedra informačních technologií. Dostupný z WWW: <http://www.trochu.kvalitne.cz/diplomka/DP_xhaum07_v4_se_zadanim.pdf>.

[6] MALATESTA, M., PEREGO, M., PESSINA, G. *A linear optical coupler for cryogenic detectors* [online]. [2000?] [cit. 2008-05-27]. Dostupný z WWW: <http://crio.mib.infn.it/wig/electronics/LTD8%20-%2099/OptoCoupler.pdf>.

[7] ŠIMÁČEK, Jiří. *Syntax analysis based on the pda with deep pushdown expansions* [online]. 2006 [cit. 2007-11-10]. 3 s. Bakalářský projekt. Dostupný z WWW: <http://www.feec.vutbr.cz/EEICT/2006/sbornik/01-Bakalarske_projekty/07-Informacni_systemy/05-xsimac00.pdf>.

[8] ZOUHAR, Petr. *Systémy pro kontrolu elektronických textů*. Brno: Vysoké učení technické v Brně. Fakulta elektrotechniky a komunikačních technologií. Ústav telekomunikací, 2008. s.24. Semestrální projekt. Vedoucí práce Václav Pfeifer.

Internetové stránky:

[9] Český statistický úřad. *Využívání informačních a komunikačních technologií v domácnostech a mezi jednotlivci* [online]. c2008, Aktualizováno dne:15.4.2008 [cit.2008-05-07]. Dostupný z WWW: <http://www.czso.cz/csu/redakce.nsf/i/domacnosti_a_jednotlivci>.

[10] Wikipedie. *Korpus* [online]. 9. 11. 2007 [cit. 2007-12-06]. Dostupný z WWW: <http://cs.wikipedia.org/wiki/korpus>.

[11] Wikipedie. *Lexikální analýza* [online]. 12. 3. 2008 [cit. 2008-05-25]. Dostupný z WWW: <http://cs.wikipedia.org/wiki/Lexik%C3%A1ln%C3%AD_anal%C3%BDza>.

[12] Wikipedie. *Plagiát* [online]. 7. 12. 2007 [cit. 2007-12-09]. Dostupný z WWW: <http://cs.wikipedia.org/wiki/Plagi%C3%A1t>.

[13] Wikipedie. *Překladač* [online]. 20. 5. 2008 [cit. 2008-05-25]. Dostupný z WWW: <http://cs.wikipedia.org/wiki/P%C5%99eklada%C4%8D>.

[14] Wikipedie. *Sémantická analýza* [online]. 22. 5. 2008 [cit. 2008-05-25]. Dostupný z WWW: <http://cs.wikipedia.org/wiki/S%C3%A9mantick%C3%A1_anal%C3%BDza>.

[15] Wikipedie. *Syntaktická analýza* [online]. 22. 4. 2008 [cit. 2008-05-25]. Dostupný z WWW: <http://cs.wikipedia.org/wiki/Syntaktick%C3%A1_anal%C3%BDza>.

[16] ZEIDMAN, Bob. *Dr. Dobb's | Detecting Source-Code Plagiarism | July 1, 2004* [online]. July 01, 2004 [cit. 2008-03-02]. Dostupný z WWW: <http://www.ddj.com/architect/184405734>.

Programy:

[17] Bloodshed Software. *Dev-C++* [počítačový program]. Ver. 4.9.9.2 [2006] [cit. 2008-03-27]. Dostupný například z WWW: <http://www.stahuj.centrum.cz/vyvojove_nastroje/delphi_a_c++/ostatni/devc/>.

[18] BLOOMFIELD, Louis A. *Plagiarism Resource Site Home Page* [online]. v2.6. c1997-2006, page last updated: March 8, 2006 [cit. 2008-05-27]. Dostupný z WWW: <http://plagiarism.phys.virginia.edu/Wsoftware.html>.

[19] CaNexus. *EVE2 Plagiarism Detection for Teachers* [online]. [2005] [cit. 2008-05-27]. Dostupný z WWW: <http://www.canexus.com/>.

[20] CFL Software Development. *CFL Software Development* [online]. 2007 [cit. 2008-05-27]. Dostupný z WWW: <http://www.copycatchgold.com/>.

[21] Glatt Plagiarism Services, Inc. *Glatt Plagiarism Program* [online]. [200-?] [cit. 2008-05-27]. Dostupný z WWW: <http://www.plagiarism.com>.

[22] HART, Benjamin, et al. *SourceForge.net: The BOSS Online Submission System* [online]. v5. c1999-2008 [cit. 2008-05-27]. Dostupný z WWW: <http://sourceforge.net/projects/cobalt/>.

[23] iParadigms, LLC. *Turnitin* [online]. c1998–2008 [cit. 2008-05-27]. Dostupný z WWW: <http://www.turnitin.com/>.

[24] KWEMOU, Emeric, KROLL, Moritz. *JPlag - Detecting Software Plagiarism* [online]. v2.0 - Build 322. [2007] [cit. 2008-05-27]. Dostupný z WWW: <https://www.ipd.uni-karlsruhe.de/jplag/>.

[25] MALCOLM, James A. *UH Plagiarism Detection Group: Homepage* [online]. v4.0. 2000-2006, last modification 4th July 2006 [cit. 2008-05-27]. Dostupný z WWW:<http://homepages.feis.herts.ac.uk/ pdgroup/>.

[26] Mediaphor Software Entertainment AG. *Plagiarism-Finder - software compares term-papers with the Web to find plagiarism* [online]. c2006 [cit. 2008-05-27]. Dostupný z WWW: <http://www.m4-software.de/>.

[27] PLCH, Jan. *Závěrečný projekt z předmětu Počítače a programování 2* [program]. Použit zdrojový kód (v porovnávaném korpusu označen jako s4.cpp)

[28] S. A. F. E. *Software Analysis and Forensic Engineering* [online]. v2.3.0. c1996–2008 [cit. 2008-05-27]. Dostupný z WWW: <http://www.safe-corp.biz/products_codesuite.htm>.

[29] Sciworth Inc. *MyDropBox* [online]. c2002–2006 [cit. 2008-05-27]. Dostupný z WWW: <http://www.mydropbox.com/>.

[30] SPAN, Georges. *Plagiarism.tk* [online]. Ver. 0.9.2. c2002, last updated: December, 2006 [cit. 2008-05-27]. Dostupný z WWW: <http://www.plagiarism.tk/>.

[31] The University of Sydney. *The Sherlock Plagiarism Detector* [online]. c2006, last updated: 22-May-2008 [cit. 2008-05-27]. Dostupný z WWW: <http://www.cs.usyd.edu.au/ scilect/sherlock/>.

[32] Vancouver Software Labs. *CatchItFirst.com* [online]. c2006 [cit. 2008-05-27]. Dostupný z WWW: <http://www.catchitfirst.com/>.

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

blok	jednorozměrné pole s počtem funkcí a bloků první, druhé a třetí úrovně
$con(a, b)$	containment (obsah) dokumentu a a b
HP	hledej podobnost (název vlastního programu)
i	porovnávaná pozice
$M(a)$	Množina základních jednotek dokumentu a
mensi	menší z hodnot nacházející se na pozici i v souboru obsah1.vys a obsah2.vys
$res(a, b)$	resemblance (podobnost) dokumentu a a b
$scon(a, b)$	symetrizovaný obsah dokumentu a a b
shoda	výsledná podobnost dvou zdrojových kódů
uroven	hlídá aktuální úroveň ve zdrojovém kódu
$vahy[i]$	váhy pro pozici i
vetsi	větší z hodnot nacházející se na pozici i v souboru obsah1.vys a obsah2.vys