

TDK-dolgozat

Csikós Zoltán

Szekeres Péter

2012

Friss tartalmak gyors megtalálása a magyar weben tartalomkategorizáláson alapuló újralátogatási algoritmussal

Budapesti Corvinus Egyetem, Gazdálkodástudományi Kar, Számítástudományi Tanszék

Készítette:

Csikós Zoltán (BCE-GTK, Nemzetközi Gazdálkodás BSc, III. évf.)

Szekeres Péter (BCE-GTK, Gazdaságinformatika MSc, II. évf.)

Konzulens: Dr. Fodor Szabina (BCE Számítástudományi Tanszék, egyetemi docens)

Budapest, 2012.03.26.

Tartalomjegyzék

1	Vezetői összefoglaló	4
2	Bevezetés	6
3	Vélemény- és hírfigyelés a weben	8
4	A crawler működése	10
4.1	Kiválasztási elv	11
4.2	Újralátogatási elv	11
4.2.1	Weblapok kategorizálása	12
4.2.2	Újralátogatás kategória alapján	27
4.3	Udvariassági elv	29
4.4	Párhuzamos feldolgozási elv	30
5	Technológiai nehézségek és alkalmazott megoldásaik	31
6	A crawler hatékonyságának mérése	34
6.1	Mennyiségi teljesítménytesztek	34
6.2	Referenciatesztek	36
6.3	Az újralátogatási algoritmus tesztelése	37
7	Összefoglalás	39
8	Ábrajegyzék	40
9	Táblázatjegyzék	42
10	Felhasznált források	43

1 Vezetői összefoglaló

I. Téma ismertetése

A folyamatosan megjelenő új illetve frissülő tartalmak hatékony és gyors összegyűjtése inputként szolgál különböző üzleti és nem üzleti felhasználásokhoz és alkalmazásokhoz.

A webes információk automatikus összegyűjtésére jellemzően speciális webrobotokat, úgynevezett crawlereket alkalmaznak. A dolgozatban saját fejlesztésű crawlerünket mutatjuk be.

II. Célok meghatározása

Kettős célt fogalmaztunk meg a crawlerünk fejlesztése során:

1. A rendszer limitált kapacitású hardveren futtatva is hatékony teljesítményt érjen el a magyar weben.
2. A crawler számára olyan weboldal újralátogatási eljárást határozzunk meg, mellyel a lehető leghamarabb megtalálhatóak a magyar weben megjelent friss tartalmak, hozzászólások.

III. A megvalósítás bemutatása

A kutatás során a célunk egy olyan crawler elkészítése volt, mely a web magyar nyelvű tartalmait megkeresi és kategorizálja előre meghatározott csoportok szerint. Az így kialakított osztályok segítségével a crawler meghatározza, érdemes-e az oldalt újralátogatni, és ha igen, akkor milyen gyakran, hogy a frissített tartalmakat vagy bővülő hozzászólásokat minél hamarabb megtalálja a rendszer. A webes médiumtípusok automatikus meghatározása a rendszer által talált tartalmak későbbi felhasználását és feldolgozását is elősegíti.

IV. A crawler hatékonysága

Optimális párhuzamosítással indítva a crawlert egyetlen szerveren, naponta körülbelül 1 200 000 db weboldal feldolgozására képes, ha lineáris becslést alkalmazunk.

Ezek alapján a crawlerünk elméletileg képes arra, hogy a legelterjedtebb webes keresők által 1 nap alatt beindexelt magyar nyelvű webes tartalom mennyiséget megtalálja és feldolgozza 1 napnál jóval kevesebb idő alatt.

A weboldal kategorizáláson alapuló újralátogatási algoritmusunkat eredményességét összevetettük egy „hagyományos” újralátogatási algoritmussal, és arra jutottunk, hogy az

általunk definiált újralátogatási szabályrendszer hatékonyabb a „hagyományos” újralátogatási algoritmusoknál, ha a friss tartalmak megtalálása a cél.

2 Bevezetés

A weben számtalan információt elérhetünk a web 2.0 technológiai és internethasználati paradigmaváltásának köszönhetően. A webet elárasztották a felhasználók által generált tartalmak, melyek igen széles „műfaji” skálán mozognak: a fogyasztói véleménynyilvánításoktól kezdve a szakmai guruk blogjain át a mindennapi hírekig.

Ezen tartalmak figyelése és szöveges/statisztikai elemzése újfajta üzleti lehetőségeket rejtenek magunkban, például:

- A weben ingyen közvéleménykutatások, felmérések végezhetőek
- Egyszerűen értékelhető a termékújítások fogadtatása
- Figyelhető a versenytársak tevékenysége
- Figyelhetőek a versenytársakról alkotott vélemények

A folyamatosan megjelenő új illetve frissülő tartalmak hatékony és gyors összegyűjtése inputként szolgál különböző üzleti és nem üzleti felhasználásokhoz és alkalmazásokhoz.

A webes információk automatikus összegyűjtésére jellemzően speciális webrobotokat, úgynevezett crawlereket alkalmaznak. A crawler egy olyan szoftver, amely letölti a listájában lévő weboldalakokat, azok tartalmából kigyűjti a benne található URL címeket, átadja a tartalmat egy feldoglozó vagy tároló egységnek, majd folytatja a következő címen lévő oldal letöltését.

A kutatás során a célunk egy olyan crawler elkészítése volt, mely a web magyar nyelvű tartalmait megkeresi és kategorizálja előre meghatározott csoportok szerint. Az így kialakított osztályok segítségével a crawler meghatározza, érdemes-e az oldalt újralátogatni, és ha igen, akkor milyen gyakran, hogy a frissített tartalmakat vagy bővülő hozzászólásokat minél hamarabb megtalálja a rendszer. A webes médiumtípusok automatikus meghatározása a rendszer által talált tartalmak későbbi felhasználását és feldolgozását is elősegíti.

Az algoritmus az erőforrások megfelelő kihasználása végett többszálúsított és képes dinamikusan változtatni egy oldal következő letöltési idejét a futás közben szerzett tapasztalatai alapján. A megtalált oldalak linkjeit, illetve a rajtuk talált releváns tartalmat, további elemzésre alkalmas formában, egy MySQL adatbázisban tárolja a kereső.

A dolgozatnak nem célja a tartalmak hatékony tárolásának és indexelésének vizsgálata. Kettős célt fogalmaztunk meg a crawlerünk fejlesztése során:

3. A rendszer limitált kapacitású hardveren futtatva is hatékony teljesítményt érjen el a magyar weben.
4. A crawler számára olyan weboldal újralátogatási eljárást határozzunk meg, mellyel a lehető leghamarabb megtalálhatóak a magyar weben megjelent friss tartalmak, hozzászólások.

A dolgozatban előbb ismertetjük a crawler üzleti felhasználási lehetőségeit, majd részletesen bemutatjuk a crawler működését, illetve ezt követően a crawlerünk hatékonyságának tesztelését.

3 Vélemény- és hírfigyelés a weben

A márka, azaz a brand napjaink vállalatainak egyik legfontosabb értéke, a marketing és azon belül a PR, hírnév és brand management terület egyre nagyobb hangsúlyt kap. A hírnév (márka) erőssége nehezen meghatározható, nehezen követhető nyomon, és emiatt nehezen menedzselhető volt, azonban a webes tartalomgenerálás fejlődésével új lehetőségek nyíltak meg: megjelentek a webes véleményelemző és különböző brand és web monitoring eszközök, melyek feladata, hogy felfedezzék és értékeljék a különböző cégekről/brandekről/termékekről szóló szöveges tartalmakat a weben, majd jelentések, összesítések formájában segítsék a vállalati döntéshozatalt elsősorban kommunikációs, de sok már, például termékfejlesztési/HR/értékesítési területen.

A márkaépítés illetve márkafigyelés előtérbe kerülését jól mutatja még például a personal branding irányzatok terjedése, illetve a magyarországi SuperBrands 2011 és MagyarBrands 2011 díjazottak¹ széles köre.

A brand monitor termékek piacát áttekintve több angol nyelvre készített megoldással, illetve magyar nyelvre is készített megoldásokkal is találkozunk:

Angol nyelvű megoldások:

- Radian6 (vezető brand monitoring szolgáltatás angol nyelven)
- OpinionCrawl² (ingyenes online véleményelemző szolgáltatás angol nyelvre)
- Thismoment brand monitor³
- Position2 Brand Monitor
- CSC: Brand monitoring services⁴
- TIQSentiVal⁵

¹ Lásd bővebben: www.superbrands.hu

² <http://www.opinioncrawl.com>

³ <http://brandmonitor.thismoment.com>

⁴ <https://www.cscglobal.com/global/web/csc/brand-monitoring.html>

⁵ <http://www.tiq-solutions.de/display/loesungen/TIQSentiVal>

Magyar nyelvű megoldások:

- BuzzMirror (már nem működő szolgáltatás)
- NewZeus (már nem működő szolgáltatás)
- BrandMonitor⁶
- OpinHu⁷

A friss webes információk üzleti felhasználására demonstrálására jó példa az a kutatás, amely a Twitter bejegyzések elemzése alapján 86,7%-os eredménnyel meg tudta jósolni a Dow Jones Ipari Átlag változását. (Grossmann [2010])

Az elterjedt keresőmotorokra (Google, Bing, Yahoo) is igaz azonban, hogy a magyar nyelvű szövegeket esetén kívánni valót hagy az indexelésük és megtalálási sebességük. A dolgozatban kizárólag a magyar nyelvű weboldalakra koncentrálnak, a célunk a megjelenő új tartalmak, hozzászólások minél gyorsabb megtalálása.

⁶ <http://brandmonitor.hu>

⁷ <http://opinhu.com>

4 A crawler működése

Az általunk tervezett crawler felépítése nagyban hasonlít a legelterjedtebb webes keresőkre, mint például a Google saját keresője, vagy az Apache Nutch nevű freeware motorja. Az internetes keresőmotorokra jellemző tipikus crawler és webtérképkezelési funkciók (Tikk [2007]) maradéktalan ellátására törekedtünk a megfelelő sebesség mellett.

A mi crawlerünk alapját is egy megfelelően felépített adatbázis adja. Itt tároljuk a már megtalált linkeket egy táblában, mindegyiket pontosan egyszer, és a róluk szerzett lényeges információkat, mint például az utolsó letöltés idejét, az akkori karakterszámot, az általunk meghatározott következő látogatás idejét. Az első indulás előtt ezt manuálisan feltöltöttük egy válogatott URL halmazzal a legismertebb magyar hírportálok és blogterek alapján. (Ezt az induláskori URL listát hívjuk a crawler seed listájának.)

Miután a szoftver meglátogatott egy weboldalt, megkezdí annak a feldolgozását, egyrészt levágja a felesleges HTML kódrészleteket a méret csökkentése és a sebesség növelése érdekében, ezután kikeresi az összes linket az oldalról, és mindegyiket leellenőrizi, hogy egy másik oldalra mutató linket találtunk-e, nem pedig egy fájl, vagy egy email címet, amik számunkra nem relevánsak.

A fentiek mellett minden egyes új URL cím esetén lefut egy egyszerű úgynevezett **arató vagy csonkoló metódus** (Tikk [2007]), mely egy adott URL címet feldarabol a „/” jelek mentén visszafelé haladva, és az így képzett további URL címeket is beilleszti az adatbázisba.

Miután ez megtörtént, az eredetileg vizsgált weboldal értékeit frissítjük. Az algoritmus újraszámolja a következő látogatási időt, illetve ha az első esetben látogattuk meg az oldalt, egy teljesen általunk tervezett és kidolgozott osztályozó meghatározza, hogy milyen fajta weboldalról van szó. Ezt az osztályozó algoritmust a későbbiekben még részletesen bemutatjuk és értékeljük. (lásd: 4.2.1 fejezet)

Utolsó lépésként előkészítjük a letöltött oldalt az adatbázisba illesztéshez. Levágjuk a további felesleges HTML szerkezeti elemeket, egységesítjük a karakterkódolást és lecseréljük a speciális HTML karaktereket. Magának a crawlernek a fejlesztését is ez az igény motiválta, hogy a friss tartalmakat egy általunk meghatározott, jól strukturált formában és osztályozva tudjuk tárolni, előkészítve őket a további feldolgozásra.

A fejezet összefoglalásaként illetve a következő fejezetek bevezetéseként a webrobotok viselkedését befolyásoló tényezők (Tikk[2007]) az alábbi módon jelennek meg a crawlerben:

- **Kiválasztási elv** (*selection policy*): meghatározza, hogy mely weboldalt töltse le a crawler a következő iterációban.
- **Újralátogatási elv** (*re-visit policy*): meghatározza, hogy egy adott weboldalt érdemes-e újralátogatni, és ha igen, mikor.
- **Udvariassági elv** (*politeness policy*): a webszervereken okozható többletterhelést szabályozza.
- **Párhuzamos feldolgozási elv** (*parallelization policy*): a párhuzamosan dolgozó robotok összehangolása.

4.1 Kiválasztási elv

Minden weboldalhoz számol a program a letöltést és osztályozást követően egy optimális újralátogatási időt, majd ezzel az értékkel frissíti az adatbázist, így a kiválasztás során a lekérdezésben elég egyetlen paraméter szerint rendezni a sorokat, mégpedig az újralátogatás szerint növekvő sorrendben.

Emellett figyelembe veszi még a program, hogy az osztályozás során adott típusba sorolt oldalakat nem kell többet újralátogatni, mivel nem keletkezik rajtuk új tartalom vagy hozzászólás többet. Ennek tipikus példája az archívum, vagy a cikkek, melyekhez nem lehet hozzászólni. Ezeket az oldalakat a lekérdezés során nem vesszük figyelembe, így nem látogatja meg őket feleslegesen a crawler újra.

4.2 Újralátogatási elv

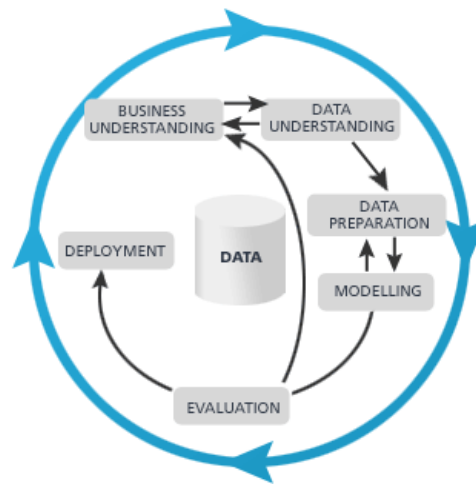
A weboldalak jelentős része folyamatosan frissül/módosul: bővül a cikkek vagy blogbejegyzések száma, újabb fórumtopikok nyílnak, stb. **A crawler feladata, hogy miután egy weboldal tartalmát illetve a rajta lévő linkeket letöltötte, meghatározza, hogy mikor kell legközelebb letöltenie az oldalt, hogy a frissült tartalmakat feldolgozza.**

Jelen crawler újralátogatási algoritmusa azon alapszik, hogy a weben található különböző oldalak besorolhatóak jól definiálható médiumtípusokba, mely típusok illetve adott típusokon lévő tartalmak pedig meghatározzák a következő újralátogatás idejét. Az újralátogatás például egyáltalán nem szükséges bizonyos cikkeknel, melyekhez nem lehet hozzászólni, ugyanakkor kifejezetten fontos a sűrű látogatás hírportálok főoldalainál, mert ezek tartalma gyakran frissül.

A következő alfejezet ismerteti a weboldal kategorizáló adatbányászati modell kialakítását, majd bemutatjuk, hogy az egyes kategóriák esetén milyen újralátogatási algoritmust definiáltunk.

4.2.1 Weblapok kategorizálása

A weboldal klasszifikációs modell kialakítása a CRISP-DM adatbányászati módszertan⁸ alapján történt RapidMiner⁹ v5.2 ingyenes eszközzel. A CRISP-DM módszertan alapja a többszörös iteráció illetve visszalépés (lásd: 1. ábra), a továbbiakban azonban nem részletezzük az iterációk és visszalépések sorát, ehelyett összefoglalóan mutatjuk be a modellépítés folyamatát a CRISP-DM módszertan alaplépései mentén.



1. ábra: A CRISP-DM módszertan lépései

(forrás: <http://www.finisco.com/crisp-data-mining-methodology.aspx>)

4.2.1.1 A szakterület/üzleti környezet megismerése

A klasszifikációs modell építése során először manuálisan megvizsgáltunk több magyar weboldalt. Ennek során hipotéziseket állítottunk fel, illetve szabályszerűségeket kutattunk. Az üzleti hírfigyelési, illetve véleményelemzési szempontok alapján elemzésre alkalmas

⁸ Cross Industry Standard Process for Data Mining: „Több nemzetközi szervezet együttműködésével kidolgozott módszertan kifejezetten adatbányászati feladatok megoldására.” (Fajsi [2010], 396. o.)

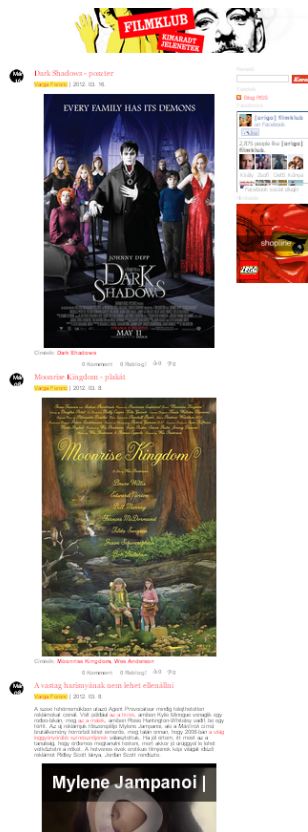
⁹ A RapidMiner v5.2 adatbányászati szoftver a <http://rapid-i.com/content/view/26/84/lang,en/> weboldaltól ingyenesen letölthető

webes szöveges médiumtípusokat (címlap, cikk, blog, fórum) definiáltunk,¹⁰ majd ezeket kiegészítettük olyan weboldaltípusokkal, melyek segítségével a crawler teljesítménye jelentősen növelhető, azáltal, hogy bizonyos típusokat, például ritkán vagy egyáltalán nem kell újralátogatni, mivel nem változik a tartalmuk.

A fentiek alapján végül 13 weblap kategóriát definiáltunk:

- **Címlap:** olyan weboldal, mely jellemzően hírportálok főoldala, hírportálok rovatainak főoldala vagy blogok főoldala. Internethasználat szempontjából úgyis meghatározható, hogy ezen oldalak a domén további tartalmainak rövid, néhánymondatos összefoglalását mutatják jellemzően képpel illusztrálva. Ezen oldalak alapján döntenek el az olvasók, hogy elolvassák-e a teljes cikket/bejegyzést, illetve ezeken az oldalakon megtalálhatóak a legfrissebb tartalmak, tehát gyakran frissülnek.

¹⁰ Üzleti hírfigyelés illetve véleményelemzés, brand monitoring szempontból olyan médiumtípusokra koncentráltunk, melyek megkülönböztetése plusz információt jelent az üzleti felhasználás során: például egy termékről szóló webes szövegek összefoglalásán túl üzletileg releváns, hogy milyen fő gondolatok jelentek meg a termékről kizárólag webes címlapokon.



2. ábra: Címlap weboldaltípus példák (www.filmklub.postr.hu és www.index.hu)

- **Linkgyűjtemény:** olyan weboldal, melyek rendkívül sok linket tartalmaznak, és ezek elsősorban kifelé, azaz más domáinekre mutatnak. Ezen oldalak olyan gráfcsúcsoknak tekinthetők a weben, mint linkek által összekötött gráfhálózaton, melyek mentén jelentősen bővíthető a crawler által bejárt oldalak köre.
- **Cikk kommentekkel:** klasszikus webes hírek és cikkek, jellemzően hírportálokon, melyekhez megjegyzéseket lehet írni.
- **Cikk kommentek nélkül:** klasszikus webes hírek és cikkek, jellemzően hírportálokon, melyekhez nem lehet megjegyzést írni.
- **Blog kommentekkel:** blogbejegyzések, melyekhez megjegyzéseket lehet írni.
- **Blog kommentek nélkül:** blogbejegyzések, melyekhez nem lehet megjegyzéseket írni.

- **Fórumfőoldal:** fórum topikokat és altopikokat felsoroló weboldal. Újralátogatási logika szempontjából a címlap kategóriába tartoznának, azonban jelentős strukturális különbségek miatt külön kategóriába tettük ezeket az oldalakat.
- **Fórum:** fórum oldal.
- **Archív:** olyan weboldal, mely egy portál vagy blog archív bejegyzéseinek linkjeit tartalmazza. A weben rendkívül sok archív oldal található. Fontos szerepe van a kategóriának crawler teljesítményében, mivel az archív oldalakat elég csupán egyszer elemeznie a crawlernek, hogy megtalálja a régebbi cikkeket és bejegyzéseket, azonban ezt követően nem kell újralátogatnia.
- **Aktívarchív:** a fenti állítás, miszerint nem kell újralátogatni az archív oldalakat, nem minden esetben igaz. Azon archív oldalak tartalma, melyek aktuális napra, hónapra vagy évre vonatkoznak, megváltozhatnak, bővíülhet cikkhivatkozásaik száma. Azonban a hónapra vagy évre vonatkozó archív weboldal az általa felölelt dátumintervallumba eső napi archív oldalak cikkhivatkozásainak összességét tartalmazza. Erre a célra hoztuk létre az aktívarchív kategóriát: a crawler, ha egy aktuális napi archív oldalt talál, akkor aktívarchívként kategorizálja, és 24 óra múlva meglátogatja újra, ekkor pedig már archív-ra állítja a kategóriáját.
- **RRS:** RSS xml oldalak.
- **Atom:** Atom xml oldalak.
- **Nem definiált (undefined):** a fenti kategóriákba nem besorolható minden további weboldal.

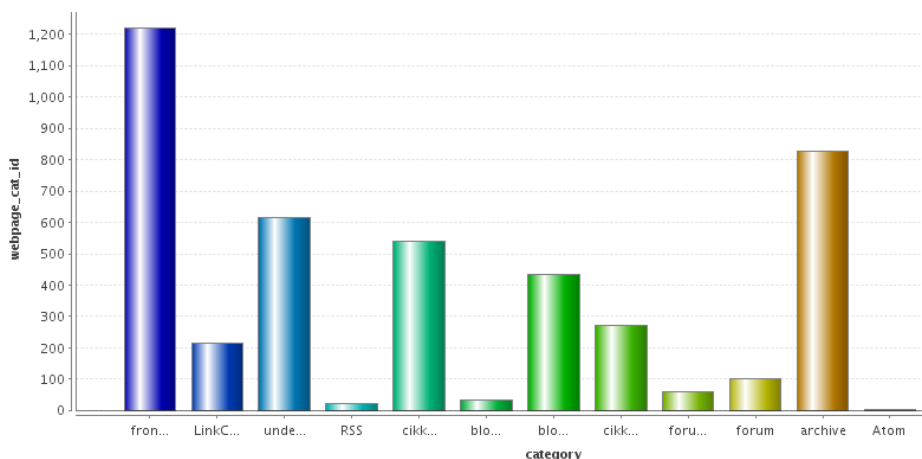
4.2.1.2 Az adatok megismerése

Az adatgyűjtés során 64 szemantikai és strukturális jellemzőt leíró attribútumot definiáltunk egy weboldal leírására. Ezeket a következő főbb csoportokba sorolhatjuk:

- **URL-en alapuló attribútumok** (például: URL hossza, URL mélysége, URL tartalmazz-e blog szót)
- **HTML head tag-en alapuló attribútumok** (például: HTML description-ben szerepel-e a forum vagy fórum szó, milyen hosszú az oldal title szövege)
- **HTML tag darabszámok** (például: hány db <h1>, <h2>, , stb. tag található az oldal HTML kódjában)

- **Szóelőfordulások** (például: a forum vagy fórum szó hányszor szerepel a teljes HTML kódban)
- **Szövegarány mutatók** (például: a <p> tagek szövege hogyan aránylik a teljes weboldal szövegéhez, a heading tagek szövege hogyan aránylik a teljes weboldal szövegéhez)
- **Link struktúrán alapuló mutatók** (például: linkek száma, a doménre – tehát befelé – mutató hivatkozások aránya az összes hivatkozáshoz viszonyítva)
- **Komment mutatók** (például: van-e komment vagy comment szót tartalmazó class-szal rendelkező HTML tagje az oldalnak)
- **Dátum alapú mutatók** (például van-e az URL-ben dátum, van-e az URL-ben évszám, van-e a weboldal szövegének első harmadában dátum)

Ezt követően 4 348 weboldalt manuálisan besoroltunk a korábban definiált 13 kategória egyikébe (lásd: 3. ábra).

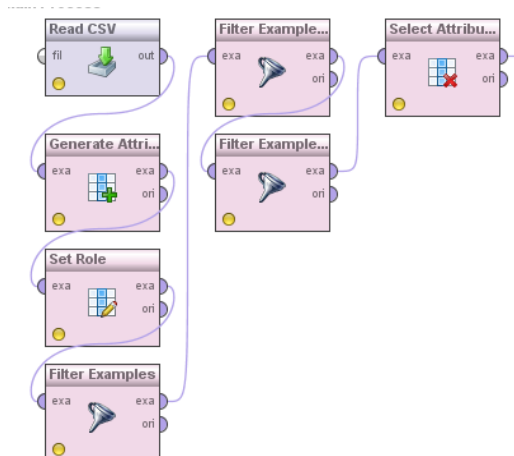


3. ábra: A weboldal kategóriák megoszlása

A $64 \times 4\,348 = 278\,272$ attribútumérték meghatározása már automatikusan, egy erre a célra Java nyelven írt robottal történt, ami egyesével letöltötte a weboldalakat, és kiszámolta az attribútumértékeket, majd elmentette azokat egy MySQL adatbázistáblába.

4.2.1.3 Az adatok előkészítése

Az összegyűjtött adatok előkészítése már RapidMiner eszközzel történt. Az adatokat CSV fájlon keresztül importáltuk.



4. ábra: RapidMiner adatelőkészítési lépések

Az adatok előkészítése során kiszűrtük az archive, activearchive, RSS és Atom típusú oldalakat, mivel ezekre a típusokra egyértelmű URL alapú szabályokat tudtunk pusztán szakmai szemmel definiálni (közel 100%-os pontossággal), így ezeket nem volt szükség tovább elemezni adatbányászati modellekkel.

A vizsgált weboldalak közül előfordult néhány, melyet az adatgyűjtő robotunk nem tudott letölteni és elemezni (például mert az oldal időközben megszűnt, vagy kapcsolódási idő túllépés történt). Ezeknek az attribútum értékei között NULL értékek szerepeltek, melyek félrevitték volna a tanuló algoritmusokat, ezért eltávolítottuk ezen rekordokat az adathalmazból. Így 3 437 rekordunk maradt az adathalmazban a modellépítéshez.

Már az adatok előkészítése során, illetve a későbbi iterációk során a modellezéshez használt attribútumok számát is lecsökkentettük 35-re, mivel a maradék 29 sokszor olyan megoldásokat (elsősorban szabályokat) eredményezett, amelyeket szakmai elemzés után a minta devianciájának vagy pedig szakmailag rossz szabálynak ítéltünk. Az attribútumok eltávolításával pontosabb modelleket tudtunk végül előállítani, melyeket emberi szakértői szemmel is igazolni tudtunk.

4.2.1.4 Modellezés és kiértékelés

Már a weboldal kategóriák létrehozása során figyelembe vettük, hogy az egyes kategóriák jól elkülöníthetők legyenek egymástól. A modell építéskor kiemelt cél volt még, hogy a kapott eredmények általunk is értelmezhetőek legyenek. Szakértői szemmel meghatározott attribútumokon kerestünk olyan szabályokat, melyeket igazolni is tudunk, de legalábbis nem tudunk elvetni logikai úton.

Így próbáltuk megakadályozni, hogy a minta esetleges devianciája elrontsa a modell későbbi használhatóságát, hiszen mindössze 3437 weboldal alapján építettünk egy a teljes magyar webre alkalmazandó modellt. **Annak érdekében, hogy a modell minél univerzálisabb legyen kiegyensúlyozott mintavételezéssel 50%-50%-os arányban osztottuk fel az adathalmazt tanító- és tesztalmazra.**

A kategorizálási feladatot tehát felosztottuk bináris osztályozási részfeladatokra, így az adott kategóriára kapott modellt szakmai tapasztalatok és szempontok alapján manuálisan módosíthattuk (például további változókat vettünk ki a modellből), melyre több ízben is szükség volt. A modell értelmezhetősége miatt döntési fákat építettünk, majd a generált döntési fát szabályokra alakítottuk, hogy könnyebben leprogramozható legyen.

Minden egyes bináris osztályozási részfeladat során először **összehasonlítottuk az alábbi 6 osztályozási módszer¹¹ ROC görbáját,¹² hogy meghatározzuk ezek közül a legjobb modellt:**

1. Naiv Bayes algoritmus
2. C4.5 döntési fa¹³ information gain kritériummal
3. C4.5 döntési fa accuracy kritériummal
4. C4.5 döntési fa Gini index¹⁴ kritériummal
5. C4.5 döntési fa nyereség arány (Gain ratio)¹⁵ kritériummal

¹¹ Mivel az ID3 és CHAID döntési fák nem alkalmazhatók numerikus attribútumokra, a RapidMiner C4.5 és CART döntési fákhöz hasonló tanuló node-ját alkalmaztuk, mely döntési fához 4-féle vágási kritérium kadható meg. Mind a 4 esetre megvizsgáltuk az előállított modellt.

¹² ROC görbe: A Receiver Operating Characteristic (ROC) analízist a statisztikai döntésemélet számára fejlesztették ki. Az egyes osztályozási modellek teljesítménye összehasonlítható a ROC görbájük alatti területek alapján. (Fajsi [2010])

¹³ A C4.5 algoritmusú döntési fa igen elterjedt, üzleti osztályozási modellek építésére is használt döntési fa algoritmus

¹⁴ Gini index: a csomópontokban mindig olyan attribútumok kerülnek a fában, amelyekkel a leghomogénebb osztályok állíthatók elő. (Bodon [2010])

¹⁵ Nyereségarány mutató (gain ratio): *„figyelembe veszi a gyerek csomópontokba kerülő tanítóadatok számát, és „bünteti” azokat az attribútumokat, amelyek túl sok gyereket hoznak létre.”* (Bodon [2010] 144. o.,)

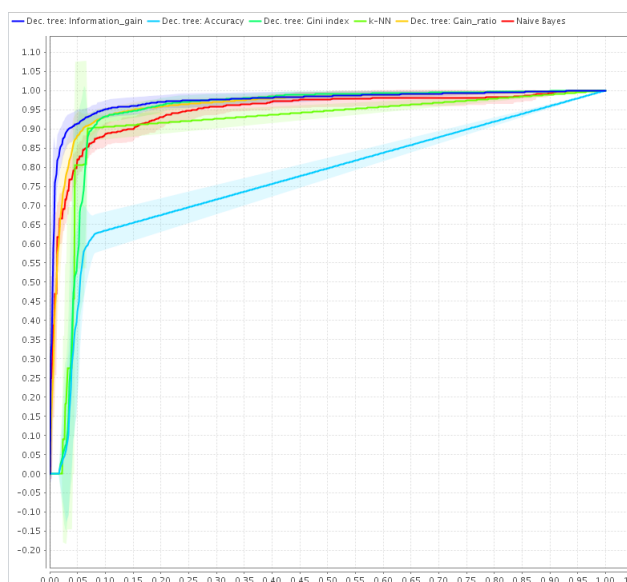
6. K legközelebbi szomszéd algoritmus

A Naiv Bayes és a K legközelebbi szomszéd (K-NN) algoritmusokat csak viszonyítási alapként használtuk, **az implementálandó algoritmus a döntési fa módszerek közül került ki az emberi értelmezhetőség és implementálási egyszerűség miatt.**

SVM¹⁶ és neurálisháló modelleket nem alkalmaztunk még referenciaként sem a modellépítés során, mert a változóink jelentős része bináris változó, és ezek a módszerek nem alkalmasak bináris változók kezelésére.

Mivel az osztályozási feladatot bináris osztályozási részfeladatokra bontottuk, ezért a CRISP-DM (Cross Industry Standard Process for Data-Mining) adatbányászati módszertan modellezés és kiértékelés lépéseit minden egyes részfeladatra külön-külön elvégeztük.

Címlap osztályozás



5. ábra: Címlap osztályozási algoritmusok teljesítményének összehasonlítása

A különböző módszerek közül az information gain módszeren alapuló döntési fa (sötétkék grafikon) bizonyult a legjobbnak a ROC grafikonok összevetése alapján. (Jelentősen azonban csak az accuracy alapú döntési fa tér el a többi módszertől.)

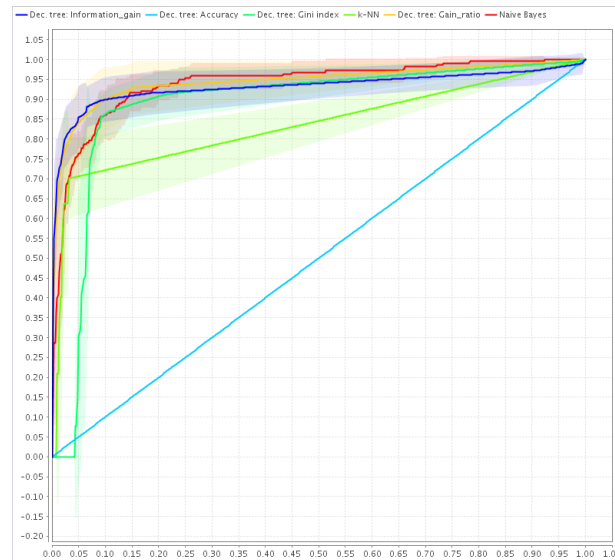
A címlap osztályozás eredményeként előálló fa pontosságát a következő ábra mutatja:

¹⁶ SVM vagy Support Vector Machine: olyan felügyelt tanulási módszer, mely szeparáló hipersíkok segítségével választja el egymástól az osztályokat.

accuracy: 92.84%			
	true false	true true	class precision
pred. false	1071	77	93.29%
pred. true	46	524	91.93%
class recall	95.88%	87.19%	

6. ábra: A címlap osztályozás pontossága

Linkgyűjtemény osztályozás



7. ábra: Linkgyűjtemény osztályozási algoritmusok teljesítményének összehasonlítása

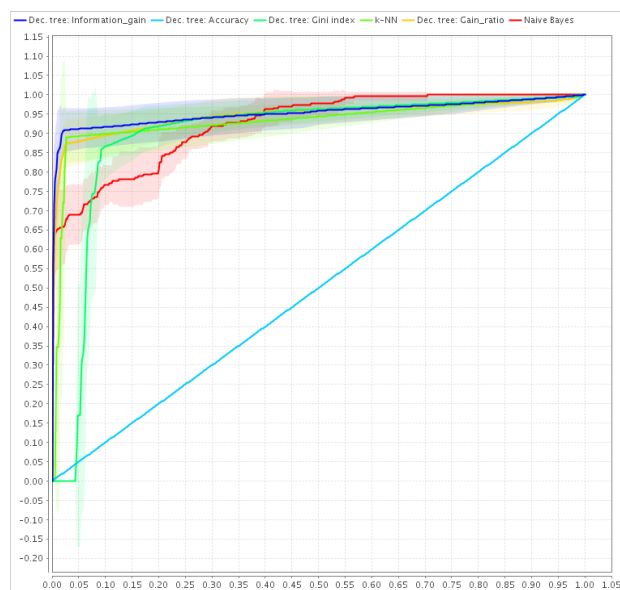
A Linkgyűjtemény kategória esetében az information gain, gain ratio alapú döntési fa és a Naiv Bayes módszer közel azonosan jó eredményeket produkált.

A gain ratio módszerrel készült fa pontosságát a következő ábra mutatja:

accuracy: 96.51%			
	true false	true true	class precision
pred. false	1578	27	98.32%
pred. true	33	80	70.80%
class recall	97.95%	74.77%	

8. ábra: A linkgyűjtemény osztályozás pontossága

Cikk kommentekkel osztályozás



9. ábra: A cikk kommentekkel osztályozási algoritmusok teljesítményének összehasonlítása

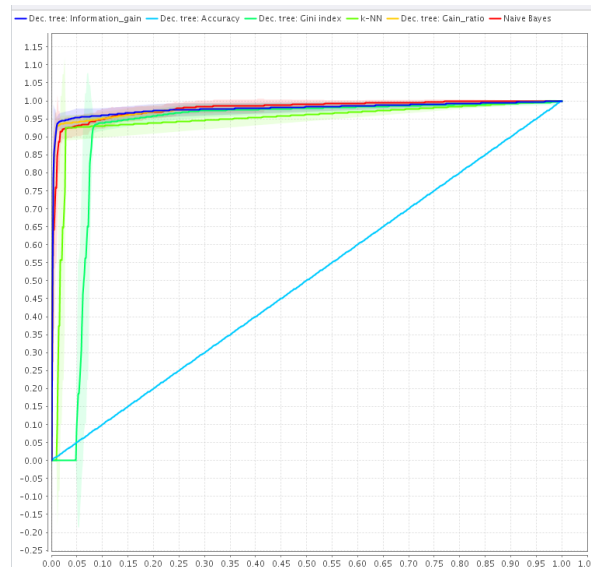
A különböző módszerek közül az information gain kritériumon alapuló döntési fa (sötétkék grafikon) bizonyult a legjobbnak.

Az előállt döntési fa pontosságát a következő ábra mutatja:

accuracy: 97.85%			
	true false	true true	class precision
pred. false	1567	16	98.99%
pred. true	21	114	84.44%
class recall	98.68%	87.69%	

10. ábra: A cikk kommentekkel osztályozás pontossága

Cikk kommentek nélkül osztályozás



11. ábra: A cikk kommentek nélkül osztályozási algoritmusok teljesítményének összehasonlítása

A különböző módszerek közül az information gain módszeren alapuló döntési fa (sötétkék grafikon) bizonyult a legjobbnak.

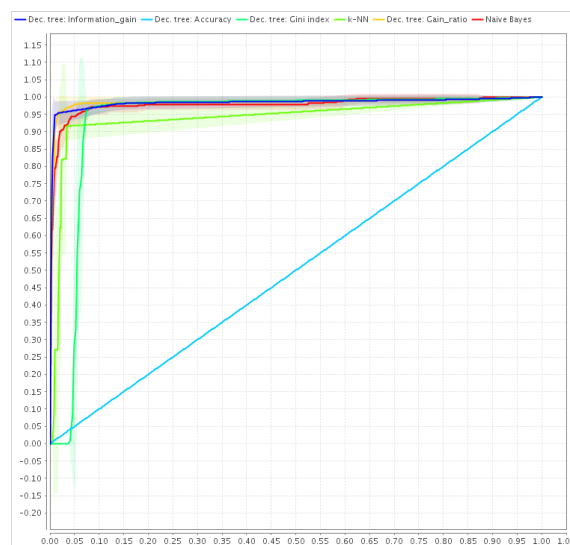
Az előállt döntési fa pontosságát a következő ábra mutatja:

accuracy: 98.08%			
	true false	true true	class precision
pred. false	1439	23	98.43%
pred. true	10	246	96.09%
class recall	99.31%	91.45%	

12. ábra: A cikk kommentek nélkül osztályozás pontossága

Bár elsőre alkalmazhatónak tűnne, a cikk kommentekkel kategória szabályainak átalakítását nem alkalmaztuk, mivel jelentősen eltérő karakterisztikákkal bírnak a komment nélküli cikkek, mint a kommentekkel ellátottak.

Blog kommentekkel osztályozás



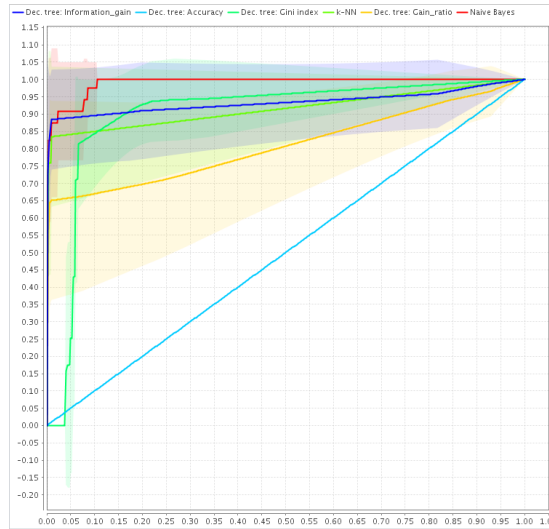
13. ábra: A blog kommentekkel osztályozási algoritmusok teljesítményének összehasonlítása

A blog kommentekkel kategória esetén az information gain kritérium bizonyult a legjobbnak. Az előállt döntési fa pontosságát az alábbi ábra mutatja:

accuracy: 98.37%			
	true false	true true	class precision
pred. false	1494	20	98.68%
pred. true	8	196	96.08%
class recall	99.47%	90.74%	

14. ábra: A blog kommentekkel osztályozás pontossága

Blog kommentek nélkül osztályozás



15. ábra A blog kommentek nélkül osztályozási algoritmusok teljesítményének összehasonlítása

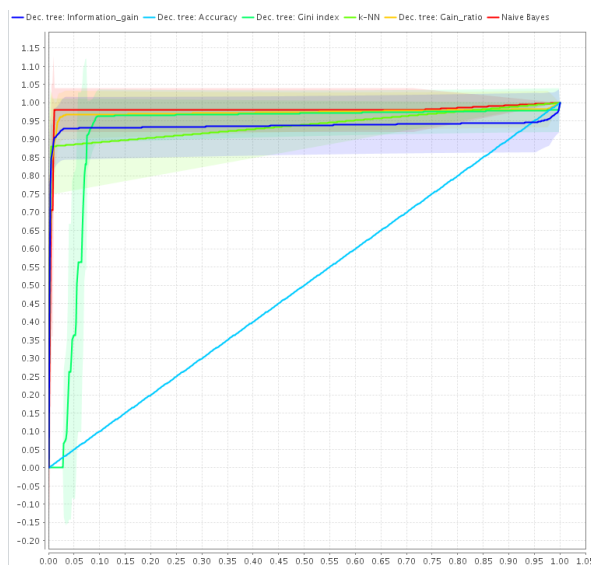
A blog kommentek nélkül kategória esetén az Naiv Bayes kritérium bizonyult a legjobbnak, az information gain kritériumot használó döntési fa volt a legjobb, amely ugyan elmarad a Naiv Bayes módszer teljesítményétől, de azt interpretálhatóság miatt ezt alkalmaztuk az implementáció során. Az előállt döntési fa pontosságát az alábbi ábra mutatja:

accuracy: 99.59%			
	true false	true true	class precision
pred. false	1695	1	99.94%
pred. true	6	16	72.73%
class recall	99.65%	94.12%	

16. ábra: A blog kommentek nélkül osztályozás pontossága

Bár elsőre alkalmazhatónak tűnne, a blog kommentekkel kategória szabályainak átalakítását nem alkalmaztuk, mivel jelentősen eltérő karakterisztikákkal bírnak a komment nélküli blogbejegyzések, mint a kommentekkel ellátottak.

Fórumfőoldal osztályozás



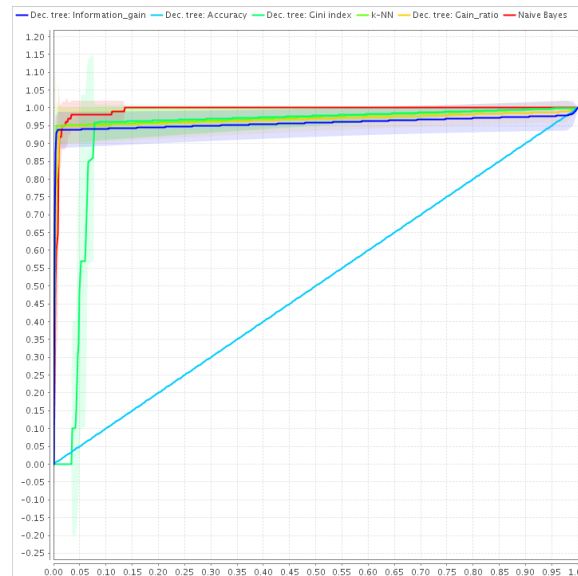
17. ábra: A fórumfőoldal osztályozási algoritmusok teljesítményének összehasonlítása

A fórumfőoldal kategória esetében az information gain, a Gain ratio alapú döntési fa és a Naiv Bayes módszer közel azonosan jó eredményeket produkált. Az information gain kritériumú döntési fa pontosságát a következő ábra mutatja:

accuracy: 99.65%			
	true false	true true	class precision
pred. false	1687	5	99.70%
pred. true	1	25	96.15%
class recall	99.94%	83.33%	

18. ábra: A fórumfőoldal osztályozás pontossága

Fórum osztályozás



19. ábra: A fórum osztályozási algoritmusok teljesítményének összehasonlítása

A fórum kategória esetében a Gain ratio alapú döntési fa és a Naiv Bayes módszer közel azonosan jó eredményeket produkált. A gain ratio kritériumú döntési fa pontosságát a következő ábra mutatja:

accuracy: 99.24%			
	true false	true true	class precision
pred. false	1667	11	99.34%
pred. true	2	38	95.00%
class recall	99.88%	77.55%	

20. ábra: A fórum osztályozás pontossága

Az egyes osztályozások pontossága (accuracy¹⁷ értéke) mind 90% fölötti, valamint mindegyik pontosság az általunk meghatározott alapérték (baseline) felett van. Alapértéknek azt a pontosságot választottuk, ha minden esetben Hamis értékeket adott volna az osztályozó.

¹⁷ Accuracy: $(TP + TN) / N$, ahol TP a true positive osztályozások, TN a true negative osztályozások száma, N pedig a teljes elemszám.

Médiumtípus	Accuracy érték	Baseline érték	Baseline fölött?
Címlap	92,84%	62,34%	I
Linkgyűjtemény	96,51%	91,39%	I
Cikk kommentekkel	97,85%	91,21%	I
Cikk kommentek nélkül	98,08%	83,70%	I
Blog kommentekkel	98,37%	86,96%	I
Blog kommentek nélkül	99,59%	98,66%	I
Fórumfőoldal	99,65%	98,20%	I
Fórum	99,24%	97,03%	I

1. Táblázat: Az osztályozások pontosságainak összefoglaló táblázat

4.2.1.5 Alkalmazás

A generált döntési fák a RapidMiner Tree to Rule node-jának segítségével if-then szabályokká alakítottuk, és a true értéket adó szabályokat implementáltuk. **Adatbányászati szabálygenerálással 8 kategóriára összesen 33 szabályt készítettünk, melyek megoszlása a következő:**

- Címlap: 14 szabály
- Linkgyűjtemény: 2 szabály
- Blog kommentekkel: 4 szabály
- Blog kommentek nélkül: 1 szabály
- Cikk kommentekkel: 6 szabály
- Cikk kommentek nélkül: 2 szabály
- Fórumfőoldal: 2 szabály
- Fórum: 2 szabály

Másik 4 kategóriára (archív, aktívarchív, RSS és Atom) manuálisan közel 100% pontosságú URL alapú szabályokat alkalmaztunk. Az előbbi 12 kategória egyikébe sem kerülő oldalak automatikusan nem-meghatározott (undefined) címkét kaptak.

A 13 kategóriát összesen 37 szabállyal fedtük le az implementáció során. (Az előzőekben megszűrt attribútumlistából 31 attribútumot használtunk fel.)

4.2.2 Újralátogatás kategória alapján

Egy weboldal kategóriájának megállapítása után a crawler eldönti, hogy milyen függvény szerint számítsa ki a következő letöltési időpontot. Minden esetben egy

újrálátogatási időközt számol percben, melyet hozzáad az aktuális időhöz, és így határozza meg az újrálátogatás időpontját.

Az egyes kategóriák számára meg van határozva a minimális és a maximális újrálátogatási időköz. A fórum illetve a kommentezhető kategóriájú oldalak újrálátogatása elsősorban az oldalon található legfrissebb hozzászólások időpontjaitól függően alakul sávok szabályrend szerint feltételezve az egyes témák fokozatos avulását és relevanciacsökkenését.

A következő táblázat ismerteti a kategóriák újrálátogatási függvényeit.

Kategória	Min. látogatási gyakoriság	Max. látogatási gyakoriság	Látogatási gyakoriság számítása
1. Címlap 2. Fórumfőoldal	5 perc	720 perc (12 óra)	Ha a tartalom változott, akkor felére csökken az újrálátogatási időköz, egyébként dinamikusan növekszik az újrálátogatási időköz.
3. Fórum 4. Cikk kommentekkel 5. Blog kommentekkel	5 perc	129 600 perc (90 nap)	Meghatározza a crawler a legfrissebb dátum aktuális korát percben (x) az oldal szövegében (ami valószínűleg a legfrissebb hozzászólás keletkezési időpontja lesz), ez alapján sávok szerint alakul az újrálátogatási időköz: ▪ Ha $x \leq 1\,440$ perc, akkor az időköz x-szel egyenlő ▪ Ha $x \geq 1\,440$ perc (1 nap) és $x < 43\,200$ perc (30 nap), akkor 1 440 perc az időköz ▪ Ha $x \geq 43\,200$ perc (30 nap) és $x < 86\,400$ perc (60 nap), akkor 10 080 perc (kb. 1 hét) az időköz ▪ Ha $x \geq 86\,400$ perc (60 nap) és $x < 259\,200$ perc (180 nap), akkor 43 200 perc (30 nap) az időköz

Kategória	Min. látogatási gyakoriság	Max. látogatási gyakoriság	Látogatási gyakoriság számítása
			▪ Egyébként 129 600 perc (90 nap) az időköz
6. Linkgyűjtemény 7. Nem meghatározott	5 perc	262 080 perc (180 nap)	Ha a tartalom változott, akkor felére csökken az újralátogatási időköz, egyébként dinamikusan növekszik az újralátogatási időköz.
8. Aktívarchív	1 440 perc (1 nap)	1 440 perc (1 nap)	Aktívarchív oldal esetében mindenképpen 24 óra az újralátogatási időköz.

2. Táblázat: Az újralátogatási algoritmus összefoglaló táblázata

A többi kategória (cikk kommentek nélkül, blog kommentek nélkül, RSS, Atom, archív) esetén **nem történik újralátogatás**, mivel ezek véleményelemzési illetve hírfigyelési szempontból releváns szöveges tartalma nem változik, legfeljebb a weboldalon található hirdetések vagy kapcsolódó cikkek listája változik.

A teljes újralátogatáshoz hozzátartozik, hogy bizonyos – főleg a crawler indításakor manuálisan betöltött URL listában (seed lista) megadott magas látogatottságú magyar - weboldalak esetén manuálisan megadtuk az újralátogatási időközt, ezeket nem változtatja a rendszer.

4.3 Udvariassági elv

A nagyobb weboldalakon már szinte mindig találunk a gyökérkönyvtárban egy robots.txt nevű egyszerű szöveges fájlt, ami a robotok számára nyújt eligazítást az oldal felépítéséről, a tiltott zónákról illetve a sitemap elhelyezkedéséről. Ennek figyelembevétele önkéntes, de az udvariassági előírások megkövetelik ezen szabályok használatát. A jelenlegi rendszerünk mellett, párhuzamosan fut egy folyamat, ami csak a robots.txt fájlokat keresi, és elemzi, majd az így kinyert adatokkal módosítja az adatbázist, lehetővé téve, hogy a keresőnk alkalmazkodjon ezekhez a szabályokhoz.

4.4 Párhuzamos feldolgozási elv

A crawler tervezése és fejlesztése során törekedtünk, hogy a program könnyen skálázható legyen, egyszerűen tudjuk finomhangolni a szálak számának változtatásával a teljesítményt a futtató gépen. Ez jelenleg manuálisan határozzuk meg, teljesítménytesztek segítségével, de a közeljövőben szeretnénk egy tanuló algoritmussal kiegészíteni a crawlert, ami segítségével automatikusan, a futási adatok elemzésével képes a program önállóan változtatni a szálak számát, és azt beállítani a legjobb teljesítményt nyújtó értékre. Ez főleg akkor válik fontossá, ha már nem csak a magyar webet szeretnénk elemezni, és szükség lesz a crawler több gépen való futtatására is, ahol az eltérő gépekből álló gépparkon rengeteg időt és energiát takaríthatunk meg a tesztelés automatizálásával.

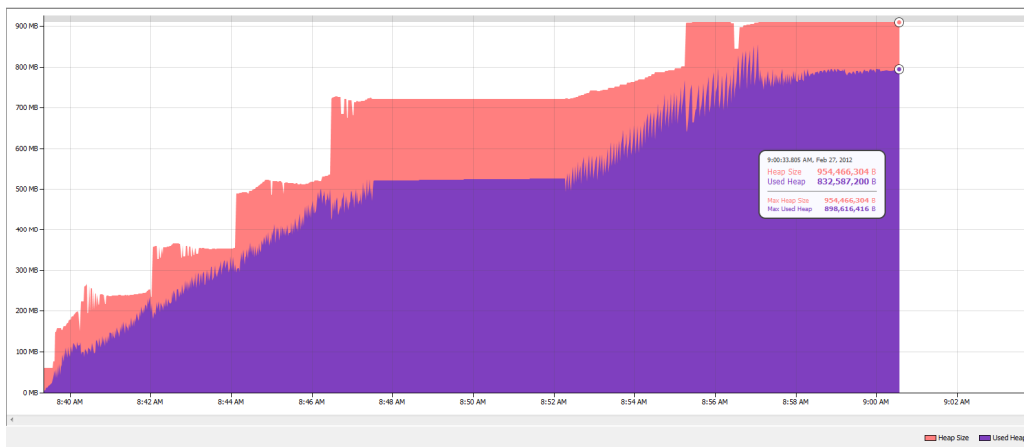
Több gépen való futtatás esetén problémás lehet az adatbázis kezelése, előfordulhatnak hibák, ha két gép egymástól függetlenül ugyanazokat a sorokat kezdi el feldolgozni. Erre egy olyan megoldást találtunk ki, hogy a sorok egyedi azonosítójának és az éppen futó gépek számának vesszük a osztás utáni maradékát, és minden gép csak a saját számának megfelelő maradékú sorokat elemezhet és szerkeszthet.

5 Technológiai nehézségek és alkalmazott megoldásaik

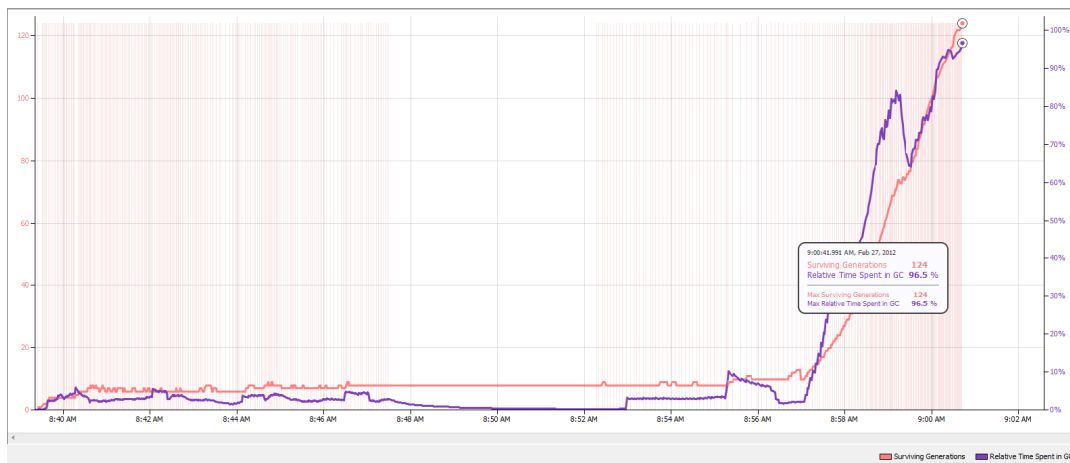
A crawler fejlesztése során többször nehézségbe ütköztünk. Jelen fejezet célja ezek közül néhány érdekesebb bemutatása, és megoldásuk ismertetése.

Az első igazi akadályt maga a többszálúsítás jelentette, kevés példa állt előttünk, és még kevesebb tapasztalatunk volt a témában, így sok utánajárást és próbálkozást kívánt a feladat megoldása. Szerencsénkre a kereső szálai aránylag egyszerű munkát látnak el, nincs szükség közöttük szinkronizációra, nem használnak közös változókat, kivéve az adatbázis kapcsolatok kezelésénél, de ez jól tervezhető és ellenőrizhető körülmények között zajlik a futás során. A megoldás végül egy ThreadPool osztály használata lett, amivel jelentősen sikerült leegyszerűsíteni a szálak kezelésének a problémáját, ezt lényegében a Java Virtuális Gép (JVM) kezeli.

A későbbiekben megjelent a memória szivárgás problémája, mely alapesetben is nehezen felderíthető és javítható probléma, masszívan többszálúsított programok esetén viszont igazi kihívás megtalálni a hiba okát. Az alábbi ábrán látható módon a program futása során szépen felélte a számára biztosított memóriát, mely magával vonta a nyelv szemétygyűjtő, Garbage Collector (GC) túlterhelődését, mely blokkolta az egész program futását, majd az végül össze is omlott.



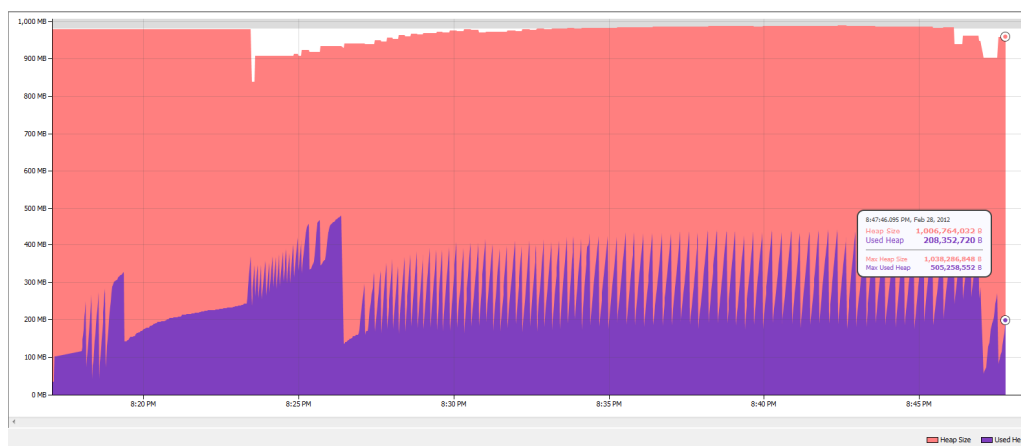
21. ábra: A crawler memóriaszivárgása



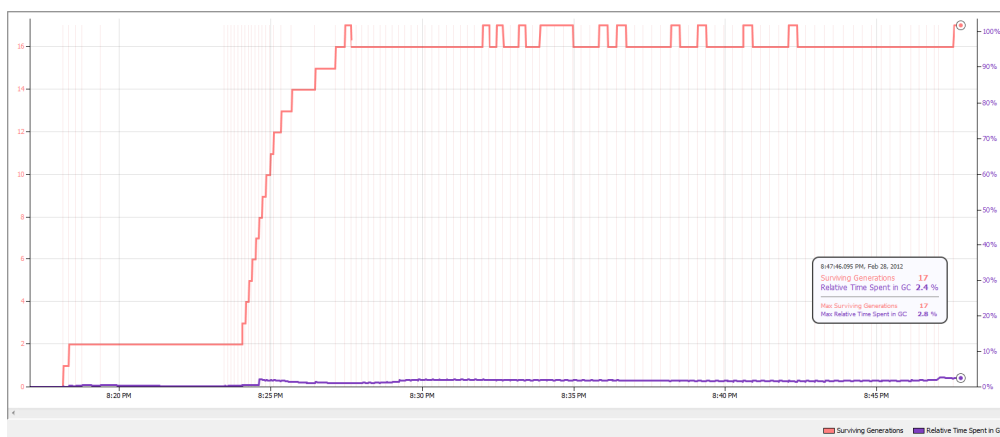
22. ábra: A szemétyűjtő (garbage collector) túlterhelődése

Látható, ahogy az átlagos pár százalékos GC terhelés felfut egészen 100%-ig, mely a memória hiányával együtt teljesen megbénítja a program további futását.

A megoldást az adatbázis lekérdezéseinek és műveletinek a megfelelő lezárása jelentette, mivel bármilyen hiba esetén ezek nem kerültek lezárásra, szépen lassan megtöltötték a memóriát. A hiba javítása után jól látható, ahogy a program memóriaigénye beáll egy szintre, és nem növekszik tovább.



23. ábra: A helyesen működő program memória igénye



24. ábra: A szemétgyűjtő (garbage collector) normális működése

Az adatbázis kapcsolatok kezelését a crawleren belül, a szálak kezeléséhez hasonló módon ConnectionPool segítségével oldottuk meg. Ennek felépítése és működésének logikája jól illeszkedett a szálak futás közbeni viselkedéséhez.

A rendszer alatt szolgáló adatbázis tábláinak motorjául a sokkal intenzívebb beillesztési műveletek miatt az InnoDB tárolási eljárást választottuk, mely jól alkalmazkodik a nagyon magas adatbázisba írás különböző igényeihez.

6 A crawler hatékonyságának mérése

Jelen fejezet célja, hogy a crawler teljesítményének tesztelését bemutassa.

6.1 Mennyiségi teljesítménytesztek

A rendszer Java nyelven íródott, többszálúsított. Minden egyes szál saját adatbáziskapcsolattal bír, és a fő szál által neki kiosztott URL-eket dolgozza fel. A rendszer párhuzamosíthatóságának mértéke paraméterezett, a crawler indításakor beállítható. Így különböző hardver konfigurációkra empirikusan meghatározható, hogy mekkora az optimális párhuzamosan futó szálak száma.

Az optimális szálszám elsősorban a CPU-tól és az elérhető hálózati sebességtől függ. Jelen teszteléshez 2 db 4 magos 2,83 GHz-es CPU-val bíró Ubuntu Linux operációs rendszerű serveren futtattuk a crawlert. A szál számot növelve azt teszteltük, hogy 1 óra alatt hány weboldalt képes feldolgozni a rendszer. A tesztelés kezdetekor 140 ezer db URL volt a crawler látogatási adatbázisában. Hogy minden egyes teszt hasonló weboldalak letöltését jelentse, a weboldalak feldolgozása után nem módosítottuk a következő letöltési időt. Az újralátogatási algoritmus lefutott, azonban a számolt értékeket nem írtuk vissza az adatbázisba, így biztosítható volt, hogy jobban összehasonlíthatóak legyenek az egyes futások, mivel a letöltendő és feldolgozandó weboldalak köre és sorrendje nem változott.¹⁸

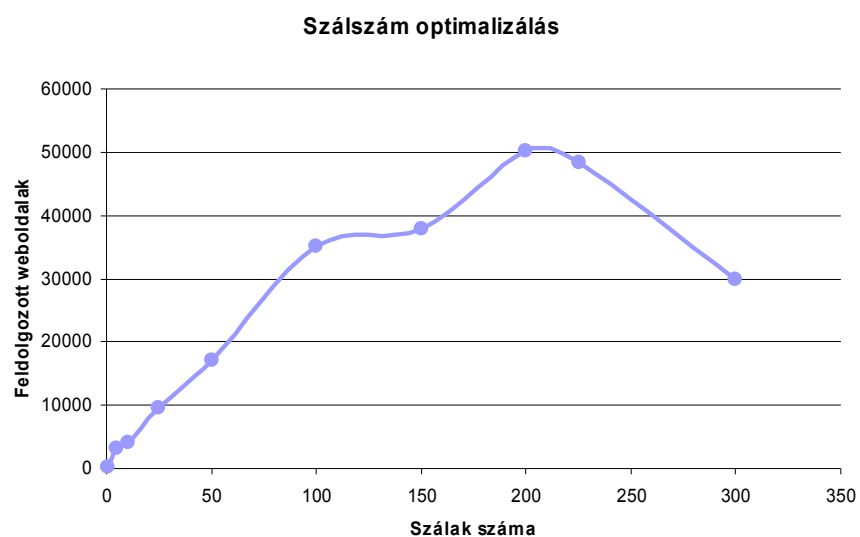
A szálszám optimalizálás eredményei:

#	Szálszám	Feldolgozott weboldalak száma (db)	Bejárési sebesség (weboldal/sec)
1	1	303	0,08

¹⁸ Amennyiben a „normálisan” működött volna a crawler a száloptimalizálás során, akkor a weboldalak sokszínűsége miatt jelentősen eltérő teljesítményekkel találkozhatunk volna, mivel weboldalak HTML kódjának hossza és az oldalon lévő hivatkozások száma jelentősen szóródik. Nem mellesleg a weboldalak feldolgozásának legtovább tartó része a csatlakozás a weboldalhoz, illetve sikertelen csatlakozás esetén az egyszeri újracsatlakozás. Ezek akár 5-6 másodpercbe is kerülhetnek, és több ilyen oldal feldolgozása jelentősen csökkentheti az átlagos teljesítményt.

#	Szálszám	Feldolgozott weboldalak száma (db)	Bejárési sebesség (weboldal/sec)
2	5	3 087	0,86
3	10	4 116	1,14
4	25	9 562	2,66
5	50	17 125	4,76
6	100	35 031	9,73
7	150	37 879	10,52
8	200	50 279	13,97
9	300	29 859	8,29
10	225	48 442	13,45

3. Táblázat: A crawler szálszám optimalizálása



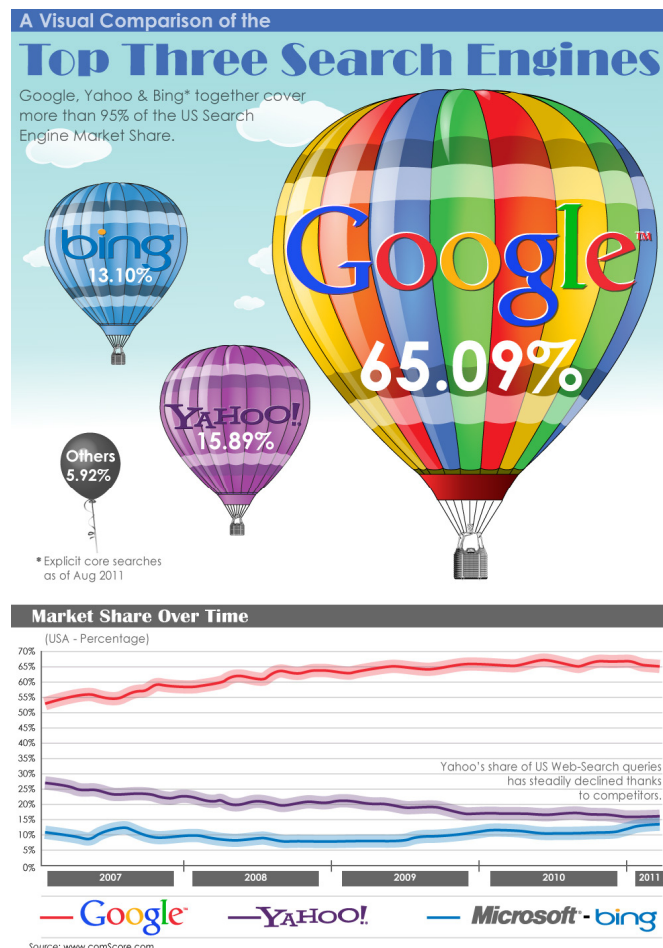
25. ábra: Az optimális párhuzamos szálszám meghatározása

Az optimális szálszámmal indítva a crawlert egyetlen szerveren, naponta 1 206 696 db weboldal feldolgozására képes, ha lineáris becslést alkalmazunk. (A teljes teljesítményméréshez hozzátartozik, hogy jelenleg a használt adatbázistábla viszonylag kis méretű, körülbelül 160 000 URL-t tartalmaz, ennek a számnak a megsokszorosodása minden bizonnyal csökkenteni fogja a teljesítményt. Az így felmerülő teljesítménycsökkenést többek között több adatbázis szerver és tábla használatával igyekszünk majd a későbbiekben kompenzálni).

6.2 Referenciatesztek

A crawler teljesítményének referenciatesztelése során különböző referenciaadatokkal vetettük össze a rendszerünk teljesítményét.

A Egyesült Államokban és Európában legnépszerűbb 3 kereső találati számait vizsgáltuk.



26. ábra: A 3 legnépszerűbb kereső piaci részesedése az USA-ban 2011. augusztusban (forrás: http://kereses.blog.hu/2011/10/12/keresopiaci_helyzet_usa_infografika)

Az egyes keresőkben az „az” kifejezésre kerestünk magyar nyelvű weboldalak között 2012.03.19-én, melyek az elmúlt 24 órában keletkeztek:

- Google 224 000 találatot adott.
- A Bing kereső ugyanarra a beállításra 75 100 találatot adott.
- A Yahoo 817 000 találatot adott, de itt fontos megjegyezni, hogy a Yahoo első 20 találata között a beállítás ellenére nem volt magyar nyelvű találat, így a tényleges magyar nyelvű találatok száma ennek a számnak a töredéke csupán.

Ezek alapján a crawlerünk elméletileg képes arra, hogy a legelterjedtebb webes keresők által 1 nap alatt beindexelt magyar nyelvű webes tartalom mennyiséget megtalálja és feldolgozza akár 1 napnál kevesebb idő alatt. Érdemes megemlíteni, hogy az általunk készített crawler a három keresőóriás hardverkapacitásával szemben egyetlen 4 magos processzorú szerveren fut.

6.3 Az újralátogatási algoritmus tesztelése

A crawlerünk újralátogatási algoritmusát úgy terveztük, hogy az képes legyen minél hamarabb megtalálni a friss magyar nyelvű tartalmakat illetve kommenteket, fórum hozzászólásokat. Hogy igazoljuk az algoritmusunk eredményességét, a weboldal kategorizáláson alapuló újralátogatási algoritmusunkat összevetettük egy „hagyományos” újralátogatási algoritmussal.

A tesztelés során a crawlerünkéből csináltunk egy hagyományos crawlert (továbbiakban: referencia crawler) az újralátogatási algoritmus módosításával: weboldal kategorizálást eltávolítottuk az algoritmusból. A referencia crawler a megtalált oldal tartalomhossz változása alapján állapította meg, hogy mikor kell legközelebb meglátogatnia az adott oldalt. Ezt leszámítva a két rendszer teljesen megegyezett.

Ezt követően 2012. március 20-án duplikáltuk az adatbázist, és elindítottuk a két crawlert egyszerre, ugyanazon a hardveren, ugyanazon feltételekkel, az előzőekben megállított optimális szálszám felével (100 szál).

2012. március 25-ig a referencia crawler 1 730 860 db weblapot dolgozott fel, míg a saját crawlerünk 2 556 569 weboldalt. A több mint 800 000-es különbség jól mutatja azt a hatékonyságjavulást, amit az elemző adott a rendszernek. Mivel ezt alkalmazva a keresőnk már nem látogatta újra azokat az oldalakat, ahol várhatóan nem frissül többet a tartalom, sikerült 47%-al több weboldalt feldolgoznunk.

Később megvizsgáltuk a crawlerek logfájljai alapján, hogy mely crawler hány aktuális tartalmat dolgozott fel. Az aktuális tartalmak megtalálását az URL-ekben előforduló dátumbélyegek alapján vetettük össze (például a 2011. január 10-én keletkezett cikkek esetén az URL legtöbb esetben tartalmazza a 2011/01/10/ vagy 20110110 karaktersort, lásd például az Origo, Index.hu vagy a Hvg.hu hírportálok cikkeit).

A futást a nagyban befolyásolta, hogy a crawlerek friss indítása miatt azok erőforrásait jelentősen lekötötte a weben keletkezett régebbi tartalmak felderítése és osztályozása. Mindez jelenleg abban nyilvánul meg, hogy a friss tartalmak megtalálása 1-2 napos csúszással történik meg. Várhatóan egy bizonyos futásmennyiség (kb. 30 nap) után több erőforrás jut majd a friss tartalmak kezelésére.

Dátum	Referencia crawler	Saját crawler
2012.03.20.	1 249	1 468
2012.03.21.	2076	1 894
2012.03.22.	35	687
2012.03.23.	24	26
Összesen	3 384	4075

4. Táblázat: Újralátogatási algoritmus tesztelésének eredményei

A fentiek alapján elmondható, hogy az általunk definiált újralátogatási szabályrendszer hatékonyabb a „hagyományos” újralátogatási algoritmusoknál, ha a friss tartalmak megtalálása a cél.

7 Összefoglalás

A bevezetőben felvázolt problémát, illetve elérendő célokként megjelölt két pontot a fentiek olvasatában teljesítettnek tekinthetjük.

Az első pontban megfogalmazott elérendő eredményt, miszerint képesek legyünk lépést tartani a magyar webes tartalom bővülésével, a napi több mint 1 000 000 oldalfeldolgozással bőven túlteljesítjük, hisz a Google keresője is csak mintegy 250 000 magyar weboldalt indexel naponta. Ez szerencsére egyben azt is jelenti, hogy a mostani hardvernél gyengébb gép is megfelelően kielégítheti az elvárásainkat, mely szintén egy fontos kritérium volt a rendszer megtervezése és megvalósítása közben.

Második célunkként a friss tartalmak mielőbbi megtalálását tűztük ki célul, ezt egy elméleti teszteléssel igazoltuk. Gyakorlatban történő teljesítését egyértelműen sajnós nem tudtunk még igazolni, de a teljesítménytesztelés során megállapított feldolgozási sebesség figyelembevételével, illetve a régebbi tartalmak indexelésének befejeztével később nem okozhat gondot a rendszernek a legfrissebb tartalmak szinte azonnali felderítése. Jelenleg leginkább a régebbi weboldalak bejárása köti le a crawlerünk kapacitásainak nagy részét. Elképzelhető, hogy a későbbiekben még korlátoznunk is kell a futási sebességét, hogy ne terhelje le feleslegesen a látogatott oldalak szervereit.

8 Ábrajegyzék

1. ábra: A CRISP-DM módszertan lépései (forrás: http://www.finisco.com/crisp-data-mining-methodology.aspx)	12
2. ábra: Címlap weboldaltípus példák (www.filmklub.postr.hu és www.index.hu)	14
3. ábra: A weboldal kategóriák megoszlása	16
4. ábra: RapidMiner adatelőkészítési lépések	17
5. ábra: Címlap osztályozási algoritmusok teljesítményének összehasonlítása	19
6. ábra: A címlap osztályozás pontossága	20
7. ábra: Linkgyűjtemény osztályozási algoritmusok teljesítményének összehasonlítása	20
8. ábra: A linkgyűjtemény osztályozás pontossága	20
9. ábra: A cikk kommentekkel osztályozási algoritmusok teljesítményének összehasonlítása	21
10. ábra: A cikk kommentekkel osztályozás pontossága	21
11. ábra: A cikk kommentek nélkül osztályozási algoritmusok teljesítményének összehasonlítása	22
12. ábra: A cikk kommentek nélkül osztályozás pontossága	22
13. ábra: A blog kommentekkel osztályozási algoritmusok teljesítményének összehasonlítása	23
14. ábra: A blog kommentekkel osztályozás pontossága	23
15. ábra: A blog kommentek nélkül osztályozási algoritmusok teljesítményének összehasonlítása	24
16. ábra: A blog kommentek nélkül osztályozás pontossága	24
17. ábra: A fórumfőoldal osztályozási algoritmusok teljesítményének összehasonlítása	25
18. ábra: A fórumfőoldal osztályozás pontossága	25
19. ábra: A fórum osztályozási algoritmusok teljesítményének összehasonlítása	26
20. ábra: A fórum osztályozás pontossága	26
21. ábra: A crawler memóriaszivárgása	31

22. ábra: A szemétyűjtő (garbage collector) túlterhelődése.....	32
23. ábra: A helyesen működő program memória igénye	32
24. ábra: A szemétyűjtő (garbage collector) normális működése	33
25. ábra: Az optimális párhuzamos szálszám meghatározása	35
26. ábra: A 3 legnépszerűbb kereső piaci részesedése az USA-ban 2011. augusztusban (forrás: http://kereses.blog.hu/2011/10/12/keresopiaci_helyzet_usa_infografika).....	36

9 Táblázatjegyzék

1. Táblázat: Az osztályozások pontosságainak összefoglaló táblázatát.....	27
2. Táblázat: Az újralátogatási algoritmus összefoglaló táblázata	29
3. Táblázat: A crawler szálszám optimalizálása.....	35
4. Táblázat: Újralátogatási algoritmus tesztelésének eredményei.....	38

10 Felhasznált források

Bárházi, E., Héder, M. [2010]: Panaszlevelek szerkezetének gépi felismerése, Magyar Számítógépes Nyelvészeti Konferencia 2010.

Berend G. és Farkas R. [2008]: Opinion Mining in Hungarian based on textual and graphical clues, in Proceedings of the 4th Intern. Symposium on Data Mining and Intelligent Information Processing, Santander, 2008.

Bodon, F. [2010]: Adatbányászati algoritmusok (letölthető: www.cs.bme.hu/~bodon/magyar/adatbanyaszat/tanulmany/adatbanyaszat.pdf, utoljára letöltve: 2012.03.19-én)

Boldi, P., Codenotti, B., Santini, M. and Vigna, S. [2004]: UbiCrawler: a scalable fully distributed Web crawler. Software: Practice and Experience 34 (8): 711–726

Cho, J. and Garcia-Molina, H. and Page, L. [1998]: Efficient Crawling Through URL Ordering. In: Seventh International World-Wide Web Conference (WWW 1998), April 14-18, 1998, Brisbane, Australia.

Cho, J. [2001]: Crawling the Web: Discovery and Maintenance of a Large-Scale Web Data. Ph.D. dissertation, Department of Computer Science, Stanford University, 2001

Fajszai, B [2010]: Üzleti haszon az adatok mélyén – Az adatbányászat mindennapjai, Alinea Kiadó – IQSYS Informatikai és Tanácsadó Zrt.

Grossman, L. [2010]: Twitter can predict the stock market (forrás: <http://www.wired.com/wiredscience/2010/10/twitter-crystal-ball/>, utoljára letöltve: 2012.03.12-én)

Han, J. – Kamber, M. [2004]: Adatbányászat – Konceptiók és technikák, Budapest, Panem Könyvkiadó

Hsu, W. H. és Weninger T. [2008]: Text extraction from the web via Text-to-Tag Ratio

Miháltz Márton [2010]: OpinHu: online szövegek többnyelvű véleményelemzése, VII. Magyar Számítógépes Nyelvészeti Konferencia

Nyékyné Gaizler, J. [2009]: Java 2 útikalauz programozóknak: 5.0, Budapest, ELTE Természettudományi Kar

Shestakov, D. [2008]: Search Interfaces on the Web: Querying and Characterizing. TUCS Doctoral Dissertations 104, University of Turku

Sántáné-Tóth [2008]: Döntéstámogató rendszerek, Panem kiadó, Budapest

Szekeres, P. [2008]: A gépi tanulás intelligens módszerei a döntéstámogatásban -A legemberközelibb technikák összehasonlítása, BCE Gazdálkodáskodástudományi Kar, Tudományos Diákköri dolgozat

Szekeres, P. [2009]: Az információszerzés új dimenziói – az adatbányászat, a szövegbányászat és a webbányászat üzleti előnyei, BCE Gazdálkodáskodástudományi Kar, Tudományos Diákköri dolgozat

Szekeres, P. [2011]: A tőzsdei kisbefektetői hangulat elemzése webes információk alapján, BCE Gazdálkodáskodástudományi Kar, Tudományos Diákköri dolgozat

Tikk, D. [2007]: Szövegbányászat, Typotex Elektronikus Kiadó Kft, Budapest