OKOSVÁROS-TECHNOLÓGIÁK V.

Szövegdokumentumok elemzése a tartalomkezelő rendszerekben

Szűcs Gábor– Sallai Gyula

NEMZETI KÖZSZOLGÁLATI EGYETEM
BUDAPEST

SZÉCHENYI 2020
MAGYARORSZÁG KORMÁNYA
Európai Unió
Európai Szociális Alap
BEFEKTETÉS A JÖVŐBE

OKOSVÁROS-TECHNOLÓGIÁK
A technológia fejlődésének irányai és hatása
V. kötet

Sorozatszerkesztő:
Sallai Gyula

Szűcs Gábor– Sallai Gyula

SZÖVEGDOKUMENTUMOK ELEMZÉSE A TARTALOMKEZELŐ RENDSZEREKBEN

Nemzeti Közszolgálati Egyetem
Közigazgatási Továbbképzési Intézet
Budapest, 2020

A kötet a Nemzeti Közszolgálati Egyetem KÖFOP-2.1.2-VEKOP-15-2016-00001
„A jó kormányzást megalapozó közszolgálat-fejlesztés” projektje keretében,
a Budapesti Műszaki és Gazdaságtudományi Egyetem Villamosmérnöki és
Informatikai Karán létesült „Okos város – okos közigazgatás” kutatóműhelyben
(2017/162/BME-VIK) készült.

Szakmai lektor:
Magyar Gábor, tszv. egyetemi docens, BME-VIK

Szerző:
Szűcs Gábor– Sallai Gyula

A kézirat lezárásának dátuma:
2018. június 30.

© Nemzeti Közszolgálati Egyetem
Közigazgatási Továbbképzési Intézet, 2020
© Szűcs Gábor– Sallai Gyula, 2020

A mű szerzői jogilag védett. Minden jog, így különösen a sokszorosítás, terjesztés
és fordítás joga fenntartva. A mű a kiadó írásbeli hozzájárulása nélkül részeiben sem
reprodukálható, elektronikus rendszerek felhasználásával nem dolgozható fel,
azokban nem tárolható, azokkal nem sokszorosítható és nem terjeszthető.

TARTALOM

1. BEVEZETÉS

Információs társadalmunk egyre nagyobb mértékben állít elő új információt, és ez igaz a közszférára is; a közigazgatási intézményekben rengeteg irat, digitális dokumentum keletkezik. Ebben a hatalmas mennyiségű halmazban meg kell oldani a heterogén tartalmak tárolását (rövid és hosszú távú megőrzését), felhasználását, megosztását, kereshetőségét, változásainak nyomon követését, egyszóval kezelését. Ezek közül kiemelt feladat a keresés hatékony megvalósítása, hiszen sokszor nehéz megtalálni egy konkrét információt (szövegrészt, az ügyhöz szorosan kapcsolódó adatot vagy tartalmat), ami az érintetteket (ügyfelet vagy ügyintézőt) érdekli. Ezenkívül van még számos olyan feladat, ami a felhasználó szempontjából láthatóan rendezetlen halmazban egyfajta rendszert, struktúrát teremt. Ilyen például a tartalomegységek csoportosítása, a tartalom összegzése, a szöveges tartalmak szétválogatása, és még hosszan lehetne sorolni: ezeket együtt röviden elemzési feladatnak hívjuk.

Az okos - közigazgatáshoz szükség van ilyen elemzési problémák megoldására, ezért a „Szövegdokumentumok elemzése a tartalomkezelő rendszerekben" című tanulmányunkban megmutatjuk, hogy ezeket hogyan, milyen kisebb lépésekben szükséges megoldani – részletesen bemutatva a technológiák hátterét. Az okos várossá fejlesztés irányába tett lépésekhez hozzátartozik a (i) tartalomkezelő rendszerek bevezetése, hogy felhasználóik számára lehetővé tegyék nagy mennyiségű tartalom rendszerezett felvitelét, tárolását, megosztását, kezelését, felhasználását, keresését – ezeket a 2. és 3. fejezetben mutatjuk be –, továbbá lényeges pont (ii) az intelligens elemzés a szöveges dokumentumokon, melyet a 4., 5. és 6. fejezetben tárgyalunk.

A 2. fejezetben a tartalomkezelő rendszerek architektúrájára és altípusaira fókuszálunk, a 3. fejezetben pedig arra, hogy a felhasználók hogyan tudják ezt használni (hogyan tudnak dokumentumokat rendszerezni, szerkeszteni, illetve hogyan tudják a saját fiókadataikat kezelni, és karbantartani az egész rendszert). A 4. és 5. fejezet szintén összefügg, mivel az előbbi az előfeldolgozási lépéseket mutatja be, míg az utóbbi a szövegelemzés részleteit. A 6. fejezet egy szoros kapocs az elemzés és a tartalomkezelés között, ugyanis a keresésről – szaknyelven szólva az „információ-visszakeresésről" – szól. Mindkét témához kapcsolható, ugyanis a tartalomkezelő rendszerek egyik fontos funkciója, hogy a felhasználó megtalálja benne, a keresni kívánt tartalmat, másrészt hasonló szövegreprezentáció szükséges mind az elemzéshez, mind pedig a kereséshez. Tanulmányunkat a 7. fejezettel, az okos közigazgatás szemszögéből készült összefoglalóval zárjuk.

Jelen tanulmány célja tehát, hogy bemutassa a tartalomkezelő rendszerek és a szövegbányászat jelentőségét az üzleti életben és a közszolgáltatásban, mindemellett az okos városhoz kapcsolódóan áttekintést adjon a témakörről, valamint néhány példával bemutassa alkalmazhatóságukat.

Szerzők:

Dr. Szűcs Gábor okleveles villamosmérnök, informatikai PhD, a Budapesti Műszaki és Gazdaságtudományi Egyetem egyetemi docense. Szakmai és tudományos életpályáját szimuláció és mesterséges intelligencia kutatásával kezdte; jelenlegi kutatási területe az adattudomány, valamint a mesterséges intelligenciához tartozó gépi látás és mély tanulás.

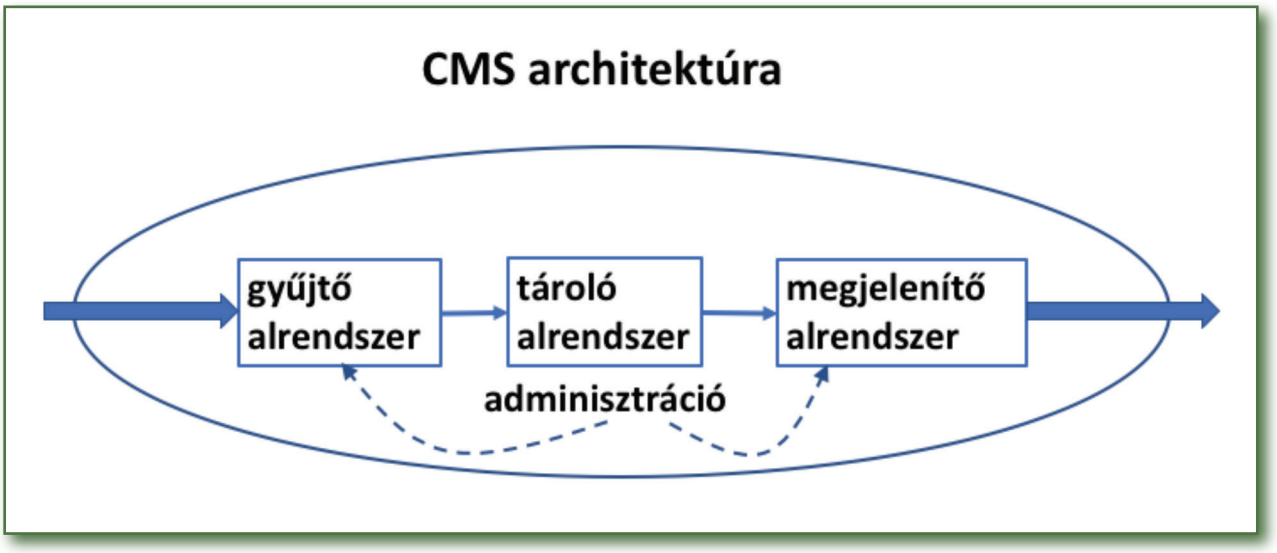
Prof. Dr. Sallai Gyula okleveles villamosmérnök, a Magyar Tudományos Akadémia doktora, a Budapesti Műszaki és Gazdaságtudományi Egyetem professor emeritusa. Szakmai és tudományos életpályája során a távközlés, illetve az infokommunikáció különböző szakterületein dolgozott. Az utóbbi években fő érdeklődési területe a jövőinternet-technológiák és azok okosváros-alkalmazásai.

2. TARTALOMKEZELŐ RENDSZEREK (CMS)

A tartalomkezelő rendszerek (CMS - Content Management System) célja, hogy felhasználóik számára lehetővé tegyék nagy mennyiségű, változatos típusú tartalom kezelését, azaz a tartalmak rendszerezett felvitelét, tárolását, megosztását, felhasználását, valamint a tartalmak közötti gyors és hatékony keresést [Boiko, 2005]. A CMS-rendszerekben az egyes tartalmakhoz való hozzáférés és a tartalmakon végezhető műveletek is jól szabályozhatók a rendszerekbe épített szerepkörök és jogosultságok segítségével. Így biztonságos, többfelhasználós környezetet nyújtanak, amely jelentősen megkönnyíti a csoportmunkát, mely lehet egy nagyvállalat összes hivatalos dokumentumának menedzselése, vagy a közigazgatásban használt hivatalos iratok kezelése. A tárolt és kezelt tartalmak rendkívül sokfélék lehetnek: szövegdokumentumok, fényképek, videók, jegyzetek, hangfelvételek, naplóbejegyzések, kapcsolati adatok, termékkatalógus elemei, újságcikkek, események leiratai, esettanulmányok, tudástár elemek stb. Ezt a fejezetet a fent említett fogalmak kifejtésére szánjuk oly módon, hogy abból részletesen kiderüljön a CMS-rendszerek működése is.

2.1 CMS-RENDSZEREK ARCHITEKTÚRÁJA ÉS ALTÍPUSAI

A CMS-rendszerek, architektúrájukat tekintve, három alapvető részre oszthatók fel. Ezek a fő feladatoknak megfelelően: a gyűjtő, tároló és megjelenítő alrendszerek [(angolul: collection, management, publishing subsystems [Boiko, 2005], melyeket az 1. ábra alapján az alább részletezünk.



1. ábra. CMS-architektúra három alrendszerrel

Forrás: a szerzők saját szerkesztése

A gyűjtő alrendszer felelős az tartalmak bekerüléséért a rendszerbe; segítséget nyújt a felhasználóknak az adatok strukturált felvitelében és/vagy előállításában, metaadatok hozzáadásában; elvégzi az esetleges formátumkonverziókat, valamint tartja a kapcsolatot a tároló alrendszerrel.

A tároló alrendszer az CMS-rendszerek legfontosabb része, a tartalmak tárolásán túl a rendszer valamennyi beállításának és adminisztratív információjának tárolásáért is felel, ahogy ez az 1. ábrán látható. Ez az alrendszer tartja a kapcsolatot az adatbázis szerverrel és/vagy a fájlrendszerrel.

A megjelenítő alrendszernek a rendszerben tárolt tartalmak felhasználásában van fontos szerepe; a tároló alrendszertől kapott adatok alapján ez a réteg állítja elő a publikációt, rajta keresztül kereshetnek és férhetnek hozzá a felhasználók az egyes tartalmakhoz.

Működésüket tekintve a CMS-rendszerek 3 félék lehetnek [Kumar, 2013], úgymint online feldolgozású, offline feldolgozású és hibrid rendszerek. Az online feldolgozású rendszerek esetében a megjelenítő alrendszer a publikációt felhasználói kérésre végzi, azaz minden egyes kérésre adott választ akkor állít össze a sablonok, az adatok és a bemenő paraméterek felhasználásával, amikor a kérést megkapja. Offline feldolgozású rendszerek esetében a lehetséges válaszokat adott időközönként vagy változások hatására előre előállítja a rendszer, és később, kéréskor ezekből szolgál ki. A mai rendszerek többsége hibrid rendszer, azaz a gyorsaság és a frissesség közötti jó kompromisszumként a ritkán változó válaszokat előre, míg a gyakran változókat a felhasználói kéréssel egy időben készíti el.

A CMS-nek több altípusa is kialakult az idők folyamán, melyek a következők:

- DMS (Document Management System): A dokumentumkezelő rendszerek elektronikus irattárak kialakítását lehetővé tévő rendszerek, melyek biztosítják az elektronikus dokumentumok létrehozásának, tárolásának, kezelésének, keresésének, továbbításának, szükség esetén megsemmisítésének, azaz a dokumentum teljes életciklusának felügyeletét [Altenhofen et al., 2004].
- DAM (Digital Asset Management): A digitális vagyonkezelő rendszereket elsősorban multimédiás tartalmak, azaz (képek, hangállományok, animációk, videók [Kovács et al., 2010] bevitelére, tárolására, metaadatokkal történő ellátására, katalogizálására, visszakeresésére használják [Regli, 2009].
- LMS (Learning Management System): A tanulásmenedzsment-rendszerek olyan webportálok, amelyekkel oktatási intézmények alakíthatnak ki tanításra és tanulásra használható virtuális tereket kurzusokkal és azokhoz tartozó tananyagokkal ([Ninoriya et al., 2011]).
- WCMS (Web Content Management System): A webes tartalomkezelő rendszerek vállalkozások, cégek és egyéb szervezetek számára teszik lehetővé tartalmak tárolását és közösségi kommunikációra alkalmas webhelyek kialakítását és menedzselését úgy, hogy a felhasználóknak nincs szükségük magas szintű informatikai tudásra [Roebuck, 2012].
- ECMS (Enterprise Content Management System, de legtöbbször csak ECM-nek rövidítik): A vállalati tartalomkezelő rendszer egy integrált CMS, amely az előzőekben felsorolt altípusok közül többet is magában foglalhat, hogy biztosítsa a vállalat számára a szervezeti tartalmak menedzselését a hatékony működés érdekében [Alalwan & Weistroffer, 2012].

A városi polgárok többségéhez legközelebb a webes CMS-rendszerek állnak, a 2.2. alfejezetben ezért ezt mutatjuk be részletesebben a felhasználók szempontjából.

A CMS-nek tehát több altípusa is létezik: a dokumentumkezelő rendszerektől a digitális vagyonkezelő rendszereken át az integrált vállalati tartalomkezelő rendszerekig.

2.2 WCMS-RENDSZEREK

A CMS-rendszerek megvalósításának ma leginkább elterjedt módja a webes tartalomkezelő rendszer (WCMS – Web Content Management System), amely nem más, mint egy webalkalmazásként implementált CMS-rendszer, amely böngészőn keresztül érhető el. Ez természetesen nem azt jelenti, hogy kizárólag publikusan, az interneten elérhető CMS-rendszerről van szó, hiszen sok vállalat belső, céges hálózatában is megtalálhatóak. Elsősorban azzal a céllal jöttek létre, hogy jelentősen leegyszerűsítsék a webes tartalmak publikálását, rendszerezését és kezelését. Egy modern WCMS-rendszer használatához nincs szükség magas szintű informatikai ismeretekre, a bonyolult folyamatok a felhasználók elől rejtve maradnak. Ezt bizonyítja, hogy mostanra ezek a rendszerek a részét képezik mindennapjainknak; nyugodtan kijelenthető, hogy nincs olyan mai internetfelhasználó, aki ne találkozna ilyen rendszerekkel. Ide sorolhatók a teljesség igénye nélkül a hírportálok, a fórumok, a wikik, a különböző közösségi portálok, online lexikonok, tudástárak, videó- és képmegosztó oldalak, online boltok, online aukciós portálok, vagy akár a sok munkahelyen használt, csoportmunkát támogató, dokumentumkezelő rendszerek is, mint például a Microsoft SharePoint.

Rengeteg WCMS létezik, sok közülük a nyílt forráskódú, közösségi fejlesztésű. Ezek rendkívül gyorsan fejlődnek, szinte hétről hétre jelennek meg újabb és újabb funkciók bennük, ráadásul a teljes forráskódjuk, és ezáltal architektúrájuk, szabadon elérhető bárki számára. Rengeteg ötletet és tapasztalatot lehet gyűjteni e rendszerek használata közben, már rövidtávon is, funkcionális és architekturális kérdésekben egyaránt. Jelen tanulmányban a két legnépszerűbb nyílt forráskódú webes tartalomkezelő rendszert: a Joomla!-t és a Drupalt ismertetjük részletesen [Patel et al., 2011].

2.2.1 Joomla!

A Joomla! [Rahmel, 2008] széles körben elterjedt univerzális webtartalom-kezelő programcsomag, amelyet készítői a legkisebb blogoktól kezdve a legnagyobb nagyvállalati portálokig minden vállalatnak ajánlanak. PHP- (Hypertext Preprocessor) alapú, adatbázisként MySQL-t használ, így széles körben, ingyenes platformokon is felhasználható; telepítése egy átlagos felhasználó számára is igen könnyű. A telepítést követően a következő főbb funkcionalitásokkal rendelkezik:

- Felhasználókezelés: regisztráció-, bejelentkezés-, profil-, szerepköralapú jogosultságkezelés.
- Lokalizációs támogatás: akár párhuzamosan többnyelvű oldal lehetőségével.
- Beépített névjegyzék: a felhasználóknak, kapcsolataik tárolására.
- Tartalmak kezelése: létrehozás, értékelés, küldés e-mail-ben, mentés pdf-be.
- Médiakezelő: képek és más fájlok feltöltésére és kezelésére.
- Bannerkezelő: hirdetések és más bannerek beszúrása, kezelése.
- Szavazások kiírása, kezelése, eredmények publikálása.
- Keresés a tartalmak között.
- RSS- (Really Simple Syndication, azaz alkalmazáson belüli szolgáltatás, ami jelzi, ha a bejegyzett weboldal frissült) támogatás: vagyis ezt a nagyon egyszerű hírmegosztást támogatja.
- Menükezelő: tartalmi hierarchiától független menü létrehozása.

- Oldalsablonok kezelése.
- Súgó, beépítve, a Joomla! használatához.
- Optimalizációs szolgáltatások: gyorsítótár, GZIP-tömörítés (GZIP egy tömörítőformátum).
- Rendszergazdai szolgáltatások: hibakereső mód, hibajelentések, adminisztrátori privát üzenetek küldése a felhasználóknak, FTP- (File Transfer Protocol) réteg a bővítmények oldalon keresztüli letöltéséhez.

Az alapfunkciók külön telepíthető bővítményekkel egészíthetők ki. Ehhez több ezer kiegészítés áll rendelkezésre a hivatalos oldalon keresztül. Ha ezek között sem lenne megfelelő, akkor – a részletes fejlesztői dokumentáció és az ehhez rendelkezésre álló Joomla!-API (alkalmazásprogramozási interfész, mely az angol Application Programming Interface rövidítése) segítségével – fejlesztőként létrehozhatók saját funkciók, melyet utána a hivatalos kiegészítés-gyűjteményben is meg lehet osztani a Joomla!-közösség többi felhasználójával.

2.2.2 Drupal

A Drupal [Tomlinson & VanDyke, 2010] az egyik legelterjedtebb rendszer a nyílt WCMS-ek között, célcsoportja a Joomla!-hoz hasonlóan széles, igyekszik a teljes piacot lefedni. Szintén PHP-alapú; adatbázisként a MySQL mellett a hasonlóan ingyenes PostgreSQL használatát is lehetővé teszi. Telepítése a Joomla!-hoz képes kicsivel bonyolultabb; kevesebb automatizmust tartalmaz, de még így is könnyen megoldható a lépésről lépésre végigvezető dokumentációnak köszönhetően. Az alaprendszer, a konkurenshez részben hasonlóan, az alábbi funkciókkal bír:

- Felhasználókezelés: regisztráció, profil, szerepköralapú jogosultságkezelés.
- Belső és külső autentikáció: bejelentkezés saját, vagy egy másik Drupal-oldal adatbázisa alapján.
- Tartalomkezelés: létrehozás, kezelés verziókövetéssel.
- Keresés a tartalmak között.
- Szavazások: kiírás, eredmények publikálása.
- Kommentek: tartalmakhoz való hozzászólások lehetősége.
- Sablonkezelés: oldalsablonok felvitele, módosítása.
- Menükezelő: többszintű, akár tartalomtól független.
- Beépített blogmotor: többforrású blogbejegyzések támogatása.
- RSS-támogatás.
- Lokalizációs támogatás: beépített fordítómodullal, többnyelvű oldal támogatása.
- Beépített fórum.
- Optimalizációs szolgáltatások: gyorsítótár, GZIP-tömörítés.
- Teljesen webalapú oldaladminisztráció.
- Rendszergazdai szolgáltatások: hibakereső mód, hibajelentések.

A fenti alapfunkciók, a Joomla!-hoz hasonlóan, itt is egy nagy közösség által karbantartott, hivatalos oldalon lévő gyűjteményből, modulok és más kiegészítők (melyek számossága több ezer) segítségével bővíthetők. Természetesen ehhez a rendszerhez is részletes fejlesztői dokumentáció áll rendelkezésre, amely segítségével tetszőleges funkció fejleszthető és megosztható.

2.2.3 Nyílt forráskódú WCMS-rendszerek hátrányai

A létező, elterjedt WCMS-rendszerek legnagyobb előnye egyben a legnagyobb hátránya is, amely jelentősen gátolja széles körű elterjedését, speciális, éles vagy ipari környezetekben – ez a multifunkcionalitás. Ezek a rendszerek igyekeznek az összes felhasználói oldalról érkező igénynek megfelelni, a vállalati weboldal funkciótól a wikin, a fórumon és a webshopon keresztül egészen az aukciós portálokig, kép- és videómegosztó oldalakig, de ez több szempontból is nehéz feladat [Vivekavardhan & Verma, 2016].

A nehézséget több összetevő adja. Az első ok az, hogy néha lehetnek olyan igények, amelyek bizonyos szempontokból ellentétes működést követelnének meg a rendszertől, ezért a megszületett kompromisszum valójában egyiknek sem felel meg igazán. Gondoljunk például arra, hogy mennyire különböző tárolási mód lehet optimális szöveges vagy kép-, esetleg videótartalmak esetében, melyiket milyen módon lehet jól visszakeresni, de ez csak egy példa a sokból.

A másik ok az, hogy mivel rengeteg igényt kell lefedni, az egyes megoldások általában nem elég aprólékosak a speciális rendszerek követelményeihez. A szükséges funkció megvalósításra kerül ugyan, de sem tartalmilag, sem működésében nem fedi le pontosan az igényeket. A legnagyobb probléma ebből a szempontból az egyes funkciók egymáshoz viszonyított kötöttsége vagy épp ennek teljes hiánya. Ez annyit tesz, hogy a rendszer tervezői által megálmodott és megvalósított munkafolyamatok, azaz a funkciók működésének és használatának módja, egyes elemeiben ugyan részletesen módosíthatók, bővíthetők, de a munkafolyamat alapstruktúrája csak nehezen vagy szinte egyáltalán nem változtatható.

A harmadik és talán legnagyobb probléma a bővíthetőségből következő általános és jelentős teljesítményromlás, sőt, alapértelmezés szerint is az ellátott funkcionalitáshoz képest gyengébb teljesítmény. Ennek oka, hogy az ilyen rendszereket az általánosságuk miatt, rengeteg bővítmény és kiegészítő fogadására készítik fel, ráadásul a fejlesztők munkájának és a rendszer átláthatóságának könnyítésére egy külön, rendszermagot elfedő réteg is megvalósításra kerül. Ennek következtében a működés során a vezérlés folyamatosan ugrál az alapmotor, a fejlesztői réteg, a bővítménykezelő és maguk a bővítmények között. Mindemellett, ha ezekből sok van, az – ebből kifolyólag – rengeteg felesleges rendszeren belüli műveletet eredményezhet, sőt a nyílt fejlesztésekből adódóan, ha esetleg a telepített kiegészítők mindegyikének forrása más, és még egy kicsit különböző szemlélettel is lettek megírva, az még további, komoly elpazarolt teljesítményhez és nehézkes, lassú működéshez vezethet.

A nyílt forráskódú rendszerek ezekre a problémákra némileg megoldást kínálnak a szabad módosíthatóságuk, kiegészíthetőségük által, de az általános szemléletű architektúra, illetve a problémák mélysége miatt ez gyakran csak olyan nagy, rendszermagot érintő átalakítások árán lenne lehetséges, hogy akkor már célszerűbb a speciális feladatokra saját WCMS-rendszer tervezésén/fejlesztésén gondolkodni, ami már a kezdetektől csak ezeket az igényeket veszi alapul, ezekre optimalizálja működését.

2.2.4 Kritériumok egy új WCMS tervezésekor

A jó teljesítmény kiemelten kezelendő egy WCMS esetében. Ennek elérésére a rendszernek minimalizálnia kell többek között az adatbázis műveleteket és az oldalak összeállítási idejét. Ennek hatékony eszköze egy gyorsítótár bevezetése, amely megfelelő időközönként statikus HTML-oldalakat generál a dinamikus tartalmakból, és ezekből szolgálja ki a bejövő kéréseket, töredékére csökkentve ezzel a kiszolgálási időket. A gyorsítótár mellett, mivel bizo-

nyos funkciók nem szolgálhatók ki ilyen módszerrel, fontos, hogy a dinamikus tartalmak a lehető legoptimálisabb algoritmusok segítségével álljanak elő a rendszerben. Törekedni kell az egyszerűségre és a gyorsaságra, mellőzve a felesleges műveleteket.

A robusztusság és a stabilitás a webes CMS-rendszerek esetében általában nagyrészt a kiszolgálószoftvereken és szervert működtető hardveren múlik. Mindkét esetben léteznek jól bevált ipari megoldások az esetleges problémák kivédésére. A hardverek és kiszolgáló szoftverek hibái ellen a legjobb megoldás egy többszörösen redundáns, elosztott, klaszteralapú rendszer használata. Ennek ellenére fontos, hogy ahol lehet, felkészüljünk a rendszer egyes részeinek kiesésére. Erre is egy gyorsítótár megléte adhat megoldást. Például egy adatbázisszerver kiesése esetén az oldalak kiszolgálása folytatódhat a gyorsítótárból, hiszen az ott található tartalmakhoz nincs szükség adatbázis kapcsolatra. Igaz ilyenkor feláldozzuk az adatok frissességét, de ilyen esetben a működés zavartalansága többnyire fontosabb. Más esetekben, például közigazgatási vagy jogi ügyintézésre szolgáló rendszerekben, magasabb prioritása lehet annak, hogy minden esetben a hatályos szabályoknak megfelelő legyen az információ: ekkor a gyorsítótár kiesése csökkent teljesítményt jelent ugyan, de minden kérést direkt módon a hatályos szabályoknak eleget tevő adatbázisból szolgálnak ki.

A közigazgatásban használandó WCMS-rendszerekben a munkavégzésnek gyorsnak, hatékonynak és egyértelműnek kell lennie, hogy a feladatok minél kevesebb kattintással elvégezhetők legyenek. A felhasználói felületeknek könnyen áttekinthetőknek kell lenniük úgy, hogy ne tartalmazzanak egyszerre az adott kontextusban szükségesnél több információt vagy kezelőszervet. Az esetleges felhasználói és más hibák esetére a rendszer, ahol szükséges, legyen felkészítve automatikus mentés, lomtár, vagy művelet visszavonása funkciókkal. Összességében a rendszer használatának kényelmesnek kell lennie, hogy soha ne bizonytalanítsa el a felhasználót.

Ahogy a fejezet elején bemutattuk: a CMS-nek több altípusa is létezik: a dokumentumkezelő rendszerektől a digitális vagyonkezelő rendszereken át az integrált vállalati tartalomkezelő rendszerekig. Vannak olyan CMS-ek is, melyek kimondottan a médiatartalomra fókuszálnak [Magyar et al., 2009], de jelen tanulmány keretei nem teszik lehetővé, hogy ezeket is részletesen tárgyaljuk. E felosztás abban az értelemben nem egzakt, hogy vannak olyan konkrét megvalósított szoftverrendszerek, melyeket nem lehet csak egy altípusba sorolni; továbbá bizonyos szakirodalmak megemlítenek még egyéb (kevésbé fontos) altípusokat is. A következő fejezetben két olyan CMS-altípust vizsgálunk meg, ami a közigazgatásban előfordulhat: a WCMS és a DMS, mivel mindkettő szöveges típusú tartalmakkal, dokumentumokkal foglalkozik. E két altípus közel áll egymáshoz; a fejlődésük során a dokumentummenedzsment-rendszerek is egyre jobban hasonlítanak a WCMS-rendszerekhez, mivel változásuk webes irányba indult el. A következő fejezetben nem is teszünk különbséget e két altípus között, hanem közösen tárgyaljuk őket, elsősorban a felhasználók által írt dokumentumokra koncentrálva.

3. FELHASZNÁLÓK ÁLTAL ÍRT DOKUMENTUMOK A CMS-BEN

Ebben a fejezetben a dokumentumkezelésre, a tartalomkezelő rendszereknél a felhasználók által írt dokumentumokra fókuszálunk. Megvizsgáljuk a dokumentumok rendszerezési lehetőségeit metaadatok alapján; bemutatjuk a felhasználók kezelését, valamint a dokumentumok szerkesztését az életciklusuk alatt; végül kitérünk a tartalomkezelő rendszerek karbantartására.

3.1 DOKUMENTUMOK RENDSZEREZÉSE METAADATOK ALAPJÁN

A közigazgatásban nagy jelentőséggel bír a dokumentumok tematikus vagy más szempontok szerinti rendszerezése, hiszen ez biztosítja az áttekinthetőséget, a struktúrát a dokumentumok nagy halmazában. Ebben a legfontosabb segítség a kategorizálás tud lenni, főleg, ha ezt a rendszer hierarchikus szinten támogatja egy-egy kategórián belül alkategóriák hozzárendelésének lehetőségével. A hierarchikus támogatásnál legtöbb esetben nem várható el a kötelező besorolás, azaz ilyenkor a kategória megadása kötelező, de az alkategória opcionális, és további rendszerezésre ad lehetőséget.

A rendszerezést segítő metaadat a címke, ami a dokumentumok tartalmát tudja jól jellemezni egy (vagy néhány) szóval [Xiao et al., 2007]. Sok olyan rendszer van, ahol lehetőség van minden dokumentumhoz több címkét is hozzárendelni. Ez – ha jól használják, és valóban a dokumentumok tartalmi kulcsszavai szerepelnek közöttük – több szempontból is hasznos adat: segít a tartalom szerinti további strukturálásban, megkönnyíti a tartalom szerinti keresést, sőt egy adott dokumentumhoz így lehetőséget biztosít témában hasonlóak ajánlására is. Helyes használata a jó megválasztásuk mellett azt is megköveteli, hogy azonos kulcstartalom esetében azonos nyelvtani formában használjuk őket, hiszen ez teszi lehetővé a pontos kapcsolatok felismerését különböző dokumentumok tartalma között. Ezt elősegítheti a címkék felvitele során az automatikus felajánlás funkció használata, amely kilistázza a már begépelt karakterekkel kezdődő, a rendszerben már meglévő címkéket.

A rendszerezést tovább segíti a címkefelhő, mely a rendszerben leggyakrabban előforduló címkék listája. Érdemes a rendszerben megvalósítani ilyen címkefelhőt, mert érdekessége abban rejlik, hogy a listában egy adott címke az előfordulási száma szerinti méretben jelenik meg, azaz minél gyakrabban fordul elő, annál nagyobb betűmérettel kerül feltüntetésre, és minél ritkábban, annál kisebbel. A teljes rendszerre kiterjedő felhő szűkíthető egy kategória vagy alkategória kiválasztásával; ilyenkor csak az ezekben publikált tartalmak címkéi jelennek meg a felhőben. Ezenkívül szabályozható a megjelenő elemek száma, amely azt jelenti, hogy a népszerűségi lista tetejéről hány elem kerüljön bele a listába. Ilyenkor mindig a bekerülő elemek közül a legnépszerűbb lesz a legnagyobb betűvel írva és a legkevésbé népszerű a legkisebbel, tehát a megjelenítés a listában megjelenő elemek között relatív, és nem abszolút kapcsolatot mutat.

A lehetséges metaadatok közül általában kötelező a szerző megadása (közigazgatásban természetesen lehetnek szerző nélküli hivatalos iratok is) – aki lehet a rendszer egyik felhasználója (alapesetben a dokumentumot

létrehozó vagy feltöltő személy nevét írják be ilyenkor szerzőként ehhez a metaadathoz) –, azonban a dokumentumot feltöltő felhasználó és a szerző lehet különböző is. Bár a szerző – alapértelmezés szerint – az, aki felviszi a dokumentumot a rendszerbe, ez jó esetben módosítható, és egy listából kiválasztható a rendszer felhasználói közül a tényleges szerző is. Fontos tulajdonság lehet még, hogy a dokumentum olvasói számára publikus-e a dokumentum szerzője.

Egy másik opcionálisan megadható metaadat a forrás, ami egy tetszőleges szöveges adat és egy hozzá tartozó e-mail-cím vagy internetes link kombinációjából állhat. Amennyiben egy dokumentumot publikussá szeretnének tenni, akkor meg kell adni egy dátumot, hogy ez mikortól legyen nyilvános. A publikálás dátuma tehát egy másik metaadat, amely lehet a pillanatnyi idő, ha azt szeretnék elérni, hogy a dokumentum a valós idő szerinti időpontban legyen publikálva, illetve lehet egy jövőbeli dátum (időpont) is, ha csak egy jövőbeli pillanatban szeretnék azt publikussá tenni (így az nem lesz elérhető egészen a megjelölt időpontig, amikor a dokumentumot a rendszer publikálja a megadott publikálási dátummal). További metaadatként automatikusan kerül elmentésre minden dokumentumhoz a felvitelének és az utolsó módosításnak az időpontjai is, valamint mindkét eseményhez az ezt végző felhasználó is.

Ha a metaadatokat nézzük, akkor közigazgatási specialitás az olyan jellegű pluszinformáció, ami egy dokumentumhoz kapcsolódó anyagokra utal (azaz a metaadat egy olyan link ez esetben, mely más dokumentumra mutat). Ez a közigazgatásban azért fordul elő gyakran, mert egy ügylet általában több dokumentumon keresztül ível át, és ezeket össze kell kapcsolni. Legjobb az lenne, ha ezek összerendelhetők lennének tartalom alapján teljesen automatikusan is, azonban a tévesen egymáshoz rendelt dokumentumok (amikor a címkék vagy konkrét tartalom alapján hasonlónak vélt, de valóságban mégis különböző dokumentumok kerülnének össze) elkerülése végett félautomatikus megoldást érdemes használni. Azaz lehetőséget kell biztosítani kapcsolódó dokumentumok hozzárendelésére a rendszerben úgy, hogy a hozzárendelés során a rendszer lehetőséget biztosítana az általa hasonlónak ítéltek közüli manuális választásra vagy ettől független továbbiak keresés útján történő kiválasztására.

3.2 FELHASZNÁLÓK REGISZTRÁCIÓJA, KEZELÉSE

3.2.1 Felhasználók regisztrációja

Egy WCMS-be való regisztrációnál – gyakran kötelező jelleggel – a felhasználói név, e-mail-cím és a kívánt jelszó megadására van szükség, vagy pedig közösségimédia- fiókkal (például facebook, Linkedin stb.) lehetséges a regisztráció. A felhasználói névnek és az e-mail-címnek a rendszeren belül minden felhasználónál egyedinek kell lennie, így foglaltság esetén a rendszer erre figyelmeztet; ilyenkor újat kell választani. A jelszót általában kétszer, egyformán kell megadni, hogy elkerülhetők legyenek az esetleges félregépelésekből eredő hibák. Biztonsági szempontból a regisztrációs oldalon még egy grafikus ellenőrzőkódot, az úgynevezett captchát [Gossweiler et al., 2009] szokták kérni, amelynél helyes begépelés szükséges a sikeres regisztrációhoz. Ez egy képként megjelenített, erősen torzított karaktersorozat, amelynek felismerése emberek számára könnyű, de egy gép számára nem triviális probléma, így elkerülhető a robotok regisztrálása a rendszerbe. Ha a kép mégis túl nehéznek tűnik, az alatta lévő link segítségével új kép kérhető. Utolsó kötelező elemként az oldal felhasználási feltételeinek az elfogadását visszaigazoló jelölőmezőt kell a legtöbb rendszernél kitölteni, ezután már sikeresen befejeződik a regisztráció. Ha valamelyik kötelező mező nem került kitöltésre, a rendszer figyelmeztet erre.

Néhány további, nem kötelező adat is megadható, például a teljes név, a születés éve vagy a felhasználó országa; ez utóbbit egy minden országot tartalmazó legördülő listából szokták felajánlani gyakran úgy, hogy a regisztráló IT-eszköz IP-címének megfelelő országot kiemelik (az IP-cím ellenőrzése, különösen közigazgatási alkalmazásban, amúgy is biztonsági előnyökkel jár). Magyarországon érdemes a lista tetejére tenni a saját országunk nevét, mivel várhatóan ez lesz a leggyakrabban kiválasztva, és ez nagyban megkönnyítheti a használatot (ezt követően már lehet az országokat betűrendben elhelyezni). Az ország megadásán kívül még a város neve is megadható. A felhasználói élmény növelése érdekében az ország kiválasztása után (esetünkben Magyarország kiválasztása esetén) egy olyan legördülő listát érdemes létrehozni, amely tartalmazza az adott ország összes települését betűrendben. A lista tetejére – itt is egyszerűsítő jelleggel – a fővárost (nálunk Budapestet) érdemes tenni, majd ezt követve a többi helységet. Magyarországon kívüli ország kiválasztása esetén – hogy ne kelljen a világ összes városát beletenni – a város megadása a városnév kézi begépelésével történhet.

További biztonsági megfontolásként, az adatok elküldését követően, a regisztrációról a rendszer e-mailt szokott küldeni a megadott címre, amely tartalmaz egy egyszer használatos egyedi linket, amellyel a regisztráció élesíthető: ezt nevezzük aktiválásnak. Amíg ez nem történik meg, az új felhasználóval nem lehet belépni a rendszerbe. Ennek elvégzésére limitált idő (legtöbbször a regisztrációtól számított 24 óra) áll rendelkezésre. Az időkereten belül nem aktivált regisztrációk automatikusan törlődnek a rendszerből. Ezzel a módszerrel egyrészt ellenőrizhető az e-mail-cím valódisága, másrészt az a tény, hogy a felhasználó a saját e-mail-címét adta-e meg.

3.2.2 Felhasználói jelszó kezelése, titkosítása

Sikeres regisztrációt, majd aktiválást követően lehetőség nyílik a rendszerbe történő belépésre. Ezt a felhasználói név és a jelszó, vagy az e-mail-cím és jelszó párosok egyikének sikeres és helyes megadásával lehet megtenni. Megadható, hogy a rendszer jegyezze meg a felhasználót, ilyenkor az adott gépen nincs szükség a későbbiekben a bejelentkezési adatok megadására, a rendszer automatikusan bejelentkezik a korábban megadott adatokkal. Néhány (ez a néhány egy viszonylag kis szám, legtöbbször 3) sikertelen belépési kísérletet követően a rendszer, biztonsági okokból, a következő alkalomtól a bejelentkezési adatok mellett egy ellenőrző kód – captcha – megadását is kéri, elkerülve ezzel a szisztematikusan próbálgató rosszindulatú programokat és robotokat.

Amennyiben a felhasználó elfelejti a jelszavát, lehetőség van annak megváltoztatására. Ehhez igényelnie kell a változtatást (ez legtöbbször a regisztrációs belépőoldal mellett vagy alatt található „elfelejtettem a jelszavam"-linkre való kattintással történik), majd a felhasználói név vagy e-mail-cím és egy captcha megadását követően a rendszer egy levelet küld az e-mail-címre, amely az aktivációhoz hasonlóan egy egyszer használatos egyedi linket tartalmaz a jelszóváltoztató oldalra, ahol a régi ismerete nélkül adható meg új jelszó. Ez a link is a kérvénytől számított limitált ideig (alapesetben 24 órán keresztül) érvényes, utána törlésre kerül. Az új jelszó megadásáig, függetlenül attól, hogy kért-e jelszóváltoztató linket a felhasználó vagy sem, a régi jelszó van érvényben. Így, ha közben a felhasználónak mégis eszébe jut a jelszó, csak figyelmen kívül kell hagynia a levelet; a link a megadott időkeret lejárata után automatikusan érvénytelen lesz. Sikeres belépést követően a rendszer arra az oldalra irányítja vissza a felhasználót, ahol rákattintott a belépés linkre.

Nemcsak az elfelejtett jelszó esetén van lehetőség annak megváltoztatására, hanem a bejelentkezést követően, a felhasználói profiloldalon is. Számos rendszer időszakosan kényszeríti is a felhasználót jelszócserére – mintegy

üzemeltetői elvárásként –, melynek vannak biztonsági előnyei. A közigazgatási alkalmazásokban fontos lehet ezt a gyakorlatot alkalmazni. Sőt itt a felhasználói név kivételével bármely, a regisztráció során megadott adat módosítható, a jelszót is beleértve. A jelszó módosításához először a régi jelszó megadására van szükség, majd kétszer azonosan kell begépelni az újat, a módosítás ezután lehet sikeres. Az e-mail-cím módosítása esetén az újonnan megadott e-mail-címre a rendszer egy aktivációs levelet küld, amiben egy (szintén egyszer használatos) egyedi link segítségével az új e-mail-cím aktiválható. Az egyedi link érvényességi ideje limitált, és aktiválásig (vagy az aktiválás elmaradását követően) a régi cím marad életben. Ez továbbra is azért fontos, hogy a rendszer megbizonyosodjon arról, hogy a felhasználó saját és létező címet adott-e meg. A többi adat minden különösebb bonyolítás nélkül szabadon módosítható vagy törölhető.

A felhasználói adatok mellett a modernebb rendszerek rövid statisztikát is megjelenítenek az oldalon végzett tevékenységekről, úgymint: a felhasználó hányszor lépett már be, mennyi időt töltött összesen az oldalon, vagy hány hozzászólást tett a rendszerben. Ha a felhasználó úgy kívánja: a felhasználói profilban nyílik lehetőség a regisztráció törlésére a GDPR-rendeletnek megfelelően [European Parliament, GDPR, 2016]; ehhez a jelszó megadása és egy captcha kitöltése kötelező. A felhasználó, amennyiben szeretné jelezni, opcionálisan megadhatja a törlés okát, ez továbbításra kerül a rendszer adminisztrátorai felé. Az űrlap elküldéséhez – további biztonsági eszközként – a regisztráció törlése gomb megnyomása előtt egy jelölőnégyzetben jelölni kell, hogy a felhasználó biztos-e a műveletben; a törlésre csak ezután kerülhet sor. A rendszer ilyenkor minden adatot töröl a felhasználóról. A felhasználó hozzászólásai nem kerülnek törlésre – egy esetleges beszélgetés folytonosságának megtartása érdekében–, ilyenkor a legtöbb rendszer a felhasználói név helyett ettől az időponttól kezdve a „törölt felhasználó" jelzéssel jeleníti meg a hozzászóló nevét.

Biztonsági szempontból rendkívül fontos a bejelentkezési és egyéb jelszóbekérési esetekben, hogy a jelszó, amennyiben lehetőség van rá, ne kerüljön átküldésre a hálózaton egyszerű szöveg formájában. Erre alapvetően két eltérő megoldás létezik.

Az egyik megoldás a nyilvános kulcsú titkosítás használata. Ennek legegyszerűbb módja, ha a jelszó-kommunikációt igénylő oldalakat nem http-, hanem https- protokollon keresztül használjuk, így a teljes oldal kommunikációja a felhasználók számára átlátszó módon nyilvános kulcsú titkosítást fog használni. Ez biztonságos megoldás ugyan, de sajnos költséges is; a problémamentes használatához ugyanis szükség van egy vagy több kiszolgálói kulcstanúsítvány beszerzésére egy hivatalos hitelesítés szolgáltatótól.

A másik lehetőség, hogy a jelszót közvetlenül nem küldjük át a hálózaton. Ebben lehet segítségünkre a kihívás-válasz protokoll [Buttyán & Vajda, 2007], melynek működése a következő: a rendszer minden jelszóbekérő mező kiadása előtt létrehoz egy login session-t, amelyben eltárol két darab véletlen karaktersorozatot (random1 és random2). A felhasználó által beírt jelszó javascript segítségével kerül a kliensoldalon titkosításra SHA1(random1 + SHA1(jelszó) + random2) formában, ahol az SHA1 az azonos nevű hashfüggvény alkalmazását [Wang et al., 2005], a „+" pedig a karaktersorozatok konkatenációját jelöli. A szerver, miután ezt megkapta, a felhasználónév alapján lekérdezi az adatbázisból a tárolt SHA1(jelszó)-t, és a login session alapján, abból kiolvasva a két randomszámot, elvégzi a hashelést, majd összehasonlítja a két karaktersorozatot. Egyezésük esetén a jelszó helyes. A random karaktersorozat alkalmazására az algoritmusban azért van szükség, mert ha csak a jelszó hash-elt verzióját küldenénk át, akkor semmivel sem kötnénk azt az adott sessionhöz, így ha azt valaki lehallgatja, egy másik sessionben elküldve is be tudna jelentkezni, azaz szinte semmivel nem nyújtana nagyobb védelmet, mint egyszerű szövegként átküldve. A véletlen karaktersorozatok alkalmazásával ez a lehetőség megszűnik, hiszen az átküldött karaktersorozat

lehallgatása után – azt egy másik bejelentkezési session-ben elküldve a nem egyező véletlensorozatok miatt – a szerveroldalon generált karaktersorozattal sem fognak egyezni, így meghiúsul a kísérlet. Az algoritmus egyetlen gyengesége, hogy a kliensoldali titkosítás miatt felhasználói oldalról szükség van JavaScript-támogatás engedélyezésére. Mivel az interneten általános, hogy rengeteg szolgáltatás épül már erre, sőt sok szolgáltatást el sem lehet érni nélküle, a felhasználó érdeke is, hogy ezt engedélyezze a böngészőjében.

3.2.3 Jogosultságkezelés

A jogosultságkezelésnél [Benantar, 2006] különböző szerepköröket lehet definiálni. Az ezekhez tartozó személyek alkotják a felhasználói csoportokat. A szerepkör nem más, mint jogosultságok egy megadott halmaza, amelyek így egy közös egységként kezelhetők. Szerepkör hozzáadása esetén ebből kifolyólag csak egy név megadására van szükség, amelyet érdemes úgy megválasztani, hogy az képes legyen jól leírni az adott jogosultságcsokor által biztosított elvégezhető műveletek körét. A szerepkörök kezelése során csupán a név módosítására, vagy, amennyiben nincs az adott szerepkörrel rendelkező felhasználó a rendszerben, annak törlésére van lehetőség. Az egyes szerepkörökhöz rendelt jogosultságok halmaza a jogosultságmátrixban adható meg.

A jogosultságmátrix – ahogyan a neve is mutatja – egy táblázat, amelynek vízszintes soraiban a rendszerben lévő jogosultságok, oszlopaiban az egyes szerepkörök vannak. A táblázat minden cellájában egy jelölőnégyzet található, amelynek a segítségével egyszerűen megadható, hogy egy adott szerepkör rendelkezik-e egy adott joggal vagy sem. Értelemszerűen a jelölt állapot jelenti a belefoglalást, így a kezelőfelületen csak a táblázatot (és egy „mentés” gombot) kell megvalósítani; a módosítások könnyebb nyomonkövethetősége érdekében, szerkesztés közben, a még el nem mentett módosításokra egy többitől elütő színt érdemes alkalmazni.

Egy CMS-rendszer jogosultságkiosztásánál a következő szerepköröket szokták definiálni:

- Látogató: Az oldalon böngésző – nem azonosított, azaz bejelentkezés nélküli – nem regisztrált felhasználók tartoznak ebbe a csoportba. Ők a rendszer publikus tartalmaihoz és oldalaihoz férhetnek hozzá, valamint regisztrálhatnak a rendszerbe.
- Felhasználó: A rendszeren belül a regisztrált – felhasználónévvel és jelszóval sikeresen azonosított, bejelentkezett – látogatókat nevezzük felhasználónak. A publikus tartalmak böngészésén túl az egyes tartalmakhoz hozzászólásokat is fűzhetnek, illetve elérhetik esetlegesen azokat a funkciókat, amelyekhez kizárólag ennek a kiemelt csoportnak van joga.
- Szerző: Olyan felhasználó, akinek van joga új dokumentumot felvinni a rendszerbe, és természetesen a sajátjait módosítani is tudja.
- Szerkesztő: Olyan felhasználó, aki a szerzői jogosultságokon túl a következő jogokkal is rendelkezik: publikussá teheti a dokumentumokat; módosíthatja az egyes oldalakon megjelenő tartalmakat; kezelheti a menüket, a forrásokat; megtekintheti a statisztikákat.
- Adminisztrátor: A legmagasabb szintű felhasználó, a rendszerben lévő összes jogosultsággal rendelkezik. Kezelheti a felhasználókat; törölheti őket, módosíthatja a szerepkörüket. Kezelheti magukat a szerepköröket és a hozzájuk rendelt jogosultságokat; új szerepeket vihet fel a rendszerbe.

A továbbiakban az adminisztrátor funkcióit tekintjük át részletesen, hiszen a többi szerepkör mind levezethető ebből a legmagasabb jogosultsággal rendelkező szerepkörből oly módon, hogy a használható funkciók ennek részhalmazai.

Az adminisztrációs (néha csak rövidítve: admin) felület csak az adminisztrátorok számára érhető el, és ennek segítségével kezelhetők a rendszer felhasználói, adhatók ki jogok, hozhatók létre új szerepkörök. A felhasználók kezelése egy listatípusú felületen lehetséges. Itt a rendszer minden regisztrált felhasználója megjelenik az összes fontos, hozzájuk tartozó adattal együtt. A megjelenített adatok általában két részre oszthatók. Az egyik része (a jelszót kivéve) a felhasználó profiladatait, a másik pedig a felhasználó rendszeren belüli állapotát, szerepkörét és statisztikai adatait jeleníti meg. A felhasználó profiladatai itt csak informatív jelleggel jelennek meg, módosításukra ezen a felületen nincs lehetőség, azt csak a felhasználó teheti meg saját profiloldalán.

Statisztikai adatok közül a következőket szokták megvalósítani: a regisztráció időpontja, az esetleges utolsó kitiltás végének időpontja, a rendszerben eltöltött összes idő, a megjegyzések száma (ahonnan egy komplexebb statisztikát is tartalmazó rendszer esetében tovább lehet jutni az adott felhasználó megjegyzéseinek listájához), valamint egy indikátor, amely megjeleníti, hogy az adott felhasználó épp online van-e vagy sem. Ezeknek az informatív és statisztikai adatoknak a megtekintésén túl az adminfelületen általában megtekinthető még az adott felhasználó állapota és a hozzárendelt szerepkör (amely értékek módosíthatók is).

A felhasználói állapotnál a következő értékeket szokták definiálni: aktív, kizárt és törölt. Egy újonnan regisztrált felhasználó – aktiválás után – alapértelmezés szerint aktív állapotú, azaz szabadon bejelentkezhet a rendszerbe. Amennyiben viselkedése alapján az adminisztrátorok úgy ítélik meg, akkor a felhasználó kizárható a rendszerből, ami annyit jelent, hogy hozzáférési adataival bizonytalan ideig nem tud bejelentkezni a rendszerbe, továbbá azonos e-mail-címmel és névvel sem tud új regisztrációt kezdeményezni (mivel nem tud bejelentkezni, ezért a rendszer szempontjából egyszerű látogatóként tud csak viselkedni). Ha az adminisztrátorok úgy ítélik meg, akkor később visszaállíthatják aktív állapotúvá. A törölt állapot nem beállítható, akkor kerül ebbe az állapotba egy felhasználó, ha ő kezdeményezi törlését saját profiloldalán keresztül.

Fejlettebb rendszerek esetén a fenti műveletek (szerepkör és állapot módosítása) a listaszerkezetben párhuzamosan is, sőt – egyszerre több felhasználó kijelölése esetén – csoportos műveletként is elvégezhetők, amely így jelentősen meggyorsíthatja és leegyszerűsítheti az adminisztratív feladatok elvégzését, főleg, ha sok felhasználót kell egyszerre kezelni. A lista általában tovább kereshető és szűrhető (szűrésre például szerepkör és állapot szerint van lehetőség), így könnyebben megtalálható a keresett és kezelni kívánt felhasználó a felhasználói név, a teljes név és e-mail-cím alapján.

3.3 DOKUMENTUMOK SZERKESZTÉSE, KEZELÉSE

3.3.1 Dokumentumok életciklusa

Egy dokumentum az életciklusa [Kiedrowicz & Stanik, 2015] során különböző állapotokon mehet keresztül, ezalatt a tartalomkezelő rendszerekben a következő állapotokat veheti fel: piszkozat, megírt, publikált, visszavont. Az állapotoknak, több szempontot is figyelembe véve, megvan a jelentőségük. Egyrészt segítik a dokumentum életciklusának – és a vele végzett munkafolyamatnak – a követését a csoportmunka során, másrészt ettől függ, hogy egy adott dokumentum mely szereplők számára érhető el.

Egy dokumentum, megírásának kezdetén piszkozat állapotban van. Ennek csupán annyi a jelentősége, hogy jelzi a rendszer többi felhasználójának, hogy a dokumentum még nincs kész, a szerző még nem kívánja jelen állapotában publikálni. Fontos, hogy ebben az állapotban a rendszer még nem követeli meg a kötelező adatok és tulajdonságok kitöltését az elmentéshez. Ha a dokumentum elkészült, és minden kötelező mező kitöltésre került, az állapot átállítható megírtra. A szerző ezzel jelzi, hogy a dokumentuma, véleménye szerint, kész a publikálásra. A dokumentumot ezután lehet publikálni, ekkor kerül át publikált állapotba. Bizonyos rendszereknél az egyszerű és gyors munkavégzés kedvéért, piszkozat állapotú dokumentum is kerülhet közvetlenül publikálásra, de csak akkor, ha minden kötelező mező kitöltésre került. Egy dokumentum állapota publikálás után már nem állítható vissza megírt állapotba, de ha mégis módosítani, javítani szeretné a felhasználó, akkor lehetősége van piszkozat állapotba tenni. Ez a fajta átjárás lehetőséget ad a szerkesztők és a szerzők közötti csoportmunka hatékonyabbá és követhetőbbé tételére.

Ha a szerző valamilyen okból meggondolta magát a publikálást illetően, a dokumentum állapota átállítható visszavontra is. Ilyenkor a dokumentum kikerül a publikus oldalakról, listákból, a rámutató link hibát fog visszaadni. Ennek az állapotnak a bevezetése is a munkafolyamat követhetőségének érdekében történik, mivel ebből az állapotból már csak újrapublikálható a dokumentum; más állapotba nem tehető. A visszavont állapot a dokumentum tulajdonságainak módosíthatósága szempontjából is kiemelt fontosságú (metaadatai nem módosíthatók). A visszavont állapot után lehetőség van végleges törlésre is.

3.3.2 Dokumentumok írása

A dokumentumszerkesztő felületet a legtöbb esetben két részre osztják: az első részében a dokumentum tulajdonságai (metaadatai) adhatók meg. Itt található továbbá a törléséhez szükséges gomb is, amelynek megnyomása esetén a rendszer felugró ablakban kér visszaigazolást, megerősítést a műveletre. A második részben a szövegszerkesztő-felület található gépelési és szerkesztési lehetőségekkel, azaz a felhasználó itt írhatja meg a dokumentum szövegét. Ennél a felületnél a legelterjedtebb (és egyben a felhasználó számára a legkényelmesebb is) a WYSIWYG-típusú (what you see is what you get [Guo et al., 2011]) editor, azaz ez teszi lehetővé a szöveg olyan jellegű formázását, hogy a publikált kinézet megegyezik a szerkesztőben látható nézettel.

Kapcsolódó dokumentumok megadását is meg szokták valósítani valamilyen felugró ablak segítségével, amelyben két listát jelenítenek meg. Az egyik lista tartalmazza az adott dokumentumhoz hozzárendelt, azaz kapcsolódó dokumentumokat, amely kezdetben üres. A másik lista pedig egy szűrhető és kereshető dokumentumlista azzal a funkcióval kiegészítve, hogy egy speciális szűrő bekapcsolásával lehetőség van a rendszer által automatikusan, címkék alapján hasonlónak ítélt dokumentumok listázására is. A második lista találati eredményei közül kell kiválasztani és áthúzni (az első listába) azokat az elemeket, melyeket a felhasználó kapcsolódó anyagként szeretne feltüntetni. Ezek jellemzően a dokumentumok hivatkozási részében felsorolt más dokumentumok, de lehet egyéb jellegű kapcsolódó anyag is, mely a közigazgatásban hasznos lehet. Ebben a listában az egyes elemek sorrendje szabályozható, az oldalon a megjelenés az itt megadott sorrendet fogja követni.

Ahogyan említettük, a dokumentumszerkesztő második részében található – gépelési lehetőségekkel – a szövegszerkesztő-felület, valamint a dokumentum szövege itt formázható és vihető fel formázottan a rendszerbe. Ennek kezelőfelülete a szövegszerkesztőkből megszokott eszközkészletet idézi, és az ott általános formázási lehetőségeket nyújtja.

Egy CMS-rendszer esetében a mentés némileg eltér a sima szövegszerkesztőkétől, mivel két opcióból választhat a felhasználó. Az egyik a sima mentés, amely a beállított állapottal felviszi a dokumentumot a rendszerbe, ez piszkozat vagy megírt lehet; a másik pedig a publikálás, mely egy felugró visszaigazolás-kérést követően, mentéssel együtt, publikálja is a dokumentumot. Ennek a két lehetőségnek egy már publikált dokumentum módosításánál van nagy jelentősége, hiszen itt ez a két művelet különbözteti meg azt, hogy csupán menti a felhasználó a módosítást, de még a régi marad publikálva, vagy rögtön nyilvánosságra hozza a módosításokat is. A szerkesztőfelület az összes elemre automatikus mentést is meg tud valósítani, amellyel a háttérben észrevétlenül menteni tudja a módosításokat, így hiba esetén elkerülhető az adatvesztés. Az automatikus mentés természetesen sosem publikál, csak sima mentést hajt végre.

A rendszerben könnyen előfordulhat olyan eset, amikor két felhasználó véletlenül azonos tartalmat nyit meg szerkesztésre, vagy valaki kitörölhet olyan tartalmat, amit valaki éppen szerkeszt. Ez nem kívánt következményekkel járhat, ezért a rendszernek erre fel kell készülnie. Ennek megfelelően az ilyen rendszerekben konkurenciakezelés kerül megvalósításra, melynek működése rendkívül egyszerű. Ha egy felhasználó megnyit egy tartalmat szerkesztésre, akkor az adott tartalmat zárolja. Ilyenkor más azt nem szerkesztheti vagy törölheti. Ha befejezte a szerkesztést, és kilép a felületről, a zárolás feloldódik, felszabadítva ezzel a tartalmat mások számára.

Azonban ezzel újabb problémák merülhetnek fel. Ha egy felhasználó írás közben véletlenül nyitva felejti a felületet egy tartalom számára, akkor más felhasználók végtelenségig várhatnak. Ez olyankor jelenthet gondot, ha a zároló felhasználó nem elérhető, másvalakinek viszont sürgősen módosítania kell valamit. Ezt a problémát úgy szokták megoldani, hogy a rendszerben bevezetik a zárolási időtartamot, ami azt jelenti, hogy – egy felhasználó szerkesztési oldalon tett utolsó tevékenységétől számítva – adott idő (néhány perc) után a zárolás automatikusan megszűnik, felszabadítva a tartalmat mások számára.

3.3.3 Webes dokumentumok és kiegészítők

Webes dokumentumoknál a WCMS-oldal alapvető struktúráját az oldal sablonja határozza meg. A sablon osztja fel az oldalt és jelöl ki adott szélességű helyeket az oldalon belül, valamint megadhatja az egyes részek stílusát is. A sablon által kijelölt, adott szélességű mezők a blokkok. Ide azok a – blokkok szélességének megfelelő – dobozok kerülhetnek, melyekben majd a különböző dinamikus tartalmi elemek láthatók. Egy új oldal létrehozásakor a két legfontosabb dolog, amit meg kell adni, az oldal neve, valamint az, hogy melyik sablonra épül. Egy oldalhoz csak egy sablon rendelhető hozzá; az oldal dinamikus tartalmának szerkezete ezen belül adható meg. Az oldal neve általában az oldal tartalmára jellemző, ez jelenik meg a menükben, vagy az oldalra történő hivatkozásoknál, esetleg sablontól függően az oldalon magán is. Emellett meg kell adni az oldal linkjét is, amely a teljes linkben az oldalt azonosító rész. Ennek értéke alapértelmezés szerint az oldal nevéből jön létre, de természetesen ez átírható tetszőleges értékre. A felvitt oldalak alapértelmezés szerint engedélyezve vannak, azaz publikusak és elérhetőek mindenki számára. Ez az állapot átállítható, ekkor az adott oldal letiltásra kerül, és az oldalt felkereső személyek nem fogják azt látni.

Egy adott blokk alapvető megjelenése blokksablonok segítségével adható meg. A sablon tartalmazza a háttérszínt, esetleges keretet, a benne megjelenő tartalom egyes szövegelemeinek formátumát; és mivel ezek nem hordoznak magukban méretinformációkat, ezért a sablon megtervezésénél figyelni kell az esetleges pixelgrafikus

elemekre, és fel kell készülni a tág mérethatárok közötti helyes és esztétikus megjelenésre. Minden blokknak megadható egy cím, amely általában a doboz tetején jelenik meg. Opcionálisan linket is rendelhetünk hozzá, ekkor a doboz fejléce kattintható lesz és a megadott linkre mutat. A blokk alján, ugyancsak opcióként, feltüntethető egy további tartalmakra mutató tetszőleges link, egyedi címszöveggel. Szövegdokumentumot tartalmazó blokk esetén beállítható, hogy a címen kívül mi jelenjen még meg. Bekapcsolható, hogy megjelenjen-e a dokumentum publikálásának dátuma és a kategóriája (ez utóbbiak helye is szabályozható), és állítható a kapcsolódó dokumentum megjelenítése is.

Az oldal és a blokk hierarchikus szerkezetű, de bizonyos esetekben függetleníthetők is egymástól. Ilyenkor a blokk mentésekor meg kell adni, hogy véglegesként szeretné-e a szerző menteni. Ez egy kétállapotú mutató, amelytől függően egy adott blokk megjelenik – vagy nem jelenik meg – egy oldalon, ha oda kitesszük. A két állapotban rejlő funkcionalitás az, hogy a függetlenség miatt egy félkész blokkot is el lehet menteni a rendszerbe anélkül, hogy attól kelljen tartani, hogy ez valahol egy oldalra véletlenül kikerül. Természetesen, ha egy oldalon már kint lévő blokkot módosítanak, akkor ez azt jelenti, hogy amíg nem kerül véglegesítésre, addig a módosítást megelőző verzió látható az oldalon mialatt a módosításokat a rendszer elmenti.

A dokumentumokhoz sokféle kiegészítő lehetőség kínálkozik. A teljesség igénye nélkül bemutatunk néhányat, úgymint a megjegyzések írása, metaadatok hozzárendelése, dokumentumok megosztása, nyomtatása.

A dokumentumokhoz – amennyiben ez ki lett alakítva a rendszerben, és a dokumentum paraméterei között engedélyezve lett – a felhasználóknak lehetőségük van megjegyzéseket írni. A legtöbb esetben ez csak bejelentkezett felhasználók számára lehetséges, látogatók csak megjegyzések megtekintésére jogosultak. Általában ezeket úgy valósítják meg, hogy minden egyes megjegyzés szövege mellett megjelenik annak a felhasználónak a neve, aki írta, a felvitel időpontja, valamint egy válasz gomb, aminek segítségével reagálhat a felhasználó, és szöveges választ adhat az eredeti megjegyzésre. Ezek a megjegyzések és válaszok egyszerű, formázásmentes szövegek, melyek nem tartalmaznak sem linkeket, sem formázási paramétereket.

A felhasználónak a dokumentum szerkesztése végén a fontosabb kiegészítőket, metaadatokat is meg kell adnia, amelyeknél az alábbi instrukciókra érdemes figyelni:

- Cím: a cím önmagában egy egyszerű formázás nélküli szöveges tartalom, amelynek külön sablon alapján definiálható a stílusa (ez lehet egyedi formázási stílus, de lehet egy egységes is; főleg a közigazgatásban fontos, hogy az egységes megjelenítés miatt ugyanazt a sablont használják). A cím hossza a rendszerbe beépített értékeknek megfelelően korlátozott. A szerkesztők munkájának elősegítése érdekében a modernebb CMS-rendszerek visszafelé pörgő karakterszámláló segítségével mutatják, hogy hány karakter beírására van még lehetőség.
- Publikálás időpontja: a publikálás időpontjának beállítására egyszerű dátumot, órát és percet beállító listákon (pl. legördülő menük segítségével) van lehetőség. Amennyiben viszont a felhasználó rögtön szeretné a dokumentumot publikálni, lehetőség van ennek a gyorsítására is egy „most” gombbal, amely beállítja az időpontot az adott pillanatra.
- Címke: a korábban már használt címkék gyakori újrafelhasználásának ösztönzésére, és az azonos jelentésű, de eltérő nyelvtani alakú szavak elkerülésére a rendszerek a címkék felvitelét automatikus kiegészítéssel segítik. Ez a funkció – gépelés közben – egy legördülő dobozban megjeleníti a begépelt karakterekkel megegyezően kezdődő címkéket, ahonnan ezek közül könnyen tud a felhasználó választani.

Az elkészült dokumentumokat meg lehet osztani, ki lehet nyomtatni. Dokumentumok ajánlására több lehetőséget is alkalmaznak: az első ilyen, amikor egy kis felugró ablak segítségével a felhasználónak lehetősége nyílik a konkrét dokumentum linkjének elküldésére e-mailben. Ehhez meg kell adni a címzett e-mail-címét, valamint opcióként egy rövid üzenet is írható mellé. A másik lehetőség a dokumentum megosztása a népszerű közösségi portálok valamelyikén. Ebben az esetben először egy listáról ki kell választani a megfelelő portált, majd az oldal átirányítja a felhasználót a kiválasztott oldal megosztó képernyőjére, ahol a tartalomra vonatkozó mezők már automatikus kitöltésre kerülnek.

Egy dokumentum felhasználásának további módja, amikor a felhasználó papíralapon szeretné azt olvasni/használni (közigazgatásban gyakori végcél, hogy az elektronikusan szerkesztett dokumentumot kinyomtatják, majd aláírásokkal hitelesítik). Ehhez külön nyomtatási funkciót rendelnek, legtöbbször nyomtatási nézettel együtt. Azért kell ennek külön funkciónak lennie, hogy ne kelljen a teljes oldal minden grafikai elemét kinyomtatni a dokumentum szövegével együtt, hanem szépen rendezve csak egy kis fejléccel kerüljön mindez nyomtatásra. Ennek a nézetnek a megtekintését teszi lehetővé például a nyomtatás-vezérlőelem (azaz az említett nyomtatási nézet). Sőt azt is meg tudják valósítani, hogy egy nyomtatás gombbal indítva automatikusan, a dokumentum számára optimális beállításokkal küldje el azt nyomtatásra.

3.3.4 Dokumentumok kezelése szerkesztői jogkörrel

A dokumentumok kezelésére minden felhasználónak – a saját tartalmak esetében – joga van; egyedül a szerkesztői szerepkörrel rendelkező felhasználóknak (és persze az adminisztrátornak) van jogosultságuk az összes dokumentumhoz hozzáférni és kezelni egy szerkesztői felületen keresztül. A szerkesztői felület olyan része a tartalomkezelő rendszereknek, amely a tartalmak magasabb szintű kezelésével foglalkozik, hiszen a szerkesztők itt vihetnek fel dokumentumokat, rendezhetnek be weboldalakat, moderálhatnak megjegyzéseket, stb. A következőkben ezeket a funkciókat tekintjük át részletesen.

A rendszerben lévő dokumentumok kezelésére szolgáló felületnél egy listában jelennek meg a dokumentumok legfontosabb adatai: a címe; a létrehozásának, utolsó módosításának és esetlegesen a publikálásnak az időpontja; a megjegyzéseinek száma; a kategóriája, az alkategóriája; a szerző és az utolsó módosítást végző felhasználó. Általában a listában található sorok elején (ritkábban a sor végén) találhatók a dokumentum szerkesztésére és megnyitására lehetőséget adó kezelőszervek. A dokumentum törlésére csak a szerkesztőfelületen adnak lehetőséget arra ösztönözve a felhasználókat, hogy alaposabban ellenőrizzék, mit törölnek, mielőtt ténylegesen törlik azt; a véletlen törlések így nagyobb számban kerülhetők el.

A szerkesztői felületen látható lista szűrhető és kereshető oly módon, hogy a keresőmezőbe beírt szó alapján a dokumentumok címében és metaadataiban kereshet a szerkesztő. A lista szűrésére a kategória, alkategória, a szerző és a dokumentumok állapota szerint, illetve ezek tetszőleges kombinációjával van lehetőség. A lista lehetőséget biztosít egyszerre több kijelölt dokumentum esetében is a kategóriájának, alkategóriájának módosítására akár párhuzamosan is. Továbbá a felületen látható lista esetében általában lehetőség van dátum szerinti sorrendezésre is úgy, hogy a legfrissebbek legyenek legelöl.

A szerkesztő szerepkörhöz tartozik a kategóriák szerkesztési joga is. Egy speciális felületet szoktak erre létrehozni, ahol kezelhetők a rendszerben a kategóriák és alkategóriák, és ez két egymás melletti listából áll. Az első listában a kiválasztható kategóriák szerepelnek; a második lista kezdetben üres, ám egy kategória kiválasztásával

az alá tartozó alkategóriák jelennek meg benne. Kategória vagy alkategória felvitelére neve alapján kerülhet sor. Szofisztikáltabb rendszerekben további adatként megadható, hogy egy adott kategória vagy alkategória használata engedélyezett-e a rendszerben. Letiltott állapot esetén ugyanis egy (al)kategória tartalma természetesen továbbra is elérhető, de új dokumentumot nem lehet ilyen állapotú (al)kategóriában publikálni.

A dokumentumokhoz fűzhető megjegyzések két típusát különböztethetjük meg: az egyik a dokumentum egészéhez csatolt megjegyzés, a másik pedig a dokumentum adott pontjába (adott dokumentumrészhez) beszúrt megjegyzés. Létre szoktak hozni megjegyzések kezelésére szolgáló felületet, amely egy szűrhető és kereshető lista: ebben általában időrendben jelennek meg a rendszerbe került megjegyzések a dokumentum egészéhez csatolt megjegyzések esetén, továbbá oldalszám szerinti sorrendben a második típusú megjegyzések esetén. A megjegyzésekről – bizonyos rendszereknél – akár táblázatformátumú információk is kérhetők, melyek tartalmazzák a megjegyzés szövegét; a felhasználót, aki felvitte; a felvitel időpontját; a dokumentum címét, amelyhez tartozik; a dokumentum kategóriáját és a megjegyzés státuszát. Jobb rendszereknél a megjegyzés szövegében lehetőség van keresni; találatokat szűrni a dokumentum, a kategóriája, a felhasználó, és a megjegyzés státusza szerint. A státusz három értéket vehet fel: új, elfogadva és elutasítva. Egy megjegyzés, a felvitelét követően, mindig az új állapotot veszi fel. A rendszerben megjegyzésszerkesztési joggal rendelkező felhasználók a listafelület segítségével, ezekből az állapotokból kiindulva, elfogadhatják vagy elutasíthatják az adott megjegyzést. Több megjegyzés együttes kijelölése esetén állapotuk egyszerre is állítható, így sok megjegyzés esetén jelentősen felgyorsítható ez a munka.

Amennyiben a szerkesztői szerepkörrel rendelkező felhasználó egyben szerző is, azaz a saját maga által írt és szerkesztett dokumentumokat is kezelnie kell (közigazgatásban ez könnyen előfordul), akkor érdemes egy olyan felhasználói felületet biztosítani számára, amely megkönnyíti a navigálást. Az ilyen jogosultsággal rendelkező felhasználók számára – a listajellegű szerkesztői felületeken túl – a rendszernek tehát, megfelelő módon, támogatnia kell az olvasói oldalakon is a kezelőszervek (mint például a dokumentum szerkesztésére vonatkozó és a megjegyzés moderálására vonatkozó kezelőszerv) megjelenítését. Így amennyiben egy felhasználónak joga van egy dokumentum szerkesztésére, a szerkesztési oldalra mutató link a dokumentum olvasói oldalán is megjelenik számára. Ez jelentős egyszerűsítés, hiszen így egy esetlegesen itt megtalált hiba esetén a felhasználónak nem kell elnavigálnia a dokumentumokat kilistázó oldalra, és ott újra megkeresni a cikket, hanem csak egyszerűen a szerkesztés linkre kattintva máris a cikkszerkesztőben találhatja magát. Előnyös, ha a rendszer ezt az egyszerűsítő funkciót még egy helyen, a megjegyzések moderálásánál is támogatja. Ha egy felhasználó rendelkezik az ehhez szükséges joggal, minden egyes megjegyzés mellett megjelenik az adott megjegyzés döntéséhez szükséges kezelő. Ilyenkor a rendszer az adott oldalról nem is navigálja el a felhasználót, csak értesíti a művelet sikerességéről, majd ennek az eredményének a megmutatásához (hogy az új megjegyzés elfogadott lett-e vagy visszautasított) frissíti az adott oldalt.

Kiegészítésként egy speciális dokumentumtípust, az űrlapot érdemes részletesebben is megvizsgálni, mivel a közigazgatási rendszerekben sűrűn előfordulnak. Az ezek kezelésére szolgáló felület, a korábbiakhoz hasonlóan, a legtöbbször szintén kereshető és szűrhető listaszerkezetű. Itt a szokásos paramétereken kívül (ki és mikor publikálta, stb.) olyan segítő információkat is megjelenítenek, hogy hányan töltötték már ki, illetve ki töltötte ki utoljára és mikor. A lista a legtöbb esetben kereshető, melynek során a megadott szót a rendszer az űrlapok nevei között keresi, továbbá a találatok további szűrése is lehetséges az űrlap típusa szerint. Az egyes sorok elején egy kezelőgombot szoktak elhelyezni, mely az adott űrlap szerkesztőfelületére visz. Ha a megvalósításkor a biztonságra szeretnének törekedni, akkor ez esetben az űrlap csak a szerkesztőfelületen belülről lenne törölhető, így ösztönözve a felhasználót, hogy pontosan ellenőrizze le, milyen űrlapot töröl, mielőtt ténylegesen törli azt.

Egy űrlap szerkesztőfelülete általában több részből áll: van egy rész, amely az űrlapok közös tulajdonságainak szerkesztésére szolgál; egy másik az űrlap előnézeti képét tartalmazza; a harmadik pedig a tartalmi elemek kiválasztását segítő és szabályozó felület. Az első két vezérlő rész, általában egymáshoz közel (egymás mellett, például a szerkesztőfelület tetején) található. A közös tulajdonságok az erre szolgáló felületen állíthatók be az adatok típusának megfelelő mezők segítségével. Ide érdemes tenni a törléshez szükséges gombot is, amelynek megnyomása esetén a rendszer felugró ablakban kér visszaigazolást a műveletről. Az előnézeti kép – az űrlap beállított tulajdonságai, valamint hozzárendelt tartalma alapján – a rendszer aktuális adatainak megfelelő, valódi kinézetét mutatja (ahogyan az egy oldalon megjelenne). A tartalmat szabályozó rész a harmadik felület (mely legtöbb esetben a fenti két vezérlő alatt helyezkedik el). Mivel a tartalmak függenek az űrlap mezőtípusaitól (pl. szövegmező, rádiógombok, jelölőnégyzet, legördülőlista-típusú mező), így ez a rész minden űrlap esetében különbözik.

3.3.5 Keresés a dokumentumokban

Egy tartalomkezelő rendszer egyik legfontosabb funkciója a keresés, ezért általában a keresőoldalon kívül is, minden oldal tetején, a fejlécben megtalálható egy egyszerű szöveges keresés indítására lehetőséget adó keresőblokk (ami a tartalomkezelő rendszer saját, beépített kereső funkciója); egy innen indított keresés is természetesen a keresőoldalra viszi a felhasználót, ott jeleníti meg az eredményeket. Részletes keresés indítására a keresőoldalon nyílik lehetőség. A keresés itt is a keresőszó megadásával történik, de számos beállítással bővíthető vagy szűkíthető a találati lista. A keresés – alapértelmezés szerint – a rendszerben lévő összes dokumentum címében, összefoglalójában, szövegében történik (illetve lehet a címkék között is, ahogy ezt lentebb még részletesebben kifejtjük).

Fejlettebb tartalomkezelő rendszerek esetében lehetőség van a találatok bővítésére a „töredékszavakra történő keresés" bekapcsolásával. Ez azt jelenti, hogy nemcsak azok a találatok jelennek meg, amelyekben a keresőszó pontosan a beírt alakban fordul elő, hanem azok is, amelyekben a keresőszó vagy annak valamely bővített, toldalékolt alakja jelenik meg. Például ha a keresőszó az „önkormányzat", akkor azok a dokumentumok is megjelennek találatként, amelyek csak az „önkormányzatokat" szót tartalmazzák, az eredeti keresőszót nem. Tovább bővíthetők az eredmények, ha a felhasználó bekapcsolja, hogy a dokumentumokon kívül, az azokhoz tett megjegyzések szövegében is keresni szeretne.

A találatok szűkítéséhez szűrni lehet egy adott időintervallumra (mikor készült el a dokumentum) vagy szerkesztő nevére. Ilyenkor a találatok közül csak azok jelennek meg, amelyekre igazak a beállított szűrési feltételek. A részletes keresőben – további lehetőségként – keresőszó megadása helyett keresni lehet címkére is; ilyenkor a megadott címkékkel rendelkező dokumentumok jelennek meg a találatok között. Természetesen ez a lista is szűrhető még a fentebb leírt feltételek segítségével. A keresés elméleti részéről és a dokumentumhalmazban való gyakorlati alkalmazásáról részletesen szó lesz tanulmányunk 6. fejezetében.

3.4 CMS KARBANTARTÁSA DOKUMENTUMKEZELÉS SZEMPONTJÁBÓL

Egy CMS-portál, működését tekintve, két állapotban lehet: karbantartási vagy normál üzemmódban [Souer et al., 2007]. A karbantartási módban az oldalra látogató felhasználók egy erre figyelmeztető oldalon találják magukat; a portálra belépni, továbbá tartalmakat elérni nem tudnak. A belépés oldalon csak és kizárólag adminisztrátorok

tudnak bejelentkezni; ők ezután a normál üzem szerinti felületre kerülnek. A karbantartási mód azért fontos, mert lehetőséget biztosít például a portált működtető fájlok cseréjére, frissítésére anélkül, hogy egy esetleges köztes állapotból eredő hiba látható lenne a külvilág számára; mindeközben tájékoztatja az állapotról az oldalra érkező felhasználókat. Karbantartásra sokféle okból lehet szükség: ilyen például a működés optimalizálása, a memóriatakarékosság, naplózás, hibakezelés, gyorsítótár bevezetése, duplikációk kiszűrése; az alábbiakban ezeket vesszük górcső alá.

Az optimális működéshez első körben törekedni kell az adatok és tartalmak megfelelő kezelésére. A rendszerben ez több mindenre is vonatkozik, a legfontosabb talán a várható terhelésre vonatkozó szempont. A portál esetében ez azt jelenti, hogy az adatbázisban lévő adatok szerkezetét elsődlegesen úgy kell optimalizálni, hogy az olvasói felület a lehető leggyorsabban és legnagyobb hatásfokkal érje el a megfelelő tartalmakat. Ez azért fontos, mert egy CMS esetében a terhelés szinte egésze ezen a felületen keletkezik (kevés felhasználónak van joga az adminisztrációs felületeket is használni). Egy másik, a funkcionális szempont, miszerint törekedni kell arra, hogy egy adott funkció használatakor a lehető legkevesebb adatbázistáblát kelljen párhuzamosan használni. Ez azért fontos, mert csökkenti a lekérdezések bonyolultságát, ezzel párhuzamosan az igényelt teljesítményüket és a lefutási idejüket.

A memóriatakarékosság szempontjából az eltárolt eredmények törlésére is van lehetőség. A kapcsolat bontása után ezeket lehetne manuálisan is törölni, de ehelyett az automatikus funkciót szokták megvalósítani, amikor a következő lekérdezés során automatikusan törlik az eltárolt eredményeket, így szabadítva fel helyet a memóriában. A lekérdezések eredményének kezelésére is több lehetőség adódik. Például az eredményhalmaz soronkénti lekérésére sorban egymás után, vagy egy adott indextől; sőt, ha szükséges, az összes eredménysor is visszakapható egyszerre.

Az adatbázisréteg az adatbázison végzett minden lekérdezést és minden vele kapcsolatos eseményt naplóz; ennek a támogatása az összes metódusba be van építve. A napló több különböző típusú bejegyzést tartalmazhat, amelyek a szerveren egy kívülről nem elérhető fájlba kerülnek. Kezelhetőbb méretű naplókat úgy lehet megvalósítani, hogy minden nap új fájlt kezd el írni a rendszer, így kisebb méretű fájlok keletkeznek.

A gyorsítótár (cache) használatának elsődleges célja az oldal és a tartalmak betöltődésének gyorsítása, valamint a szerveroldalon fellépő terhelés csökkentése; ez a két cél szorosan összefügg. A lassú betöltődést elsősorban a sok adatbázis-művelet eredményezi, illetve ez is okozza a legnagyobb terhelést a szerveren, így a cél elsősorban az adatbázis-műveletek számának csökkentése. Ezt a legkönnyebben úgy lehet megoldani, ha a gyakran látogatott lapok és megtekintett tartalmak bizonyos időközönként a szerveren előre összeállításra kerülnek egy html-fájlba, így a látogatók általi lekéréskor már nincs szükség adatbázis-művelet végrehajtására. Ennek a műveletnek a megvalósítása egy további problémát vet fel. A kérdés, hogy mi történjen abban az időintervallumban, amikor a rendszer éppen frissíti a gyorsítótárban lévő html-fájlt, mert ekkor előfordulhat, hogy valaki a frissítés közepén egy még nem teljesen elkészült fájlt kér le. Lehetne egy adatbázisban tárolt változó segítségével állítgatni minden frissítés előtt és után, hogy használható-e a cache. Ezzel az a probléma, hogy így rendszeresen előfordulna, hogy bizonyos kéréseket nem a gyorsítótárból szolgál ki a rendszer; így jelentősen romlik a teljesítmény. Ennek elkerülésére megoldást jelenthet két cache váltott használata, azaz amíg az egyikbe írunk, addig a másikból olvasunk. Itt további kérdés, hogy mi alapján döntsünk arról, hogy melyik fájl az, amit éppen használnunk kell. Erre elsőként kézenfekvő megoldást nyújt a fájl dátuma: amelyik frissebb, azt kell használni; mindezt egy központi helyen kell tárolni, hogy mikor, melyik gyorsítótár használható. Mivel normális működés során a tartalmak a gyorsítótárból futnak, egy esetleges adatbázisszerver kiesés esetén is, frissülés nélkül ugyan, de teljesítménycsökkenés nélkül tud a portál tovább üzemelni, így a cache a gyorsításon kívül a rendszer várható rendelkezésre állását is hatékonyan növeli.

A rengeteg dokumentum között könnyen előfordulhat, hogy az egyik duplán (két különböző helyen, de ugyanolyan tartalommal) jelenik meg a rendszerben, sőt, kettő helyett akár többször is szerepelhet ugyanaz a tartalom. Ennek kiküszöbölésére egy duplikációk kiszűrését végző metódust kell megvalósítani. A módszer lehet teljesen automatikus, de legtöbb esetben csak duplikációt jelző megoldást alkalmaznak, mely alapján a szerkesztő már könnyen eltávolíthatja a kevésbé fontos helyről a másodpéldányt.

A továbbiakban – kicsit eltávolodva a tartalomkezelő rendszerek működésétől – ezúttal a tartalomra, azon belül is a szöveges dokumentumokra fókuszálunk. A következő fejezetekben bemutatjuk, hogy hogyan vihető végig az előkészítő lépésektől a végeredményig az ilyen típusú tartalmak elemzése.

4. SZÖVEGFELDOLGOZÁS

Ebben a fejezetben bemutatjuk a szövegelemzési feladatok előkészítő lépéseit (szövegfeldolgozást), és azt az utat, ahogy a folytonos, nem strukturált szövegből a számítógépek által is értelmezhető strukturált reprezentáció készül, valamint felvillantjuk az elemzési feladatok körét is.

4.1 SZÖVEGREPREZENTÁCIÓ-ÉPÍTÉS

A következőkben részletes kifejtjük, hogy az elemzési feladatokhoz milyen előfeldolgozási lépéseket kell megtenni a szöveggel kapcsolatban. A tanulmányunkban bemutatni kívánt elemzési feladattípusok (osztályozás, csoportosítás, információkinyerés, információ-visszakeresés, összegzéskészítés) a szövegbányászati szakirodalomhoz [Tikk, 2007] tartoznak, és ezekkel az 5. fejezetben foglalkozunk részletesen. Azonban ahhoz, hogy az elemzést el lehessen végezni, a számítógépeknek tudniuk kell a szöveget megfelelő formátumú adatként kezelni; azaz a szövegdokumentumok reprezentációját kell előállítanunk.

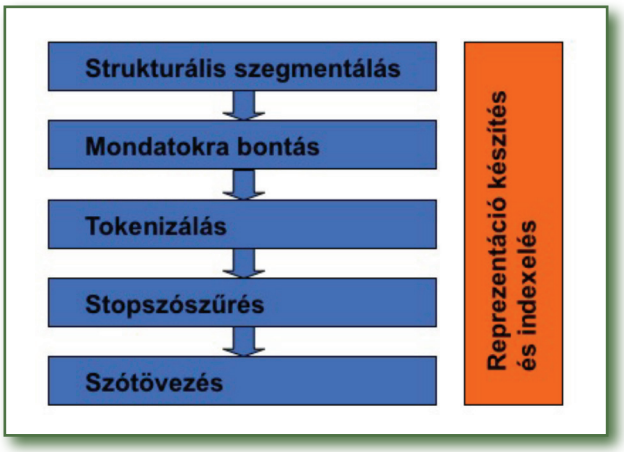
Az (informatikai értelemben vett) szövegelemzés [Tikk, 2007] elvégzéséhez szükséges tehát egy előfeldolgozási (más néven előkészítési) fázis, ahol a számítógépes reprezentációkészítés és indexelés folyik. Indexelés alatt azt értjük, hogy megjelöljük azokat a szemantikus építőelemeket, melyek a szövegdokumentumot alkotják, és mindezzel együtt eltároljuk az építőelemek helyét a dokumentumon belül.

A legegyszerűbb reprezentációs megoldás a szózsákmodell [Salton & McGill, 1983]. Ez a dokumentumokat szavak halmazának tekinti, ahol az elemek többszörösen is szerepelhetnek (ellentétben a hagyományos halmazzal, ahol az elemek csak egyszer szerepelhetnek). Azonban a szózsákmodell egy alacsony szintű modell. A dokumentumok számítógépes reprezentálásának három, széles körben elterjedt magasabb szintű megközelítése modellje van [Baeza-Yates & Ribeiro-Neto, 1999]: halmazelméleti, algebrai és valószínűségi (valószínűségelméleti) modell. A halmazelméleti modellnél halmazműveleteket használnak. Az algebrai modellnél a dokumentumokat algebrai objektumokként (vektor vagy mátrix) ábrázolják [Berry et al., 1999], algebrai műveletekkel hasonlítják őket össze (a legelterjedtebb ilyen modell: a vektortérmodell [Chowdhury, 2010], melyről még a 6. fejezetben részletesen szó lesz). A valószínűségi modellek pedig a dokumentumokat valószínűségi eseményként kezelik, és a hasonlóságot feltételes valószínűségi becslésként határozzák meg.

Az előkészítés (azaz a reprezentációkészítés és indexelés) során a folytonos szöveget kisebb részekre kell feldarabolni (a 2. ábrán látható módon), mely az összes lépést mutatja: strukturális szegmentálás, mondatokra bontás, tokenizálás. A feldarabolás után az előkészületi műveletek tovább folytatódnak (stopszószűréssel, szótövezéssel), ezekről a későbbiek során még részletesen szó lesz. Az indexelés során a feldarabolt rész helyét is eltároljuk, és ezzel a legvégén előáll a szöveg reprezentációja.

Strukturális szegmentálás folyamán a szöveget strukturális egységekre (kötet, rész, fejezet, szakasz, bekezdés stb.) kell bontani. Ez a felbontás erősen függ a dokumentumok típusától; más kell könyvekhez, más újságcikkekhez, oktatási segédletekhez, és más egyedi megoldások szükségesek a közigazgatásban használt dokumentumokhoz.

Felmerülhet a kérdés, hogy a felbontás mennyire automatizálható. Az egyedi típusok miatt sajnos nehezen, de természetesen minden típusra ez külön megoldható. Így főleg a felhasználónak kell megmondania, hogy adott dokumentumtípusnál milyen strukturális egységek értelmezhetők (ehhez mindenképpen emberi munka szükséges). Miután ezeket egy szakértő definiálta, az egységhatárok felismerésére már kisebb programok, eljárások írhatók.



2. ábra. Előfeldolgozási lépések szövegelemzéshez

Forrás: a szerzők saját szerkesztése

Az előkészítés következő lépése a mondatokra bontás, ami már viszonylag jól automatizálható. Mondathatárokat a mondatzáró írásjelek segítségével lehet meghatározni. Vannak persze kivételek: dátum, sorszám, URL, e-mail-címek; ezeknél olyat is mondathatárnak állítanak be, ami valójában nem az. Itt szabályalapú megoldásokkal lehet javítani a felismerésen.

Tokenizálásnál tisztázni kell, hogy mit értünk token alatt: egy karaktersorozat konkrét dokumentumbeli előfordulását. Például a következő mondat: „lenni vagy nem lenni" 4 tokent tartalmaz, ahol a „lenni" kétszer is előfordul, míg a többi token csak egyszer. Az azonos karaktersorozatot tartalmazó tokenek osztályát típusnak hívjuk, és a típusok összessége alkotja a nyers szótárt (előző példában 3 elemet tartalmaz a szótár: „lenni", „vagy", „nem"). A dokumentumot tehát tokenek sorozatára kell bontani, ami azért nem triviális, mert néhány karaktersorozat esetén nehéz eldönteni, hogy hol húzzuk meg a token határát. Például az 1848. március 15. tekinthető-e 1 tokennek? Ráadásul a tokenizálás nyelvfüggő is, hiszen a magyarban nincs aposztróf egy szón belül, francia nyelv esetében pedig van. De nemcsak az aposztrófos szavaknál nehéz a döntés, hanem példának okért a kötőjeles szavaknál is.

Az eddig bemutatott három lépéssel párhuzamosan elkezdődhet az indexelés, azaz a tokentípusok (általánosabb értelemben vett szavak) szempontjából történő ábrázolás elkészítése. Ebben a folyamatban a szavakhoz információkat gyűjthetünk: melyik dokumentumban szerepel, annak melyik strukturális egységében, stb. A különböző reprezentációs megoldások más-más információkat rendelhetnek a szavakhoz. A tokenizálás végére általában előáll egy invertált állomány (reprezentáció), mely persze a későbbi lépések folyamán még módosulhat. A strukturális szegmentálás, mondatokra bontás, tokenizálás három lépéséből a legfontosabb a harmadik, hiszen ez minden esetben szükséges, míg az előző két lépés a feladattól függően akár el is hagyható. A legtöbb alkalmazási példánál tehát tokenizálással kezdődnek az előfeldolgozási lépések, ezért ezt a fázist – külön kiemelve – később egy alfejezetben tárgyaljuk.

A következő lépés a stopszó (angolul: stop words, vagy magyarul még: töltelékszó, tiltott szó) szűrés. Ennek lényege a leggyakrabban használt szavak eltávolítása a nyers szótárból, elhagyásuk ugyanis csökkenti a tárolás méretét (a szótár elemszáma kisebb lesz). Ez az elhagyás az osztályozást, csoportosítást nem befolyásolja, ezért az elemzéses feladatoknál ezt meg lehet tenni; befolyásolná viszont a keresést, ezért a kereséses feladatoknál nincs stopszószűrés.

Utolsó lépésként még a szótövezést (Sharma, 2012) szokták elvégezni az elemzési feladatok előtt, hogy jelentősen csökkentsék a modellméretet. Míg nyelvészetben a szó lemmájának (normalizált vagy szótári alak) előállítása a cél – ahol mindig értelmes szóalakot kapnak –, addig az informatikában a szavak szótövének előállítása a feladat, amely néha (köznapi értelemben) értelmetlen eredményt adhat.

A stopszószűrés és szótövezés két olyan fontos lépése az előkészítésnek, ami később az erősen befolyásolhatja, hogy az elemzés mennyire jó, ezért egy későbbi alfejezetben visszatérünk e két lépés részletezésére.

4.2 TOKENIZÁLÁS

Ahhoz, hogy a tokenizálást tetszőleges dokumentum esetében végre lehessen hajtani (függetlenül attól, hogy a dokumentum egy sima szövegfájl, vagy éppen PDF, akár egy weblap), a dokumentumokat tisztítani kell, hogy egységesen ugyanolyan típusú tiszta szöveg jöjjön létre. Ennek megfelelően a nem sima szövegfájlok esetén a beolvasás után el kell távolítani a felesleges, tartalomhoz nem kapcsolódó elemeket (főleg a weblapoknál van sok ilyen, ott a HTML-címkéket kell kiszűrni), amelyek a dokumentumok indexelésénél problémát jelenthetnének. Ezután – a tisztítási procedúrát folytatva – a rendszer a szavakat kisbetűssé alakítja, hogy az azonos szavak ugyanazon címke alatt legyenek, függetlenül attól, hogy kis- vagy nagybetűvel kezdődnek. Erre azért van szükség, mert különben a rendszer külön kezelné az azonos kifejezéseket, ami problémát okozhatna a későbbi lépések során.

A tokenizálás – mint szövegfeldolgozási lépés – során a program a dokumentumot úgynevezett tokenek sorozatára bontja. A token nem más, mint egy karaktersorozat konkrét előfordulása a dokumentumban, amely többnyire egy-egy szót jelent. A tokenizálás látszólag egyszerű feladat, hiszen a szavakat általában szóközök határolják, és ezt felhasználva a szavak listája könnyen előállítható. Ez a megközelítés azonban részben figyelmen kívül hagyja a gyakorlatban jelentkező problémákat, mivel előfordulhatnak olyan kifejezések, amelyek ugyan tartalmaznak szóközöket, mégsem szabad, vagy célszerű felbontani (például New York, Szabó János). Amennyiben az alkalmazás az ilyen kifejezések precíz kezelését is megkívánja, célszerű előre eldönteni, hogy melyek azok a kifejezések, amelyeket tovább szabad bontani, és melyek azok, amelyeket nem. A fentiek megvalósításához általában kifinomult eljárásokra van szükség, ezért néhány rendszer a tokenizálás problémáját leegyszerűsíti, és a szóközt határolójelnek tekinti. Ez ugyan elméletileg kisebb pontatlanságokat eredményezhet, az elemzés végeredményét azonban számottevően nem befolyásolja. A tokenizálás kimenete a dokumentum szövegében előforduló kifejezések listája. Ez a lista még redundáns, hiszen egy adott szó különböző előfordulásait, valamint felesleges kifejezéseket is tartalmazhat. A redundancia megszüntetése a következő előfeldolgozási lépés feladata.

4.3 STOPSZÓSZŰRÉS ÉS SZÓTÖVEZÉS

A stopszószűrés egy olyan előfeldolgozási lépés, mely a tokenizálás után, de még a szótövezés előtt végezhető el. Stopszavaknak a gyakran előforduló tartalmi információt nem hordozó kifejezéseket (névelők: a, az; kötőszavak: és) tekintjük. A stopszavak szinte minden szövegben előfordulnak, így elhagyásukkal a szótárméret csökkenthető, és a stopszószűrésnek a vektortérmodell dimenziószámának csökkentésében is van szerepe. A szűrést a program egy szöveges állományban eltárolt általános stopszólista alapján végzi. A legtöbb esetben az adott nyelvhez tartozó lista egy manuálisan jól megírt válogatott szavak listája, amelyek a korpuszokban a leggyakrabban előfordulnak. Ebbe a listába beleveszik azokat a szavakat is, amelyek a vizsgált korpusz minden dokumentumában előfordulnak (az ilyen szavaknak még az információ visszakeresésben sincsen jelentősége). Ezek a szavak nem diszkriminatívak; ha elhagyjuk őket, akkor az így kialakuló szócsökkentés nem befolyásolja negatív irányban a későbbi elemzéseket. Míg a legtöbb nyelvhez tartozó stopszólista megtalálható az interneten, addig vannak olyan módszerek is, mely az adott korpuszhoz tartozó stopszavak listáját automatikusan generálja. A stopszólista elkészítésének nagyon egyszerű módja a szó dokumentumfrekvenciáján alapul (a dokumentumok száma, amelyben a szó megjelenik), ahol az egész korpuszon gyakran megjelenő szavak kezelhetők stopszavakként. Pontosabban: ha egy adott határértéknél magasabb a szónak a dokumentumfrekvenciája, akkor azt a módszer stopszónak állítja be.

A szótövezés célja – amint azt elnevezése is sugallja – a szavak szótövének meghatározása; az esetek többségében ez az eljárás nem értelmes szótári alakot ad eredményül, ellentétben az úgynevezett lemmatizálással. Míg nyelvészetben: a szó lemmájának (normalizált vagy szótári alak) előállítása a cél, ahol mindig értelmes szóalakot kapnak, addig az informatikában a szavak szótövének előállítása az elsődleges szempont, amely néha értelmetlen eredményt adhat (A szótövezésnek speciális, primitív módja a csonkolás, amikor csupán a szóvégződését vágjuk le. Ennél rögtön látható, hogy az eredmény legtöbbször nem egy értelmes szó (például „tavak" szóból csonkolt „tav" karaktersorozat; de a komplexebb szótövezésnél is kaphatunk hasonlóan nem értelmes, informatikai értelemben vett szótövet). A szótövezés két szempontból is nagy jelentőséggel bír. Egyrészt a stopszószűréshez hasonlóan dimenziószám-csökkenést eredményez a vektortérmodellben (több szóalakhoz szótövezés után már csak egy szóalak tartozik), amelynek köszönhetően a szótár memória- és tárigénye is csekélyebb lesz. Másrészt azért is hasznos, mert az azonos jelentéskörben használt, különböző toldalékokkal (ragokkal, képzőkkel stb.) ellátott szavakat a modell nem különbözteti meg, csak az úgynevezett kanonikus alakjukkal dolgozik tovább. Ez persze bizonyos fokú szemantikai veszteséget okoz, ami azonban sok esetben nem jár hátrányos következményekkel, sőt, gyakran kimondottan előnyös. Több módszercsalád is van a szótövezési feladat megoldására:

- az algoritmikus módszerek nyelvspecifikus átírószabályok alkalmazásával dolgoznak,
- a szótármódszerek a szavakat és szótövüket tartalmazó szótárakat használják,
- a statisztikai módszerek pedig az adott nyelv statisztikai sajátosságainak kihasználásával működnek.

5. SZÖVEGELEMZÉS

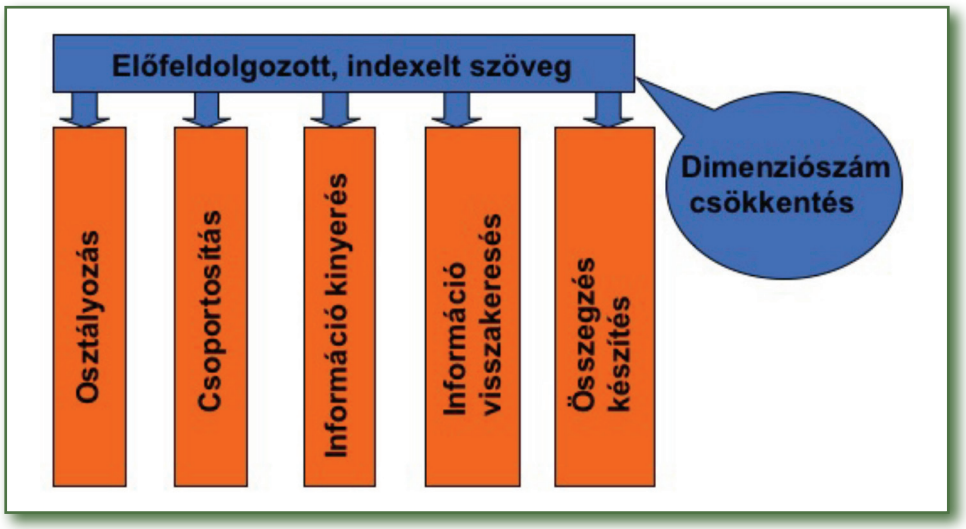
Ebben a fejezetben mutatjuk be az előkészítő lépések után előálló szövegreprezentáción elvégezhető elemzési feladatok körét. Míg az előző fejezet a dokumentumok leírására született megoldásokat foglalja össze a kis egységekre bontástól kezdve a szavak átalakításáig, addig ez a fejezet azt mutatja be, hogy ezekből az építőkövekből hogyan lehet különböző célok érdekében elemzési feladatokat megoldani.

5.1 SZÖVEGELEMZÉSI FELADATTÍPUSOK

A szövegelemzési feladatok tipizálása kétféleképpen is történhet, melyeket a szakirodalom két nagy irányvonala, a természetes nyelvű feldolgozás és a szövegbányászati technikák területein belül szokták bemutatni.

Az első a természetes nyelvi feldolgozás (angolul: Natural Language Processing, röviden NLP) szakirodalmához tartozik, ahol háromféle szintet különböztetnek meg: a morfológiai, szintaktikai és szemantikus elemzést. A morfológiai esetben az alapszavak (például tulajdonnevek) meghatározása a cél; a szintaktikus elemzésnél a nyelvtani szerkezetek (például névszói csoport) azonosítása a feladat; a szemantikai elemzés során pedig a szavak, mondatok jelentését kell meghatározni körülírással [Cambria & White, 2014].

Az elemzési feladatok másik tipizálását a szövegbányászat szakirodalma adja; jelen tanulmányunkban ezt az irányvonalat követjük. Az előző fejezetben bemutattuk az előfeldolgozási lépéseket. Az előfeldolgozott, indexelt szövegen sokféle elemzési feladatot lehet végrehajtani [Tikk, 2007]; ezek a következő nagy csoportokba sorolhatók (3. ábra): osztályozás, csoportosítás (más néven klaszterezés), információkinyerés, információ-visszakeresés és összegzéskészítés. A továbbiakban ezeket a feladattípusokat vesszük sorba.



3. ábra. Szövegelemzési feladattípusok

Forrás: a szerzők saját szerkesztése

Az osztályozás lényege, hogy előre definiált csoportokba kell az elemeket (szövegdokumentumokat) besorolni. Osztályozáson belül megkülönböztethetünk egycímkés (single-label) és többcímkés (multi-label) osztályozást. Az első esetén minden dokumentumot pontosan egy kategóriához kell rendelni, míg többcímkés osztályozásnál, minden egyes dokumentum esetében, az összes előre definiált kategóriára vonatkozóan meg kell adni, hogy beletartozik-e, avagy sem. Az osztályozásnál a kategóriák függetlensége szerint megkülönböztetünk egyszerű osztályozást, melynél a kategóriák függetlenek, valamint hierarchikusat, melynél a kategóriák egymásba ágyazott rendszert alkotnak (ekkor a gyerekkategóriába eső dokumentumok a szülő kategóriájába is beletartoznak). Továbbá az osztályozás során a vezéreltség alapján megkülönböztetünk dokumentumvezérelt osztályozást (document-pivoted categorization: DPC), ahol egy adott dokumentumhoz keressük az összes hozzá tartozó kategóriát, illetve kategóriavezérelt osztályozást (category-pivoted categorization: CPC), ahol egy adott kategóriához keressük az összes beletartozó dokumentumot. Az osztályozásnál érdemes két olyan konkrét feladatot is bemutatni, amely a közigazgatási rendszerek esetében kiemelten hasznos lehet; ezekre a következő alfejezetekben részletesebben kitérünk.

A klaszterezés feladata az előre nem definiált csoportokba való besorolás; azaz csoportokat kell képezni. Ez a felügyelet nélküli gépi tanulómódszerekhez tartozik, amikor nincs tanítóhalmaz. Célja, hogy úgy kell az elemeket (szövegdokumentumokat) klaszterekbe sorolni, hasonlóság alapján, hogy az azonos csoportba kerülő elemek egymáshoz hasonlóak, míg külön klaszterbe került elemek egymástól különbözőek legyenek. Klaszterezéssel ma már nagyon sok alkalmazási területen foglalkoznak. Az internet világában egyre nagyobb szerepet kap a felhasználók böngészési szokásainak elemzése, amely a különböző szolgáltatások fejlesztését jelentősen megkönnyíti a személyre szabottság kritériumának előtérbe helyezésével [Szűcs & Móczár, 2013]. Kulcsfontosságú a klaszterezés felhasználása a kommunikációs hálózatok és a közösségi rendszerek területén is. Az internetes keresőkben ugyancsak nagy segítséget nyújt a klaszterező eljárások alkalmazása, amelyek segítségével a hasonló találatok egy csoportba sorolhatók, így a felhasználó rövidebb idő alatt érheti el a számára releváns információkat tartalmazó weboldalakat, dokumentumokat. Több kiterjedt ügyfélkörrel rendelkező vállalatnál alkalmaznak klaszterező algoritmusokat az ügyfelek kategorizálására, amely segít a hatékony üzleti stratégiák kidolgozásában. A klaszterezés az előbbieken túl az élet számos más területén is nagy népszerűségnek örvend. A vegyészetben például a kémiai vegyületek szerkezet szerinti csoportosításához, az orvostudományban pedig a betegségek tünetegyütteseinek azonosításához használják [Iványi, 2005].

Az információkinyerés [McCallum, 2005] fontos szövegrészek kigyűjtését és strukturálatlan szövegekből strukturált információ előállítását jelenti. Először meg kell találni az adott feladat szempontjából a releváns részeket, majd ezeket strukturált módon – legtöbbször táblázatos formában – kell a felhasználó rendelkezésére bocsátani.

Az összegzéskészítés [Jing & McKeown, 1999] célja hosszú szövegrészek automatikus tartalmi összefoglalásának elkészítése. Ennek két típusa a kivonatolás és az összefoglalás-generálás. Az első esetében olyan összegzéskészítés a feladat, amely kizárólag eredeti szövegrészt tartalmaz. Az összefoglalás-generálás (abstraction) pedig egy olyan szintetizált összegzéskészítés, amely olyan szövegrészt is tartalmaz, ami nem volt benne az eredeti szövegben.

Információ-visszakeresés (information retrieval: IR (Baeza-Yates & Ribeiro-Neto, 1999]) fogalma alatt azt értjük, hogy egy felhasználónak ki kell elégíteni az információs igényét egy adott strukturálatlan dokumentumhalmazon. Azaz meg kell találni számára a kérdéséhez releváns dokumentumokat. A tartalomkezelő rendszerek esetén ez egy kiemelt funkció, így ezt a témát külön és részletesen a 6. fejezetben tárgyaljuk.

5.2 LEVELEK AUTOMATIKUS OSZTÁLYOZÁSA

Az emberi kommunikációnak ma a leggyakrabban használt eszköze az e-mail. A levelek automatikus osztályozása az üzleti szférában és a közszférában is hatékonyabbá teheti a munkát. A vállalati dolgozók levelezésében a levéláradat szétválogatása gyorsabb szervezeti reakciót eredményez, a közszférában pedig például automatikus ügyfélirányításra alkalmas. Ennek jelentősége tudatában mutatunk be egy lehetséges rendszert levelek automatikus osztályozására. Az előkészítéshez szükséges egy nagy korpusz (dokumentumhalmaz), amin majd tanulhat a rendszer; ez lehet például a letöltött levelek halmaza havi bontásban külön szövegfájlokban elmentve. Az előfeldolgozás során ebből a szövegfájlból kell elkészíteni a levelekben használt szavak szótárát és az invertált állományt (melyik szóból melyik levélben hány darab található), valamint a levelek metaadatait. A metaadatok tartalmazzák a levelek fejlécében található információkat, tehát a küldő, a téma, az azonosító (ID) és a reply-to-ID, amennyiben van.

Az előkészítési fázishoz a korábban bemutatott lépéseket kell megvalósítani, és a szótövezett szavak listája lesz a szavak szótára. Szótövező algoritmus megvalósításához magyar nyelven a Snowball- [Porter 2001] és a Tordai-féle szótövező (Tordai & De Rijke, 2005) jöhet szóba. Ez utóbbinak több verziója (Light1, Light2, Medium, Heavy) is van attól függően, hogy mennyire erősen vágja vissza a szavak végét (a toldalékokat); továbbá figyelembe veszi a szótőjelölt hosszát, és azt is, hogy tartalmaz-e érvényes mássalhangzó-magánhangzó kombinációt. Magát a szótövezést az alábbi 9 lépésben végzi el:

1. eszközhatározó esetrag törlése;
2. gyakori esetragok törlése;
3. speciális esetragok törlése;
4. további esetragok törlése;
5. transzlatívusz esetrag törlése;
6. birtokrag törlése;
7. egyes számú birtokragok törlése;
8. többes számú birtokragok törlése;
9. többes szám törlése.

A szövegelőkészítés, majd a levelek megfelelő reprezentációjának (jelen esetben algebrai modelljének) megvalósítása után következik a tanulási fázis, ahol az osztályozáshoz egy tanulómodellt kell használni. Az SVM-modell (Support Vector Machines [Boser et al., 1992]) teljesen megfelelő a célra, széles körűen elterjedt, és szövegek osztályozására több helyen is használják (bináris osztályozásra képes, de több SVM-modellt kombinálva többosztályos problémára is alkalmazzák). A Boser, Guyon és Vapnik által javasolt alapváltozata lineáris szeparálásra képes (azaz a tanulási fázis alatt megtanulja, hogy milyen szeparáló hipersíkkal lehet szétválasztani a bináris osztályozásnál a két osztályba tartozó pontokat a reprezentációs térben), de egy kiterjesztett változata nemlineáris szeparálásra is lehetőséget nyújt. Lineárisan szeparálható esetben végtelen sok lineárisan szeparáló hipersík létezik, amelyek mindegyike hibátlanul szétválasztja a tanítópontokat. A szeparálás minősége azonban az egyes megoldások esetén jelentősen eltérő lehet. Az az elválasztás eredményez jobb általánosító képességet, amely a két osztályba tartozó tanítópontok között, a tanítópontoktól a lehető legnagyobb távolságra helyezkedik el. A lineáris bináris osztályozási feladat megoldását adó SVM ezt az „optimális" szeparátor síkot határozza meg. A pontok közt középre elhelyezett szeparáló felületet a tanítópontoktól egy margó, azaz egy biztonsági sáv választja el. A tanulás után már csak a szeparáló hipersíkot kell alkalmazni az ismeretlenek osztályozására, így a levelek automatikus besorolása megoldható.

Az ilyen implementált osztályozóval tehát a rendszer képes a levélüzeneteteket felbontani, a fejléc-információkat kinyerni. A legtöbb gondot azonban a folyamatosan fellépő karakterkódolások szembenállásai okozhatják, ha a levelek nincsenek egységesen kódolva. Ennek az akadálynak a leküzdéséhez például az ékezetes karakterek konvertálásával lehet eljutni. Bármilyen előre definiált osztályok (például spam és nem spam) osztályozási feladata megoldható, csak tanulóállományként meg kell adni minden osztályból elég sok mintát (elég hosszú szöveget), és az elkészült rendszerrel megoldható lesz a levelek automatikus osztályozása.

5.3 NYELVFELISMERÉS

Egy másik osztályozási probléma, ami gyakran előfordul: a nyelvfelismerés (más néven nyelvdetektálás), melynek alapvető feladata egy adott hanganyag vagy szöveg nyelvének meghatározása. Az előbbi legjellemzőbb alkalmazási területe elsősorban a beszédfelismerés, az utóbbié pedig a szövegbányászat (mi csak az utóbbi, nyelvdetektálással foglalkozunk). A detekciós algoritmusokkal szemben támasztott legfőbb elvárás pedig, hogy hibatűrő legyen, mivel a szövegek többsége (különösképpen az internetes oldalakon találhatóké) általában sok helyesírási, valamint gépelési hibát tartalmaz. Természetes követelmény továbbá, hogy minél több nyelv felismerését tegye lehetővé, ugyanakkor működjön hatékonyan.

Írott szövegek nyelvének megállapítására az egyik legelterjedtebb módszer az n-gram-alapú felismerés [Canvar & Trenkle, 1994]. Az n-gram nem más, mint a szöveg egy *n* hosszúságú karakterlánca. Egy szó n-gramjait úgy határozhatjuk meg, hogy a benne előforduló *n* szomszédos karakterből sorozatot képzünk az összes lehetséges módon (ha a szó rövidebb, mint n hosszúságú, akkor szóközökkel egészítjük ki *n* hosszúságúra).

Az n-gram-alapú nyelvdetektálás lényege a fentiek tükrében a következő. Az algoritmus minden nyelvre előállít egy úgynevezett nyelvi profilt, amely általában egy néhány tízezer karakterből álló célnyelvi szöveg n-gramjait, valamint azok gyakoriságát tartalmazza. Ezt követően rendezi az n-gramokat elfordulási gyakoriságuk szerint, majd ugyanezt a műveletsort elvégzi arra a dokumentumra is, amelynek a nyelvét meg szeretnénk határozni. Ezek után a nyelvi profilok és a dokumentum profiljának összehasonlítása következik, amihez az algoritmus az első körülbelül 300 leggyakoribb n-gramot veszi alapul (ezek általában nagyon pontosan jellemzik az élő nyelvek többségét). Ha egy adott n-gram mindkét profilban szerepel, akkor kiszámítja a rangsorban elfoglalt pozíciójuk különbségét, egyébként egy előre rögzített maximális értékkel kalkulál. Végül minden nyelvi profilra összeadja az egyes n-gramokra kapott eredményt, és a vizsgált dokumentumot ahhoz a nyelvhez sorolja, amelyik profiljára nézve ez az összeg a legkisebb (azaz a leghasonlóbbra dönt).

6. INFORMÁCIÓ-VISSZAKERESÉS

Ebben a fejezetben részletesen áttekintjük az információ-visszakeresés (angolul: information retrieval) kérdéskörét, nevezetesen a dokumentumok keresésére kifejlesztett számítógépes megoldásoknál használt matematikai modelleket, ennek az elméleti és gyakorlati oldalát. A szöveg- és a linkalapú rangsorolás után a felhasználók szempontjából részletesen bemutatjuk a keresési funkciókat a tartalomkezelő rendszerekben, kitérve az egyszerű és összetett kereséstől kezdve a felhasználói viselkedésalapú keresésig.

6.1 INFORMÁCIÓ-VISSZAKERESÉS MODELLJEI

Ahogyan azt az előző fejezetekben a szöveges dokumentumok elemzésénél és feldolgozásánál láttuk: a szövegbányászat területén alkalmazott módszerek használata a dokumentumok számítógép által értelmezhető reprezentációját igényli. Szöveges dokumentumok reprezentálására az elmúlt évtizedekben számos modellt dolgoztak ki, melyek közül a gyakorlatban a következő három megközelítés terjedt el a leginkább: az algebrai, a halmazelméleti és a valószínűségi modellek [Baeza-Yates & Ribeiro-Neto, 1999] – ezeket alább részletezzük:

- Az algebrai módszerek esetében a dokumentumokat vektorokkal vagy mátrixokkal reprezentálják, összehasonlításuk pedig algebrai műveletekkel történik. Nagy előnye ezen modelleknek, hogy lehetőséget kínálnak a dokumentumok rendszerezésére, ezért az ilyen algoritmusok gyakran dolgoznak vektortérmodellben leírt dokumentumokkal.
- A halmazelméleti megközelítés alapját a halmazműveletek adják, amelyek nagy segítséget nyújtanak többek között a keresőrendszereknél felmerülő feladatok megoldásánál. A legismertebb ilyen modell a Boole-modell, mely a felhasználó által beírt keresett szöveget Boole-kifejezésként várja, és elemi halmazműveletek („and", „or", „not") segítségével oldja meg a visszaadott találatok előállítását [Lashkari et al., 2009].
- A valószínűségi modellek esetében a dokumentumok egy-egy valószínűségi eseménynek feleltethetők meg, hasonlóságuk pedig valószínűségi becslésként számítható ki [Jones et al., 2000].

Az ismertetett módszerek alkalmazhatósága nagymértékben függ a felhasználási területtől. Míg az egyik modell kiválóan alkalmas lehet egy adott probléma megoldására, könnyen előfordulhat, hogy a másik egyáltalán nem használható ugyanarra a célra. Az előnyök egyesítésére is akad példa: a vektor- és a Boole-modell kombinálásával hozták létre például a kiterjesztett Boole-modellt [Salton et al., 1983]

Mivel a megközelítések közül a leghatékonyabb és legelterjedtebb az algebrai módszerhez tartozó vektortérmodell [Salton et al., 1975], ezért a következőkben ezt a modellt ismertetjük részletesen. A vektortérmodellben a dokumentumokat sokdimenziós vektorok reprezentálják, amelyek egy-egy pontnak felelnek meg a vektortérben, a tér minden egyes dimenziója pedig az egyes szavakat (szaknyelven szólva: terminusokat) reprezentálja (tehát a dimenziószám megegyezik a dokumentumokban előforduló különböző szavak számával). A dokumentumvektorok a dokumentumok egyedi szavaihoz rendelt súlyértékekből állnak, amelyek az egyes terminusok fontosságát

írják le. A keresőkifejezéshez kiszámított vektor és az ugyanabban a vektortérben található dokumentumvektorok között már számolhatók hasonlósági értékek, és ezen értékek alapján rangsorolhatók a dokumentumok, így a dokumentumlista csökkenő relevancia-sorrendben adható vissza a felhasználónak [Lee et al., 1997].

A vektortérmodell előnyei közé tartozik, hogy matematikailag könnyen kezelhető, valamint a dokumentumok hasonlósága a [0,1] intervallumban folytonos értékkel jellemezhető. E tulajdonságok miatt előszeretettel alkalmazzák többek között információ-visszakereső rendszerekben, mivel lehetőséget ad a részleges illeszkedés mérésére is. Hátránya viszont, hogy a dokumentum szavait egymástól szemantikailag függetlennek [Lewis, 1998] tekinti, hosszabb szövegek esetén pedig a tér dimenziószámának nagy értéke miatt a feldolgozás hatékonysága csökkenhet. A vektortérmodellnél súlyozásra több módszer is létezik, de a legjobb és egyben leggyakrabban használt a „tf-idf"-módszer [Soucy & Mineau, 2005], mely két fő szempontot vesz figyelembe a súlyok meghatározásánál, nevezetesen:

- a dokumentum szavai mennyire jellemzik jól az adott dokumentumosztályt;
- az adott dokumentumosztályt milyen mértékben különböztetik meg más témájú dokumentumoktól.

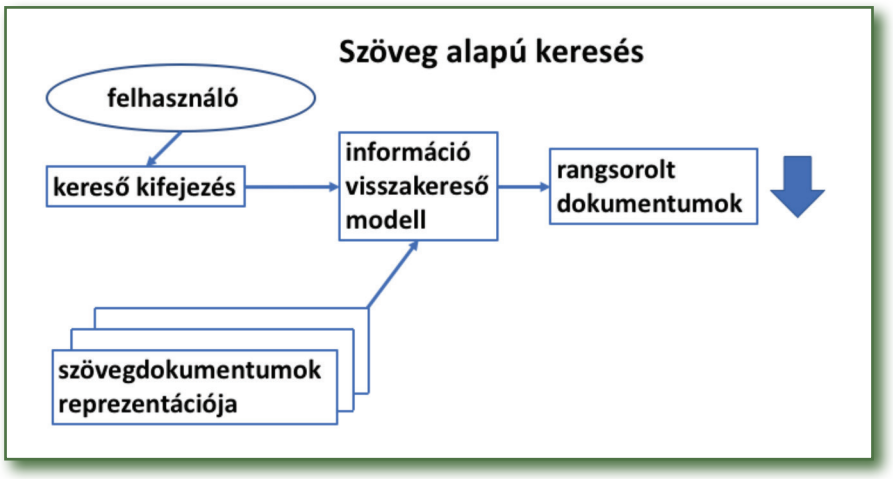
A fenti elvvel összhangban egy adott kifejezéshez rendelt súlyérték:

- nagy lesz, ha az egy adott dokumentumban gyakran, ugyanakkor a teljes korpuszban ritkán fordul elő;
- kicsi lesz, ha az egy bizonyos dokumentumban ritkán, vagy a korpusz több dokumentumában gyakran szerepel;
- nagyon kicsi lesz, akár nullához közeli értéket vesz fel, ha az (majdnem az) összes dokumentumban előfordul.

6.2 SZÖVEGALAPÚ RANGSOROLÁS

Ha kevés kulcsszóból álló keresőkifejezés segítségével akarunk nagy szöveges dokumentumhalmazban egy keresést elvégezni, akkor a találati halmazunk túlságosan nagy lesz. Az eredményben a felhasználó a megfelelő információt úgy találhatja meg a legegyszerűbben, ha a rendszer gondoskodik a visszakapott tartalom olyan jellegű rendezéséről, hogy a találati lista elején szerepeljenek a felhasználó számára relevánsabb tartalmak, majd ezután az egyre kevésbé értékesek (kevésbé relevánsak).

A szövegalapú keresésnél egy ilyen rangsorolás megvalósítását láthatjuk a 4. ábrán. Ez viszonylag egy egyszerű probléma, mert ebben az esetben a rendszer azt veszi figyelembe, amit a felhasználó az aktuális keresés során megadott keresőkifejezésként (opcionálisan még megadhatók a keresési paraméterek). A rendezés a megadott feltételekre történő illeszkedés mértéke alapján valósul meg. Az információ-visszakereső modell a szövegdokumentumok reprezentációját figyelembe véve keresi a hasonlóságot a keresőkifejezés és a visszaadható dokumentumok között, majd hasonlóság alapján rangsorolja azokat úgy, hogy azok kerüljenek legelőre, melyek a legnagyobb valószínűséggel relevánsak a felhasználói igényre.



4. ábra. Szövegalapú keresés

Forrás: a szerzők saját szerkesztése

Ennél a fajta keresésnél csak a szöveg illeszkedését vizsgáljuk; a felhasználó által megadott keresőkifejezés és a dokumentumok szövegrésze vesz részt az illeszkedés vizsgálatban (a 6.1. alfejezetben vázolt matematikai modell csak a szövegalapú kereséshez kapcsolódik).

6.3 LINKALAPÚ RANGSOROLÁS

Ha a szövegen kívül a szövegközti hivatkozásokat (linkeket) is figyelembe szeretnénk venni, akkor erre már más módszerek kínálkoznak. A linkelemzést alkalmazó eljárások más jellegű és bonyolultabb algoritmust használnak, mint a hasonlóságmértéken alapuló szövegalapú keresés rangsoroló algoritmusa. Ezek az internetes keresés során visszakapott weboldalak rendezésére szolgálnak (a weblinkek kihasználásán alapulnak, amelyek az összekapcsolt weblapok viszonyát fejezik ki), azonban bármilyen más hypertext jellegű dokumentumhalmazra is alkalmasak (például kereszthivatkozások, vagy egyéb utalások felelhetnek meg a linkeknek). A két legismertebb, linkeken alapuló keresőmódszer a HITS (Hyperlink-Induced Topic Search [Kleinberg, 1999]) és a Google által kifejlesztett PageRank-algoritmus [Page et al., 1999].

A HITS alapötlete a releváns weblapok felbontása két csoportra: a központi (hub), illetve a szakmailag megbízható (authoritive) oldalakra. Központinak tekinthető egy linkgyűjtemény vagy egy katalógus, míg szakmailag megbízható (illetve a továbbiakban röviden csak megbízhatónak hívjuk) egy tanulmányokat, publikációkat tartalmazó weblap. A besorolás a két csoportba rekurzívan történik; egy lap központi, ha sok megbízható oldalra mutat, illetve megbízható, ha sok központi oldal mutat rá. Az oldalak „megbízhatóságát" és központiságát reprezentáló számok meghatározása iterációval történik. Kezdetben minden érték ugyanaz, majd minden iterációban egy oldal központisága az általa mutatott oldalak megbízhatóságának összege, míg egy lap megbízhatósága a rá mutató oldalak központiságának összege lesz (persze az iterációk között normalizálni kell még a számértékeket, hogy az eljárás során ne legyenek olyan nagyok az értékek, amelyeket a számítógépek már nem tudnak kezelni). A HITS-algoritmus az adott témára keres, ezért sokkal pontosabb és relevánsabb eredményt ad, viszont mivel minden egyes

kérdésnél ki kell számolni az oldalak megbízhatóságát és központiságát reprezentáló értékeket, ezért kevésbé hatékony, és így ez nem is terjedt el annyira, mint a PageRank-algoritmus.

A PageRank-algoritmus lényege, hogy minden egyes oldalra előre kiszámoljuk annak fontosságát – azaz a PageRank értékét – a web véletlenszerű bejárásának szimulálásával, és ezután ezen érték alapján rendezzük a weblapokat. Témafüggetlen keresést valósít meg a felhasználók webes viselkedésének matematikai alapon történő modellezésével. Tevékenységük leírható egy sztochasztikus Markov-folyamatként, aminek a stacionárius eloszlása számítható. Ez az eloszlás megadja annak a valószínűségét, hogy a felhasználó egy adott oldalon tartózkodik, ami az adott lap PageRank értéke lesz. Larry Page és Sergey Brin (a Google alapítói) arra a feltételezésre építettek a PageRank megalkotásakor, hogy a weboldalak készítői általában azokra az oldalakra linkelnek a saját lapjukról, amiket jónak tartanak, vagyis minden link felfogható egy-egy szavazatként a céloldalra [Langville & Meyer, 2011]. Minél több szavazatot kap egy oldal, annál fontosabb, de azt is figyelembe kell venni, hogy a szavazatot leadó oldal mennyire fontos. Ebben a rekurzív definícióban az a fontos oldal, amire fontos oldalak mutatnak, így a PageRank nem más, mint a fontosság számszerűsítése. A korai alapötlet szerint kezdetben minden oldalnak egy egységnyi szavazata van, amit egyenlően szétoszt az általa mutatott oldalak között, és egy oldal PageRankje megegyezik a kapott szavazatok számával. Ahhoz azonban, hogy ez az eljárás jól definiált legyen, be kell vezetni egy *d* csillapítótényezőt: az oldalak a szavazatukból csak *d* részt osztanak tovább és *(1-d)*-t pedig megtartanak. Ez a csillapítótényezővel kiegészített megoldás terjedt végül el, ahol *d* egy 0,85 körüli konstans érték.

6.4 KERESÉS TARTALOMKEZELŐ RENDSZEREKBEN

A tartalomkezelő rendszereknek, különösen a webes rendszereknek – a felhasználók nagy száma végett – hatalmas mennyiségű tartalmat kell kezelniük. A Web2.0-nak [O'reilly, 2009] köszönhetően a felhasználók amellett, hogy mások számára megosztják saját tartalmaikat, minden bizonnyal szeretnének ők is a számukra érdekesek között böngészni; azonos érdeklődési körű felhasználókat találni, velük közösséget alapítani, vagy akár már létező közösségekhez csatlakozni. Ezen funkciók mindegyikéhez szükséges egy alaposan átgondolt keresőszolgáltatás, hiszen ilyen nélkülözhetetlen funkciót szívesen használnak a felhasználók, amikor a nagyszámú tartalmak között szeretnének eligazodni és valamit meg akarnak találni. Ezért a felhasználói felületnek egyszerűnek, átláthatónak és könnyen kezelhetőnek kell lennie; az adatbázis-hozzáférést pedig érdemes minél egyszerűbbre és gyorsabbra tervezni, szem előtt tartva a felhasználó érdekeit, aki valószínűleg nem szívesen várakozik több másodpercet egy-egy keresés eredményére.

6.4.1 Egyszerű keresés

Egyszerű keresési funkciónál a felhasználó megad egy szót vagy kifejezést, és azt a rendszer a dokumentumok szövegében keresi meg, majd visszaadja azokat a dokumentumokat, melyben ez megtalálható (szofisztikáltabb keresőrendszerek még kiemeléssel be is jelölik, hogy a dokumentum mely részében olvasható a keresőkifejezés). Ez a szolgáltatás mind a felhasználóoldali megjelenésben, mind a keresőmotor funkcionalitásában átlátható, egyszerű felépítésű; a közigazgatási munkafolyamatok során általában teljes mértékben megfelel egy ilyen bonyo-

lultságú keresőfunkció. Alapesetben az ilyen keresők csupán egy szövegbeviteli mezőből és egy keresési funkciót meghívó gombból állnak. Modernebb keresők már rendelkeznek gépelésalapú keresésifunkcióval is, melynek lényege, hogy a keresés már gépelés közben elkezdődik, és a begépelt szó függvényében folyamatosan frissül az eredményhalmaz.

A szövegmezős keresés során a keresőkifejezés és a dokumentumok közti illeszkedés vizsgálatánál többféle mód is létezik. A teljesség igénye nélkül itt felsorolunk néhányat:

- szóra vagy kifejezésre történő keresés pontos egyezéssel;
- keresés szó- vagy kifejezésrészletre oly módon, hogy egy jokerkarakter jelzi a hiányzó részeket (ami lehet bármilyen karaktersorozat),
- adott szövegkörnyezetben egymáshoz fizikailag közel álló szavak vagy kifejezések keresése;
- keresés szótőképzéssel: azaz megadott alapszó származtatott alakjait is elfogadja az illesztőmotor;
- keresés szinonima kifejezésekkel: azaz a keresés és az illesztés a rokon értelmű szavak között is zajlik (ehhez tezaurusz állomány építése szükséges).

6.4.2 Összetett keresés

Ha a felhasználó nem tudja pontosan azt a szót vagy kifejezést, amire a keresési igénye irányul, hanem ennél csak árnyaltabb információkkal rendelkezik (például: tegnap délután töltötték fel azt az iratot/dokumentumot) akkor is lehetőséget kell adni számára, hogy megtalálja a releváns információt [Croft, 2010]. Erre szolgál a részletes, más szóval összetett keresés. Az összetett keresésnél tipikusan a következők között lehet keresni: dokumentum neve (azaz fájlnév), dokumentum címe, dokumentum létrehozásának ideje („idő" alatt mindig a dátumot és az adott napon belüli pontos időpontot értünk), dokumentum utolsó módosításának, feltöltésének ideje, dokumentum szerzőjének neve; és a lista még bővíthető. Ezen listaelemek között vannak szabad mezők, ahova bármilyen karaktersorozat beírható (például a fájlnévben való keresésnél beírhatja a felhasználó, hogy „gépjármű" és akkor az összes olyan dokumentum halmaza lesz az eredmény, ahol ez a szó a fájlnév bármely részében előfordul), és vannak az úgynevezett szűrők, amelyek kötött rendet biztosítanak. Ilyen szűrő például a dokumentum létrehozásának ideje: ha a felhasználó egy adott időintervallumra szeretné leszűkíteni a találati halmazt, akkor meg kell adni az intervallum elejét és végét a kívánt tartomány szűréséhez. Ha mindkét dátum megadásra került, akkor azon dokumentumok kerülnek bele az eredményhalmazba, amiket a két időpont által meghatározott intervallumban hoztak létre. Lehetőség van arra is, hogy csak az egyik időpontot adja meg a felhasználó; ekkor értelemszerűen csak a megadott időpont alapján történik a szűrés (csak egy alsó, vagy csak egy felső korláttal az időpontra vonatkozóan). A különböző tulajdonsággal rendelkező dokumentumtípusok esetén más és más szűrőket biztosítanak a találati halmaz csökkentése érdekében. Általánosságban elmondható minden szűrőre, hogy tetszőleges kombinációban bevonhatók, illetve kizárhatók a keresésből. Értelemszerűen minimum egy szabad mezőnek vagy szűrőnek benne kell lennie, hogy legalább egy tulajdonság alapján lehessen keresni.

A szövegtípusú szűrők közül a szerzőre történő keresést érdemes különválasztani a többi ezzel azonos típusú szűrőtől (például cím), hiszen míg az előbbinél csak személyre, addig az utóbbinál általánosan lehet keresni a megadott keresőkifejezéssel. A különválasztás azt jelenti, hogy a két típushoz külön, egy-egy szövegbeviteli mező tartozik. A tartalomkezelő rendszerek használata során előfordulhat, hogy valaki nem dokumentumot, hanem

konkrét felhasználót keres (itt a felhasználó alatt nemcsak egy dokumentumot író szerzőt értünk – mert lehet, hogy nem is írt egyetlen dokumentumot sem –, hanem ténylegesen a felhasználó személyét a hozzá tartozó publikus adatokkal). A felhasználókeresésénél további szövegmezőket lehet felajánlani a felhasználói névre (usernév), illetve teljes névre való keresés során, sőt a keresés finomítása érdekében szerepelhetnek még további szűrők is, mint az e-mail, a felhasználó neme (férfi/nő), esetleg születési év is.

Az összetett keresésnél az egyes attribútumokra vonatkozó szűrők különbözősége maga után vonja egy újabb hierarchiaszint lehetőségét. Ennek a megvalósítására két irány létezik:

- Az egyik esetben többes kiválasztással akár egyszerre több típusú attribútumban is kereshetünk a felületen megadható opciók segítségével. Például a *Tóth* szóra történő keresésnél akár egymás mellett megtalálhatjuk az eredményhalmazban az ilyen nevű fájlokat és az ilyen nevű felhasználók dokumentumait.
- A másik megoldás a hierarchiaszintek bevezetése és a grafikus felhasználói felületen való megjelentetése, és ezáltal átláthatóbbá tétele. Ebben az esetben a felhasználónak először el kell döntenie, hogy milyen attribútum érdekli, majd a választástól függően jeleníti meg a rendszer számára a továbbiakat, ahol majd keresni tud. Ez a megoldás átláthatóbb felületet eredményez, könnyebben meg lehet valósítani, ellenben több interakciót igényel a felhasználótól, és nincs lehetőség több attribútum együttes keresésére.

6.4.3 Viselkedésalapú keresés

A felhasználó nem csupán a keresőfelületen tudja befolyásolni, hogy mely találatokat és milyen sorrendben kapjon vissza, de a felhasználói profil preferenciabeállításai is hatással lehetnek a keresési eredményekre [Agichtein et al., 2006], amennyiben a rendszerben kiépítésre került. A preferenciákat a felhasználó általában direkt módon meg tudja adni. Az intelligensebb módszer az, ha ezt nem kell közvetlenül megtennie, hanem a rendszer adaptívan kitalálja a felhasználó viselkedéséből. A következőkben összeszedtük, hogy hogyan változik a preferencia a felhasználó különböző viselkedéseinek hatására:

- Felhasználó megnéz egy dokumentumot: ekkor a megnézett dokumentum címkéinek és a szerzőjének súlya növekszik.
- Felhasználó letölt egy tartalmat: ebben az esetben a letöltött tartalom címkéinek és a szerzőjének a súlya szintén növekszik.
- Felhasználó módosítja egy dokumentumnak a címkéjét: címke hozzáadásával az adott címke súlya növekszik, míg címke törlésekor annak súlya csökken.
- Felhasználó megjegyzést ír egy tartalomhoz: ekkor az értékelt tartalom címkéinek, illetve a szerzőjének a súlya változik az értékeléstől függően.
- Könyvjelző hozzáadása: azaz könyvjelzők használatával a felhasználó megjelölhet számára releváns tartalmakat. Létrehozásakor az ahhoz tartozó tartalmak címkéinek, illetve szerzőjének súlya értelemszerűen nagymértékben növekszik.
- Könyvjelző törlése: ennek hatására a könyvjelzőhöz tartozó tartalom összes címkéinek, illetve szerzőjének súlya csökken.

Ez a fajta intelligens mód már valóban „okos" közigazgatássá tudná tenni az ilyen rendszerek használatát. Mindehhez azonban nem elég csupán a statisztikai adatokat tárolni, amiben csak összegzett adatok vannak nyilvántartva (az összes felhasználóra), hanem az előzőekben felsorolt felhasználói cselekedeteket (tartalom megnézése, megjegyzés írása, könyvjelző hozzáadása, stb.) is tárolni kell. Azaz, ha személyre szabottabban akarjuk a keresési eredményeket rendezni, akkor felhasználónként le kell menteni az egyéni cselekedeteket, amiből a későbbiekben következtetések vonhatók le, és felhasználhatók lesznek a keresési eredmények rendezése során.

7. ÖSSZEFOGLALÓ – A KÖZIGAZGATÁS SZEMPONTJÁBÓL

Ebben az összefoglaló fejezetben összeszedjük, hogy milyen tartalomkezelő rendszerre lenne szükség a közigazgatásban a dokumentumok kezeléséhez, továbbá bemutatjuk azt a néhány hasznosulási lehetőséget, amely a szövegelemzésből származhat a tartalomkezelő rendszereknél, kitérve az alkalmazási és továbbfejlesztési lehetőségekre is.

Az okos közigazgatásban a jó kormányzást megalapozó közszolgálat-fejlesztés érdekében érdemes olyan webes tartalomkezelő rendszert bevezetni (tervezni vagy egy meglevőt kiválasztani és testre szabni [Rohilla, 2017]), mely nagy terhelhetőségű, és a közigazgatásban dolgozók kiszolgálására lett specializálva. Ennek megfelelően a rendszer tartalmi egységelemének a szöveges dokumentumokat érdemes választani. A források gyűjtéséről az erre kiképezett dolgozóknak kell gondoskodniuk (persze az is előny, ha a rendszer használatának elsajátításához nem szükséges hosszú tanfolyamokat tartani, hanem az alapfogalmak megismerése után minden magától értetődően kezelhető), a rendszernek pedig alkalmasnak kell lennie a több forrásból is érkező tartalmak befogadására. A dokumentumok közös szerkeszthetősége alapkövetelmény (több szerkesztő számára is elérhetőnek kell lennie), a megjelenítésért pedig mindig a szerkesztők által könnyen és szabadon használható dinamikusan weboldalak a felelősök. Hibrid típusú rendszert érdemes választani, így egyaránt elérhető a jó teljesítmény és a közel valós idejű publikálás; valamint fontos követelmény még a rendszerrel szemben, hogy a tartalmaknak jól kereshetőeknek kell lenniük. Mivel a közigazgatásban a hivatalos iratok nagy mennyiségben fordulnak elő, ezért kiemelten fontos a biztonság. A rendszernek biztonságos jelszókezelést kell megvalósítania, ügyelnie kell a személyes adatok védelmére, mentesnek kell lennie minden olyan kiskaputól, amely az ismert típusú támadásoknak engedhet teret. Továbbá teljesítménycsökkenés nélkül kell elviselnie a rendkívül széles határok között mozgó terhelésváltozást is, ami függ a futtató hardvertől is (felhőalapú rendszerek esetén a nagy terhelésingadozás megoldható skálázhatósággal).

A rendszernek biztosítania kell, hogy a látogatók és felhasználók azokhoz a fájlokhoz ne férjenek hozzá, amelyekhez nem rendelkeznek megfelelő jogosultsággal. A biztonság egyrészt technikai, másrészt emberi tényezőkből áll; technikai szempontból fontos, hogy ellenálljon a különböző támadásoknak, mint például az SQL injection, script injection, űrlapmódosítás elküldés előtt, URL-támadás. Ha az emberi tényezőket vizsgáljuk, akkor megállapítható, hogy a legtöbb felhasználó nem figyel a biztonságra, ezért a rendszernek ebben is tudnia kell segíteni. Erre egy lehetséges eszköz egy jelszóházirend bevezetése, amely az üzemeltetők által meghatározott bonyolultságú jelszavak használatát teszi csak lehetővé. Például kötelezővé teheti a kis- és nagybetűk, valamint számok szerepeltetését a jelszóban; előírhatja a jelszavak minimális hosszát, de akár nem alfanumerikus karakterek felhasználását, és a rendszeres jelszócserét is kötelezővé teheti. Érdemes jól átgondolni, hogy ezekből a lehetőségekből milyen kombinációt alkalmaznak, hogy a biztonsági szempont ne menjen a felhasználói élmény rovására.

További fontos szempont, hogy a jelszavaknak biztonságosan kell közlekednie a hálózaton. Erre azért kell figyelni, mert sajnos az átlagfelhasználók nagy része azonos jelszavakat használ több különböző szolgáltatás eléréséhez is, így egy jelszó megszerzésével több szolgáltatáshoz is hozzáférést szerezhetnek. A jelszavaknak tehát nem,

vagy valamilyen titkosított formában szabad csak közlekednie a hálózaton, hogy egy esetleges hálózati forgalmat lehallgató program ne férhessen hozzá. Azonban a biztonság megőrzése mellett a rendszernek lehetőséget kell biztosítania egy esetlegesen elfelejtett jelszó esetén a felhasználó azonosítására és új jelszó megadására, ahogy azt a tanulmányunkban részletesen bemutattuk.

A rendszernek karbantarthatónak (lásd a 3.4. CMS karbantartása alfejezetet), továbbá naplózásra és statisztikák gyűjtésére is alkalmasnak kell lennie. Ez utóbbihoz használható például a Google Analytics, mely az amerikai Google Inc. weboldal statisztikákat és látogatottsági adatokat gyűjtő szolgáltatása. Használata ingyenes és egyszerű; rövid regisztrációt követően mindösszesen egy rövid programkód beillesztését igényli a monitorozandó weboldalba. A szolgáltatás használata, amennyiben az adott közigazgatási környezetben megengedhető egy más által üzemeltetett elemző szolgáltatás beillesztése, több szempontból is előnyös. Egyrészt nincs szükség ezeknek a statisztikáknak a rendszeren belüli naplózására, valamint a szolgáltatás függetlensége által harmadik fél számára is hiteles adatokat biztosít a weboldallal kapcsolatban. A Google Analytics az adatok gyűjtésén túl azok elemzésében is komoly segítséget nyújt a rengeteg beépített jelentés és grafikon segítségével, de egyedi elemzések és egyszerűbb adatbányászati feladatok elvégzésére is támogatást kínál. A rendszer lehetőséget biztosít arra, hogy csak az adatokból generált jelentések megtekintésére jogosult felhasználót definiáljunk. A portál működése során a szükséges rövid programkód, a statisztika pontosságának növelése érdekében, csak az olvasói vagy regisztrált felhasználói jogkörrel rendelkező felhasználók számára kerül beszúrásra, így kiszűrhető a szerkesztők és az adminisztrátorok tevékenykedése az adatokból, melyek torzítanák a statisztikákat.

A tanulmányunkban bemutatott módszerek több alkalmazási területen is használhatók az okosváros-koncepciót érintően: egyrészt a polgárok szempontjából, a szövegelemzés felhasználható automatikus szövegfeldolgozásra, másrészt az államigazgatásban is hasznosulhat, mert a szövegbányászat alkalmas hírfigyelésre, ügyfélszolgálati tevékenységre, e-kormányzatnál automatikus ügyfélirányításra, digitális könyvtárakban pedig például könyvek keresésére; mindegyik a városban lakó polgárok ügyfélelégedettség-növelését célozza meg.

A szövegbányászat hasznosulhat az üzleti életben is, hiszen ennek a természetes közege a Web, amit a cégek és magánemberek többek között információk publikálására használnak, és weboldalak a látogatottságért versengenek. Egy nagy látogatottságú, bejáratott weboldalt azonban nem csak információk közlésére lehet hasznosítani. Ha széles közönséghez jut el, akkor a rajta elhelyezett hirdetések sok látogatót elérhetnek. Egy reklám annál nagyobb értékkel bír, minél több ember látja, ezért egy sűrűn látogatott weboldalon szívesen hirdetnek a hirdetők, és tulajdonosa is több pénzt elkérhet az ott elhelyezett hirdetésekért.

A szövegbányászat alkalmazásának fontos irányát mutatja a Financial Times tengerentúli napilap, mely már korábban elkezdett használni egy hírek üzleti témáival kapcsolatos érzéseket, érzelmeket figyelő, speciális keresőmotorral összekapcsolt kísérleti programot. Ez a fajta kereső több szempont (cég, hely, személy és téma) szerinti kutakodást biztosít, és segítségével a gazdasági hírek társadalmi hatását lehet nyomon követni. A szolgáltatás keresési funkcióját témakör, intézmény, helyszín és személynév alapján lehet szűkíteni [Devitt & Ahmad, 2007].

Ahogy az előző példánál láthattuk: az érzelemelemzés az online információkeresés gyakorlatát is megváltoztathatja. Az érzelemelemző szoftverek nyelvhasználatunk olyan finom rétegeinek megértésére törekednek, amelyekhez már a legmodernebb szövegbányászati újítások szükségesek. Egy szó jelentését és érzelmi töltetét ugyanis nem mindig lehet eldönteni a szótári definíció alapján: ezeket nemegyszer épp az a kontextus adja hozzá egy nyelvi alakhoz, amelyben előfordul. Néhány innovatív alkalmazás ezért nemcsak azt vizsgálja, hogy a szavak önmagukban milyen emocionális telítettséggel rendelkeznek, hanem azt is, hogy milyen többletjelentést nyernek a ma-

gasabb szintű nyelvtani szerkezetekben. A legújabb algoritmusok nemcsak az adott témával kapcsolatos érzéseket elemzik, hanem a legbefolyásosabb véleménynyilvánítókat is képesek azonosítani. Jelenleg olyan előrejelző algoritmusokat próbálnak fejleszteni, amivel például meg lehet jósolni, hogy egy fontos lap vezércikke milyen hatással lesz a cégek tőzsdei szereplésére. Az érzelmi alapú keresőalgoritmusok továbbfejlesztésével jelentős gazdasági előrelépési lehetőség kínálkozik a vállalatoknál.

A főbb területeket áttekintve az tapasztalható, hogy a szövegbányászat még sok kihívást tartalmazó, innovatív technológiai területnek számít. A piaci szereplők közül egyre többen felismerik jelentőségét és alkalmazzák azt piackutatásaik során a gazdasági felméréseik tökéletesítéséhez [Fajszi et al., 2010], valamint használják ajánlórendszereknél is. Ezek olyan speciális információszűrő rendszerek, ahol termék- és felhasználóprofilokat építenek tanuló algoritmusok segítségével, majd a modellek alapján olyan tartalmat ajánlanak a felhasználónak, amely nagy valószínűséggel érdekes lehet számára. Ezek miatt fontos, hogy az okos város polgárai is élvezzék ennek előnyeit.

Az okos közigazgatásban használhatók mindazok a módszerek, melyeket tanulmányunkban bemutattunk. Felmerülhet a kérdés, hogy mitől lehet még okosabbá tenni a bemutatott módszereket a közigazgatás szempontjából. Ha a keresőalgoritmusokat tekintjük, lehetőség van még a javításukra: hagyományos módszernél az adott felhasználótól független általános és összegzett statisztikai adatok alapján történik a relevanciameghatározás, és ez alapján a sorrendezés; a felhasználó egyéni cselekedeteit is figyelembe vevő algoritmus adaptívabb tudna lenni (és már el is indult a fejlődés ebben az irányban). További fejlődési irányként olyan intelligens megoldási módot is el lehet képzelni, amely a felhasználók közti hasonlóságot is ki tudja használni. A rendszer tudná vizsgálni és hasznosítani, hogy egy adott felhasználó mennyire hasonló információkat keres, mint egy másik. A hasonlóságot befolyásolhatja minden olyan tevékenység, melyet a felhasználók a rendszerben végeznek (mely dokumentumokat nézték meg, töltötték le, melyekhez írtak megjegyzéseket). Ez alapján, ha egy felhasználókeresési folyamatánál találunk egy másik olyan felhasználót, aki nagyon hasonló hozzá, akkor érdemes ezt figyelembe venni (például a kereséskor egy hasonló felhasználó által letöltött tartalmat érdemes előrébb sorolni a rendezés során). Mindez a találati halmazban nagy eséllyel növeli a keresés hatékonyságát. Ez az adaptivitás és intelligens megoldási mód fontos, hiszen a közigazgatási informatikai rendszerek sikerességéhez nagymértékben hozzájárulhat, hogy a rendszereket használó ügyintézők, háttérben dolgozó személyek meg vannak-e elégedve a rendszerrel.

IRODALOMJEGYZÉK

[1] Agichtein, E., Brill, E., & Dumais, S. (2006): *Improving web search ranking by incorporating user behavior information*. In: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval. 19-26. o.

[2] Alalwan, J. A., & Weistroffer, H. R. (2012): *Enterprise content management research: a comprehensive review*. Journal of Enterprise Information Management, 25(5). 441-461. o.

[3] Altenhofen, C., Hofmann, H. R., Kieninger, T., & Stanišić-Petrović, M. (2004): *Results of a Survey about the Use of Tools in the Area of Document Management*. In: Reading and Learning. Springer, Berlin, Heidelberg. 328-354. o.

[4] Baeza-Yates, R. & Ribeiro-Neto, B. (1999): *Modern Information Retrieval*. New York: ACM Press, Addison Wesley.

[5] Benantar, M. (2006): *Access control systems: security, identity management and trust models*. Springer Science & Business Media.

[6] Berry, M. W., Drmac, Z., & Jessup, E. R. (1999): *Matrices, vector spaces, and information retrieval*. SIAM review, 41(2). 335-362. o.

[7] Boiko, Bob (2005): *Content Management Bible,* John Wiley & Sons.

[8] Boser, B. E., Guyon, I. M., & Vapnik, V. N. (1992): *A training algorithm for optimal margin classifiers*. In: Proceedings of the fifth annual workshop on Computational learning theory. ACM. 144-152. o.

[9] Buttyán L., Vajda I. (2007): *Kriptográfia és alkalmazásai*. Typotex kiadó, Budapest.

[10] Cambria, E., & White, B. (2014): *Jumping NLP curves: A review of natural language processing research*. IEEE Computational intelligence magazine, 9(2). 48-57. o.

[11] Canvar, W. B. & Trenkle J. M. (1994): *N-Gram-Based Text Categorization,* Proceedings of the 3rd Annual Symposium on Document Analysis and Information Retrieval. Las Vegas, Nevada, USA. 161–175. o.

[12] Chowdhury, G. G. (2010). *Introduction to modern information retrieval*. Facet publishing.

[13] Croft, W. B., Metzler, D., & Strohman, T. (2010): *Search engines: Information retrieval in practice* (Vol. 283). Reading: Addison-Wesley.

[14] Devitt, A, & Ahmad, K. (2007): *Sentiment Polarity Identification in Financial News: A Cohesion-based Approach*. In Proceedings of the 45th annual meeting of the association of computational linguistics (ACL). 984-991. o.

[15] European Parliament, Council of the European Union (2016): The General Data Protection Regulation (EU) 2016/679 ("GDPR") on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC, 27 April 2016. 1–88. o.

[16] Fajszi B., Cser L., Fehér T. (2010): *Üzleti haszon az adatok mélyén – Az adatbányászat mindennapjai,* Alinea Kiadó – IQSYS.

[17] Gossweiler, R., Kamvar, M., & Baluja, S. (2009): *What's up CAPTCHA?: a CAPTCHA based on image orientation*. In: Proceedings of the 18th international conference on World wide web. ACM. 841-850. o.

[18] Guo, H., Mao, N., & Yuan, X. (2011): *Wysiwyg (what you see is what you get) volume visualization*. IEEE Transactions on Visualization and Computer Graphics, 17(12). 2106-2114. o.

[19] Iványi Antal (2005): *Informatikai algoritmusok II.,* ELTE Eötvös Kiadó, Budapest.

[20] Jing, H. & McKeown, K. R. (1999): *The decomposition of human-written summary sentences,* In: Proc. of SIGIR-99, 22nd ACM Int. Conf. on Research and Development in Information Retrieval. Berkeley, USA. 129-136. o.

[21] Jones, K. Sparck, Steve Walker, and Stephen E. Robertson. (2000): *A probabilistic model of information retrieval: development and comparative experiments: Part 2.* Information processing & management Vol. 36, no. 6. 809-840. o.

[22] Kiedrowicz, M., & Stanik, J. (2015): *Selected aspects of risk management in respect of security of the document lifecycle management system with multiple levels of sensitivity*. Information Management in Practice, (eds.) B. F. Kubiak and J. Maślankowski. 231-249. o.

[23] Kleinberg, J. M. (1999): *Authoritative sources in a hyperlinked environment*. Journal of the ACM (JACM), 46(5). 604-632. o.

[24] Kovács, A. F., Magyar, G., Szűcs, G. (2010): *Collaborative Media Content Management,* 19th International Conference on Information Systems Development (ISD), August 25 - 27, 2010, Prague, Czech Republic.

[25] Kumar, H. R. (2013): *Types of WCMS, Online & Offline Processing, And Hybrid Systems,* SooperArticles, January 22, 2013.

[26] Langville, A. N., & Meyer, C. D. (2011): *Google's PageRank and beyond: The science of search engine rankings*. Princeton University Press.

[27] Lashkari, A. H., Mahdavi, F., & Ghomi, V. (2009): *A Boolean model in information retrieval for search engines.* In: Information Management and Engineering, 2009. ICIME'09. International Conference on. IEEE. 385-389. o

[28] Lee, D. L., Chuang, H., & Seamons, K. (1997): *Document ranking and the vector-space model*. IEEE software, 14(2), 67-75. o.

[29] Lewis, D. D. (1998): Naive (Bayes) at forty: The independence assumption in information retrieval. In: European conference on machine learning. Springer, Berlin, Heidelberg. 4-15. o.

[30] Magyar G., Kardkovács Zs., Szűcs G. (2009): *Médiatartalom-kezelés és -szolgáltatás (Media content management and services),* Hiradástechnika, Vol. LXIV., 2009, különszám. 91-95. o.

[31] McCallum, A. (2005): *Information extraction: Distilling structured data from unstructured text*. Queue, 3(9), 4. ACM.

[32] Ninoriya, S., Chawan, P. M., Meshram, B. B., & Vjti, M. (2011): *CMS, LMS and LCMS for e-learning*. International Journal of Computer Science Issues, 8(2). 644-647. o.

[33] O'reilly, T. (2009): *What is web 2.0*. O'Reilly Media, Inc.

[34] Page, L., Brin, S., Motwani, R., & Winograd, T. (1999): *The PageRank citation ranking: Bringing order to the web*. Stanford InfoLab.

[35] Patel, S. K., Rathod, V. R., & Parikh, S. (2011): *Joomla, Drupal and WordPress-a statistical comparison of open source CMS.* In: Trends in Information Sciences and Computing (TISC), 2011 3rd International Conference on. IEEE. 182-187. o.

[36] Porter, M. F. (2001): Snowball: A language for stemming algorithms. Internetes elérhetőség: http://snowball.tartarus.org/ (Letöltés dátuma: 2019.06.21)

[37] Rahmel, D. (2008): *Beginning Joomla!: From novice to professional*. Apress.

[38] Regli, T. (2009): *The state of digital asset management: An executive summary of CMS Watch's Digital Asset Management Report.* Journal of Digital Asset Management, 5(1). 21-26. o.

[39] Roebuck, K. (2012): *Web Content Management Systems (WCMS): High-impact Strategies-What You Need to Know: Definitions, Adoptions, Impact, Benefits, Maturity, Vendors.* Emereo Publishing.

[40] Rohilla, N. (2017): *Selection of Appropriate Web Content Management System (WCMS)*. International Journal of Engineering and Computer Science, 6(6).

[41] Salton, G. & McGill M. J. (1983): *An Introduction to Modern Information Retrieval,* McGraw-Hill.

[42] Salton, G., Fox, E. A., & Wu, H. (1983): *Extended Boolean information retrieval*. Communications of the ACM, 26(11). 1022-1036. o.

[43] Salton, G., Wong, A., & Yang, C. S. (1975): *A vector space model for automatic indexing*. Communications of the ACM, 18(11). 613-620. o.

[44] Sharma, D. (2012): *Stemming algorithms: a comparative study and their analysis*. International Journal of Applied Information Systems, 4(3). 7-12. o.

[45] Soucy, P., & Mineau, G. W. (2005): Beyond TFIDF weighting for text categorization in the vector space model. In IJCAI, Vol. 5. 1130-1135. o.

[46] Souer, J., Honders, P., Versendaal, J., & Brinkkemper, S. (2007): *Defining operations and maintenance in web engineering: A framework for CMS-based web applications.* In: Digital Information Management, 2007. ICDIM'07. 2nd International Conference on (Vol. 1.). IEEE. 430-435. o.

[47] Szűcs G. & Móczár Z. (2013): *Iterative Text Clustering of Search Results,* International Journal of Computers, 7:(4). 127-134. o.

[48] Tikk, D. (2007): *Szövegbányászat*. TypoTEX, Budapest.

[49] Tomlinson, T., & VanDyke, J. (2010): *Pro Drupal 7 Development*. Apress.

[50] Tordai, A., & De Rijke, M. (2005): *Four stemmers and a funeral: Stemming in Hungarian at clef 2005.* In: Workshop of the Cross-Language Evaluation Forum for European Languages. Springer, Berlin, Heidelberg. 179-186. o.

[51] Vivekavardhan, J., & Verma, M. K. (2016): *Open Source Content Management System for Content Development*. Library Waves, 2(1). 6-14. o.

[52] Wang, X., Yin, Y. L., & Yu, H. (2005): *Finding collisions in the full SHA-1*. In: Annual international cryptology conference. Springer, Berlin, Heidelberg. 17-36. o.

[53] Xiao, C., Wu, D., Guo, S., & Chen, X. (2007): *Index Algorithm of CMS Metadata Based on XML*. Computer Engineering, 7, 022.

A Nemzeti Közszolgálati Egyetem kiadványa

Kiadó:
Nemzeti Közszolgálati Egyetem
Közigazgatási Továbbképzési Intézet
www.uni-nke.hu

Felelős kiadó:
Prof. Dr. Kis Norbert rektorhelyettes
Címe: 1083 Budapest, Üllői út 82.

Olvasószerkesztő:
Kelemen Dóra

Tördelőszerkesztő:
Mikes Vivien

ISBN (PDF): 978-963-498-114-5