Rajzpályázat 2002 – nyeremények többmilliós értékben!

CHIP

SZÁMÍTÁSTECHNIKA ÉS KOMMUNIKÁCIÓ

www.chipmagazin.hu

XIV. évfolyam, 9. szám • 2002. szeptember • 1196 Ft • Előfizetéssel 952 Ft

Hangos film

TESZTEK

- 8 DVD-meghajtó
- 15 profi és félprofi hangkártya
- Tizennyolc 17 colos monitor
- 9 i845E chipkészletű alaplap

Fénypontjaink

A hónap nethelye:
uno.hu

DVD-je:
Vanília égbolt

játékai:
Neverwinter Nights •
Warcraft III

Adat-visszaállítás s. k.
GetDataBack és DiskExplorer

Ikszpés tippcsokor
A Windows XP hangolása

Forró vasak
512 bites Matrox Parhelia •
Rambus RIMM memóriás Asus alaplap

Zenés programok
Hangkártyák körítése

Új Windowsok jönnek...
Longhorn és Palladium

Éteri technológiák
IEEE 802.11, Bluetooth, MMS

A CHIP Magazinnak ebben a számában két CD található.
Ha hiányoznának belőle, kérje az eladótól!

A diákok réme

A számítástechnika fejlődésével az írott művek előállítási folyamata egyszerűbbé, publikációjuk pedig gyorsabbá vált. Méltán hasonlítják az internet hatását a könyvnyomtatáséhoz, hiszen az interneten bármilyen mű könnyen, gyorsan és olcsón közkinccsé tehető.

Pataki Máté

A neten a szellemi művek olyan tárháza jött létre, amely egyetlen eddig létező könyvtárhoz sem hasonlítható, sem méretében, sem elérhetőségében, sem használatának módjában. A digitális adattárolás mindemellett a végletekig egyszerűsíti a művek egészének másolását vagy egyes részeinek átvételét, így megkönnyíti a plágiumok létrehozását is. Ez a legfőbb oka annak, hogy egyre több másolatkereső rendszert fejlesztenek ki napjainkban. Az ilyen rendszereket azonban egy sor más feladathoz is lehet alkalmazni.

● Használható egyetemeken házifeladat-beadó rendszerként, amelyben regisztrálhatók a korábbi évek munkái. A beérkező új dolgozatokat azonnal másolatkeresési szűrésnek vethetjük alá.

● Tartalom-ellenőrzésre is alkalmas olyan adatbázisban, ahová szerzői jog védelme alá eső dokumentumokat fel lehet vetetni.

A program ezek után például a neten lévő digitális könyvtárakat figyeli, és figyelmeztet az illegális dokumentumokra.

● Netes keresőrendszer motorjában egy adott dokumentumhoz hasonlóakat lehet vele keresni. Esetleg megadja az adott dokumentumban lévő idézetek forrását.

● Számtalan irodalmi felhasználása is elképzelhető, például fordítások összehasonlítása, különböző források vizsgálata stb.

● Kereszthivatkozásokat automatikusan létrehozó szoftver is fejleszthető a segítségével: adott bekezdéshez automatikusan hozzákapcsol másik dokumentumban lévő hasonló vagy vele azonos bekezdést.

● Annak megállapítására is alkalmas lehet, hogy adott dokumentumot milyen nyelven vagy nyelveken írtak. Egy adatbázist feltöltünk különböző nyelvű dokumentumokkal (minden nyelvhez egy szószedet), és a kimenet megadja, hogy az adott dokumentum (esetleg dokumentumrész) melyik nyelvű mintafile-hoz milyen mértékben hasonlít.

A másolatkereső rendszerek alapvetően két csoportra oszthatók. Az egyik típus két dokumentum szövegét hasonlítja össze szóról szóra. Ez nagyon pontos, de lassú algoritmus, ezért nagyobb mennyiségű dokumentum esetében nem alkalmazható, hiszen túl időigényes lenne ezek páronkénti összehasonlítása.

A másik megoldás, amellyel az alábbiakban részletesen foglalkozunk, a szövegdaraboláson alapuló összehasonlítás. Itt sokkal gyorsabb az összehasonlítás, a tárigénye is kisebb – bár ezért pontatlanabb hasonlóság-kimutatással fizetünk. Nézzük, mi történik általában egy ilyen elven működő másolatkereső rendszerben!

A legelső lépés a dokumentumok beszerzése. Itt a formázási paraméterekre (betűtípus és -méret, sorköz, új oldal stb.) nincs szükségünk, ez nem érdekes a dokumentumok szövege közötti hasonlóság szempontjából. Ezért a legegyszerűbb, ha sima szövegfile-t (txt) használunk. Minden olyan dokumentum, amelyik nem ilyen formában

A másolatkereső rendszerek általános felépítése

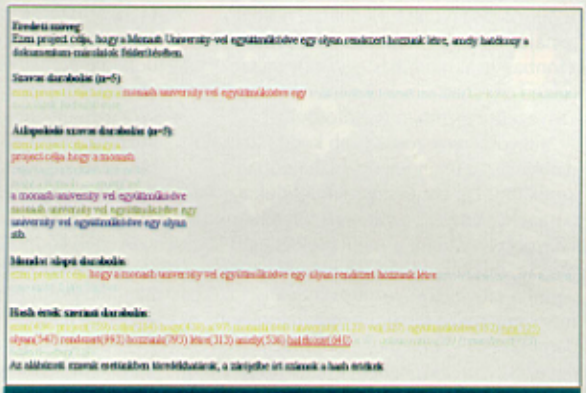
található, egy ezt megelőző lépésben konvertálásra kerülhet.

A sima szövegfile-okat ezután fel kell darabolni kisebb részekre, úgynevezett töredékekre. Ezek hossza pár szótól pár tíz szóig változhat. Hogy ezt a darabolást mi szerint végezzük, arra majd később látunk példát.

Az ezt követő lépésben a töredékeket el kell tárolni egy adatbázisban. Mivel szöveges formában sok helyet foglalnának el, ezért nem az eredeti töredékek kerülnek az adatbázisba, hanem ujjlenyomatuk, amelyet tömörítő eljárással kapunk az eredeti töredékből. Az adatbázis feltöltése az ujjlenyomatokkal tetszőleges számú lépésben történhet, ehhez minden új dokumentumot fel kell darabolni, majd a töredékek ujjlenyomatát el kell tárolni.

Ha később kíváncsiak vagyunk arra, hogy két dokumentum között van-e egyezés, csak le kell kérdeznünk az adatbázisból, hány közös töredéke van két dokumentumnak. A lekérdezést akármikor elvégezhetjük, akár minden újonnan beérkezett dokumentum eltárolása után is.

Ha rendelkezésünkre állnak az eredeti dokumentumok, az összevetést megkönnyítendő a hasonlónak ítélt file-okat egymás mellett megjelenítve szemléletessé tehetjük az eredményeket. Netes keresőrendszerben azonban nem kell feltétlenül megtartanunk az eredeti dokumentumokat, hiszen elég, ha az URL-t tároljuk, és egy linket teszünk oda a felhasználónak.



Darabolási módszerek. Azonos színnel jelöltük az azonos töredékekbe kerülő szavakat

A rendszer lelke

Most már el tudjuk helyezni a daraboló eljárást a teljes folyamatban. Nézzük meg részletesebben, milyen elvek alapján darabolhatjuk fel dokumentumainkat?

A könnyebb érthetőség kedvéért mindegyik darabolási eljárásra adunk egy-egy példát.

● *Mondatonkénti darabolás.* Kézenfekvőnek tűnhet, hogy a szövegben lévő mondatok alkossák a töredékeinket. Csakhogy ez a módszer felvet néhány problémát, ugyanis sokszor nem állapíthatók meg egyértelműen a mondathatárok. Azok a mondatok például, amelyek rövidítést tartalmaznak, több töredékre hullnak szét. Például Franz Kafka A per című művéből való mondat, „Valaki megrágalmazhatta Josef K.-t.", két töredékből áll. Ugyancsak két töredékre bomlik a következő mondat (ha a vesszőt nem tekintjük mondathatárnak): „Nem azért, hogy megtudjon valamit, hanem hogy elmozdítsa K.-t." Látható, hogy a „-t." töredék kétszer fog szerepelni az adatbázisunkban, amit rendszerünk egyezésként érzékel, holott a két mondat teljesen különböző.

Kísérletezhetünk azzal, hogy a vesszőt mondatvégnek számítjuk az egyik esetben, és nem vesszük figyelembe a másikban.

Ez ismét érdekes eredményhez vezet. Míg előbbi esetben átlagban 3,5 szó jut egy töredékbe, addig utóbbiban majdnem 12. Ezt az eredményt egy magyar nyelvű Szent Biblia alapján kaptuk, de más szövegek is nagyon hasonló eredményre vezettek.

Bár magyar nyelvű szövegeknél ez a különbség szembetűnőbb, mint az angol nyelvűeknél, ez a probléma ott is fellép.

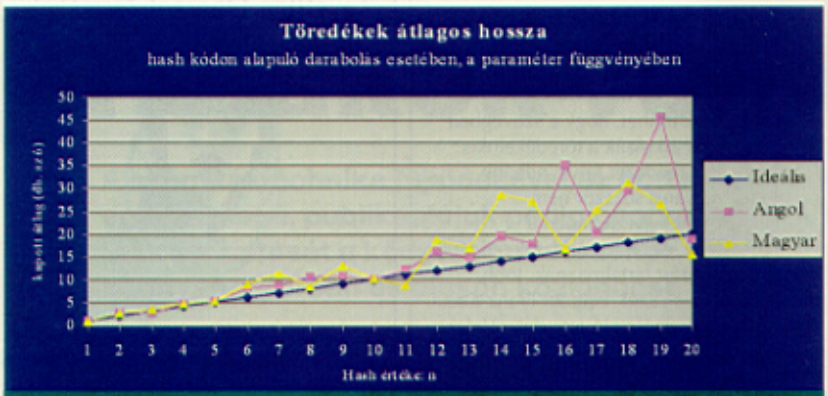
● *Szavas darabolás.* A szavas darabolás során *n* darab szó kerül egy töredékbe.

A szöveget úgy osztjuk fel, hogy *n* szavanként új töredéket kezdünk: az első töredék az első szótól az *n*-edikig tart, a második az *(n+1)*-ediktől a *2n*-edikig és így tovább. Ennek az algoritmusnak viszont van egy óriási hátránya.

Előfordulhat, hogy két szövegben van egyezés, de ezt nem fedi fel a módszerünk, mert az előtte lévő tartalom miatt nem ugyanott kezdjük darabolni a szöveget. Ez már akkor is problémát okoz, ha két

Másolatkereső rendszerek

- CopyCatch System: www.copycatch.freeserve.co.uk
- EVE Plagiarism Detection System: www.canexus.com
- Glatt Plagiarism Screening Program: www.plagiarism.com /screen.id.htm
- Plagiarism.org – the Internet plagiarism detection service for authors & education: www.plagiarism.org



Töredékek átlagos hossza hash kódon alapuló darabolásnál, a paraméter függvényében

teljesen azonos szövegnek két-, illetve háromszavas a címe. Ezt nevezzük fázisproblémának. A probléma kiszűrésének egyik módja az átlapolódó szavas darabolás.

● *Átlapolódó szavas darabolás.* Ez a darabolási eljárás hasonlít a szavas daraboláshoz, de itt minden szónál kezdődik egy új töredék, amelyek egyaránt *n* darab szóból állnak. Ezzel az eljárással ki tudjuk szűrni a szöveg esetleges eltolódását. Ebből természetesen az is következik, hogy minden szó *n* számú töredékben lesz benne, és hogy a szavas daraboláshoz képest *n*-szer annyi darab keletkezik.

Az átlapolódó szavas darabolás *n=1* esetében azonos a szavas darabolás *n=1* beállításával, és azt eredményezi, hogy minden szó külön töredéket alkot.

● *Hashkódon alapuló darabolás.* Ez az eljárás is paraméterezhető, paraméterét jelöljük *n*-nel. Egyszerű és gyors hash függvényt használ annak megállapítására, melyik szó legyen egy töredék határa. Ehhez minden szóra kiszámít egy számértéket, például úgy, hogy összeadja a szó betűinek ASCII kódját.

Ha ez a szám maradék nélkül osztható *n*-nel, ez a szó lesz a töredékhatár. Az eljárásból következik, hogy ha egy szó egyszer töredékhatár, akkor mindig az lesz. A következő töredék a töredékhatár után álló szóval kezdődik, és az első olyan szó zárja le, amelynek értéke maradék nélkül osztható *n*-nel – ez lehet akár egyben a töredék kezdőszava is.

A hash kódon alapuló darabolásra is elmondható, hogy *n=1* esetében azonos a szavas darabolás *n=1* beállításával, és azt eredményezi, hogy minden szó külön töredékbe kerül. Ez érthető, hiszen akármilyen értéket is kapunk egy szóra, eggyel biztos, hogy maradék nélkül osztható, azaz minden szó töredékhatár lesz.

A szavas darabolást a gyakorlatban nem használják. A másik három eljárás közül pedig a felhasználási területtől függően kell választani. Nézzünk néhány példát!

A mondatonkénti darabolás elsősorban rövid idézetek azonosságának kimutatására alkalmas. Nagyobb terjedelmű, hasonló témájú szövegek összehasonlításakor azonban gyakran hibás egyezést mutat, hiszen ugyanaz a mondat más és más szövegkörnyezetben is előfordulhat.

Nagyobb egyezést a hash kódon alapuló darabolással lehet legmegbízhatóbban kimutatni. Ha ennél magasra állítjuk a paraméter értékét, az átlagos töredékhossz nagyobb lesz, mint a mondatok átlagos hossza, miáltal jobban figyelembe lehet venni a szövegkörnyezetet. Ennek is megvan a maga hibája: a mondatonkénti darabolással ellentétben nem hibás egyezést mutat ki, hanem éppen hogy elsiklik a rövidebb egyezések felett.

Például ha véletlenül egy vagy két hosszabb töredék keletkezik, amelyek az egyező rész előtt és után lévő részbe is belelógnak, akkor nem fogja kimutatni az egyezést. Ez abból a tulajdonságából adódik, hogy ha sokáig nem jön olyan szó a szövegben, amely töredékhatár lehet, akár ötven-száz szavas töredékek is kialakulhatnak. Ennek valószínűsége különböző paramétereknél más és más, sőt a nyelvtől is erősen függ. Ennek alakulását láthatjuk cikkünk grafikonján, magyar és angol nyelvű szövegek esetében.

A vízszintes tengelyen a különböző hash értékeket, a függőlegesen a keletkezett töredékek átlagos hosszát ábrázoltuk.

Jól megfigyelhető, hogy például ha 16-ot adunk meg paraméternek, a töredékek átlagos hossza angol szövegeknél 35 szó lesz – tehát teljesen alkalmatlan rövidebb egyezések kimutatására.

Az átlapolódó szavas darabolás a legmegbízhatóbb módszer.

Paraméterétől függően több felhasználási területen is jól használható. 2-t vagy 3-at választva paraméternek (két- vagy háromszavas töredékek) akkor fog hasonlónak ítélni két szöveget, ha azokat azonos stílusban írták, hiszen a pár szavas kifejezéseket és szóösszetételeket fogja kimutatni. Ez akár a szövegek stílus vagy téma szerinti osztályozására is alkalmazható.

Közepesnek mondható paraméter (8–10) már tökéletesen kimutat pár mondatos egyezést, ráadásul a mondathatárokat se veszi figyelembe.

Van viszont egy nagy hátránya ennek a darabolási eljárásnak, mégpedig a keletkező töredékek mennyisége, amely nagyon megnöveli a keletkező adatbázist, ezáltal a benne való keresés idejét. Emiatt ezt az eljárást csak kisebb dokumentumgyűjteménynél lehet használni.

Házi feladatokat ellenőrző rendszerhez például tökéletesen alkalmas, de netes keresők motorjában már nem lehetne alkalmazni.

A töredékek tömörítése hash kódolással

Mint azt az előzőekben láttuk, a dokumentumkezelő és -feldolgozó rendszerek megvalósításakor fontos a szövegtárak méreteiből fakadó korlátok kezelése, mert a kisebb adatbázis kevesebb helyet foglal, és gyorsabb benne a keresés. Különösen igaz ez a netes alkalmazásokra. A hash algoritmus régi és jól bevált módja a szövegek kódolásának. Fő alkalmazási területe a kriptográfia, de szövegek általános célú tömörítésére is kitűnően alkalmazható.

Lényege, hogy egy akármekkora karakterláncból egy számot generál. Ugyanabból a szövegből mindig ugyanazt a számot fogja adni, különböző szövegekből pedig lehetőleg különbözőt.

Utóbbi feltétel legtöbbször teljesül, de néha mégis azonos számot kapunk különböző szövegekre. Szerencsére megfelelően nagy szám és megfelelő hash algoritmus alkalmazása esetén ritkák az ilyen szövegpárok, azaz az ebből adódó hiba elhanyagolható.

Érdekességként megjegyezzük, hogy a közismert MD5 hash algoritmus, amelyet egyes rendszereknél a jelszavak titkosítására használnak, alkalmas szövegek összehasonlítására is. Ráadásul ehhez nem kell az eredeti 128 bites kódot használni, bőven elegendő 32 bit, azaz 4 byte.

Ez átlagosan 80 karakteres töredékek esetén 5%-os tömörítési arányt jelent (4/80 byte = 5%).

Egyre több ilyen elven működő dokumentum-összehasonlító program jelenik meg a neten, sőt egyre több egyetem vizsgálja ezzel a módszerrel a diplomák és disszertációk eredetiségét. Egyes rendszerek a gyors keresést arra használják, hogy leszűkítsék a páronként összehasonlítandó dokumentumok számát, és a következő lépésben a lassabb, de biztosabb szavankénti összehasonlítást végzik el.

Valószínűleg nem kell sokat várnunk arra sem, hogy a magyar oktatási intézményekben is megjelenjenek hasonló rendszerek.