

TDK-dolgozat
Kiss András Károly

Pécsi Tudományegyetem
Természettudományi Kar
Matematikai és Informatikai Intézet

A plágiumkereső szoftverek kiskapui

Kiss András Károly

2. évfolyam

Témavezető: Horváth Zoltán
egyetemi tanársegéd

Pécs, 2012.12.12.

Tartalomjegyzék

1 Bevezetés.....	1
2 Célkitűzések.....	2
3 A plagizálás potenciális veszélyei	3
4 Anyag és módszer	5
4.1 Részletek a vizsgált dokumentumokból	5
4.2 Google	8
4.3 KOPI Plágiumkereső.....	10
4.4 Plagiarism Detector	13
4.5 Plagiarism Checker	16
4.6 Plágiumkeresés ujjlenyomatok segítségével.....	18
4.7 Dokumentumok összehasonlítása	21
4.8 Külföldi kutatások eredményei	25
5 Eredmények	27
6 Irodalomjegyzék	31

1 Bevezetés

Napjainkban egyre fontosabb szerepet tölt be az információ a társadalom életében. Könnyedén, pár kattintással, akár néhány másodperc alatt juthatunk hozzá a világ bármely pontjáról származó információhoz. Akadályt már a nyelvi korlátok sem jelentenek, hiszen egyre kifinomultabb, pontosabb fordítóprogramok látnak napvilágot [12]. Ez a felgyorsult és leegyszerűsödött információszerzés vezetett ahhoz, hogy a hallgatók igen gyakran sajátjukként tüntetik fel mások szellemi termékeit. A plagizálás sajnos egyre inkább mindennapossá válik. Az Internet adta lehetőséggel visszaélő hallgatók száma évről évre emelkedik és ez a tendencia folytatódni fog, mindaddig, amíg nem teszünk ellene [17, 24].

Mit is jelent a plágium? Plágiumról beszélünk, minden olyan esetben, amikor a más szerzőtől származó tartalmat használunk fel munkánkhoz anélkül, hogy annak igazi szerzőjét megjelölnénk. Az idegen szerző munkáját minden esetben jelölni kell, a tartalmat megfelelő hivatkozással kell ellátni [21]. Szó szerinti idézés esetén a szerző megjelölésének elmulasztása, illetve az idézőjelek hiánya már plágiumnak minősül [11]. Persze a parafrázálásról sem szabad megfeledkeznünk, más munkájának tartalmilag hű átírása esetén is kötelesek vagyunk megjelölni az átfogalmazott szöveg forrását, ennek elmulasztása szintén plágium [20].

Rendkívül fontos természetesen a bírák és a felsőoktatási intézmények egyhangú támogatása a plágium elleni küzdelemben, hogy csak azok munkái kerüljenek elfogadásra, akiknek leadott dolgozata valóban saját kutatásra épül, továbbá önálló gondolatokat rejt. Joggal feltételezzük, hogy a vizsgálat elmaradása esetén a hallgatók egy része élni fog ezzel a lehetőséggel [6]. A vizsgálat elmulasztása, egy későbbi lebukás az egyetem hírnevére is rossz fényt vethet. Az ellenőrzésnek meg kell történnie a szakdolgozat készítését megelőző évek során leadott dolgozatok esetében is, sőt a középiskolás házi dolgozatok esetében is javasolt. Mindez sok munkát és plusz energiát igényel, de ezen erőfeszítések megtétele biztosíthatja egy plágiummentes oktatás létrejöttét, megszüntetve ezzel napjaink intézményeinek egyik fő problémáját [22].

2 Célkitűzések

A jelenleg használatban levő plágiumkereső szoftverek hiányosságainak feltérképezése után célként tűztem ki egy olyan szoftver kifejlesztését, melyben a más programoknál talált kikapukat bezárom. Eddigi munkám során tesztelt programok számos esetben hibáztak és engedtek át plágiummal megtöltött dokumentumokat. A tévedés lehetőségét csak növeli, ha az eredeti dokumentum nyelve különbözik a plágiummal élő hallgató nyelvétől, plágiummentesnek tüntetve fel akár teljesen értelmetlen, idegen nyelvről fordító segítségével alkotott szövegeket is. Komoly az elmaradás a fordítási plágiumok szűrése terén. Szoftveremet úgy építem fel, hogy az képes legyen kiszűrni ezt, lépést tartva ezáltal, a kor igényeivel és a fordító programok fejlődésével. Persze további hiányosságokkal is találkoztam. A jelenlegi plágiumkereső szoftverek képtelenek kiszűrni a tartalmilag hű másolást. Néhány szó megváltoztatása után már plágiummentesnek találták a másolt szöveget. Szoftveremnek képesnek kell lennie felismerni a parafrázisát is, hiszen csak a leglustább diákok élnek szó szerinti másolással. Programomban szeretnék minél több egyszerűsítő funkciót beépíteni, melyekkel a hétköznapi felhasználók számára szeretném a program használatát megkönnyíteni. Külön hangsúlyt fektetek a felhasználói felületre, mivel az általam tesztelt szoftverek grafikus kezelői felülete nem nyerte el a tetszésem.

3 A plagizálás potenciális veszélyei

Szeretnék rávilágítani arra, hogy a plágium egy veszélyes eszköz, amely esetenként akár vissza is üthet későbbi karrierünk folyamán. Mint az köztudott, az elmúlt években egyre többet hallani a plágium okozta problémákról, melyek szenvedő alanyai napjaink közéleti szereplői. Azon állításom alátámasztására, hogy egy a diplomamunkában elkövetett plágium karrierünk egészére kihatással lehet, összegyűjtöttünk pár ismertebb esetet, amelyben neves egyetemet végzett, ismert emberekről derült ki, hogy más munkáját jogtalanul felhasználva szereztek diplomát. Az esetek nagy port kavartak hazánkban, és rontottak a botrányba keveredett emberek megítélésén. Ezen eseteket érdemes átgondolni, hiszen bárki, aki jogtalanul, esetleg helytelenül használta fel más munkáját, könnyedén kerülhet egy hasonló eset középpontjába. Különösen igaz ez a köztisztviselők esetében, hiszen nem csak a rosszakarók, de a szenzációhajász média is fényt deríthet egy-egy ilyen visszaélésre.

Talán a legismertebb botrány Schmitt Pál nevéhez fűződik. 2012. január 11-én a HVG internetes oldalon jelent meg az akkori köztársasági elnök doktori disszertációját tárgyaló cikk. Az 1992-ben írt és a Semmelweis Egyetem bizottsága által elfogadott mű nagy része egy bolgár sportkutató és –diplomata, Nyikolaj Georgiev munkájának szó szerinti fordítása. Egyes becslések szerint Schmitt 215 oldalas művének mintegy 180 oldala a bolgár kutató munkája, további egy oldal pedig Hristo Meranzov és Georgiev munkájának átvétele. Később fény derült arra is, hogy a fennmaradó 45 oldal sem teljes egészében saját munka, ebből nagyságrendileg 17 oldal egy német kutató, Klaus Heinemann tanulmányából származik. A doktori disszertáció ráadásul nem felel meg az alapvető formai követelményeknek sem. A plágiumgyanú bebizonyosodása után Schmitt Pál doktori címét elvették, továbbá államfői posztjáról is lemondani kényszerült [28].

Az elmúlt év másik nagy botránya Gyurcsány Ferenc nevéhez fűződik. Az 1984-ben leadott „A Balaton –felvidék szőlészete és borászata” című szakdolgozatnak nem csak a címe, de a tartalma is megegyezik majdani sógora, Rozs Szabolcsnak munkájával. Mindezt az egykori miniszterelnök munkájáról kiadott bírálatnak és Rozs Szabolcs művének összehasonlítása segítségével állapították meg. Saját bevallása szerint az átfedés a közös forrásoknak köszönhető, mivel azokat a Rozs család házában gyűjtötte össze. Rozs Szabolcs nem tud arról, hogy Gyurcsány Ferenc olvasta volna szakdolgozatát, de nem zárja ki annak

lehetőségét. A plágiumvádát bebizonyítani nem sikerült, mivel mindkét dolgozat eltűnt a Pécsi Tudományegyetemről, így Gyurcsány Ferenc is megmenekült a felelősségre vonás alól [29].

Plágiumgyanúba keveredett Semjén Zsolt, miniszterelnök helyettes is. A nemrégiben, szintén a HVG internetes oldal szerkesztői által kirobbantott botrány még nem zárult le végérvényesen. Először az 1991-ben megvédett teológiai doktori disszertáció körül találtak egyezéseket. A dolgozat megközelítőleg 40%-a magyar, német és angol források átvétele, fordítása. Legnagyobb terjedelemben Helmuth von Glasenapp-tól vesz át részleteket, ez a dolgozat hatodát teszi ki. Az egyezések többsége idézőjelek nélküli, szó szerinti fordítás, illetve rövidítés. A továbbiakban kiderült, hogy nagy mennyiségű egyezés található Semjén Zsolt dolgozata, illetve konzulense, Molnár Attila Károly későbbi írása között, pedig nem jelölték még egymást társszerzőként. Noha az ügy eddig nem vont magával pozícióvesztést, a gyanú mindenképpen foltot ejt a róla kialakult képen [23].

Az említett botrányok kínosak voltak a velük kapcsolatban álló intézményeknek is, hiszen a nyomozás, illetve az igazság kiderítése sok esetben lehetetlen. A megelőzés hatásosabb ezekben az esetekben, mint az utólagos vizsgálat. A következő fejezetben tárgyalt programok ellenőrzése során ezeket az eseteket próbálom rekonstruálni, bemutatva, hogy ezek a programok képtelenek az ilyen és a hasonló esetek megelőzésére.

4 Anyag és módszer

Az általam vizsgált szoftverek közül kettő meglehetősen könnyedén félrevezethető a szavak sorrendjének megváltoztatásával, míg egy harmadikat a szöveg hosszával sikerült megtéveszteni. Kutatásom kezdeti szakaszán a Google keresőmotort és 3 ingyenesen elérhető szoftvert vizsgáltam. A következő programokat teszteltem:

- KOPI Plágiumkereső
- Plagiarism Detector
- Plagiarism Checker

A kutatásom alapját öt dokumentum képezte, melyeket megvizsgáltam a Google és a három plágiumkereső szoftver használatával is. Az első dokumentum egy Wikipédiáról kimásolt szöveget tartalmazott, a második egy nagy terjedelmű, Internetről származó, 38 oldalas cikk, míg a harmadik egy IEEE által indexelt cikk fordítószoftverrel készített változata, ami a fordítás után értelmetlen szöveggé alakult. Vizsgáltam továbbá 2 korábbi szakdolgozatot is, ezeket az Internetről töltöttem le. Ahol szükséges volt, ott más dokumentumokat is felhasználtam a teszteléshez, kettő teljesen megegyező, egy részben egyező és egy az előbbiektől teljesen különböző dokumentumot a saját művek közti egyezőség ellenőrzésének teszteléséhez. A dokumentumokban közös, hogy nem tartalmaz saját munkát, tartalmuk az internetre mások által feltöltött anyagból származik. Ezeket a dokumentumokat azért készítettem, hogy alátámasszam segítségükkel azon állításomat, hogy a plágiumkereső szoftverek működése nem elégíti ki a mai kor igényeit.

4.1 Részletek a vizsgált dokumentumokból

Szükségesnek tartom ismertetni a szövegek egyszerűségét, azért, hogy az Olvasó megérthesse, miért kiábrándítóak a programok által közölt eredmények. A szövegrészletek megismerése után senki sem fog kételkedni azon megállapításban, hogy ha a programok ilyen nyilvánvaló és leplezetlen plágium esetében kudarcot vallottak, akkor hasonlóképpen eredménytelenek lesznek az egyetemi alkalmazás során is.

Elsőként vessünk egy pillantást a Wikipédiáról másolt szövegre. Amint azt láthatjuk, az eredeti szöveget módosítás nélkül emeltük át a saját dokumentumunkba.

„A kutya a farkas házasításával (domesztikációjával) jött létre, amint azt mitokondriális DNS-adatok is bizonyítják. A házasítás kezdetének időpontját tudományos viták övezik, de általában 10 000–100 000 évvel ezelőtre teszik, vagyis a kutya a középső kőkorszak vagy az újkőkorszak óta társa az embernek. A házasítás kezdeteiről semmilyen dokumentum sem maradt fent, ezért főképp feltételezésekre hagyatkozhatunk. A németországi Oberkassel környékén feltártak egy hozzávetőleg 33 ezer éves kutya állkapcsot, amely a legkorábbi ismert házasított állat maradványa lehet, illetve Szibériában egy koponyát. A két lelet azt a feltételezést támasztja alá, hogy a domesztikáció egyszerre több helyen ment végbe.

Létezik olyan vélemény is, hogy a kutyák az emberrel együtt jelentek meg hozzávetőleg 125 ezer évvel ezelőtt. Ennek a véleménynek azonban ellentmond, hogy olyan vadkutyát nem találtak, amely nem elvadult volna. A legtöbb lelet Izraelből és Palesztina területéről származik, ami a kutya délnyugat-ázsiai eredetét látszik igazolni. A koponyaszerkezet alapján viszont a kelet-Ázsiai kialakulás és a kínai farkastól való származás is lehetséges. A mitokondriális DNS-vizsgálatok alapján is a kelet-ázsiai teória a valószínűbb, ugyanis innen származtak a legváltozatosabb minták. Viszont mára már bizonyos, hogy a kutya kizárólagos őse a farkas. (Konrad Lorenz egy ideig bizonyos kutyaféléknek más őst, az aranysakált tulajdonította, de később visszavonta elméletét. A kutyaajták közötti genetikai variáció nem kisebb, mint a farkasok és a kutyák közötti (0,2%).” [10]

A 38 oldalas dokumentum több cikk egyesítésével lett létrehozva. Ebből egy részlet:

„Az augusztusi 74,6 százalékos szobafoglaltság a változatlanul erős külföldi keresletnek köszönhető, az év nyolcadik hónapjában 21 százalékkal több éjszakát töltöttek külföldi vendégek a főváros szállodáiban, mint tavaly ilyenkor. Feltűnő, hogy ennek ellenére az augusztusi átlag szobaár nem nőtt, azonos maradt a múlt évvel - állapítja meg a szövetség.

Az MSZÉSZ szerint az adatok azt igazolják, hogy a Malév csődje miatt, a fapados légitársaságok térnyerésével átrendeződik a főváros turisztikai piaca: nő a kevesebbet költő turisták száma, illetve részaránya, emiatt a szobaárak stagnálnak vagy alig nőnek. Az átrendeződés vesztesei a magasabb kategóriájú szállodák, kedvez viszont a helyzet a háromcsillagos hoteleknek.

Az idén az első nyolc hónapban a magyarországi szállodák átlagos szobafoglaltsága 47,7 százalék volt, 0,9 százalékponttal több a múlt événél. A bruttó átlag szobaár csupán 4,3 százalékkal nőtt, lényegesen elmaradt a KSH által jelzett 6,6 százalékos inflációtól.”[1]

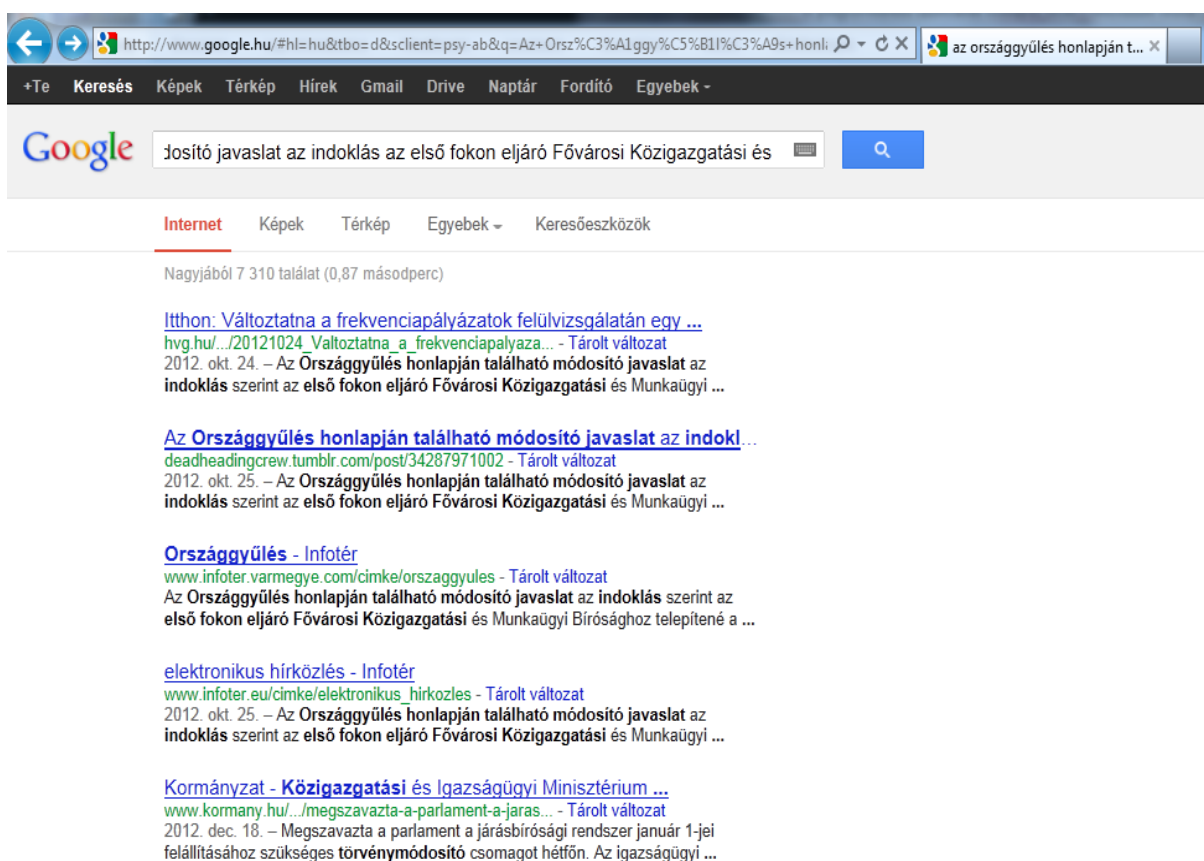
Végül néhány sor a fordító segítségével kreált szövegből:

„Ez a 2020-as évre, valamint ifjú Tom és Sara várják első gyermeküket. Együtt kiválasztásával a legújabb high-tech babakocsi, szedés ki a kiságy, és díszítését a gyerekszobában, akkor letölteni az újszülött kérelem suite azok univerzális kommunikátor, az általuk használt, amit mi hívjuk a SmartPhone 20,0. Mielőtt az esedékesség napján veszik a telefont egy túra a ház, hagyta, hogy a telefon szenzorok és gépi tanulási algoritmusok létrehozása fény és hang ujjlenyomatok minden szobában. Amikor letelepedni Tom Jr. le az első nap otthon, hogy helyezze a SmartPhone 20,0 az ágyában. Megértése, hogy a kiságy, ahol a baba alszik, a SmartPhone aktiválja a hirtelen csecsemőhalál szindróma (SIDS) alkalmazása és használja a beépített mikrofon, gyorsulásmérők és egyéb érzékelőkkel figyeli kis Tommy szívverés és a légzés. A Baby álláspont app elemzi az élő video stream, hogy Tommy ne fordítsa fel a gyomra a helyzetben, hogy az orvosi folyóiratokban is jelentés hozzájárul a bölcsőhalál. Persze, a legjobb gyakorlatok gyermeknevelés úgy tűnik, hogy gyorsan változnak, de Tom és Sara nem túlságosan aggódnak, mert az újszülött kérelem suite frissíti magát a legújabb orvosi eredményeket. A csend Tommy aludni, a SmartPhone 20,0 játszik zenét, kipróbálása a különböző beállításokat, és tanulás megfigyelés, amely a zene leginkább megnyugtató ehhez a csecsemő.”[3]

A két szakdolgozathoz a szerzői jogok védelme miatt szándékosan nem idézek. Nem szeretnék már elfogadott szakdolgozatokat sem meggyanúsítani, sem pedig lebuktatni. Ezen okból kifolyólag a dolgozatok forrását, szerzőjét és címét is elhallgatom, elvégre ezeknek semmi jelentősége a plágiumkeresésben. A továbbiakban szakdolgozatok néven fogok ezekre a munkákra hivatkozni.

4.2 Google

A Google (1. ábra) 1998-ban kezdte meg működését, napjainkra pedig a világ leghasználtabb internetes keresőrendszerévé nőtte ki magát [4]. Megjelenése megkönnyítette a releváns tartalmak keresését, így a plagizálók egyik fő eszközévé vált. A Google-t két okból vizsgáltam. Egyrészt kíváncsi voltam, hogy visszafelé is működik-e, tehát képes-e megtalálni a dokumentumunk forrásait, másrészt, tökéletes viszonyítási alap a plágiumkereső szoftverek által produkált eredményének értékeléséhez.



1. ábra: Plágiumkeresés a Google használatával [5]

Elsőként a Wikipédiáról másolt szövegre kerestem rá. A Google használatát plágiumkeresésre ellehetetleníti az a tény, hogy maximum 32 szavas szövegre lehet rákeresni. Ennél hosszabb bekezdés esetén, a keresés nem fog lefutni. A szöveget mondatonként ellenőrizve a keresőmotor számos találatot adott, az eredeti forrást is megtalálta.

A 38 oldalas dokumentum vizsgálatakor szintén mondatokra kerestem rá. A keresés ebben az esetben is sikerrel végződött, a Google bármely mondat esetében megtalálta a szöveg forrását. Az ellenőrzés mondattöredékek esetében is jól működött. Az egyetlen dolog, amire ügyelni kellett, az, hogy valóban másolt szövegre keressek rá. Lehetséges, hogy az általunk plágiumnak gondolt mondaton felül további részletek is át lettek véve a forrásból. Ilyenkor nehézséget jelenthet, hogy keresőmotorunk nem veti össze a találatokat a dokumentummal, csak annak az egy mondatnak az egyezőségére koncentrál. Ezt az összehasonlítást nekünk kell elvégeznünk saját kezűleg.

Fordítási plágiumok kiszűrése a Google segítségével meglehetősen nehézkes. Bármelyik mondatot másoljuk be a keresősávba, a keresés semmilyen esetben sem végződik eredménnyel. Az ilyenkor szokásos teendők, mint a kulcsszavak cseréje, általánosítása vagy csökkentése sem segít. Megoldást jelenthet a problémára a szöveg újbóli fordítása, mely meg is jelenik opcióként. Ezt saját kezűleg kellett elvégezni. A fordított szöveget újból bemásolva a keresőbe, már meg találja annak forrását.

A szakdolgozatok vizsgálatakor a legnagyobb nehézséget a vizsgálandó mondatok kiválasztása és a szöveg hossza okozta. Bármely mondat beillesztésekor a Google ad találatokat, de ezek összevetése kevés esetben járna tényleges eredménnyel. A Google minden esetben elvégzi a keresést, a felhasználó felelőssége viszont a megfelelő mondatok kiválasztása és az összehasonlítás elvégzése. Ez túl nagy terhet ró egy egyetemi oktatóra.

4.3 KOPI Plágiumkereső

Az általam vizsgált első szoftver az MTA SZTAKI Elosztott Rendszerek Osztálya által fejlesztett és üzemeltetett KOPI online Plágiumkereső (1. ábra). A kutatások 2001-ben kezdődtek el, majd 3 évnyi munka után, 2004-ben elindult a szolgáltatás magyar és angol nyelven, 2011-ben pedig fordítási plágiumok felderítésére képes algoritmussal bővült. A program az egyetlen magyar fejlesztésű plágiumkereső szoftver.

sztaki kopi

SzótárKOPINDAKereső

Plágiumkeresés és dokumentumkezelés

FeltöltésDokumentumaimPlágiumkeresőFutó keresésekÜzenetek

Kérem válasszon plágiumkeresési formát:

■ Egynyelvű keresés - dokumentumok összehasonlítása:

☐ minden felhasználó dokumentumaival

■ Többnyelvű keresés (tesztüzem) - dokumentumok összehasonlítása:

☐ az angol Wikipédiával

☐ a magyar Wikipédiával

Plágiumkeresés indítása >>

Cím	Szerző	Feltöltés dátuma	
a plagium	asd	2012.11.03.	Törlés

Amennyiben valamelyik dokumentumot mégse szeretné, hogy részt vegyen a keresésben, eltávolíthatja a "törlés" gombbal a listából.

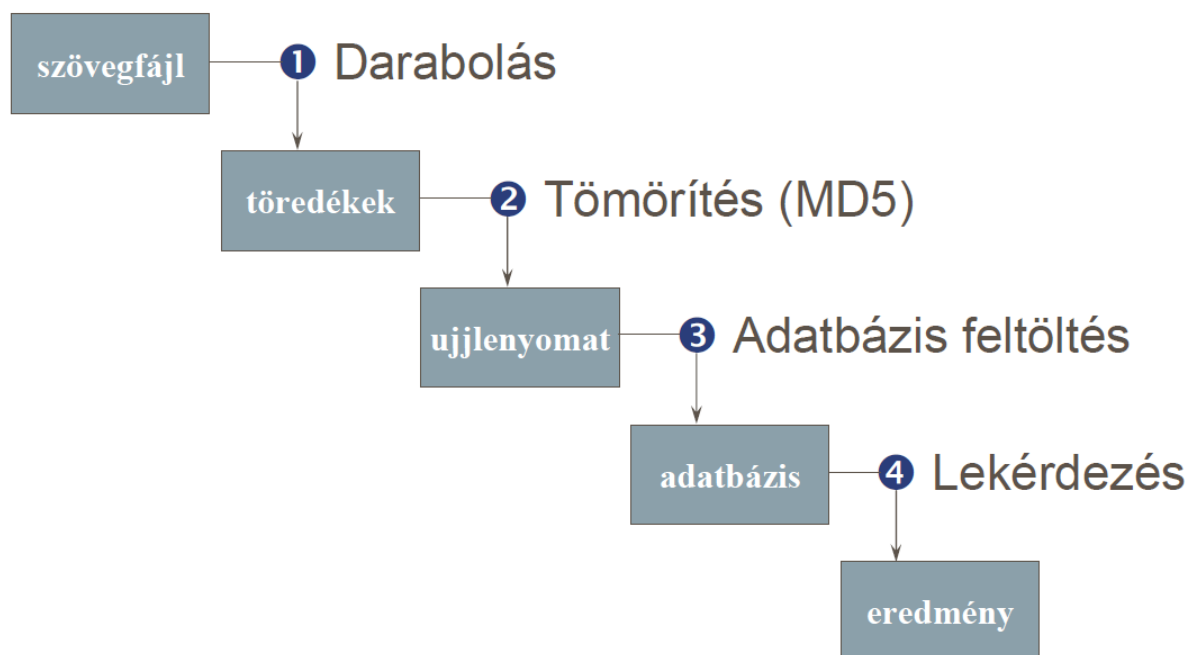
Plágiumkeresés indítása >>

2. ábra: A KOPI Plágiumkereső felhasználói felülete [8]

Használatához regisztráció szükséges. A bejelentkezés után a vizsgálandó dokumentumokat fel kell tölteni, megadni a címét és szerzőjét. Pár perces konvertálást követően, a számunkra megfelelő keresés kiválasztása után kezdhetjük is az ellenőrzést. Itt több lehetőség közül választhatunk. Egynyelvű keresés esetén az általunk feltöltött dokumentumok között kereshetünk átfedést, továbbá összehasonlíthatjuk a mi dokumentumunkat a mások által feltöltöttekkel. Többnyelvű keresés esetén a dokumentumunkat a magyar, illetve az angol

10

Wikipédia szövegével vethetjük össze. Ez csak egy szűk halmaza az Interneten fellelhető dokumentumoknak, másolni pedig – mint azt bizonyítom is- sok más helyről is lehet.



3. ábra: A KOPI Plágiumkereső működése [8]

A KOPI Plágiumkereső működése négy lépésből áll (2. ábra), legyen az egy saját dokumentum, vagy a Wikipédia valamely cikke. A forrás szöveget a program feldarabolja 40-60 karakternyi töredékekre. A töredékek tárolása ebben a formában nagyméretű adatbázist eredményezne, ezért a töredékekből úgynevezett ujjlenyomatokat hoz létre egy tömörítési eljárással. Ennek két előnye is van, egyrészt ebben a formában jóval kisebb helyet foglal, másrészt ezekből az ujjlenyomatokból nem állítható elő az eredeti dokumentum. A program az adatbázisba felkerülő ujjlenyomatok között keres egyezést egy lekérdezés futtatásakor [8].

A korábban említett dokumentumok ellenőrzése meglehetősen érdekes eredményeket adott. Elsőként a Wikipédiáról kimásolt szöveget ellenőriztettem. A felhasználók dokumentumaival való összevetés még kielégítő eredményt adott. Valaki már előttem töltött fel az enyémmel egyező anyagot, ezt a hasonlóságot könnyedén kimutatta. Ez után újra kellett futtatni a programot, hogy a magyar Wikipédia szövegével is összevethessük. A plágiumkereső 26 mondatnyi egyezést talált csak, ami a szöveg 68%. Ez elenyésző, ahhoz képest, hogy a dokumentum teljes tartalma a Wikipédiáról származott. Az angol Wikipédiával való összevetés értelemszerűen nem hozott eredményt.

A második dokumentum az internetről másolt hosszú szöveg volt. A feltöltött munkákkal való összevetés itt is jól működött, 100%-os egyezést mutatva, egy másik általam feltöltött dokumentummal. A másik két ellenőrzés viszont nagyon lassan futott le, a futás majd egy órát vett igénybe, az eredmény pedig az, amire előzetesen számítani lehetett. A program nem talált egyezést, mivel a szöveg nem a Wikipédiáról származott.

Harmadikként az IEEE által indexelt cikken végeztem el az ellenőrzést. Ez egy értelmetlen szöveggé vált a fordítást követően, így még csak hasonlót sem talált az adatbázisban, plágiummentesnek nyilvánítva a szöveget. Ezzel megegyező eredményt adott a kereső a magyar és az angol Wikipédiával való összehasonlítás eredményeként, ami nem meglepő, hiszen a forrásszöveg nem ezekről az oldalakról származott.

A szakdolgozatok ellenőrzése azonos, 99%-os eredményt adott. A program mindkét szakdolgozat esetében talált pár mondatos egyezést. Ezek egy része csupán a tulajdonnevek egyezésének köszönhető, míg a többi valóban plágiumgyanús mondat volt. Ezek többsége fordítási plágium volt. Átfutott a program viszont olyan mondatokon, amelyeket a többi szoftver plágiumnak titulált. Akárcsak a 38 oldalas dokumentum esetében, a szakdolgozatok ellenőrzése is meglehetősen lassan futott le.

Szükségnek éreztem tesztelni a saját dokumentumok összehasonlítása funkciót. A funkció csak ennél a programnál jelenik meg, ráadásul rendkívül hasznosnak találtam, mivel gyakori, hogy a hallgatók egymás írását másolják le. Ehhez két teljesen egyező, egy részben egyező és egy teljesen különböző dokumentumot töltöttem fel. A dokumentumok összehasonlításának eredménye hűen tükrözte a valóságot, ez a funkció helyesen működött.

A program részletes vizsgálata során igen hamar falakba ütköztem. A program 10 ellenőrzés lefuttatása után nem engedett több keresést lefuttatni, azzal az indokkal, hogy a napi keret lejárt. Folytatva a munkát a program már a havi keret elhasználását írta ki. A napi korlát megegyezik a havi korláttal, míg éves szinten 120 lekérdezést engedélyez. Ezen felül a program használatáért fizetni kell. A lekérdezések számát megköszörözi, hogy egy dolgozat összevetését a többi dolgozattal és a Wikipédiával nem lehet egyidejűleg elvégezni. Egy évfolyam dolgozatainak kijavítása, ellenőrzése akár több hónapot is igénybe vehet, emellett pedig magas költségeket ró az egyetemekre.

4.4 Plagiarism Detector

A következő vizsgált szoftver a Skyline, Inc. által fejlesztett Plagiarism Detector (3. ábra) volt, melyet a készítők számos egyetemmel együttműködve hoztak létre. A program fejlesztése 2005-ben kezdődött el, majd több díjat is nyert 2008 és 2010 között. Számos nyelvet támogat, köztük több világnyelvet, de sajnos a magyart nem. Több tízezer felhasználóval rendelkezik, használata főleg az Európai Unió és az Amerika Egyesült Államok területén terjedt el.

Generated Reports: click once to view short details, double-click to view detailed analysis

Filter Settings:

Check type:	Date:	Sorting:	View Style:
☒ - show all	☐ - Filter by Date:	☐ - by generation time	☐ - detailed\chart
☐ - show internet only	2012. november 3. ▾	☒ - by plagiarism %	☒ - compact\compare
☐ - show database only		☐ - by references %	

Diagram:	Date:	Checked Document Name:	Check Type:	Plagiarism:	Original:	Quoted:	Linked:
	2012.11.03. 17:58:16	wikipedia.docx	Internet	79%	21%	0%	0%
	2012.10.24. 22:32:33	sokoldal.docx	Internet	19%	81%	0%	0%
	2012.10.24. 21:49:51	google_fordito.docx	Internet	0%	100%	0%	0%

4. ábra: Eredmények a Plagiarism Detector-ban [19]

Használat előtt a programot le kell tölteni, majd telepíteni, ezután kezdhethetjük el használni. Regisztrációt nem igényel. Az ellenőrzés kivitelezését egy varázsló segíti. Először meg kell adnunk a vizsgálni kívánt dokumentum elérési helyét, majd el kell döntenünk, hogy az adatbázisban, az interneten, esetleg mindkettőben egyszerre szeretnénk keresni. A keresés lefutása után a program egy másik ablakban mutatja meg az eredményt. Itt a listázás feltételeit változtathatjuk, annak függvényében, hogy mire vagyunk kíváncsiak. Megadhatjuk például, hogy milyen típusú kereséseket mutasson, a keresések dátumát, mi szerint legyenek rendezve, hány százalék felett mutassa, illetve megváltoztathatjuk a nézetet is. A program ábrákkal szemlélteti, hogy a szöveg hány százalékát találta eredetinek illetve plágiumnak, továbbá megadja a linkelt és az idézett sorok százalékos arányát is.

A Plagiarism Detector működésének fő lépései:

1. A dokumentum szövegének feldarabolása
2. A darabok elküldése egy keresőmotornak
3. A keresés találatainak letöltése
4. A lehetséges források összevetése a szöveggel
5. Az eredetiségvizsgálat eredményének elkészítése

A program működése 5 lépésből áll. A megadott dokumentumot a KOPI Plágiumkeresőhöz hasonlóan apró 40-60 karakteres darabokra vágja, ezeket a darabokat pedig elküldi egy keresőmotornak. A program a Google-t, a Bing-et és az AltaVista-t használja. Sosem tárolja a keresett dokumentumot és nem is publikálja azokat. A keresés eredményeit letölti a gépre, ezek a lehetséges források. Ezeket a program összehasonlítja az általunk megadott fájlal, majd ezekből elkészíti az eredetiség vizsgálat eredményét. A böngészővel megnyitva még részletesebb adatokat kaphatunk a keresés részleteiről. A program minden esetben a saját erőforrásainkat használja, így a rendszer túlterhelése okozta viszontagságoknak nem leszünk kitéve.

Elsőként itt is a Wikipédiáról másolt szövegen futtattam le a keresést. A program valóban – a gyártók oldalán említetteknek megfelelően – egy-két percen belül lefutott. Ez jóval kevesebb, mint a korábban vizsgált KOPI Plágiumkereső futási ideje. A Plagiarism Detector az eredményben is felülmúlta vetélytársát, noha az eredmény továbbra sem tökéletes. A program 79%-ban határozta meg a másolt szöveg mennyiségét, ami nagyságrendekkel jobb a KOPI 68%-os eredményéhez képest, de még mindig rossz, ahhoz viszonyítva, hogy a dokumentum 100%-a plágium, illetve, hogy a készítők a honlapon 99%-os eredményt ígérnek. Az eredmény már csak azért is érdekes, mivel a forrásszöveget megtalálta az Interneten, az összehasonlítás során mégsem találta teljesen egyezőnek őket.

Folytatva a tesztelést, elvégeztem a plágiumkeresést a 38 oldalas dokumentumon is. Elsősorban arra voltam kíváncsi, hogy mennyivel hosszabbodik a keresés időtartama, továbbá, hogy mennyiben befolyásolja az eredményességet a szöveg hossza. A futamidő a várakozásaimnak megfelelően hosszabbodott, de a növekedés nem volt drasztikus. A program 3 perc alatt lefutott. Visszaemlékezve a KOPI Plágiumkereső által produkált, megközelítőleg egy óras ellenőrzésre megállapíthatjuk, hogy a Plagiarism Detector nagy terjedelmű szöveg esetén is gyorsan lefut. A program talált egyezést, de közel sem mindet. A szöveg 81%-át eredetinek találta, és csak 19%-nál fedezett fel plágiumgyanút.

Ezután az IEEE által indexelt idegen nyelvről fordított cikket vetettem ellenőrzés alá. A futtatás után a program által készített jelentésben szereplő eredmény kiábrándító. A kereső átengedett a szűrésen egy teljesen értelmetlen, fordító segítségével kreált szöveget, 100%-ban eredetinek titulálva azt, nem találva egy mondatnyi plágiumot sem.

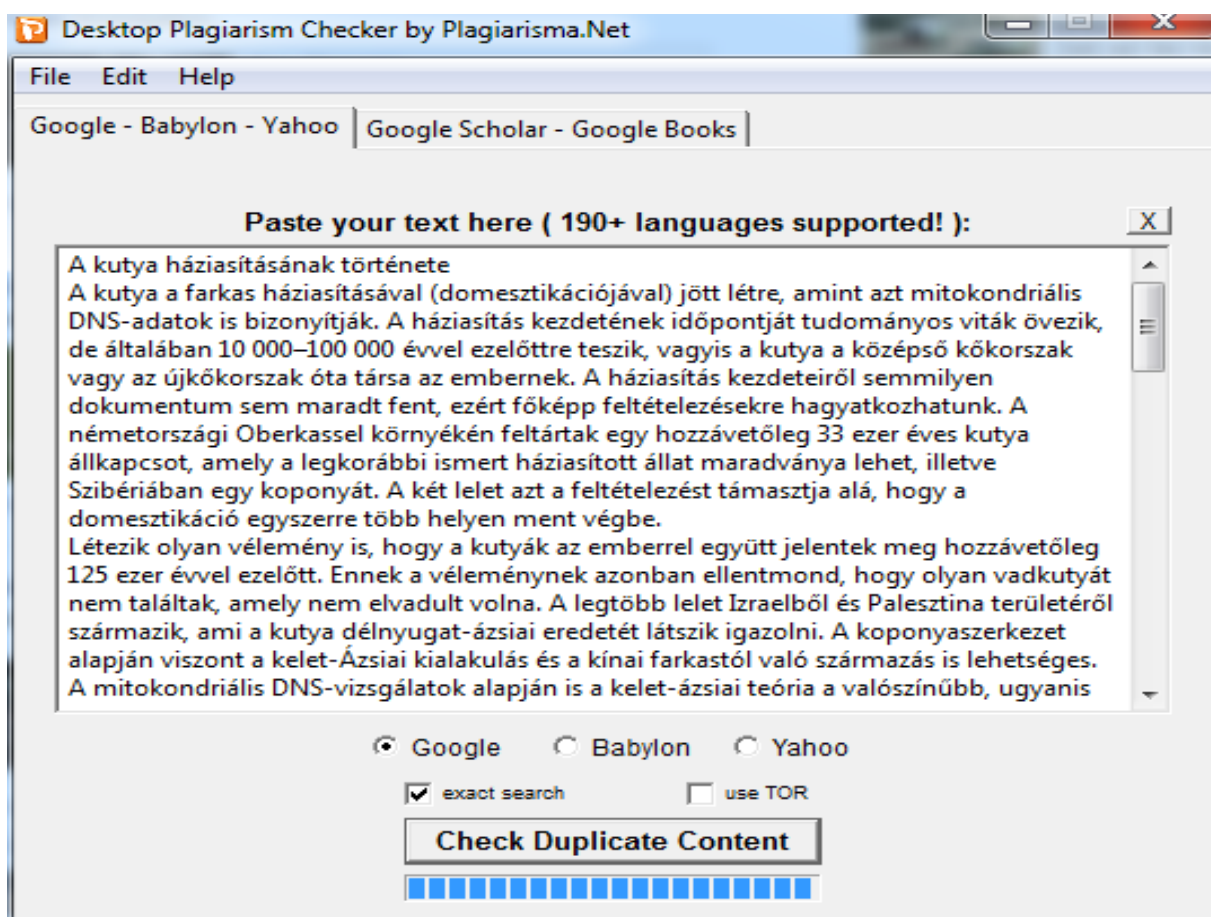
A Plagiarism Detector használatával egyszerűen lehetett a szakdolgozatokat leellenőrizni. A vizsgálandó dokumentum kiválasztása után a keresés néhány perc alatt lefutott. A dokumentumot 98%-ban eredetinek találta a plágiumkereső szoftver, de talált pár mondatos egyezéseket is. A lehetséges forrásokat meg is jelenítette a megfelelő helyen. Ezzel szemben viszont nem találta meg azokat a fordítási plágiumokat, amelyeket a KOPI korábban felfedezett.

A Plagiarism Detector a plágiumkeresésen felül lehetőséget ad az idézések ellenőrzésére is. Ezt az általam készített dokumentumok minimális átalakítása után végeztem el. A változtatás lényege abban rejlett, hogy a szöveget lehivatkoztam. Az idézések vizsgálata lényegében mennyiségi ellenőrzés, melynek eredménye megmutatja, hogy a tartalom mekkora hányada van helyesen lehivatkozva. Ez a funkció nem működik jól. Helyes és egyforma hivatkozással ellátott dokumentumok esetében, ez egyiknél felismeri azt, míg a másiknál nem. Ez súlyos hiba, hiszen közel sem mindegy, hogy a szövegünk plágium vagy idézet.

Vizsgálataimat a program demó verzióján végeztem el. Munkámat ez két ponton korlátozta. Egyrészt a demo program tíz futtatást engedélyez. Ez alatt korlátlan mennyiségű ellenőrzést végezhettem, így csak arra kellett figyelnem, hogy ne zárjam be az alkalmazást idő előtt. Másrészt ebben a verzióban le van tiltva az adatbázisban való, illetve a kombinált keresés. Kutatásom során ezekre nem volt szükségem, hiszen az általam használt dokumentumok az Interneten is fellelhetőek voltak.

4.5 Plagiarism Checker

A harmadik vizsgált plágiumkereső szoftver a Plagiarisma.Net által készített Plagiarism Checker (4. ábra) volt. A program vásárlói neves, főleg amerikai középiskolák és egyetemek, mint a Benjamin Franklin High School, The University of Michigan, Harvard University stb. A programhoz van letölthető alkalmazás, de elérhető online is. Több tucat nyelvet támogat, de magyarul sajnos nem elérhető.



4. ábra: A Plagiarism Checker keresés közben [18]

Ha a programot online szeretnénk használni, akkor a <http://plagiarisma.net/> oldalon történő regisztrálás után már kezdhethetjük is a plágiumkeresést. Az ellenőrizendő szöveget be kell másolni a megfelelő ablakba, majd a számunkra tetszőleges keresőmotor - Google, Babylon vagy Yahoo - kiválasztása után indíthatjuk is a keresést. A kereső egyesével vizsgálja meg a mondatokat, az eredmény az eredeti mondatok és az összes mondat aránya százalékban kifejezve. Az egyes szövegrészek eredményeit külön megjeleníti. Az alkalmazás működése hasonló. Itt meg is nyithatjuk a szöveget, nem feltétlen kell nekünk bemásolni. Az ellenőrzést lefuttatva az online kereséshez hasonló módon írja ki az eredményt. A keresés folyamata mindenben megegyezik az előbb vizsgált Plagiarism Detector esetében bemutatottakkal.

Az ellenőrzést – mint mindenhol máshol - itt is a Wikipédiáról másolt szöveggel kezdtem. A futási ideje megegyezett az korábban vizsgált Plagiarism Detector futási idejével. A keresést a Yahoo keresőmotor kiválasztásával lefuttatva egy eredeti mondatot talált, a szöveg 91%-át pedig plágiumnak minősítette. Ugyanezt a szöveget ellenőrizve a Google is eredetinek minősítette az előbb említett mondat túlnyomó részét, de a mondat utolsó két szavát már másoltak vélte, 92%-ban plágiumnak minősítve a szöveget. Végül lefuttattam a Babylon-nal is a keresést. Ez már a teljes szöveget plágiumnak vélte, tökéletes eredményt adva.

Ezután következett a 38 oldalas dokumentum. A szöveg importálása kisebb nehézségeket okozott, igen lassan történt meg, de ez indokolható a szöveg hosszával. A futási idő mindhárom keresőmotor esetében hasonló volt, illetve megközelítőleg ugyaannyi, mint a Plagiarism Detector esetében. Érdekes, hogy a különböző keresőmotorokkal elvégzett keresések eredménye itt is eltérést mutat. A Google segítségével végzett ellenőrzés 94%-ban találta plágiumgyanúsaknak a szöveget. Ezután elvégeztük az ellenőrzést a Yahoo használatával is, melynek eredménye, hogy a teljes szöveg plágium. Ugyan ezt az eredményt kaptam a Babylon keresőmotorral végzett ellenőrzés eredményeként is.

Következőként az IEEE által indexelt angol nyelvről fordított szövegen végeztem el a plágiumkeresést, mindhárom keresőmotor segítségével. Kijelenthetem, hogy a Plagiarism Checker nincs felkészítve fordítási plágiumok felderítésére. A Google kiválasztása után elvégzett keresés nem találta meg a szöveg angol megfelelőjét, így a szöveget teljes egészében eredetinek találta. Ugyanezen feladtnál a Yahoo segítségével elvégzett keresés sem fedezett fel plágiumot, majd a Babylon is plágiummentesnek minősítette a szöveget.

Utolsóként a szakdolgozatok ellenőrzését végeztem el. A keresések mindkét dolgozat esetében mindhárom keresőmotor használatával közel azonos eredményt adtak. A munkák 98-99%-a szoftverek szerint eredeti. Ezt a csekély különbséget, az eltérő keresőmotoroknak tudhatjuk be. A vizsgálat végén a Plagiarism Detector segítségével végzett ellenőrzés során kapott linkeket kaptuk eredményül. A Plagiarism Checker sem találta meg azokat a szövegrészleteket, amelyek fordítási plágium eredményeként jöttek létre.

4.6 Plágiumkeresés ujjlenyomatok segítségével

Egyetemi tanulmányaim alatt megtanultam, hogy egy algoritmus készítése közben, 3 fő lépést kell végrehajtani. Első lépésként meg kell adni pontosan az elvégzendő feladatot. Egyszerű lépésnek tűnik, de közel sem az. Ezután következik a feladat algoritmizálása. Ez a legnehezebb feladat, számos lehetőség közül kell kiválasztani a feladat elvégzésére legalkalmasabb megoldást. Utolsó lépésként meg kell határozni a készített algoritmus határait, hogy véletlen se próbáljak meg elvégeztetni vele olyan feladatot, amelynek elvégzésére képtelen.

Az algoritmus feladatát már meghatároztam korábban is. Keresse meg a plágium előfordulásait a szövegben.

Az algoritmizálás már nem fog ilyen könnyen menni. A fellelhető algoritmusok közt kevés van, amely bármely nyelv esetén működik, de még kevesebb olyan, amely a tesztelések során bízató eredményt adott. Ezen kevés lehetőség közül kell kiválasztani a feladat számára legmegfelelőbbet.

Bármely ellenőrzés elvégzése előtt ajánlott végrehajtani néhány előkészületi lépést. Először is törölni kell a keresés szempontjából irreleváns információkat [14]. Ilyenek lehetnek például az írásjelek, a helykihagyások, kis- és nagybetűk, felsorolások. Ezután érdemes kitörölni a felesleges szavakat is. Ezek közé soroljuk azokat a szavakat, melyek önmagukban nem bírnak semmilyen jelentéssel, például a névelők, kötőszavak [2]. Ezekkel a lépésekkel leegyszerűsíthető a szöveg, ezáltal csökken annak mérete, ami a plágiumkeresés futásidejének csökkenését eredményezi, de vigyázni kell, hogy csak olyan részek kerüljenek törlésre, amelyek valóban nem befolyásolják a szöveg tartalmát [9].

Most, hogy a dokumentum mérete a lehető legkisebbre csökkent elvégezhető a dokumentum feldarabolása. A választott módszer nagyban befolyásolja a keresés eredményét és sebességét, ezért fontos, hogy ismert legyen a feladat elvégzésére legalkalmasabb megoldás [30]. Legígéretesebbnek a hash-elt töréspontok szerinti darabolás átfedéssel ígérkezik a szerte a világban elvégzett tesztek alapján [16].

Legyen k (1) értéke:

$$k = \sum_n^1 x_n, \quad (1)$$

ahol x értéke a vizsgált egység első, második,..., n -edik karakterének ASCII kódtábla szerinti értéke. Természetesen n minden esetben megegyezik az egységben szereplő betűk számával.

Legyen m :

$$m \in N, \quad (2)$$

m (2) értékét érdemes a nyelv grammatikájának tulajdonságai alapján megválasztani [13]. A magyar nyelv (3) esetében

$$m = 16 \quad (3)$$

egy jó választásnak tűnik az elvégzett tesztek alapján.

Legyen a függvény (4) értéke, mely eldönti, hogy szükséges-e töréspontot beszúrni a vizsgált egység után:

$$f(k) = k \bmod m. \quad (4)$$

Töréspontot abban az esetben kell beszúrni (5), ha a vizsgált egységnél:

$$f(k) = 0.$$

(5)

Ekkor azok a szavak lesznek, a rész elemei, amelyek hash értéke a legutóbbi töréspont óta vizsgálva lett [15]. Az átfedés létrehozására (6), az eljárást meg kell ismételni egy másik:

$$p \in N \neq m$$

(6)

számmal is, m helyére minden esetben p -t helyettesítve. A p értékének megválasztása is a használt nyelvtől függ.

Következő feladat az ujjlenyomatok elkészítése. Erre két okból van szükség. Egyrészt így kevesebb helyett vesz igénybe a dokumentum tárolása, másrészt az eredeti dokumentum nem lesz visszaállítható [25]. Az ujjlenyomatok készítésénél négy dologra kell odafigyelni. Az első a generátor kiválasztása. Fontos, hogy az egyező darabokhoz generált ujjlenyomatok minden esetben egyezzenek, míg a különböző szövegrészletekhez tartozó ujjlenyomatok különbözzenek. Ez után következik a bemenő adatok méretének kiválasztása, mikor is megadásra kerül az adatok mérete, melyekből az ujjlenyomatok el fognak készülni. Harmadik lépésként meg kell adni, hogy hány ujjlenyomat képviselje a dokumentumot. Ez lehet egy konstans, esetleg függhet a szöveg hosszától. Utolsó lépésként pedig meg kell határozni, melyek legyenek azok az adott hosszúságú szövegrészletek, melyekből az ujjlenyomat készül [27].

A leírt eljárást minden vizsgálandó dokumentumon el kell végezni. Az ujjlenyomatok elkészülte után azokat el kell tárolni egy hash táblában, a dokumentumok azonosításához szükséges adatokkal együtt. Abban az esetben, ha két dokumentum a megadottnál több megegyező ujjlenyomattal rendelkezik, akkor minden kétséget kizáróan az egyik dokumentum szövegének egy részlete megegyezik a másik vizsgált dokumentumával [26].

4.7 Dokumentumok összehasonlítása

Egészen eddig nem sok szó eset a dokumentumok összehasonlításáról, pedig ez a lépés is ugyanolyan fontos akárcsak a többi. A Google használatával végzett keresés esetében megtapasztalhattuk, hogy hiába ismerjük a lehetséges forrásokat, ha azok összevetése a szövegünkkel lehetetlen. A lehetséges források mennyisége általában túl nagy ahhoz, hogy saját kezűleg végezzük ezt el. Minden plágiumkereső szoftvernek képesnek kell lennie összehasonlítani a dokumentumokat, hiszen, mint azt már észrevehettük az elsődleges cél a plágiumgyanú felderítése, nem pedig a lehetséges források megtalálása. Utóbbira egy egyszerű Google keresés is képes volt. Az összevetés mondja ki az ítéletet a dokumentum ügyében, éppen ezért terheli nagy felelősség a programozót akkor, amikor ezt készíti el. Számos megoldás van az összevetésre, ezek közül fogok egy megoldást bemutatni egy rövid program segítségével, mely általam készített dokumentumokat hasonlít össze.

Első lépésben létrehozok egy Dolgozat nevű struktúrát. Ennek definíciója a következő:

```
struct Dolgozat {  
    string fajlnev, neptun, nev;  
    set<string> szoparok;  
    Dolgozat(string const& fajlnev, string const& neptun, string const& nev);  
};
```

(1)

Ennek a struktúrának (1) a használatával fogom eltárolni a dolgozatainkat. A fájlnev, a Neptun-kód és a név eltárolása string formában fog megtörténni. Ezek értékeinek inicializálására fogom használni a Dolgozat függvényt. Létrehozok továbbá egy stringeket tároló set konténert is szoparok néven. Ez lesz a dolgozatban előforduló szópárok halmaza [31].

Lássuk a meghívott Dolgozat függvényt:

Dolgozat::Dolgozat(string const& fajlnev, string const& neptun, string const& nev):

```
fajlnev(fajlnev), neptun(neptun), nev(nev){
    ifstream is(fajlnev.c_str());
    string szo, elozo;
    is >> elozo;
    while (is >> szo) {
        szoparok.insert(elozo+szo);
        elozo = szo;
    }
}
```

(2)

A függvény a megadott fájlból beolvassa az ott található szöveget ifstream használatával. A szöveget szavanként egy while ciklus segítségével eltároljuk páronként a szoparok nevű stringeket tartalmazó set típusú konténerbe.

A vezerlofajlbol nevű függvény az létrehozáshoz szükséges adatokat szolgáltatja:

```
list<Dolgozat> vezerlofajlbol(char const *vezerlofajl){
    ifstream is(vezerlofajl);
    string sor;
    list<Dolgozat> dogak;
    while (getline(is, sor)) {
        istringstream strs(sor);
        string fajl, neptun, nev;
        strs >> fajl >> neptun;
        strs.ignore(1);
        getline(strs, nev);
        dogak.push_back(Dolgozat(fajl, neptun, nev));
    }
    return dogak;
}
```

(3)

A korábban létrehozott vezérlőfájl segítségével a függvény (3) beolvassa a szükséges adatokat. A while ciklus addig fut, amíg a beolvasandó sorok el nem fogynak. A korábban bemutatott Dolgozatok függvény hívásához szükséges változók inicializálása után meg is hívja azt, majd a hívott függvény elvégzi a már leírt feladatokat.

A hasonlóságok eltárolásához a Hasonlosag elnevezésű struktúrát (4) használom:

```
struct Hasonlosag {  
    Dolgozat const *d1, *d2;  
    double m_d1, m_d2;  
    Hasonlosag(Dolgozat const& d1, Dolgozat const& d2);  
};
```

(4)

A struktúrában hivatkozok azokra a dolgozatokra, amelyeket összevetettem, illetve tárolom a hasonlóságok mennyiségét. Ezeknek a változóknak az értékét a következő függvény (5) adja meg:

```
Hasonlosag::Hasonlosag(Dolgozat const& d1, Dolgozat const& d2)
```

```
: d1(&d1), d2(&d2){  
  
    set<string> metszet;  
  
    set_intersection(d1.szoparok.begin(), d1.szoparok.end(),  
        d2.szoparok.begin(), d2.szoparok.end(),  
        inserter(metszet, metszet.begin()));  
  
    m_d1=metszet.size() / (double)d1.szoparok.size();  
  
    m_d2=metszet.size() / (double)d2.szoparok.size();  
  
}
```

(5)

A metszet nevezetű konténerben eltárolom a közös szópárokat, illetve ezek méretét is. A közös szópárokat a `set_intersection` parancs használatával válogatom ki a megfelelő paraméterek megadásával. Végül pedig kiszámítom, hogy a létrehozott metszet mekkora része az eredeti szövegnek.

A `hasonlit` nevű függvény (6) adja az összehasonlítandó dolgozatokat. Minden dokumentumot összevetek minden dokumentummal, az eredményt egy `Hasonlosag` nevű struktúrát tároló vektorban mentem el. A függvény definíciója a következő:

```
vector<Hasonlosag> hasonlit(list<Dolgozat> const& dogak){
    vector<Hasonlosag> hasonlok;
    hasonlok.reserve(dogak.size()*(dogak.size()-1)/2);

    for (list<Dolgozat>::const_iterator d1=dogak.begin(); d1!=dogak.end(); ++d1) {
        list<Dolgozat>::const_iterator d2=d1;
        while (++d2 != dogak.end())
            hasonlok.push_back(Hasonlosag(*d1, *d2));
    }

    return hasonlok;
}
```

(6)

Most, hogy minden szükséges eljárást és struktúrát ismertettem, következzen meglehetősen egyszerű main függvény (7):

```
int main(){
    list<Dolgozat> dogak = vezerlofajlbol("input.ctr");
    vector<Hasonlosag> hasonlok = hasonlit(dogak);
    system("pause");
    return 0;
}
```

(7)

A program értékét annak egyszerűsége és gyorsasága adja. Másodpercek alatt elvégzi nagyobb mennyiségű és terjedelmű dokumentumok összehasonlítását is, az eredmények pedig megfelelő rendezés és kiírás segítségével könnyedén áttekinthetővé tehető.

4.8 Külföldi kutatások eredményei

A főleg angol nyelvű cikkek olvasása során szinte kivétel nélkül mindegyik megemlíti, hogy a plágium kezelésének két lehetséges módja van: megelőzés vagy felismerés. Megelőzés kategóriájába sorolják a diákok megtanítását a helyes idézésre, a plágium fogalmának és veszélyének ismertetését, illetve a szigorú és következetes büntetést. Felismerés halmazba soroljuk a plágiumkereső szoftvereket [7]. Úgy gondolják, és eme véleményüket én is osztom, hogy ezeket az eszközöket be kell vetni az élet minden színterén és korosztálya esetén.

A külföldi kutatók véleménye szerint az írott szöveg eltulajdonítása csak a jéghegy csúcsa. Amint sikerült minden esetben működő megoldást találni erre a problémára, tovább kell lépni és folytatni a fejlesztést annak érdekében, hogy fel lehessen deríteni a tartalom eltulajdonítását is. Nem csak a plágiumkereső szoftverek, de a plágiummal élők is egyre felkészültebbé válnak, újabb és újabb kérdések és problémák fognak a felszínre kerülni.

Talán az előbb említett előzetekintésnek az útmutatását követi az a próbálkozás, amelyben szemantikai egyezéseket keresnek. A WordNet adatbázisban a szemantikai kapcsolatok kialakításával tárolják a szavakat azoknak szinonimáival együtt. A szavakat szófajaik szerint is csoportosítják, mivel azok különböző grammatikai szabályoknak kell, hogy megfeleljenek. Érdekesnek és hasznosnak találom ezt a megközelítést, talán a jövő plágiumkereső szoftvereinek alapja is lehet a szemantikai egyezések keresése, persze ehhez más nyelveket is implementálni kell az adatbázisba.

A külföldi kutatások érintik az emberi jellemzőket is. Vannak próbálkozások olyan plágiumkereső metódusok megírására is, amely az írás stílusát veszi alapul. Teszik ezt, mert hívei úgy gondolják, hogy akárcsak a kézírásnak, az elektronikus dokumentumnak is van egyedi ismertető jele. A kötőszavak használata, a formázás, a használt szinonimák, de akár még a betűméret is árulkodó lehet.

Szerte a világban megjelent az igény a plágiumkeresésre a programozásban is. Míg hazánkban az egyetemi plágiumon kívül szó sem esik a szellemi lopásról, addig külföldön ezen a területen is komoly kutatások folynak. A programozók szeretnék megvédeni forráskódjukat a jogtalan felhasználóktól.

5 Eredmények

A vizsgálatok elvégzése után szeretném röviden összegezni az általam elért eredményeket. A vizsgálatok során szerzett információkat és tapasztalatokat a program készítése során hasznosítani fogom. A vizsgált programok kiskapui útmutatást adnak, illetve meghatározzák a fejlesztés fő irányait.

	Wikipédiáról másolt szöveg	38 oldalas cikk	IEEE által indexelt cikk fordítása	szakdolgozatok
Google	100%	100%	0%	0%
KOPI	68%	0%	0%	2%
Plagiarism Detector	79%	19%	0%	1%
Plagiarism Checker	91-92-100%	94-100-100%	0-0-0%	1-1-2%

1. táblázat: A kapott eredmények összegzése

Mint azt a táblázat (1. táblázat) jól mutatja, ezek a szoftverek tökéletesen kielégítik azok vágyait, akik még mindig más munkáiból szeretnének szakdolgozatot írni. A programok használatát nem ajánlom azoknak, akik érdemben kívánnak tenni a plágium ellen, mivel a vizsgált programok meglehetősen könnyen kijátszhatóak. Az elvégzett tesztek eredményei alapján már tudom, hogy milyen kiskapukat kell bezárni. Az ellenőrzések során megbízhatóan és helyesen működő funkciók esetében programom célja a vizsgált programoknál tapasztalt eredményesség megtartása. A fontos fejlesztések a keresések futtatása során gyengén teljesítő funkciók terén lesznek. Programom fő célja a tapasztalt hibák kijavítása, a hiányosságok

pótlása, illetve egy könnyen kezelhető, mindenki számára átlátható grafikus kezelői felület készítése.

A KOPI Plágiumkeresőnél vizsgált, adatbázissal támogatott megoldást hasznosnak tartom. A fejlesztett program is rendelkezni fog egy adatbázissal, ami a korábban vizsgált dokumentumokat fogja tárolni. A korábbi években beadott munkákat nem szabad hagyni feledésbe merülni, hiszen a diákok előszeretettel vesznek alapul olyan szakdolgozatokat, amelyek korábban már megfeleltek a követelményeknek.

A KOPI által kínált, feltöltött dokumentumok összehasonlítása egymással funkciót mindenképpen szeretném megtartani. A tesztek során rendkívül megbízhatóan teljesített. Fontosságát annak köszönheti, hogy meglehetősen magas az olyan esetek száma, amikor a hallgatók egymásról másolnak. Természetesen ez a funkció nem is annyira a szakdolgozatok, mint inkább a beadandó feladatok esetében lehet hasznos, ahol többen kapják ugyanazt a témát.

Saját fejlesztésű szoftverembe szeretnék beilleszteni, egy az idézés mennyiségét mérő algoritmust. Praktikus mivolta miatt készítem el a programomhoz ezt a funkciót. A szakdolgozatokhoz szükséges igénybe venni mások munkáját, de nem mindegy, hogy ezt miként és milyen mértékben tesszük. Egyrészt a forrás megjelölésének elmulasztása plágium, másrészt a tanulmányi és vizsgaszabályzat minden esetben meghatározza, hogy a szakdolgozat hány százaléka lehet más munkásságának az idézése. A helyesen lehivatkozott forrásból származó szöveg mennyiségét fogja mérni algoritmusom, megkönnyítve ezzel az oktató munkáját.

A potenciális források keresését meglehetősen szűk halmazon végzi el a KOPI Plágiumkereső, a magyar és angol Wikipédián kívül számos helyről származhat a forrásszöveg. A másik két szoftver már a teljes webet vizsgálja. Ez a követendő megoldás a források keresésére. Az általam kapott eredmények gyengék, ezért ezen a téren sok fejleszteni való van. A források keresésénél a programnak nem csak szó szerinti másolás, hanem parafrázálás esetén is működni kell, csak ebben az esetben lehetünk biztosak abban, hogy a szakdolgozat valóban önálló munka eredménye.

A legelhanyagoltabb területnek a fordítási plágiumok vizsgálata mutatkozott. A Plagiarism Detector és a Plagiarism Checker teljesen inkompetens idegen nyelvű forrás esetén, míg a KOPI Plágiumkereső csak az angol Wikipédia szövegéről fordított szövegeket képes megtalálni. Világunkban, ahol a nyelvtudást megkövetelik és elvárják, a fordítási plágiumok száma növekvő tendenciát mutat. A fordítók fejlődésével tovább fog emelkedni. Éppen ezért, a fejlesztésben kulcsfontosságú és kiemelt szerepet fog kapni a plágium e formájának kiszűrésére alkalmas algoritmus elkészítése.

	Kezelhetőség	Értelmezhetőség	Futási idő	Használható
Google	+	-	+	ha ismert a plágium helye
KOPI	-	-	-	ha a forrás a Wikipédia
Plagiarism Detector	+	+	+	nagy terjedelmű dokumentumok esetében
Plagiarism Checker	+	-	+	rövid átfedések felderítésére

2. táblázat: A felhasználóban keltett benyomás

Ezeknél a programoknál rendkívül fontos a felhasználóbarát grafikus felület elkészítése, támogatva ezzel, az újhoz nehezen alkalmazkodó réteget. A KOPI Plágiumkeresőt leszámítva ez mind a három vizsgált program esetében sikerült (2. táblázat). A Google használatát már korábban megszokhattuk, a Plagiarism Detector használatát egy varázsló segíti, míg a Plagiarism Checker fejlesztői a felület egyszerűségével érték el ezt. Az értelmezhetőség területén a Plagiarism Detector jár élben, melynek fejlesztői színes diagramokkal és ábrákkal segítenek a kapott eredményt értelmezni. A Google-nál az összehasonlítás hiánya az

értelmezhetőségen jelenik meg, míg a Plagiarism Checker esetében pont a már korábban említett egyszerűség teszi nehezkessé az értelmezést. A KOPI Plágiumkereső jelentése bonyolult, a közölni kívánt eredmények pedig alig kivehetők. A keresés a KOPI-tól eltekintve, minden esetben hamar lefutott.

Korábban említettem, hogy a programok használatát nem ajánlom senkinek, ezért szeretnék megemlíteni néhány kivételt. Biztosak lehetünk abban, hogy ha ismerjük a plágium helyét, akkor a Google minden esetben meg fogja találni annak a mondatnak az eredeti forrását. A KOPI használatát csak akkor ajánlom, ha biztosak vagyunk benne, hogy a plágium forrása a magyar vagy angol Wikipédia szövege. A Plagiarism Detector nagy, míg a Plagiarism Checker kis terjedelmű szövegek esetében bizonyult hatásosnak.

6 Irodalomjegyzék

1. Árban sereghajtók a magyar hotelek, MTI:
http://hvg.hu/utazas/20121024_Arban_sereghajtok_a_budapesti_hotelek [2012.10.16.]
2. Efstathios Stamatatos (2011) – Plagiarism Detection Based on Structural Information
3. Generation Smartphone, Siewiorek, D:
http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=6281134&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxppls%2Fabs_all.jsp%3Farnumber%3D6281134
[2012.10.16]
4. Google kereső, Wikipédia: http://hu.wikipedia.org/wiki/Google_keres%C5%91
[2012.11.20.]
5. Google: <http://www.google.hu/> [2012.11.20.]
6. Harc az egyetemi plágium ellen, Metazin: <http://www.metazin.hu/node/1641>
[2012.09.27.]
7. José L. Bernal-Agustín, Rodolfo Dufo-López (2011) – Free Web-Based Tools with Which to Detect Plagiarism
8. Kopi Plágiumkereső: <http://kopi.sztaki.hu/> [2012.10.13.]
9. Krisztián Monostori, Raphael Finkel, Arkady Zaslavsky, Gábor Hodász, Máté Pataki (2002) – Comparison of Overlap Detection Techniques
10. Kutya, Wikipédia: <http://hu.wikipedia.org/wiki/Kutya> [2012.10.16.]
11. Maria Kashkur, Serge Parshutin, Arkady Borisov (2010) – Research into Plagiarism Cases and Plagiarism Detection Methods
12. Martin Potthast, Alberto Barrón Cedeno, Benno Stein, Paolo Rosso (2010) – Cross-Language Plagiarism Detection
13. Martin Potthast, Alberto Barrón-Cedeno, Andreas Eiselt, Benno Stein, Paolo Rosso (2011)– Overview of the 2nd International Competition on Plagiarism Detection
14. Maxim Mozgoyov, Kimmo Fredriksson, Daniel White, Mike Joy, Erkki Sutinen (2005) – Fast Plagiarism Detection Systems
15. Mohammed Alzahrani, Salha and Salim, Naomie (2008) - Plagiarism detection techniques
16. Narayan Shivakumar, Hector Garcia-Molina (1995) - SCAM: A Copy Detection Mechanism for Digital Documents
17. Pataki Máté (2011) – Fordítási plágiumok keresése

18. Plagiarism Checker: <http://plagiarisma.net/> [2012.10.27.]
19. Plagiarism Detector: <http://www.plagiarism-detector.com/> [2012.10.17.]
20. Plagiarism, Wikipedia: <http://en.wikipedia.org/wiki/Plagiarism> [2012.09.27.]
21. Plagiarism: <http://www.plagiarism.org/> [2012.09.27.]
22. Plágium, Wikipédia: <http://hu.wikipedia.org/wiki/Pl%C3%A1gium> [2012.09.27.]
23. Plágiumgyanú Semjén Zsolt doktorija körül, György Miklós (2012):
http://hvg.hu/itthon/20121118_plagiumgyanu_Semjen [2012.10.20.]
24. Rebecca Moore Howard (2007) – Understanding „Internet plagiarism”
25. Roman Lukashenko, Vita Graudina, Janis Grundspenkis (2007) – Computer-Based Plagiarism Detection Methods and Tools: An Overview
26. Salha M. Alzahrani, Naomie Salim, Ajith Abraham (2012) – Understanding Plagiarism Patterns, Textual Features and Detection Methods
27. Sangeetha Jamal (2010) – Plagiarism Detection Techniques
28. Súlyos plágiumgyanú Schmitt Pál doktori értekezése körül, György Miklós (2012):
http://hvg.hu/itthon/20120111_Schmitt_doktori_disszertacio_plagium [2012.10.20.]
29. Újabb plágium ügy, HT (2012): <http://helyitema.hu/belfold/34366> [2012.10.20.]
30. Zdenek Ceska, Chris Fox (2009) – The Influence of Test Pre-processing on Plagiarism Detection
31. Zip tartalma: forráskód (plagium.cpp): www.eet.bme.hu/~czirkos/infocpp/plagium.zip [2012.12.8.]