MISKOLCI EGYETEM

GÉPÉSZMÉRNÖKI ÉS INFORMATIKAI KAR

AUTOMATIZÁLT KÉRDÉSGENERÁLÁS ANNOTÁLT SZÖVEGBŐL

Ph.D. Disszertáció

KÉSZÍTETTE:

Bednarik László

okleveles mérnök-informatikus

AKI DOKTORI FOKOZAT ELNYERÉSÉRE PÁLYÁZIK

HATVANY JÓZSEF INFORMATIKAI TUDOMÁNYOK DOKTORI ISKOLA
ALKALMAZOTT SZÁMÍTÁSTUDOMÁNY TÉMATERÜLET
ADAT- ÉS TUDÁSBÁZISOK, TUDÁSINTENZÍV RENDSZEREK TÉMACSOPORT

Miskolc, 2012

NYILATKOZAT

Alulírott Bednarik László kijelentem, hogy ezt a doktori értekezést magam készítettem, és abban csak a megadott forrásokat használtam fel. Minden olyan részt, amelyet szó szerint vagy azonos tartalomban, átfogalmazva más forrásból átvettem, egyértelműen, a forrás megadásával megjelöltem.

Miskolc, 2012. október 20.

Bednarik László

A disszertáció bírálatai és a védésről készült jegyzőkönyv megtekinthető a Miskolci Egyetem Gépészmérnöki és Informatikai Karának Dékáni Hivatalában, valamint a doktori iskola weboldalán az Értekezés menüpont alatt: *http:// www.hjpdh.iit.uni-miskolc.hu.*

TÉMAVEZETŐ AJÁNLÁSA

Bednarik László: ”Automatizált kérdésgenerálás annotált szövegből” című Ph.D. értekezéséhez

Bednarik László hosszú évek óta oktatóként dolgozik a Sárospataki Főiskolán informatikai témakörbe eső tantárgyakat vezetve. Nagy tapasztalattal rendelkezik az oktatás, az oktatásmódszertan területén. Először levelezős hallgatóként találkoztam vele, ahol feltűnt kitartásával és szorgalmával. Pozitív tapasztalataim alapján örömmel vállaltam el témavezetését, amikor 2004-ben jelentkezett a doktori képzésre. Témakörként olyan területet jelöltünk ki, mely egyrészt kapcsolódott a tanszék korábbi kutatási munkáihoz, másrészt köthető Bednarik László oktatásban szerzett tapasztalataihoz is. Az automatizált kérdésgenerálás témakörét választottuk ki, mely szorosan illeszthető az oktatási keretrendszerekhez, a szövegbányászathoz és az intelligens rendszerek eszköztárához is. Mindemellett a téma erősen gyakorlat orientált is, hiszen működő, adott tananyagon bemutatható mintarendszer kidolgozását tűztük ki célul.

A feladat megoldásához a jelöltnek több terület eszköztárát kellett megismernie és továbbfejlesztenie. Elsőként az e-learning jelenleg elérhető megoldásait és a jövőbeli irányait kellett feldolgozni. Ezt követően egy napjainkban felfutó ágazatnak, az ontológiának a lehetőségeit és módszereit kellett elsajátítani a tananyagok absztrakt modelljének felépítéséhez. Ezután a szövegbányászat fontosabb eredményeinek megismerése következett. A modell felvázolását követően az egyes szövegbányászati algoritmusokat kellett elemeznie a jelöltnek, majd ezen algoritmusok optimalizálását végezte el. A kutató munka végén egy működő mintarendszert kellett kidolgoznia. A jelölt egy sok területre kiterjedő elemzést készített el a kapcsolódó nemzetközi irodalomról. A dolgozat különböző részterületeken elért eredményeket fog össze, melyek közül kiemelném a neurális hálózaton alapuló osztályozó és a BIRCH alapú klaszterező modulok kidolgozását. A jelölt kitartását dicséri, hogy leterhelő munkahelyi feladatok mellett el tudta érni a kitűzött célt. Az elvégzett munka értékét nagyban növeli, hogy valós tesztekkel sikerült bemutatni a kidolgozott architektúra és rendszermodell működőképességét: az automatikusan generált kérdések és a manuálisan előállított kérdések szorosan korrelálnak egymással a vizsgázók által megszerzett pontszámokat illetően.

Az értekezés tézisei és a témához kapcsolódóan megjelent publikációk igazolják, hogy a jelölt sikeresen végrehajtotta a kitűzött célt. A jelölt a kutatás eredményeit nemzetközi szinten is publikálta, eleget téve a Hatvany József Informatikai Tudományok Doktori Iskola publikációs követelményeinek. Az eredmények igazolják, hogy a jelölt képes az elvárt, önálló kutatómunkára, munkáját a kitartás és a gyakorlat orientáltság szem előtt tartása jellemzi. Az értekezés Bednarik László saját eredményeit tartalmazza és a Hatvany József Informatikai Tudományok Doktori Iskola által megkövetelt tartalmi és formai követelményeknek mindenben megfelel. Mindezekre tekintettel a jelölt számára a Ph.D. cím odaítélését messzemenően támogatom.

Miskolc, 2012. október 20.

dr. habil. Kovács László
tudományos vezető

KÖSZÖNETNYILVÁNÍTÁS

Az értekezés a Hatvany József Informatikai Tudományok Doktori Iskola keretein belül, a Miskolci Egyetem Comenius Főiskolai Kar Informatikai Tanszékén 2004-ben kezdett kutatómunkám eredményeit foglalja össze. Köszönettel tartozom mindazoknak, akik lehetővé tették, illetve közvetlen vagy közvetett módon megkönnyítették a dolgozat elkészülését.

Köszönet illeti **Dr. Tóth Tibor** professzort, aki a Doktori Iskola vezetőjeként, sok hasznos tanácsot nyújtott.

Hálás vagyok tudományos vezetőmnek, **Dr. habil. Kovács László** egyetemi docensnek, aki figyelmem az adat- és tudásbázisok, tudásintenzív rendszerek témacsoport felé fordította és precíz kutatómunkára ösztönzött.

Nagyon köszönöm a társ-témavezetőmnek, **Prof. Dr. Juhász Imre** egyetemi tanárnak, aki professzionális tudásával, segítőkészségével nagymérték segítette a dolgozat elkészülését.

Köszönöm a Miskolci Egyetem Comenius Főiskolai Kar dékánjának és a Reál Tudományok Intézet kollektívájának a szakmai és erkölcsi támogatást.
Végül szeretném megköszönni családom és barátaim támogatását.

Tartalomjegyzék

Rövidítések jegyzéke

ADL	Advanced Distributed Learning – Fejlett elosztott tanulás
ASCII	American Standard Code for Information Interchange – Amerikai szabványos kód az információcseréhez
AQG	Automatic Question Generation – Automatizált kérdésgeneráló
BFS	Best First Search – Legjobbat legelőször keresési stratégia
BIRCH	Balanced Iterative Reducing and Clustering using Hierarchies – Kiegyensúlyozott iteratív csökkentés és klaszterezés hierarchiák segítségével
CAM	Content Aggregation Model – Tartalomhalmozási modell
CBT	Computer Based Training – Számítógép alapú oktatás
CF	Clustering Feature – Klaszterezési jellemzők
CRF	Conditional Random Field – Feltételes valószínűségi mező
CPN	Counter-Propagation Network – Szembeterjedéses hálózat
CURE	Clustering Using REpresentatives – Klaszterezés reprezentáló elemek segítségével
DITA	Darwin Information Typing Architecture – Technikai dokumentációk írását és publikálását támogató XML-architektúra jelölőnyelv
EAQC	Enhanced Automatic Question Creator – Továbbfejlesztett automatikus kérdés előállító
GIFT	General Import Format Technology – Általános importálási formátum technológiája
HAC	Hierarchical Agglomerative Clustering – Hierarchikus egyesítő klaszterezés
LAN	Local Area Network – Lokális hálózat
LCMS	Learning Content Management System – Tartalomkezelő rendszer
LMS	Learning Management System – Tanulmányi keretrendszer
LO	Learning Object – Tananyagelem
LOM	Learning Object Metadata – Tanulási objektum metaadat
MCV	Model, Control and View – Modell, vezérlő és nézet
MOODLE	Modular Object-Oriented Dynamic Learning Evrironment – Moduláris objektum-orientált dinamikus tanulási környezet
ODL	Object Definition Language – Objektum definíciós nyelv
OWL	Web Ontology Language – Web-ontológiák nyelve
PLE	Personal Leraning Environment – Személyes tanulási környezet

PWN	Princeton WordNet – Princeton szóbank
QTC	Quality Treshold Clustering – Minőségi küszöbérték klaszterezés
RDF	Resource Description Framework – Erőforrás-leíró nyelv
RVP	Ranking Voted Perceptron – Rangsorolási szavazó perceptron
SCO	Sharable Content Object – Megosztható tartalomegység
SCORM	Sharable Content Object Reference Model – Megosztható tartalom objektum hivatkozási modell
SGML	Standard Generalized Markup Language – Szabványos általános jelölőnyelv
SOM	Self-Organising Map – Önszervező háló
SVM	Support Vector Machine – Szupportvektor gép
TEI	Text Encoding Initiative – Szövegkódoló kezdeményezés
WAN	Wide Area Network – Nagy kiterjedésű hálózat
WBT	Web Based Training – Web alapú oktatás
XML	Extensible Markup Language – Kiterjeszthető jelölő nyelv
XSLT	Extensible Stylesheet Language Transformations – XSL nyelv transzformációs szabványa

1. fejezet

Bevezetés

A szövegbányászat, mely az 1990-es évek végén jelenik meg, a számítástudomány szöveges elektronikus dokumentumok feldolgozásával és elemzésével foglalkozó szakterülete [5]. Kialakulása a digitalizált vagy nyomtatott elektronikus dokumentumok és médiaanyagok számának növekedésével indult el, amit az Internet megjelenése és térhódítása tovább fokozott. A fejlődés hatására jelentősen megnőtt az elektronikus szöveges tartalmak mennyisége, amelynek nemcsak tárolása okoz problémát, de hatékony feldolgozásának, rendszerezésének kérdése is egyre inkább előtérbe kerül. A szöveges dokumentum speciális adathalmaz, mely általában strukturálatlan formában áll rendelkezésre. Ugyanakkor gazdag információtartalommal rendelkezik. A definíció szerint a magas információsűrűségű, kevés szerkezettel rendelkező szöveget strukturálatlan adatnak nevezik. A szövegbányászat technikai megoldás arra, hogy a nagymennyiségű strukturálatlan adathalmazból kinyerjük a felhasználó számára fontos információkat. Meryll Lynch elemzése szerint az üzleti információk 85%-a strukturálatlan, illetve gyengén strukturált adat, például emlékeztetők, prezentációk, hírek, weboldalak, ügyfélszolgálati tevékenység jegyzetei stb. formájában áll rendelkezésre [Blumberg, 2003]. Alapvető nehézségei közé tartozik, hogy a szövegben előforduló természetes nyelvek emberek közötti kommunikáció céljára alakultak ki és fejlődtek, melynek szabályai, nyelvtana bonyolultabb a programozásban elterjedt reguláris nyelvtannál [Fajszi, 2004]. Emiatt a számítógépek számára nehézséget jelent a különböző helyesírási variációk kezelése vagy a kontextus felismerése. A szövegbányászat célja szöveges dokumentumok elemzésére és feldolgozására szolgáló algoritmusok fejlesztése, amelyek az emberi nyelv tudását ötvözik a számítógép nagy feldolgozási kapacitásával [Fan, 2006].

Az informatikával támogatott oktatási eszközök fejlődésének egyik fontos jövőbeli irányát jelentik az adaptív és szemantika-orientált számítógépes oktató rendszerek. Napjainkban a kutatások fő jellemzője, hogy a hagyományos oktatási keretrendszerek mellett megjelennek a témakör ismeretanyagát tároló tudásbázisok, szemantikai adatbázisok. Ezen adatbázisok tudásanyagára építve rugalmasan kezelhető a tananyag. Az adaptivitás elsődlegesen a hallgatói igényekhez történő alkalmazkodás képességét jelenti. A rugalmasság a rendszer több funkciójában is megjelenik. A hallgatók tudásának ellenőrzése, a számonkérés a szemantika-orientált oktatási keretrendszerek egyik fő funkciója. A kérdések megalkotása önmagában is összetett feladat, hiszen illeszkednie kell a tématerülethez, az ellenőrzés céljához és a hallgató aktuális állapotához [16].

A Ph.D. munkám keretében végzett kutatásom fő célja egy szemi-automatizált kérdésgeneráló mintarendszer megtervezése és megvalósítása. A mintarendszer feladata annotált szöveges dokumentumból kérdések és válaszlehetőségek előállítása. A kidolgozott rendszermodell feleletválasztós és kiegészítő kérdéseket generál annotált szövegből, külső szemantikai adatbázis felhasználása nélkül. A dokumentum mondatainak annotálásával megvalósítottam a kérdésekre nem használható mondatok kiszűrését, valamint a mondatok típusaira vonatkozó szűrési lehetőségeket. Az *A. 7* Függelék annotációkkal ellátott dokumentumrészletre mutat példát.

A rendszermodell elsőként szövegbányászati előfeldolgozási lépéseket hajt végre, majd saját szemantikai adatbázis épít fel a szavak tartalom alapú klaszterezésén keresztül. A rendszer egyik alapvető modulja a kiemelt szó meghatározását végző osztályozási modul, melyben neurális háló alapú módszer került kidolgozásra. Az általam végzett kísérletek igazolják, hogy a kifejlesztett új koncepció alkalmazható a jóval időigényesebb manuális kérdés-előállítás helyett. Az új módszerben alkalmazott többféle hangolási lehetőséggel (klaszterezéssel kapott szótávolság,

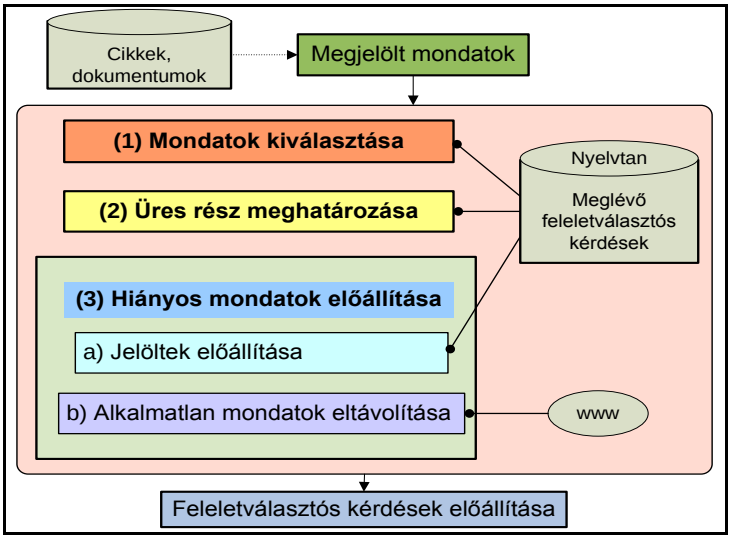
fogalomhierarchia, szubjektív koordináták stb.) pedig elérhető, hogy a kérdések az emberi kérdésekhez hasonlóan tükrözzék a tesztelés alapjául szolgáló tananyag lényegét. Ennek hatását a tesztek eredményei is igazolták.

1.1. Előzmények és kapcsolódó munkák

Az automatizált kérdésgeneráló (Automatic Question Generation, AQG) módszerekhez kapcsolódó vizsgálatok az 1990-es évekig nyúlnak vissza. Ezt megelőző időszakban az ide vonatkozó kutatások a kérdésgenerálás szemantikai aspektusára fókuszáltak, amelyek módszertani bázisként szolgálnak a mai automatizált rendszerekhez. Az AQG egyik jellemzője, hogy más területen elterjedt eszközök speciális integrációját valósítja meg. Ezen területek közé tartozik a szövegbányászat, természetes nyelvi feldolgozás, szemantikus tudásmenedzsment, pedagógiai és szoftverfejlesztés. Módszertani szempontból vizsgálva, a kérdéstípusok [Bloom, 1956] a következő komplex szintek alapján különböztethetőek meg: egyszerű felidézés, megértés, alkalmazás, elemzés, szintézis és kiértékelés.
Az alkalmazott belső algoritmust tekintve, széles körben elterjedt módszer a szabályon alapuló implementáció. Ebben az esetben az emberi szakértők által előre definiált formulák rendelődnek a mondatokhoz. Az algoritmusok második típusa valamilyen tanulási algoritmust használ ahhoz, hogy felismerje a konverziós szabályt, és megtanulja a különböző bemeneti mondatok fontosságát.
Az automatikus kérdésgenerálás során a kutatások legfontosabb feladata a legrelevánsabb fogalmak azonosítása a természetes nyelvű szövegekben. A fogalmak kinyerésének alapötlete Luhn kutatásán [Luhn, 1957] alapul, aki statisztikai eszközöket talált a szavak jelentésbeli viszonyának feltárására. Edmundson [Edmundson, 1963] továbbfejlesztette ezt a módszert a szavak kombinációjának figyelembe vételével, szógyakoriságokkal és a szavak pozíciójával. Kupiec és társai [Kupiec, 1995] kibővítették ezt a módszert az akronimával és megfelelő főnévvel. Frank [Frank, 1999] specifikus tárgykörben megalkotta a kulcsszavak kinyerését, melynek során naiv Bayes osztályozást használt a szógyakoriságra alapozva, és a szó első előfordulásának helyzete explicit paraméterként jelenik meg.
Az automatizált kérdésgeneráló rendszerek létrehozásának kutatásában a domináns megközelítést Miller [Miller, 1995] javasolta, melyben hat kérdéstípus különböztethető meg a WordNet [WordNet, 2010] tudásbázisra alapozva: definíció, szinonim, antonim, hipernim, hiponim és feleletválasztós. A szinonima típusú kérdés esetén a szinonimák kinyerését követeli meg, az antonímia típus esetén a szavak ellentétes jelentését nyerik ki. A hipernima és hiponima fogalmak összefüggnek a kapcsolatok specializációjával és általánosításával.
Coniam [Coniam, 1997] a későbbiekben kifejlesztett automatikus generálási módszert a feleletválasztós kérdésekhez. Meghatározta azokat a szavakat, amelyeknek a szófaj része és az előfordulási gyakorisága megegyezik a hiányos mondatnál a helyes választás szavaival. A szakértők a különböző szótípusokra vonatkozó kérdések előállítása érdekében kiválasztják a mondatokat, az üres részeket és a hiányos mondatokat. Ezeket a kiválasztásokat empirikusan végzik el.
A feleletválasztós kérdések előállítása három lépésből áll:

- kiválasztani a gépi tanuláson alapuló mondatokat,
- megbecsülni a feltételes véletlenszerű mezőn alapuló hiányos részeket,
- előállítani a statisztikai mintán alapuló megtévesztő mondatokat.

A feleletválasztós kérdések előállításának lépéseit az 1.1. ábra szemlélteti [Coniam, 2007].



1.1. ábra: Feleletválasztásos kérdések előállításának menete

Az első lépésben a rendszer kiválaszt néhány mondatot, melyek illeszkednek a vizsgálandó témakörhöz. Ebben a fázisban a mondatokat a szövegből a tanulási preferenciák módszerrel nyerik ki. A tanulási preferencia egy módszer a minták rendezéséhez, osztályozásához. A második lépésben a rendszer megbecsüli a hiányos részeket, felhasználva a CRF (Conditional Random Field) módszert [Collins, 2007]. A CRF olyan keretet biztosít a diszkriminatív valószínűségi modellek felépítéséhez, amellyel szegmentálhatóak és címkézhetőek a mondatok [Lafferty, 2001]. A harmadik lépésben a rendszer létrehozza a hiányos mondatokat. Ebben a fázisban a mondatok jelöltjeit a már létező feleletválasztós kérdések statisztikai mintája alapján generálják. Mitkov [Mitkov, 2003] megközelítésében a hiányos szavak az adott kulcsfogalmakon alapulnak, amelyet a WordNet felhasználásával valósítottak meg. A kérdések felépítését olyan szabályra alapozták, ahol a mondatokat kérdőszavas mellékmondattá alakították. Sumita 2004-ben [Sumita, 2004] javasolt egy hasonló elvű automatikus generáló módszert a feleletválasztós kérdések előállítására. Ebben a módszerben a mondatokban szereplő ige került kiválasztásra a kérdés előállításában. Munkájában a modul három fő lépésből áll:

- a feldolgozandó mondat kiválasztása,
- meg kell határozni a mondat üres részét és
- generálni kell a hiányos mondatokat.

A mondatok kiválasztása, az üres helyek és a hiányos mondatok meghatározása a gépi tanulási módszerek segítségével történik statisztikai és diszkriminatív modellek felhasználásával [Sumita, 2004].

Brown és kutatótársai 2005-ben kifejlesztettek egy rendszert [Brown, 2005], amely képes olyan dokumentumot létrehozni, amely megfelel a felhasználók tudásszintjének és megfelelő feleletválasztós és kiegészítő kérdéseket állít elő. A WordNet segítségével definíciókat, szinonimákat, antonimákat, hipernimákat és hiponimák típusú kérdéseket hozhatunk létre, ahhoz, hogy a szóismeretet fejlesszék felhasználói statisztikák kiértékelésével.

Hoshino és szerzőtársai [Hoshino, 2005] olyan feleletválasztós kérdésekhez javasolt generálási módszert mutattak be, amely szintén a gépi tanuláson alapul. A dokumentum megfelelő mondatainak a struktúrájuk alapján történő kiválasztásához a már létező feleletválasztós kérdések szavait és szófajait vizsgálták. Ehhez a preferenciatanulás módszerét használták. Az olyan kérdéseknél, amelyek nyelvtani tudásra kérdeznek rá, azok a mondatok a megfelelőek, amelyeknek

hasonló nyelvtani struktúrájuk van. Ezért a tanulási szakaszban a preferenciatanulást úgy hajtják végre, hogy a már létező feleletválasztós kérdésekben előforduló szavakat és szófajokat használják. Chen [Chen, 2006] nyelvtani teszteket épített fel a web-en lévő mondatok kinyerésének átalakításával. Az átalakítást úgy végezték, hogy alkalmazták a kézzel készített mintákat, és ezeket felhasználták feleletválasztós kérdések létrehozásához és hibajavító tesztekhez. Rus és szerzőtársai [Rus, 2007] minták, sablonok és speciális jelölő nyelv segítségével vezettek be egyedi módszereket a kérdésgeneráláshoz. A mintákat szemantikai, lexikális és szintaktikai struktúrák jellemzik, míg a sablonok módszereket írnak le a kérdésgeneráláshoz. Lin és szerzőtársai [Lin, 2007] feleletválasztós kérdéseket automatikusan létrehozó rendszert készítettek. Olyan kérdésekre fókuszáltak, amelyek meghatározzák a melléknevet és olyan kérdéseket állítottak elő, amelyek hiányos részei melléknevek. A válaszlehetőségeket a Thesaurus vagy a WordNet használatával készítették. Ezeknek a kutatásoknak egyik problémája az, hogy a mondatok néha egyszerűek vagy túl bonyolultak ahhoz, hogy a tudástesztekben felhasználható kérdéseket reprezentáljanak. Gütl [Gütl, 2008] olyan rendszert valósított meg, amely elvégzi a dokumentumok automatikus összefoglalását ahhoz, hogy azonosítsa a kulcsfogalmakat és ez előállít kiegészítő teszteket és limitált válaszokat.

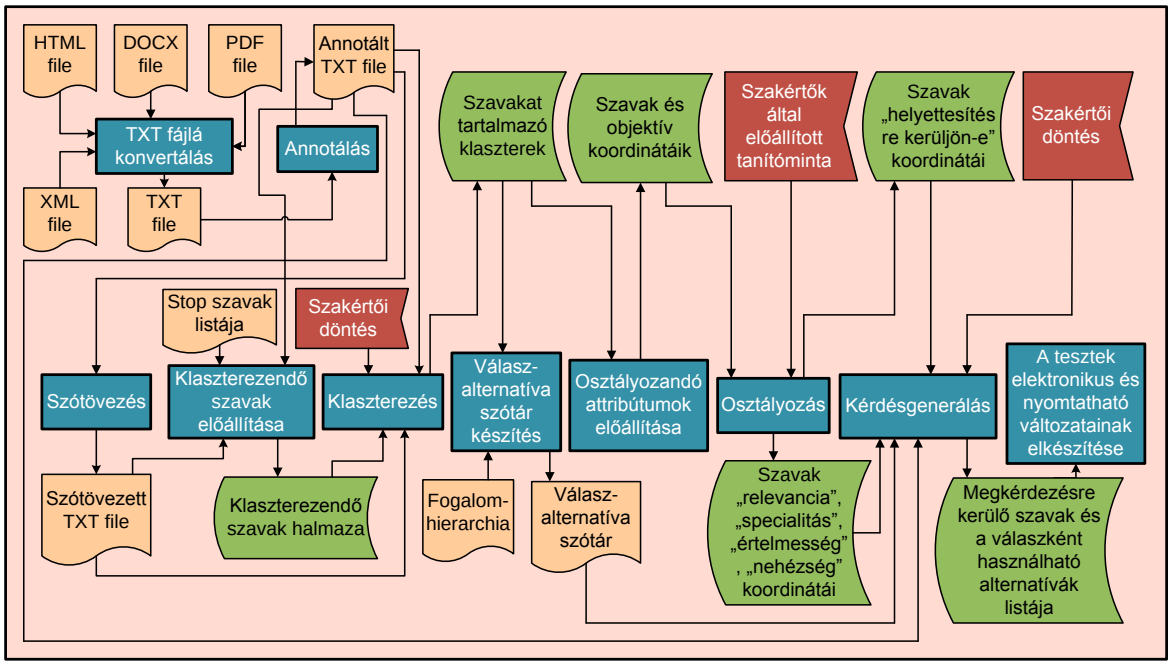
A kutatások jelenlegi állásának kiértékelése azt mutatja, hogy azok a megközelítések, amelyek valamilyen gépi tanulási módszert alkalmaznak, erőteljesen függnek a tudásmintában alkalmazott tudásterülettől és oktatási anyagtól. A legtöbb olyan statisztikai vagy szemantikai módszert alkalmaz, amely a kijelölt feltételek csak egy részét tudja teljesíteni. Az egyik kulcsprobléma az, hogy ezek a módszerek specifikus, domináns nyelvekre korlátozottak az alkalmazott nyelvtani modulok miatt. Másrészt az általuk használt szemantikai tudásbázisok csak azon az adott nyelven használhatók. Munkám elsődleges célja rugalmas és nyitott keretrendszer megtervezése, implementálása és gyakorlati működésének igazolása, amely a magyar nyelvet használja a kérdések és válaszlehetőségek előállítására.

1.2. Az értekezés célja és hatásköre

A disszertáció mintegy határterületként az oktatási, szövegbányászati és tudásmérnökségi diszciplínákat foglalja magába. Az értekezés célja olyan rendszermodell megtervezése és megvalósítása, amely képes annotált magyar nyelvű szöveges dokumentumból, megadható mondattípusok alapján, feleletválasztós és kiegészítő kérdések automatikus előállítására.
A rendszermodell az alábbi fő funkcionális modulokat foglalja magába:

- **előfeldolgozás**, feladata az eltérő szerkezetű és formátumú dokumentumoknak – egységesítési, formalizálási és normalizálási eljárások alkalmazásával – a kérdésgenerálás számára egységesen kezelhető és lényegkiemelést támogató strukturált formára alakítása,
- **klaszterezés**, feladata a dokumentum szavainak szeparált csoportokba szervezése, ahol az azonos csoportba sorolt szavak – a definiált kritériumok értelmében – egymáshoz minél hasonlóbbak legyenek, míg a különböző csoportba soroltak egymástól minél különbözőbbek legyenek,
- **osztályozás**, feladata olyan szabályrendszer feltárása, mely képes az egzaktul meghatározható objektív koordinátákkal definiált térben adott szavakat automatikusan elhelyezni az emberi megérzésekre alapuló szubjektív koordinátákkal definiált térben,
- **kérdésgenerálás**, feladata a bemenetként kapott dokumentumból a kérdésként szolgáló mondatok meghatározása, a mondatokból a kérdést reprezentáló szó kiemelése, valamint a kérdésre adható lehetséges válaszok automatikus előállítása.

Az 1.2. ábrán bemutatott rendszermodell a kidolgozott kérdésgeneráló mintarendszer funkcionális rendszertervét mutatja.



1.2. ábra: A rendszermodell funkcionális rendszerterve

A kidolgozott mintarendszer az alábbi tulajdonságokkal rendelkezik:

- támogatja a különböző bemeneti fájlformátumokat a helyi fájlrendszerből és az Internetes erőforrásokból,
- magyar nyelvű támogatás,
- a tudásterülettől és dokumentumstruktúrától való függetlenség,
- tesztkérdések készítése annotált szövegből,
- válaszlehetőségek előállítása belső, saját fejlesztésű szemantikai adatbázisból,
- annotáció támogatása,
- feleletválasztós és kiegészítő feladatok támogatása,
- elektronikus és nyomtatható feladatlapok készítése,
- konfigurálhatóság, modularitás és kiterjeszthetőség,
- együttműködési képesség a létező e-learning rendszerekkel.

A szakirodalomban jelenleg fellelhető automatizált kérdésgeneráló rendszerek jelentős részét idegen nyelvű szövegek kezelésére fejlesztették ki. A magyar nyelvűek kísérleti stádiuma, illetve a felsorolt jellemzőknek csak részleges támogatása miatt szükséges volt saját fejlesztésű, gyakorlati életben alkalmazható rendszer kidolgozására.

1.3. Útmutató az értekezéshez

Minden fejezet *irodalmi áttekintéssel* kezdődik, majd ezt követi a *saját elmélet* megalkotása az ehhez szükséges módszereket felhasználva. A fejezet befejező része a *mintaprogram* bemutatása, amely igazolja a megtervezett és kidolgozott elméletet.

A *bevezetés fejezet* három alfejezetre tagolódik. Az első alfejezet rövid történeti áttekintést nyújt a kérdésgeneráláshoz szükséges adatokról és feladatokról, majd az automatizált kérdésgeneráló kérdések létrehozásának kutatására fókuszálva az alapvető megközelítéseket tárgyalja napjainkig. Meghatározza azokat a problémákat, amelyek még mindig bizonyos korlátokat támasztanak a használhatóság vonatkozásában. A kutatás céljának megfogalmazása után a megvalósított rendszermodell funkcionális rendszertervét mutatja be.
A *második fejezetben* az e-learning fogalma és fejlődésének ismertetése után a tananyag tartalmának felépítéséről és kiválasztásáról olvashatunk. A digitális tananyagoknak az újrafelhasználhatósága, más rendszerekkel való elérhetősége szabványosítással valósítható meg. A két legfontosabb szabvány: SCORM (Sharable Content Object Reference Model) és a LOM (Learning Object Metadata). A jelenlegi kereskedelmi és ingyenes e-learning keretrendszerek összehasonlítása és kiválasztásainak fontos szempontjait tartalmazza a fejezet.
A *harmadik fejezet* a legismertebb digitális dokumentumformátumok, metaadatok alapvető fejlődési tendenciáival és a metaadatok megadási formátumával foglalkozik. Részletezi a metaadatoknak az XML (Extensible Markup Language) alapú szöveges dokumentumformátumoknál történő felhasználási lehetőségeit. A fejezet bemutatja, hogy az elterjedt XML alapú technológiák felhasználásával – mint például a DITA (Darwin Information Typing Architecture) modell – hogyan valósítható meg a tantárgy oktatási modellje.
A *negyedik fejezet* bemutatja a kidolgozásra került rendszer moduljait, valamint a közöttük lévő kapcsolatokat jelentő adatáramlásokat. Kidolgoztam egy dokumentumrendszer szerkezeti modellt is, amely a rendszermodell bemeneti és kimeneti adatait gráf struktúrájú adatbázisban tárolja.
Az automatikus kérdésgenerálás megvalósíthatóságának fontos feltételét képezi a dokumentum szavainak megfelelő klaszterekbe szervezése. Ezt fejti ki részletesen az *ötödik fejezet*.
A *hatodik fejezet* az osztályozás szabályrendszerének feltárási folyamatát mutatja be, mely képes az objektív térben adott szavakat automatikusan elhelyezni az igényeink szerint definiált szubjektív térben. A hálózat betanítását felügyelt tanítási módszerrel végeztem el. Az osztályozó rendszer hatékonyságát tesztdokumentum alapján határoztam meg.
A *hetedik fejezetben* ismertetésre kerül a kérdésekre adható lehetséges válaszok automatikus előállításának folyamata.
A *nyolcadik fejezet* első része a kérdések automatizált előállításával foglalkozik. A megvalósított kérdésgeneráló mintarendszer gyakorlati alkalmazhatóságát a tesztek eredményei igazolják. A második rész az eredmények kiértékelését tartalmazza részletesen.
A *kilencedik fejezet* az új tudományos eredményeket és a további kutatási feladatokat foglalja össze.

2. fejezet

E-learning keretrendszerek

2.1. Az e-learning fogalma, története

Az e-learning, olyan számítógépes hálózaton elérhető nyitott képzési forma, amely a tanítási-tanulási folyamatot megszervezve, a számítógépes interaktív oktatószoftvert egységes keretrendszerbe foglalva, a tanuló számára hozzáférhetővé teszi [Forgó, 2005].
Az e-learning korai formája a CBT (Computer Based Training) [Barron, 1993], azaz számítógéppel segített tanulás, ahol az oktatóanyag valamilyen digitális adathordozón található. Így a tananyag hordozhatóvá vált, és a hallgató annyiszor lejátszhatta, ahányszor szükséges volt a tanulás során. Helytől és időtől való függetlenség jellemzi. Hátránya, hogy a tanár és a tanuló között semmilyen kapcsolat nem épült ki, és csak az adathordozón található tartalmat közvetítette.
Az információs technológiák fejlődése során lehetőség nyílt arra, hogy az elektronikus tanulás valódi képzésmenedzsmenttel társuljon. Ezt az oktatástípust WBT-nek (Web Based Training) [Helic, 1994], azaz számítógépes hálózaton keresztül megvalósuló tanulásnak nevezték. Az oktatásban megjelenik a hálózati kommunikáció, a hallgató a tanárral e-mail, chat, fórum, videokonferencia formájában tartja a kapcsolatot. Az e-learningről az 1990-es évek elejétől beszélhetünk, amikor az Egyesült Államokban először megjelent, mint képzési forma. Maga az e-learning elnevezés Európában az 1990-es évek végén honosodott meg. Az oktatás jellemzője, hogy az információ továbbítása informatikai hálózaton – LAN (Local Area Network), WAN (Wide Area Network), Internet vagy intranet – keresztül történik [15]. A kommunikáció és a tanítási-tanulási folyamatok tervezése, szervezése, kivitelezése és értékelése is ezen a számítógépes hálózaton keresztül valósul meg a hallgató, a tanár, és az oktatásszervező között.
A digitalizáció, – amely kezdetben a helyhez kötött médiumokkal történő tartalom feldolgozást és kommunikációt forradalmasította – napjainkra a hálózati kommunikációs formák merőben új részterületeit alakította ki [Forgó, 2009]:

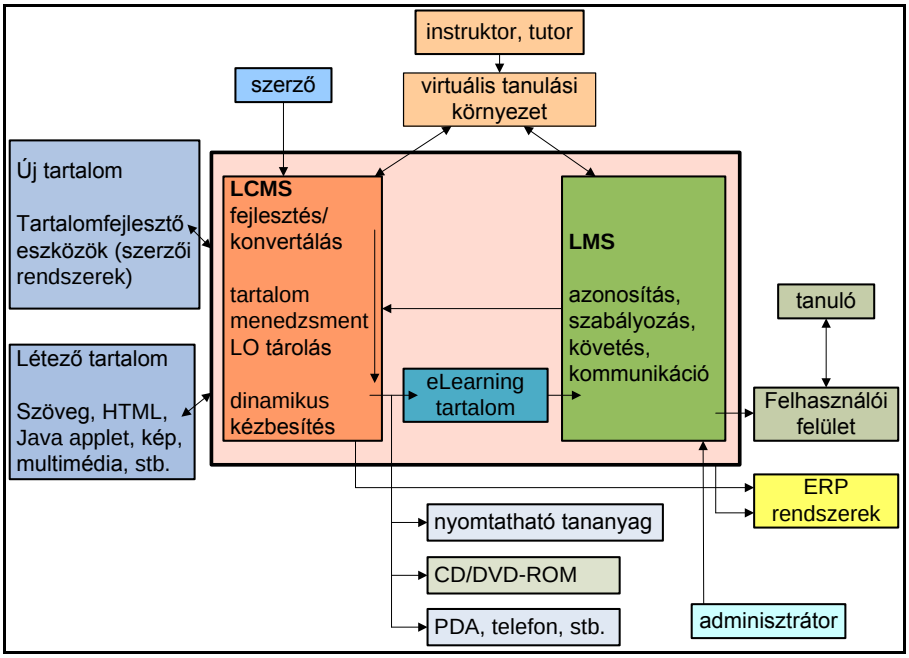
- Web 2.0-án alapuló társas-közösségi szerveződési és tanulási formákat és tanulóközpontú webes környezetet (e-learning 2.0),
- digitális technológián alapuló televízión keresztüli interaktív tanulást (t-learning),
- vezeték nélküli (mobil) telefónia általi információ- és ismeretszerzést (m-learning).

Az e-learninggel támogatott vegyes típusú oktatásban a tér- és időbeli korlátokat már digitális (off-line, on-line) technológia-technika révén valósítják meg, melyek a papíralapú tananyagok mellett kezdetben komplementer és napjainkban egyre inkább alternatív módon vannak jelen az elektronikus tanulásban [Benedek, 2003].
Az internet megjelenése és szolgáltatásainak széleskörű terjedése – webes felületen (Web 1.0) – nemcsak a gazdaságra és kommunikációs formákra hatott, hanem a tanulás eszköztárának szélesítéséhez is elvezetett. Kezdetben a tanulási tartalmak szöveges, képi illusztrációkkal ellátott, multimédiás anyagok formájában voltak elérhetők. A tanulásszervező programok (Learning Management System, LMS), a tartalom közreadásán és az adminisztrációs lehetőségeken túl már olyan eszközt is tartalmaztak, amely a tanulási folyamatot keretek közé szervezve lehetőséget adtak a hallgatói aktivitás növelésére [Forgó, 2005].
A Web 2.0-án alapuló társas-közösségi szerveződési forma kialakulását követően a tanulási formákban is megjelent az e-learning 2.0, a tanulóközpontú webes környezet formája. A Web 2.0

típusú tanulás elméletét a konnektivizmus – a hálózatalapú tanulásfelfogás – írja le, mely a digitális korszak tanuláselméletének fogható fel [Siemens, 2005].

2.2. Az e-learning tanulmányi keretrendszere és tartalomkezelő rendszere

Az e-learning tanulmányi keretrendszere az LMS [Weaver, 2008], amely az alapvető oktatási szervezési feladatoktól kezdődően a tananyagok megjelenítéséig a legkülönbözőbb tanulmányi funkciókat ellátja. Az LMS rendszer kiválasztásakor fontos szempontok a következők: a hallgatói jelentkezés automatikus kezelése, automatikus számlázási lehetőség, előre elkészített tananyagok importálása, nem az LMS-ben lévő tananyagok kezelése, más rendszerekhez való integrálhatóság és a távoli adminisztráció lehetősége. Mivel az LMS rendszereket intézményeknek megfelelően hozzák létre, nem volt biztosított a hallgatók számára az átjárás lehetősége. Ennek megkönnyítésére tanulóközpontú szemléletre épülő személyes tanulmányi környezetet valósítottak meg, aminek a neve a PLE (Personal Leraning Environment). A rendszerek tartalmazzák a naplóírás és olvasás lehetőségét, egy helyen több weblap tartalmának figyelését és kapcsolatok ápolását. A hallgató létrehozza saját, egyéni tanulmányi környezetét, aminek segítségével kapcsolódik a kívánt oktatói keretrendszerekre.
A tartalomkezelő rendszerhez (Learning Content Management System, LCMS) [Schluep, 2003] olyan szoftverek tartoznak, amelyeket elektronikus kurzusok fejlesztéséhez használnak. Az LCMS megkönnyíti a szerzőnek a munkáját, segít létrehozni és tárolni az oktatási egységeket, segít az adminisztrációban és a tanfolyamok, leckék, oldalak engedélyezésében. Az LCMS-nek biztosítania kell: a közös munkafelületet a tartalom létrehozásához és módosításához, a tanfolyam alapanyagainak tárolását, és a tartalom újrahasználhatóságát, dokumentumok importálást és a kurzusok, leckék, elemek importálását és exportálását.
Az LMS és LCMS rendszerek önállóan, egymástól függetlenül is működhetnek, de magas szinten szervezett e-learning környezetben a két keretrendszer alkalmazása szerves együttműködésben valósul meg, melyet a 2.1. ábra szemléltet [Papp, 2005].



2.1. ábra: Az e-learning rendszer felépítése

2.3. Az IEEE e-learning szabványai

Manapság egyre több digitális tananyagot készítenek. A tananyagoknak az újrafelhasználhatóságát, más rendszerekkel való elérhetőségét biztosítani kell, ami szabványosítással elérhető. Amit a szabványoktól elvárunk: együttműködési képesség, újrahasznosíthatóság, kezelhetőség, elérhetőség, tartósság, megengedhetőség.
A LOM [Holzinger, 2001] nemzetközileg elfogadott e-learning szabvány, amit 2002. június 12-én deklarált az IEEE 1484.12, 1484.14 (Institute of Electrical and Elektronics Engineers) szervezet. Tananyagelemek, tanulási egységek (Learning Object, LO) leírására alkalmazzák. Maga a szabvány az LO méretére nem ad meghatározást, de ez a legkisebb értelmes önálló egység. A tanulási egységet jellemző leírást nevezzük metaadatoknak. A metaadatok segítségével a tananyagelem minden lényeges tulajdonsága leírható. A szabvány meghatározza a LOM szintaktikai és szemantikai jellemzőit. A tananyagelemek tartalmazhatnak bármilyen multimédiás elemet, tanulási célt, instrukciós szoftvert és tanulási eseményt. A LOM célja, hogy meghatározza azokat a minimálisan szükséges jellemzőket, amelyekkel a tanulási egységek könnyen kezelhetők, minősíthetők és visszakereshetők. A LOM tananyagelemekből és a velük kapcsolatos erőforrásokból nyert adategyüttes, amely lehetővé teszi:

- a tanulási egységek cseréjét és megoszthatóságát különböző oktatási-tanulási rendszerek között,
- a tanárok és tanulók számára a tananyagelemek keresését, felhasználását, minősítését,
- a tananyagok moduláris fejlesztését.

A tananyagelemet leíró adatelemek különböző kategóriákba sorolhatók. A LOM kategóriáit a 2.1. táblázat mutatja.

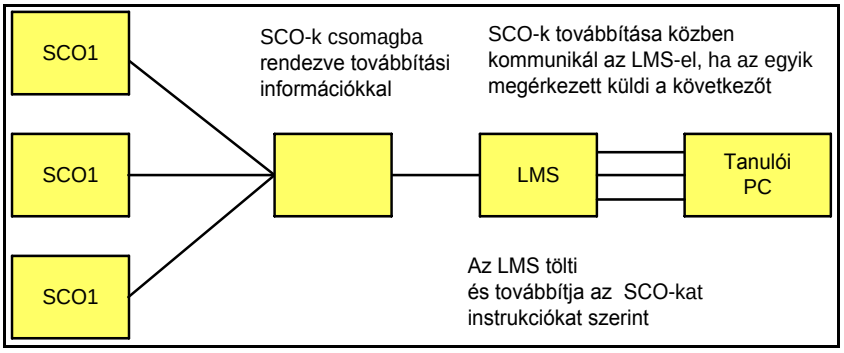
Kategória neve	Leírás
Általános	A tananyagelem általános leírására szolgál.
Életciklus	Az erőforrások életciklusával kapcsolatos tulajdonságok.
Meta-metaadat	Magáról a metaadatról ad információt.
Technikai	Az erőforrások technikai jellemzői.
Oktatási	Oktatási és pedagógiai tulajdonságok.
Tulajdonjogok	Szellemi tulajdonjogok és felhasználói jogok feltételei.
Kapcsolat	Más tananyagelemekhez való kapcsolódást jellemzi.
Megjegyzés	Megjegyzések a szolgáltatások oktatási használatával kapcsolatban.
Besorolás	A tananyagelemek kapcsolata másik besorolási rendszerhez.

2.1. táblázat: LOM kategóriái

A SCORM napjainkban a legáltalánosabban elfogadott e-learning szabvány. 1997-ben az ADL (Advanced Distributed Learning) kezdeményezést az Amerikai Egyesült Államok Védelmi Minisztériuma, a Fehér Ház Tudományos és Műszaki Irodája és Munkaügyi Minisztérium hozta létre. A különböző szabványajánló szervezetek által kidolgozott ajánlásokat egységes rendszerbe foglalja. Ez a modell a SCORM [Chu, 2005].
A SCORM alapján elkészített tananyagoknak meg kell felelniük:

- újrafelhasználhatóságnak: a tananyag könnyen módosítható legyen,
- elérhetőségnek: a tananyagok kereshetők és könnyen elérhetők legyenek,
- platformfüggetlenségnek: a tananyagok a különböző hardver- és szoftverkörnyezetben használhatók legyenek,
- tartósságnak: ha a futtató rendszer változik, kevés módosítással felhasználható legyen a tananyag.

Az ADL a SCORM szabvány együttes felépítését, alkalmazásának módját egy nyilvános dokumentációban teszi közzé. A szabványt négy fájlban írták le, ezek a következők: áttekintés, tartalomhalmozási modell, futásidejű környezet, sorrend és navigáció. Az első fájl általános tájékoztatást ad az ADL szerepéről, küldetéséről, a többi fájlban található a szabvány.
A tartalomhalmozási modell (Content Aggregation Model, CAM) a szerzői rendszerek gyártóinak és a tartalomfejlesztőknek ad eligazítást. A SCORM-ban a tananyagok úgynevezett megosztható tartalomegységekből, SCO-ból (Sharable Content Object) épülnek fel. Az SCO-k a legkisebb címezhető egységek, amelyeknek az újrafelhasználhatóságot az biztosítja, hogy nem tartalmaznak hivatkozást külső elemre, illetve más SCO-ra. A megosztható tartalomegységek tananyagelemekből épülnek fel. A tananyag szerkezetét XML fájl írja le, amely tartalmazza az összes SCO metaadatait.
A SCORM (Sharable Content Object Reference Model) a tanulási technológiát két funkcionális elemre osztja. A kulcs elemek az LMS és a SCO. Az SCO az újrahasznosító tanulási elem szabványosított formája. LMS bármilyen rendszer lehet, ami tanulói információt tárol, elindíthat és kommunikálhat a tanulási tananyagelemekkel és értelmezni képes azokat az utasításokat, amelyek megmondják, hogy melyik elem következik. A SCORM modell további elemei olyan eszközök, amelyek tananyagelem-sorozatokat hoznak létre és a tanulás nagyobb egységeivé szerkesztik össze. A 2.2. ábra a SCORM szabvány modelljét mutatja [Fogarasi, 2009].



2.2. ábra: SCROM szabvány modellje

2.4. E-learning keretrendszerek tudásfelmérő moduljai

Az e-learning keretrendszer olyan szoftver, melynek segítségével számítógépes hálózaton keresztül személyre szabott tanulási folyamat végezhető és szervezhető [Hauser, 2005]. A keretrendszerek használatának és fontosságának egyre növekvő szerepe van a jelenlegi internet-alapú társadalomban. Az első e-learning rendszereket csak passzív prezentációjú rendszerekhez lehetet használni, kötött tartalommal és tananyaggal. Mivel a hallgatóknak különbözőek a képességeik és igényeik, ezért a kötött tartalom alacsony hatékonyságot eredményez. Az oktatás-tanulás minőségének javítása érdekében olyan kereskedelmi vagy ingyenes szoftverek készültek, amelyek biztosítják a magas szintű interaktivitást és személyre szabhatóságot. Az e-learning keretrendszereknek fontos komponense a *tesztmodul*, amely különböző szerepeket tölt be a rendszeren belül:

- a modul használhatóvá válik a felhasználók számára,
- mérni lehet az elért tudásszintet,
- visszajelzést ad oktatók és a fejlesztők számára, hogy mennyire sikeres az oktatási tananyag és a módszer elkészítése.

A jelenlegi kereskedelmi és ingyenes e-learning rendszerek, mint az ILIAS, MOODLE (Modular Object-Oriented Dynamic Learning Evrironment), OLAT tartalmaz kifinomult tesztmodult

különböző kérdéstípusokkal. A kérdéseket tartalmazó feladatlapok megtervezése és megvalósítása manuálisan történik, így a kérdőív elkészítése relatíve drága feladat. Másik hátránya a keretrendszereknek, hogy megadott szöveges dokumentum alapján nem képesek automatikusan kérdéseket előállítani.
A szakirodalom áttekintése során számos keretrendszerrel találkoztam, ezek a következők: ATUTOR, BAZAAR, CLAROLINE, DOCEBOLMS, DOKEOS, FLE3, GANESHA, ILIAS, INTERACT, KEWL, LAMS, MANHATTAN, MOODLE, OLAT, SAKAI, SPAGETTI LMS, STUD.IP, TINYLMS. A keretrendszerek kiválasztásánál fontos szempont, hogy milyen szolgáltatásokat nyújtanak a felhasználók számára. A legfontosabb szempontok a keretrendszerek vizsgálatánál: platformok támogatottsága, testreszabhatóság, adminisztrátori hatáskör, nyelvi változatok, szabványok, eszközrendszer, modulok, *tesztek*, szerepkörök és támogatás [Papp, 2005].
A szempontok közül elsősorban a tesztmodult vizsgáltam a kérdésgenerálás szempontjából, figyelembe véve a programozhatóságot és automatizálhatóságot. Minden rendszer lehetőséget ad valamilyen formában tesztek készítésére, a kivitelezésben azonban jelentős különbségek vannak. A közös metszet a tesztek típusait tekintve a feleletválasztós, az igaz-hamis valamilyen realizációja és a szöveges válasz. A részletgazdagabb rendszerek ezt kiegészítették a párosítás lehetőségével, a beágyazásos kérdéssel, a sorrendbe rendezéses és valamilyen numerikus kérdéstípussal. A rendszerek túlnyomó többsége a kérdéslapokat közvetlenül jeleníti meg a kurzusfelületen s alig akadt keretrendszer, amely a kérdéseket tesztadatbázisokban tárolja. A MOODLE a kivételek közé tartozott. Másik fontos szempont a tesztek exportja/importja. A rendszerek többsége csupán a saját formátumát ismeri. Van olyan, hogy több formátum exportja és importja támogatott a kérdéstípusok korlátozása mellett. Ilyen a MOODLE, amely a piacvezető kereskedelmi rendszerek a WEBCT és a BLACKBOARD kérdésformátumát is kezeli.
Magyarországon a felsőoktatásban a MOODLE keretrendszer terjedt el. Rendkívül dinamikusan fejlődő, s rohamosan terjedő rendszer, mely előtérbe helyezi a könnyű használhatóságot a telepítéstől a közösségi életig. A MOODLE keretrendszert, mint alkalmazást, számos tudásfelmérő modul egészíti ki, melyek közül az alábbiakat emelem ki.
A *Tesztmodul* segítségével lehetséges a rendszeren belül manuálisan kérdéseket rögzíteni, illetve azokból teszteket generálni. A Tesztmodul alapját a kérdésadatbázis adja, melyben tetszőleges számú és adott típusú kérdés vehető fel. A kérdésadatbázisban tárolt kérdésekből generálhatók kérdéssorok, akár véletlenszerű kiválasztás, vagy akár rögzített kiválasztás segítségével. A *Felmérésmodul* használatával elektronikus alapú felmérést, esetleg értékelést tudunk készíteni és publikálni a felhasználók felé. A rendszer felhasználói a feltett kérdésre kiválaszthatják a megadott lehetőségek közül a számukra leginkább megfelelőt. A felmérés végén az adott jogosultságú szereplő kiértékelheti a válaszokat, értékelheti a felmérést. A *Feladatmodul* a tanulási folyamatot és a hatékony felkészülést támogatva a rendszerben lévő tanár szerepkörű felhasználók a tanulók számára önállóan megoldható feladatokat oszthatnak ki, melyeket a keretrendszeren keresztül publikálnak, kapnak rá megoldást, és a rendszer felületén is értékelik a beérkező megoldásokat. A *Fogalomtár* segítségével szószedetet állíthatunk össze, melyet különböző szempont szerint tudunk rendezni, illetve csoportosítani. A felvett szavakhoz, kifejezésekhez magyarázatot rendelhetünk, melyhez természetesen keresési funkció is tartozik.

Az elektronikus tanulási környezet kialakításához ma már a keretrendszerek széles választéka áll rendelkezésre. A nagy szoftvergyártók üzleti alapon fejlesztett rendszerei mellett egyre nagyobb számban találhatók meg az ingyenes, szabad forráskódú, de egyre több szolgáltatást nyújtó rendszerek is.
A szabad forráskód előnyei:

- a fejlesztés folyamatossága,

- a szolgáltatás folyamatossága,
- a termék testreszabható, új lehetőségekkel bővíthető,
- díjmentesen használható.

A legfontosabb szabad forráskódú e-learning keretrendszerek összehasonlítását a 2.2. táblázat mutatja, amely a 2011-ig fellelhető kínálatot foglalja össze [LMS rendszerek, 2008], [Alshomrani, 2012].

LMS	Kompatibilitás	Tesztmodul	Használat
MOODLE	Linux, Unix, Windows, Mac OS X, FreeBSD, Támogatja a PHP, MySQL, Apache rendszert. Verzió 2.2.	Minden teszttípust támogat, Kérdéslapokat közvetlenül jeleníti meg, Biztosítja a tesztek exportját/importját.	Naponta átlagosan 500 alkalommal töltik le. 66.719 regisztrált webhely, Összesen 6 millión felüli kurzus.
ILIAS	Linux, Unix, Windows, Mac OS X, FreeBSD, Támogatja a PHP, MySQL, Apache rendszert. Verzió 4.2.1.	Minden teszttípust támogat, Kérdéslapokat közvetlenül jeleníti meg, Biztosítja a tesztek exportját/importját.	115 ország és 2246 szervezet használja, Tesztreszabható főmenük, Könnyen használható.
CLAROLINE	Linux, Unix, Windows, Mac OS X, FreeBSD, Támogatja a PHP, MySQL, Apache rendszert, Verzió 1.8.	Minden teszttípust támogat, Kérdéslapokat közvetlenül jeleníti meg, Részben biztosítja a tesztek exportját/importját.	35 nyelven érhető el, több mint 80 országból vannak felhasználói.
BODINGTON	Linux, Microsoft, Mac OS X, UNIX. Támogatja a PHP, MySQL, Apache rendszert.	Teszttípusokat csak részben támogatja, Részben biztosítja a tesztek exportját/importját.	A Leeds Egyetem, UHI Millennium Institute és a University of Oxford közös fejlesztése.
DOKEOS	Linux, Unix, Windows, Mac OS X, FreeBSD, Verzió 2.0, Támogatja a PHP, MySQL, Apache rendszert.	Teszttípusokat csak részben támogatja, Kérdéslapokat közvetlenül jeleníti meg, Adatok importálhatók CSV vagy XML fájlokból.	Harminc nyelven, ezernél is több szervezet használja. A Ghent Egyetem és a brüsszeli Vrije Universiteit közös fejlesztése.
.LRN	Linux, Microsoft, Mac OS X, UNIX, Támogatja a PHP, MySQL, Apache rendszert.	Teszttípusokat csak részben támogatja, Kérdéslapokat közvetlenül jeleníti meg.	Közel félmillió felhasználó 18 országban.
ATUTOR	Linux, Unix, Windows, Mac OS X, FreeBSD, Támogatja a PHP, MySQL, Apache rendszert, Verzió 2.0.3.	Teszttípusokat csak részben támogatja, Kérdéslapokat közvetlenül jeleníti meg, Részben biztosítja a tesztek exportját/importját.	Világszerte több, mint 17.000 regisztrált telepítés.
OLAT	Microsoft Windows, Mac OS X, Linux, Solaris, UNIX. Támogatja a PHP, MySQL, Apache rendszert.	Minden teszttípust támogat, kérdéslapokat közvetlenül jeleníti meg, Biztosítja a tesztek exportját/importját.	Népszerű az európai felsőoktatási közösségekben.
SAKAI	Microsoft Windows, Mac OS X, Linux, Solaris, UNIX, Támogatja a PHP, MySQL, Apache rendszert.	Teszttípusokat csak részben támogatja, Kérdéslapokat közvetlenül jeleníti meg.	Világszerte számos elismert egyetem használja.

2. 2. táblázat: Szabad forráskódú e-learning keretrendszerek

A 2.2. táblázatban ismertetett keretrendszerek részletes elemzése alapján megállapítható, hogy azok nagyon sok funkciót biztosítanak a felhasználók számára. A kérdésgenerálás nézőpontból vizsgálva a *tudásfelmérő modulokat* megállapítottam, hogy a kérdések megszerkesztése és az ehhez tartozó válaszok megalkotása manuálisan történik. Ez sok időt vesz igénybe, aminek a költsége rendkívül magas. Ezen okok alapján célszerűvé vált egy szemi-automatizált kérdésgeneráló rendszer megvalósítása. E rendszer összekapcsolása a meglévő keretrendszerekkel – mint például a MOODLE keretrendszerrel – a programozhatósági és az automatizálási szempontokat is figyelembe véve további kutatási feladatot jelent.

A tananyagfejlesztés kísérleti megvalósításához 2003-ban a ME Comenius Főiskolai Karon egy projektet valósítottam meg. Ebben a projektben sor került az adott képzési témához kísérleti e-learning tananyag készítésére és kipróbálására [15]. Az oktatásban használható e-learning rendszer *tesztmodul* részét terveztem meg, mely tartalmazza a tanuláshoz szükséges oktatási segédletet. Ezen oktatási segédletet feleletválasztós kérdések és válaszok formában az adatbázisban tároltam le. Ezt felhasználva a hallgató több feladatból álló tesztet készíthet véletlenszerű választással. A beérkező válaszokat a rendszer rögzíti és értékeli. Mindez az Interneten keresztül zajlik. Az alkalmazás arra is képes, hogy több tantárgyból válasszon kérdéseket. A rendszer adminisztrálása a hálózaton keresztül is történhet. A tantárgyakhoz szükséges oktatási segédletet a ME CFK szerveren tároltam azért, hogy bármikor elérhető legyen. A rendszer alappillérét két tantárgy képviseli. A vizsgasorok készítésénél az oktató meghatározza, hogy milyen jellegű kérdéseket szeretne az adott vizsga során feltenni a hallgatónak. A milyenség jelenti a kérdések nehézségi fokát, elméleti vagy gyakorlati voltát. Az oktatónak először archiválnia kell a vizsgasort, kinyomtatni és csak ezután lesz elérhető a vizsgázó számára. A vizsgasor kérdéseit és a helyes válaszokat a vizsga után a hallgató ellenőrizheti a számítógéppel [17].

2.5. E-learning keretrendszerek összefoglalása

Az e-learning fogalma és fejlődésének ismertetése után az e-learning tanulmányi keretrendszert és a tartalomkezelő rendszert mutatta be a fejezet. Röviden elemeztem a két legfontosabb e-learning szabvány a SCORM és a LOM feladatait. A jelenlegi ingyenes e-learning keretrendszerek összehasonlítása és kiválasztásainak szempontjait tartalmazza a fejezet, kiemelve a *tesztmodul*t.

3. fejezet

Digitális dokumentumok metaadatai és keretrendszere

3.1. A dokumentumok metaadatai

A hatékonyság egyre döntőbb szemponttá válik nemcsak a hagyományos szövegszerkesztésben, hanem a dokumentum szerkesztési folyamatában is. Az alapvető cél a technológia fejlődésében többek között az automatizált szöveggenerálás, a már meglévő dokumentumok újrahasznosítása és rugalmas megjelenítése. Ezen cél elérésére új módszereket kell kifejleszteni a szerkesztési és dokumentumkezelési eljárásokban. Az új módszerek új dokumentummodelleket kívánnak meg, az alapdokumentumokat ki kell egészíteni hozzáadott metaadat információs elemekkel [6].
A metaadat elnevezés [Timpf, 1996] arra utal, hogy ezen elemek adatok az adatokról, ahol teljesül, hogy

- a metaadatok is adatok,
- a metaadatok relatív szereppel bírnak,
- a metaadatok leíró adatok.

A metaadat kezelés fejlődése során a kapcsolódó tárolt információ mennyisége és jellege is folyamatosan növekedett. A metaadatok implicit vagy explicit formában jelennek meg. A kezdeti időszakban az implicit mód dominált, amikor is a laza szétcsatolt és heterogén tárolás a jellemző. Az explicit metaadatok alkalmazásának első fő formája az adatbázis-kezeléshez kapcsolható [12]. Az adatbázis-kezelő rendszerek metaadatai általában nagymennyiségű információt tartalmaznak, olyan témakörökben, mint az adatbázis felépítése, biztonsági információk vagy üzleti szabályok és megszorítások. Ezeket az információkat ugyanolyan alakban tárolják, mint a normál üzleti adatokat, és ugyanolyan parancsfelülettel lehet azokat kezelni. Az adatbázisoknál laza csatolás figyelhető meg az adatok és metaadatok között.
Szorosabb kapcsolat valósul meg a napjainkban elterjedő adattárolási modellben az XML formátum használatánál. Az XML formátumnak sok előnye van a hagyományos formátumokkal szemben, mint például a rugalmasság, a szabványosítás és az erős feldolgozás [11]. Míg a korábbi formáknál erősen korlátozott volt a metaadatok jelentési köre, addig az XML formánál a tervező szabadon definiálhat egyedi metaadatokat, melyek a normál adatokhoz hasonlóan azokkal együtt jelenne meg a dokumentumban. Míg a korábbi modelleknél a keretrendszer szeparálta az adat és metaadat közötti határt, addig az XML-nél az alkalmazásra van bízva a szeparáció megvalósítása. E rugalmasság következtében a metaadatok szerepe is jelentősen megnövekedett az utóbbi időben. Mint, ahogy a korábbi elemzések is mutatják [4], a metaadat nélküli dokumentumok automatizált feldolgozása jelentősen összetettebb feladat, mint a metaadatokkal támogatott dokumentumok kezelése. Ezen metaadatok pótolhatják a még hiányzó szemantikai háttér tudásbázis-hiányát, lokális információkat biztosítva. Természetesen ez a megoldás csak egyfajta kompromisszumnak tekinthető, hiszen számos buktatót is rejt magában.
Összességében, az XML szintű metaadat kezelés előnyei a következőkben foglalhatók össze:

- rugalmasság,
- egyszerű kezelhetőség,
- kis számítási, kezelési költség.

A hátrányok, veszélyek oldalán az alábbi jelenségek adhatók meg:

- lokális értelmezés,

- inkonzisztencia,
- hiányos információk.

Ezen hiányosságok csak globális ontológia szerver megvalósulásával lesznek majd megoldhatók.

3.2. Metaadat megadás formátumai

Az adatkezelés rendszerekben az átadandó információkat úgynevezett információatomokra bonthatjuk fel. Az információatomot egy hármassal reprezentálhatjuk [Lassila, 1999], melynek elemei: szubjektum, amiről állítunk valamit; predikátum, amit állítunk; és az érték vagy objektum. Ugyan az adat, mint esetleg önálló jelsorozat jelenik meg, használatához mögötte mindig ott kell állnia a jelentésnek is. Önállóan álló 36-os szám nem hordoz információt, csak a kapcsolatrendszerével, kontextusával kibővített rendszer alkalmas új ismeret kódolására. A kódolás azonban többszintű folyamat, melynek egyfajta modelljét tükrözi Sieber [Sieber, 2006] modellje. Ebben a modellben a legalacsonyabb szint az adatpéldány, a jelsorozat szintje. Például a 36 megjelenhet arab és római számjeggyel, balra vagy jobbra dőlt számokkal. Ezen jelek mindegyike egy-egy külön példányt képvisel. Az értelmezés középső szintjén az azonos kódolási rendszerhez tartozó példányokat fogjuk össze. Így például egy-egy külön elem lesz a római és az arab számot használó reprezentáns. Az adat harmadik szintje az absztrakt jelölő elem, mint például a 36, mint szám, mely összefogja az összes lehetséges reprezentációs alakot. Az értelmezés következő szintjén az absztrakt elemhez társulnak a kontextusbeli kapcsolatai, megadva az információ atomot. Az elemi szemantikai szint után értelmezhetők további szintek, melyek meghatározzák a lokális, domain szintű és a globális kapcsolatrendszereket. Az értelmezés említett elemei már a pragmatikai és apobetikai oldalakhoz köthetők.

Az adatérték jelentésének, azaz kontextusának meghatározása természetesen további adatelemeket igényel. Ezen adatelemeket, mivel nem magát az alapértéket adják meg, hanem annak környezetét, metaadatoknak nevezik. A metaadatoknak fontos szerepe van a kontextus megadásán keresztül az intelligens, automatizált adatfeldolgozás hátterének biztosításában.

A metaadatok jelenléte mindig is fontos eleme volt a különböző adattárolási rendszereknek. Az egyszerűbbnek tekinthető relációs adatkezelés esetében [Date, 1995], [Kovács, 2009] az adatbázisséma elemei szolgálnak a metaadatok tárolására. Ebben a körben a kulcs, idegen kulcs, szerkezet és integritási elemek adják meg a letárolható információatomok körét. A metaadat-kezelés alapvetően rejtve marad a végfelhasználók előtt, habár lehetőségeiket nagyban befolyásolják ezen elemek. A relációs adatbázis-kezelésen belüli metaadat-kezelés további lényeges vonása, hogy az adatok és metaadatok szeparáltan kezelhetők és a metaadatok felhasználása problémakör-független értelmezésű, emiatt nem az információkezeléshez, hanem az adatkezeléshez kapcsolható.

A relációs modell igencsak korlátozott metaadatrendszerével szemben nagy előrelépésnek tekinthető a szemi-strukturált, XML alapú adatrendszerek metaadat-kezelése. Az XML [Bosak, 1999] adatkezelés egyik fontos jellemzője, hogy egységesen, közösen tárolja az alapadatokat és a hozzá kapcsolódó metaadatok leírását, annotációkat. A XML formátum esetén a legnagyobb lehetőséget az adja, hogy a szerkezeti elemek alapvetően tetszőlegesen alakíthatók ki, a szerkezeti elemek azonosítása is tetszőleges névvel történhet. Ezen lehetőséget kihasználva az adatelemek jelentését a befoglaló elemek elnevezésén keresztül, vagy a hozzá kapcsolható direkt annotációkon keresztül adhatjuk meg. Az XML szabvány ugyan lehetőséget ad tetszőleges kontextus-információk tárolásra, azonban nem biztosít eszközöket a magasabb szintű feldolgozási műveletekre, nem értelmezhető például automatikusan a specializáció relációja.

A valós problémakörhöz, domain-hez közelebb álló metaadatrendszerek elsőként a szemantikai adatmodellekben jelentek meg. Az ODL (Object Definition Language) modell [Catell, 1997] az objektum orientált modellezési világ eszközeit építette be a metaadat rendszerébe. A kidolgozott rendszer nagyon jól kiszolgálja a szoftverfejlesztés igényeit, azonban nem biztosít kellő kifejező erőt az általánosabb, absztraktabb problémáknál. A későbbi leíró nyelveknél a gráf alapú reprezentációs formák terjedtek el, mivel ezek alkalmasabbak a rugalmas elem- és kapcsolatrendszer megadására. A gráfban a csomópontok rendszerint a fogalmakat, az élek pedig a relációkat jelölik.
Napjainkban, ezen egyedalapú modellek kiterjesztésére az ontológiamodellek terjedtek el az adatok kontextusának megadásánál. Az ontológia-rendszerek tudják biztosítani a magas szintű adatkezelési ismeretek hátterét [Sloman, 2005]. Az ontológiai-rendszerek fontosabb jellemzői: rugalmas kapcsolatrendszer és fogalomrendszer, domainfüggetlen eszközrendszer, logikai eszközökkel való feldolgozhatóság, formálisan ellenőrizhető és feldolgozható adatrendszer.
Az egyik legfontosabb ontológia leíró nyelv az RDF (Resource Description Framework) nyelv [Lassila, 1999]. Az RDF feladata a problémakör fogalmainak semleges, átfogó, az automatikus feldolgozást támogató leírása. Az RDF nyelv, a korábban említett információatomokhoz hasonlóan az alábbi elemekből épül fel:

- erőforrások (egyed- vagy tulajdonságazonosító),
- literálok,
- állítások.

Az állítások egy (s, p, o) hármassal írhatók le, ahol s egyed erőforrás, p egy tulajdonság és o erőforrás vagy literál. Az állítás megfeleltethető információatomnak. A RDF terminológiában az állítás elemeit szubjektumnak, predikátumnak és objektumnak is nevezik. Az RDF nyelv továbbfejlesztéseként jött létre az OWL (Web Ontology Language) ontológialeíró nyelv [Goble, 2005]. Az OWL nyelv legfontosabb új eleme, hogy szélesebb körű integritási megkötéshalmazt tesz lehetővé, élesebben különbséget tesz a fogalmak különböző szintjei között (példányok bevezetése) és épít a formális feldolgozásra. Az OWL keretrendszer támogatja többek között a leíró logika apparátusát [Nardi, 2002], melyen keresztül lehetőség nyílik a felépített kontextus ellenőrzésére, új tények levezetésére. Az OWL támogató keretrendszerek között az ismertebbek közé tartozik a Protege, a Pellet és a Kaon.
Az adataink automatikus feldolgozásának hatékonyabbá tételéhez az alaprendszert kibővítő, a kontextust megadó metaadatrendszert kell létrehozni. Az annotáció ezen feladatát napjainkban az ontológia alapú keretrendszerre támaszkodva érdemes elvégezni [Fensel, 2001].

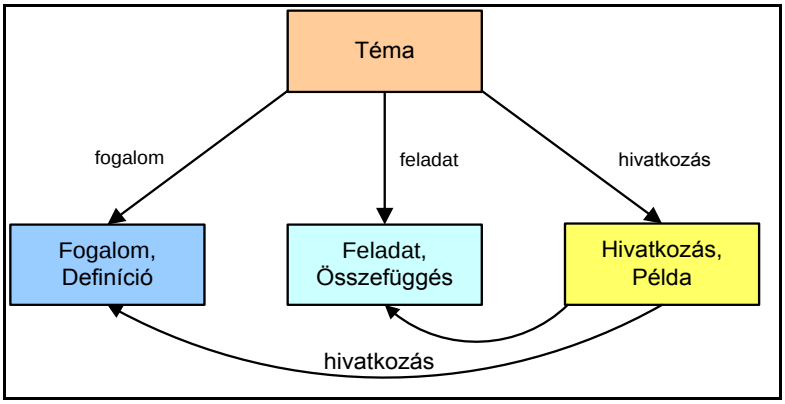
3.3. XML alapú digitális dokumentumformátumok

A dokumentum feldolgozásához és megjelenéséhez kapcsolódóan számos metaadatkezelő módszert és szabványt fejlesztettek ki az elmúlt évtizedekben. A szöveges dokumentumkezelés területén az XML alapú technológiák terjedése jellemzi a piaci helyzetet. A korábban alkalmazott technológiáktól eltérően ezen szabványok egyre növekvő mértékben valósítják meg a modularizálás és újrahasznosíthatóság követelményrendszerét. A tartalom és forma leírásának elkülönítése már a régebbi jelölőnyelves megoldásoknál, mint például a Latex nyelvnél, sikeres megoldássá vált. Az XML alapú dokumentumreprezentációs technológiák lényegi újítása elsődlegesen a tartalom modularizálásban áll. Az XML nyelv jellegéből fakadóan a leíró dokumentum elemei nemcsak formátumozási információkat tartalmaznak, hanem a tartalom jellemzésére vonatkozó elemeket is. Ilyen elemek lehetnek többek között a *definíció, utasítás,*

kérdés elemei is. Az XML alapú általános dokumentumleíró modellek közül a TEI (Text Encoding Initiative), LATEX, DITA és DOCBOOK szabványok tekinthetők a legelterjedtebb szabványoknak. Ezen dokumentumformátumok lehetőséget adnak az automatikus kérdésgeneráláshoz szükséges metaadatok tárolására a forrásdokumentumon belül. Mivel a felsorolt szabványok közül a DITA szabvány biztosítja a legnagyobb funkcionalitást a dokumentumspecifikus metaadatok terén, ez kerül felhasználásra a mintarendszer keretében.

3.3.1. DITA keretrendszer

A DITA szabványt az IBM cég jelentette meg 2001-ben, nem sokkal az XML szabvány 1999-es publikálása után. A DITA szabvány közvetlen ősének az SGML (Standard Generalized Markup Language) nyelv tekinthető [Doyle, 1999]. Az SGML, mint általános jelölő nyelv, elsődlegesen a formátum szabályozására szolgál, és elsődleges célja általános, minden kimeneti eszközre kiterjedő formátumspecifikációs nyelv definiálása. Az SGML nyelv sikertelenségét elsődlegesen az okozta, hogy elemi szinten kívánt minden ismert eszközhöz leíró nyelvet létrehozni. Mivel az elérhető eszközök köre állandóan bővül, a szabvány aktualizálása reménytelenül nagy munkát kíván az implementációt megvalósító szoftvercégektől. A DITA szabvány ezzel szemben az XML alapokra építkezik, és nem kívánja az egyes kimeneti eszközök formátumelemeit teljes részletességgel specifikálni, megmarad az általánosság szintjén, és a részleteket az eszköz-specifikus meghajtókra hagyja.
A DITA rendszerben a jelölő elemek fontos része a tartalom kezelésére szolgál. A legfontosabb egység a téma, mely önálló feldolgozási egységet is jelent. A téma többek között tartalmazza a címet, a leírást és a testet. A témának alapvetően három specializációja létezik (3.1. ábra): fogalom (definíció), feladat (tevékenység, leírás) és hivatkozás (link).



3.1. ábra: DITA modell

A fogalom *(concept)* esetében a felépítés a címből, prológból a fogalomtörzs összetevőiből áll. Különálló összetevő a hivatkozási rész *(reference*), amelyet hozzákapcsolnak más forrásokhoz. A témákat nagyobb formai egységek fogják össze, mint például a fejezet, könyvrész vagy könyv. A könyv mellett természetesen más dokumentumformátumok is támogatottak. A könyv felépítését a leíráshoz kapcsolt dokumentumtérkép tartalmazza. A dokumentumtérkép *(map)* megadja a témák tartalmazási és hivatkozási kapcsolatrendszerét. A fejlesztés hatékonyságának növelésére a szabvány lehetővé teszi a témák közötti származtatás, öröklés mechanizmusát is [4]. A legfőbb előnye a DITA megközelítésnek, hogy a munkát különálló modulokra képes bontani, ami megfelel az általános MCV (Model, Control and View) szoftverfejlesztési modellnek. Az MCV koncepcióval összefüggésben a dokumentumok leírását a következő összetevőkre lehet bontani:

szövegtartalom, megjelenítés, és szerkezet. Azzal a céllal, hogy általánosabb szerkezetet hozzanak létre, az összetevőket összekapcsolják bizonyos navigációs- és kötőelemeken keresztül. A DITA megközelítés legfőbb előnyeit 3.1. táblázat mutatja.

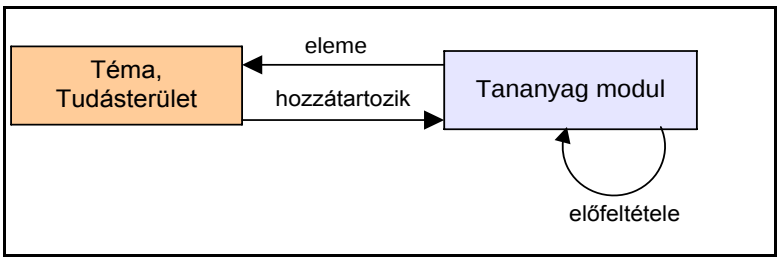
Fogalom	Eredmények
Az egység szerkezete	A munka felosztása kisebb egységekre.
Elválasztás	Párhuzamos munkacsoportok.
Egységesítés	A modulok kombinálása egy egységgé.
Specializálás	Könnyű létrehozása a speciális dokumentumtípusoknak.
Újrahasznosítás	Költség megtakarítás, fejlesztés.
Rugalmas szerkesztés	Támogatja a különböző megjelenési formákat.
Szabványosítás	Állandó technikai támogatás.

3.1. táblázat: A DITA előnyei

3.4. A kidolgozott tankönyvmodell

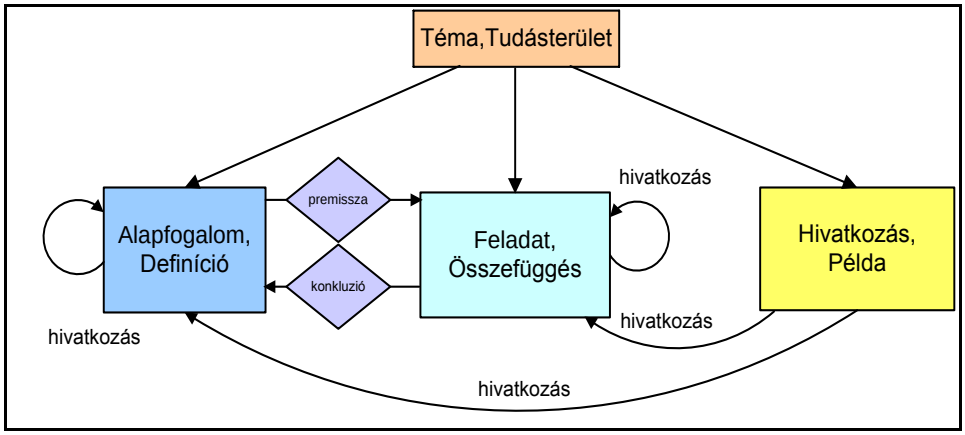
3.4.1. A tankönyvmodell DITA kerete

Kidolgoztam a tankönyvmodell DITA keretében egy tantárgy oktatási modelljét. Az automatizált tartalomkezelés és kérdésgenerálás a DITA keretében megfelelő struktúrákkal támogatható. Mivel a tankönyv tartalma tantárgyhoz vagy oktatási témakörhöz kapcsolódik, a tankönyv szerkezete szoros kapcsolatban áll a tananyagok tartalmi szerkezetével is. A tankönyv oktatási modell megtervezésénél konkrét tantárgyat, a felsőfokú műszaki képzés keretében oktatott tudásanyag modellezését választottam ki, ahol elvégeztem a terület fogalmainak és a fogalmak közötti szemantikai viszonyoknak a meghatározását. Az oktatási modell feladata a megszerezhető képességeken keresztül kapcsolatot teremteni az oktatott tantárgy ismeretelemei között. A modellre épülő alkalmazás lehet kérdésgeneráló rendszer is. A tudásforrást az oktatáshoz kidolgozott előadásanyag és a kötelező irodalom adja. A finomítás fázisa több alszakaszra bontható: elsőként az informális taxonómiát kell kialakítani, majd a váz finomítására, bővítésére kerül sor, és végül az oktatási modell formális leírását kell megadni. Az általános fogalmak meghatározását követően kerül sor az egyre speciálisabb fogalmak definiálására [10]. Valamely tantárgy kidolgozása során először annak a tematikáját kell elkészíteni, meghatározva azokat a fontosabb témaköröket, amelyek a modellben a téma, tudásterületnek feleltethető meg. Az órák keretében ezeket lebontjuk részterületekre, és értelmezzük a feladatokat, összefüggéseket az alapfogalmak meghatározásának segítségével.
A témakörök a tantárgy *tudásterület* osztályai. A következő lépés a tudásterület felosztása *altudásterületre*. Bizonyos esetekben az altudásterületet is további részterületekre kell bontani. Ezzel elkészült az elsődleges taxonómia. Ezután következik az alapfogalmak, definíciók, feladatok, összefüggések és példák meghatározása és a tudáselemek közötti kapcsolatok feltárása. A téma, tudásterület és a tananyag kapcsolatát a 3.2. ábra szemlélteti.



3.2. ábra: Téma, Tudásterület és tananyag kapcsolata

A tananyag oktatási modul az eleme reláción keresztül kapcsolódik a tudásterülethez. A tananyag eleme a tudásterület, illetve az adott tudásterület hozzátartozik a tananyaghoz. A modellben az osztályok ábrázolása téglalappal történik. Az összes nyíl 1-N és 0-N kapcsolatot jelöl.
A tudásterület belső szerkezetét a 3.3. ábra mutatja [Vas, 2007].

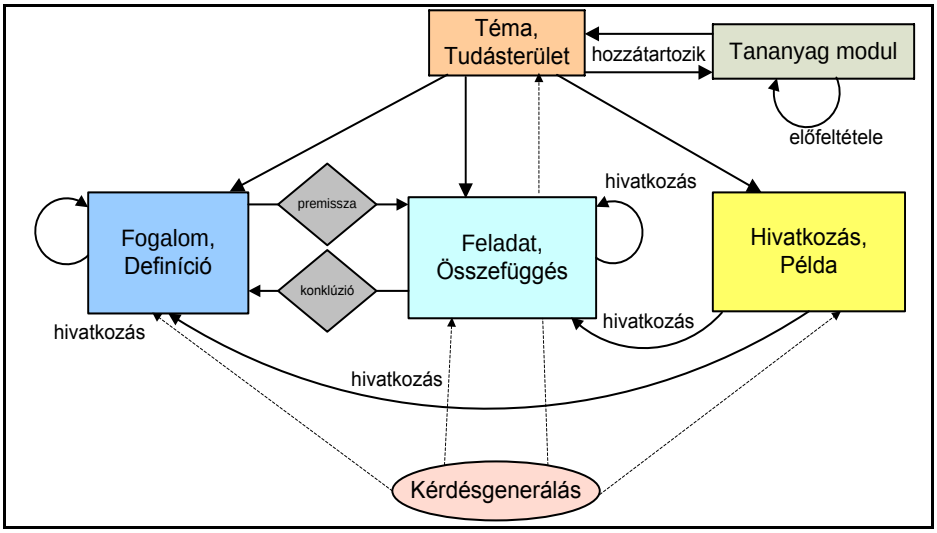


3.3. ábra: A tudásterület belső szerkezete

A modellben a következő tudáselemek szerepelnek:

1. Fogalom, Definíció,
2. Feladat, Összefüggés,
3. Hivatkozás, Példa.

Ha a tudáselemek tartalmára vonatkozó szemantikai összefüggéseket vizsgáljuk, akkor érdemes megvizsgálni a feladat, összefüggés mindkét oldalát és meghatározni a kapcsolatokat azokkal a fogalmakkal, melyeket felhasznál a feladat, összefüggés, illetve azokkal a fogalmakkal, melyekre következtetünk a feladat, összefüggés alapján. Az egyes fogalmak nem csak egy összefüggéshez kapcsolódhatnak. Ezeket a kapcsolatokat osztályok formájában kell felvenni a modellbe, melyektől speciális tulajdonság várható el. A kapcsolat mindkét oldalán megjelenő jellemzőket megfelelően le kell írni, kifejezve a fogalmak közötti hierarchiát. Ezt a relációtípust a *premissza*, *konklúzió* kapcsolóelemek *Fogalom és a Feladat*, *Összefüggés* osztályok segítségével kell leírni [2].

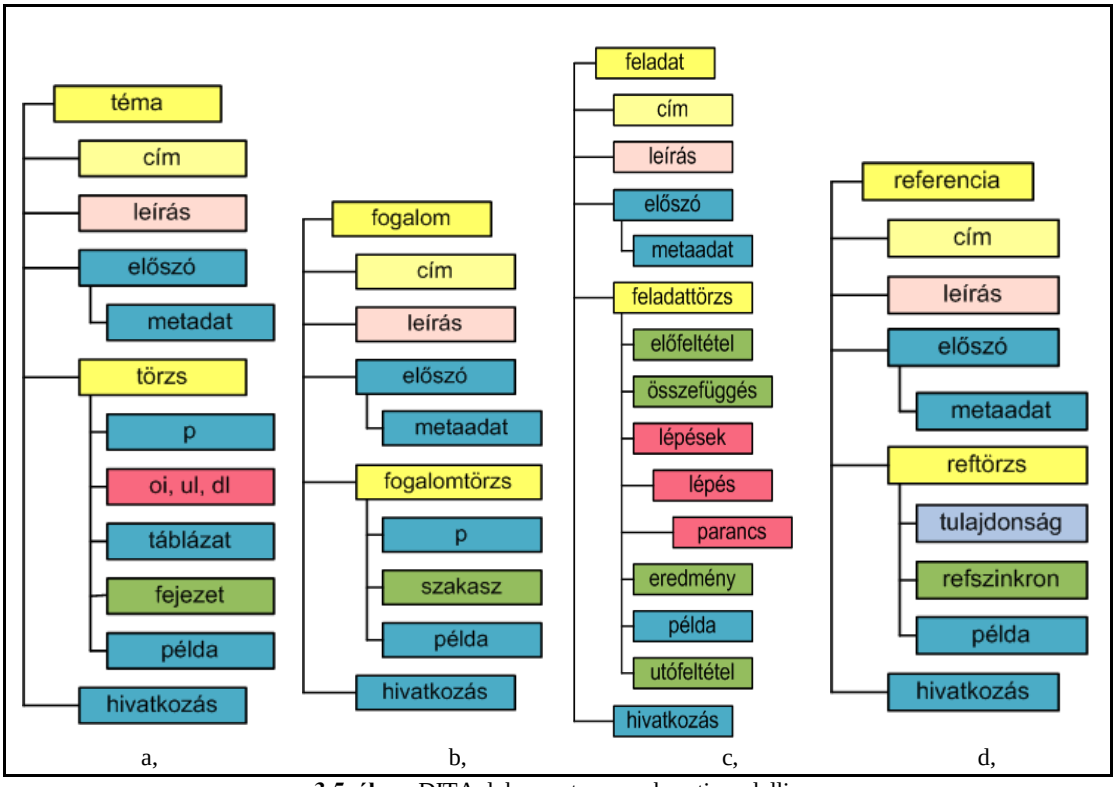


3.4. ábra: Tankönyv oktatási modellje

A hivatkozás relációval megadhatjuk, hogy az adott tudáselem, mely másik tudáselemre épül. Hivatkozhatunk fogalomra, feladatra, illetve példára a másik két tudáselem bármelyikére.
Ha a tudásterületet és az altudásterületet meghatároztuk és kialakítottuk a belső szerkezetet, akkor elkészült az adott tankönyv oktatási modellje (3.4. ábra).

3.4.2. A kidolgozott dokumentumszerkezet

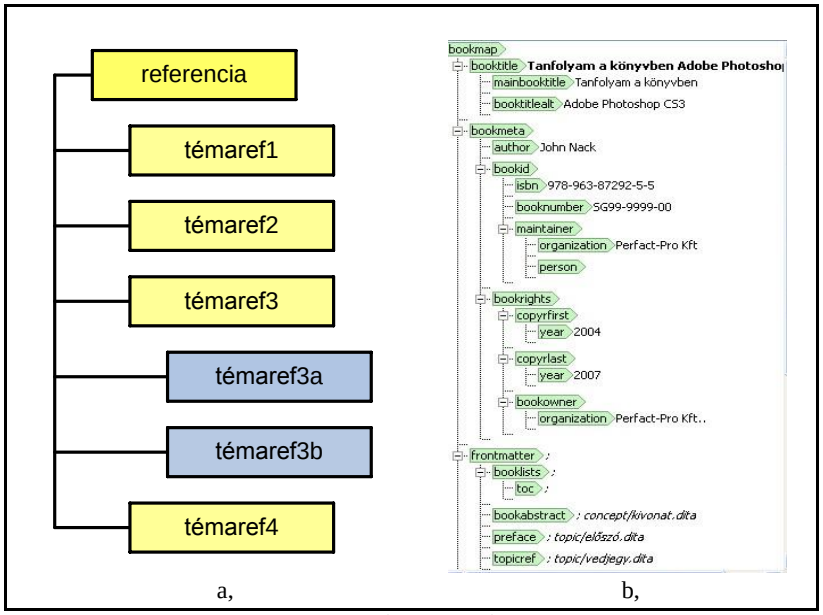
A DITA szabvány alapja a három átfogó információs kategória, melybe szinte az összes technikai adat besorolható: fogalom, feladat és a referencia. A tématípusok tartalmazzák a szakterületek alaposztályát, ahonnan öröklik az egységes központi struktúrát (lásd a 3.1. ábrát) a DITA modell cím, leírás, prológ és test információs típusaival. A DITA dokumentum tartalmaz legalább egy témát, melyet a 3.5.a ábra mutat. A téma szerkezete behatárolt, milyen sorrendben, mennyi elem jelenhet meg és hogyan van elhelyezve. A törzs részei az alaptémában bármilyen sorrendben megjelenhetnek, korlátozás nélküli számban. A téma specializációja (a fogalomban, a feladatban és a hivatkozásban) több korlátozást tesz a törzsrészben, a számukban és a megjelenésük sorrendjében [6].



3.5. ábra: DITA dokumentum szerkezeti modellje

A *fogalom* típus speciális téma elemnek a neve (3.5.b ábra). A bekezdés, lista, táblázat száma nincs korlátozva, de ezek közül egyik sem lehet az első szakasz vagy a példa után. A szakaszok és a példák bármilyen sorrendben lehetnek. A DITA információtípusai közül a *feladat* típus speciális témaelemnek a neve (3.5.c ábra). A feladat típus tartalmaz előfeltételt és összefüggést, mindkettő egy rész specializációja, majd ezt követi a sorrendbe helyezett lista specializációja és a lépések. Minden lépés a következőképpen épül fel: parancs, választható információ, példa, választási lehetőségek és az eredmény. Ezen lépések összességét követi a feladat eredménye, példa és a feladat utófeltétele. A *hivatkozás* szintén speciális típus, amely tartalmazza a témaelem nevét és szerkezetét. A 3.5.d ábrán látható, hogy a gyökérelem át van nevezve hivatkozásra, a törzselem

pedig hivatkozás törzsre. A hivatkozás törzs speciális elemeket tartalmaz, amely az elemre vonatkozó tulajdonságtípusokkal rendelkezik. Minden tulajdonság tartalmazza az elem típusát, értékét, valamint leírást. A tipikus megjelenítése általában táblázatszerű formátum.
A dokumentumok elkészítését követően a DITA *térkép* segítségével lehetőség nyílik a dokumentumok összeszerkesztésére fontossági sorrend alapján, melyet a 3.6.a ábra szemléltet. A DITA térkép a dokumentumtémát szervezheti web-es (HTML), nyomtatható (PDF) és súgó felületté. Így a tartalmat újrafelhasználja, egyedi forrásokká téve azokat. Speciális szervezési forma a *könyvtérkép* készítése (3.6.b ábra), melynek elkészítéséhez megfelelő DITA elemek nyújtanak támogatást. A DITA szabványos elemeit felhasználva egy digitális tankönyv- mintarendszert valósítottam meg. A fastruktúra legfelső szintjén helyezkedik el a könyvtérkép *(bookmap)*. A könyv fejrésze címet és a könyvre vonatkozó metaadatokat tartalmaz. A törzsrésze tartalmazza a fejezeteket, melybe a korábban elkészített fájlok kerülnek. A befejező részben irodalomjegyzék, ábrajegyzék, függelék, szószedet jelenik meg [13].



a, b,

3.6. ábra: DITA térkép- és tankönyvmodellje

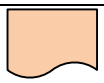
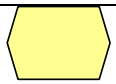
3.5. Digitális dokumentumok metaadatainak és keretrendszerének összefoglalása

Ebben a fejezetben bemutattam a legismertebb digitális dokumentumformátum metaadatait és a magasabb rendű információk lekérdezésének támogatására szolgáló metaadatok alapvető fejlődési tendenciáját.
A DITA modell alapvető elemeit felhasználva egy digitális tankönyv modelljét terveztem és valósítottam meg. A kidolgozott tankönyvmodell DITA kerete bemutatja, hogy az elterjedt XML alapú technológiák segítségével hogyan bővíthető a szöveges dokumentum leíró nyelve ontológia generálására alkalmas elemekkel. A technológia az ontológiák megfelelő elterjedése esetén jelentős hatékonyságjavulást eredményezhet az információkezelés területén.

4. fejezet

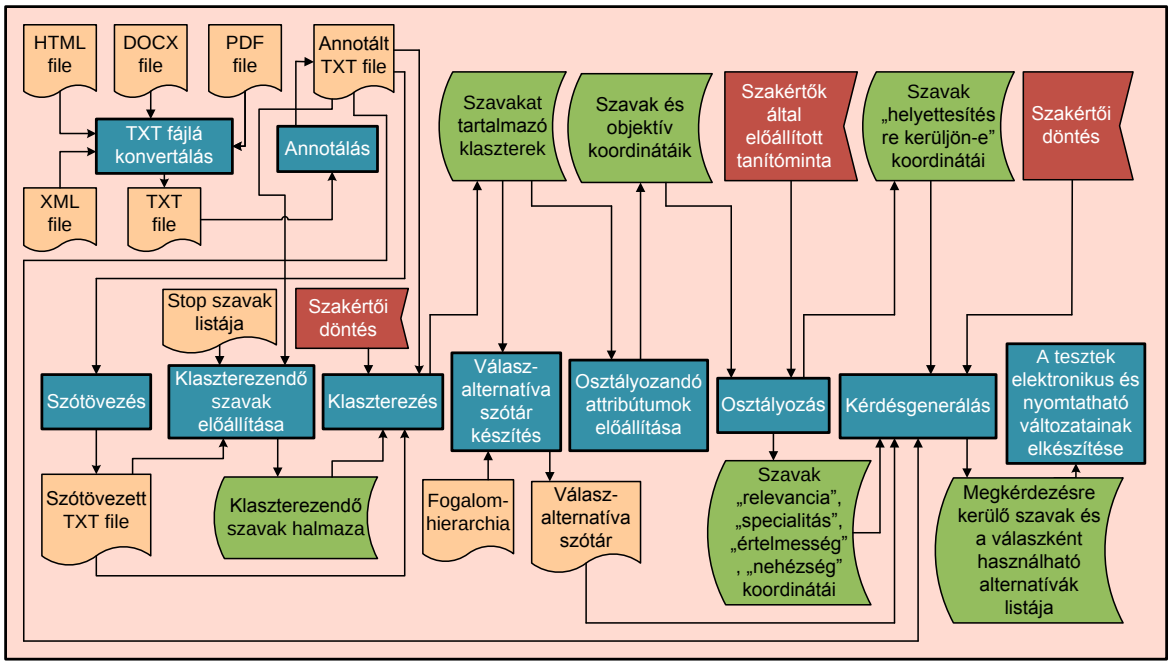
Automatizált kérdésgenerálás rendszerterve

A feladat megvalósítására olyan rendszermodellt dolgoztam ki, melyben több egymással adatkapcsolatban álló alrendszer együttes összehangolt működése biztosítja a feladat megoldását. Annak érdekében, hogy a rendszer egyes moduljai a későbbi fejlesztések során módosíthatóak, cserélhetőek legyenek minden modul számára bemeneti, illetve kimeneti interfészeket dolgoztam ki. A bemeneti interfészek definiálják azokat az adatokat, amelyeket a modulok igényelnek a feladatuk elvégzéséhez, míg a kimeneti interfészek definiálják azokat az adatokat, amelyeket a moduloknak szolgáltatniuk kell, a hozzájuk csatlakoztatott modulok számára. A rendszerterv elkészítése során a 4.1. táblázatban összefoglalt modellelemeket használtam.

Modellelem jele	Modellelem funkciója
	Szöveges, illetve bináris fájlban tárolt adat.
	Belső adatreprezentációban tárolt adat. Más alkalmazások számára nem elérhető.
	Felhasználói interakcióból származó adat.
	Jól definiált feladatot ellátó modul.
	Döntési folyamatot reprezentáló modul.

4.1. táblázat: A rendszerterv elkészítése során használt modellelemek

A Ph.D. munkámban megtervezett automatizált kérdésgenerálás feladatát ellátó rendszermodell funkcionális rendszertervét a 4.1. ábra szemlélteti [14].



4.1. ábra: A rendszermodell funkcionális rendszerterve

4.1. A kidolgozott kérdésgenerálás irodalmi áttekintése

4.1.1. A kérdésgeneráláshoz szükséges adatok és műveletek

Minden kérdésgenerálási feladat általános alakja hat paraméterrel jellemezhető. Az egyes paraméterek jelölése, valamint értelmezése az alábbiakban kerül összefoglalásra [Piwek, 2008]:

- S: a forrásdokumentum, mely alapján a kérdés előállításra kerül,
- A: a kérdésre adható érintett információs elem,
- Q: az előállított kérdés.
- SA: a forrásdokumentumot jellemző tulajdonságok és annotációk csoportja:
    - a dokumentum formátuma (pl. írott, hang alapú, képi stb.),
    - a dokumentum nyelve,
    - a dokumentum szerkezetének leírása,
    - szövegtan, mondattan, szótan,
    - szemantikai címkék (pl. szervezetek neve).
- AA: a kérdésre adható lehetséges válaszokat jellemző tulajdonságok és annotációk csoportja:
    - a válaszok formátuma (pl. írott, hang alapú, képi stb.),
    - a válaszok nyelve,
    - a válaszok szintaktikai és szemantikai leírása.
- QA: az előállított kérdést jellemző tulajdonságok és annotációk csoportja:
    - a kérdés típusa (pl. feleletválasztós, kiegészítő, stb.)
    - a kérdés hossza,
    - a kérdés témája,
    - a kérdés nyelve,
    - a kérdés stílusa.

Több kérdés esetén ezeknek a paramétereknek a listájával írhatóak le az előállított kérdések a következő formalizmus szerint [Piwek, 2008]:

$$(S_1, A_1, Q_1, SA_1, AA_1, QA_1), \dots, (S_i, A_i, Q_i, SA_i, AA_i, QA_i), \dots, (S_n, A_n, Q_n, SA_n, AA_n, QA_n).$$

4.1.2. A kérdésgenerálás során alkalmazott kérdéstípusok

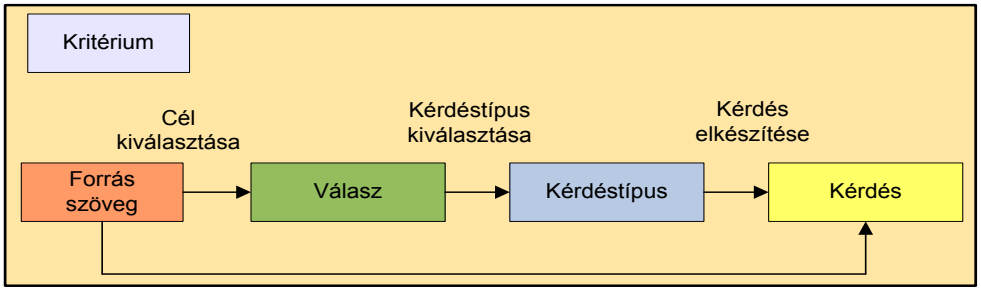
George Miller és munkatársai a Princeton Egyetem Kognitív Tudományi Laboratóriumában az angol nyelv szavai és fogalmai köré szerveződő lexikális szemantikai hálózatot készítettek [Miller, 1990]. Ez volt a WordNet-nek nevezett szinkron nyelvi tudást reprezentáló speciális szemantikai hálózat. A *wordnet* szónak ma már sok nyelvben létezik egyfajta köznévi használata, mely az eredeti angol nyelvű WordNet (Princeton WordNet, PWN) felépítését követő nyelvi adatbázisokra utal.

A wordnet olyan elektronikus lexikális szemantikai adatbázis, melyben a nyelvi fogalmak hálózatba szerveződnek. A fogalmakat szinonimahalmazok (synsetek), a közöttük lévő kapcsolatokat szemantikai relációk (hipernima, meronima, antonima stb.) reprezentálják. A wordnetekben minden synsethez tartozik rövid szöveges definíció is. Természetesen ez nem a gép, hanem az emberi felhasználó számára fontos, az adott jelentés könnyebb azonosításához.

A legfontosabb, a synsetek összetételét meghatározó szemantikai reláció a szinonímia, mely azonban – a többi relációval ellentétben – nem synsetek között, hanem magukon a synseteken belül a terminusok közt áll fenn. George Miller a szinonimitást a következőképpen definiálja a WordNet számára: két kifejezés egymással szinonim egy A nyelvi környezetben, ha az A-ban az egyiket a másikkal felcserélve a mondat igazságértéke nem változik. A szinonímia szimmetrikus reláció, ugyanis ha X szinonimája Y-nak, akkor Y is szinonimája X-nek [Miller, 1990].
A WordNet a főnévi synsetek között a következő alaprelációkat értelmezi: antonima, hiponima és meronima. A főneveket és más szófajokat összekapcsoló relációk az attribútumérték-relációk, melyek valójában főnév és melléknév közti relációk [Miháltz, 2003].
Az antoníma reláció ellentétet, szembenállást fejez ki, melyet azonban nehéz pontosan definiálni. Az antoníma szimmetrikus reláció, mely nem a szójelentések – tehát a synsetek – hanem ezek elemei, azaz a szavak között áll fenn. Könnyen belátható, hogy sokszor egy antonima relációban álló szópár szinonimáira nem állítható egyértelműen, hogy ezek is egymás antonímái lennének. Emiatt az antonima relációban álló synsetek elemei esetén egyértelműen specifikálni kell, hogy az antonímia a synsetek mely tagjai között áll fenn [Vossen, 1999].
A főnévi fogalmak között a legfontosabb reláció a hipernima (inverze: hiponima), mely hierarchikus alá-/fölérendeltséget, specifikus/generikus, faj/nem, IS-A öröklődési viszonyt fejez ki. Speciális altípusa a hipernímia reláció, mely tulajdonnevekhez kapcsolódó, individuumoknak megfelelő és általánosabb, osztályoknak megfelelő fogalmak között állhat fent [Prószéky, 2008].
A WordNet-en elérhető információk alapján hatféle kérdés hozható létre: *definíció, szinonim, antonim, hiperném, hiponém* és *feleletválasztós kérdések*. Ahhoz, hogy kinyerjünk valamilyen adatot a WordNet-ből ki kell választani az adott szónak a megfelelő jelentését. Adott szóhoz hozzá van rendelve több szó is a WordNet rendszerben. Leggyakrabban a beviteli rész csak az adott szót tartalmazza egy beszéd részeként [Brown, 2004].
A *definiáló kérdés* a szóra vonatkozó definíciót kínálja fel. Ez elérhető a WordNet jegyzet részéből. A rendszer kiválasztja az első definíciót, ami nem tartalmazza magát az adott szót. A *szinonima* típusú kérdésekben a kérdéses szót összekapcsolja a rokon értelmű szavával és a rendszer kinyeri a WordNet-ből az adott szónak a rokon értelmű szavát. Ehhez két eljárást használ. Az egyik módszernél kiválasztja azt a szót, amihez az adott szinonima tartozik. Ehhez megvan a szó és megvannak hozzá a többi szinonimák, valamint a szinonimáknak a szinonimái is. Ezeket a WordNet megjelöli, mint elérhető és választható rokon értelmű szavakat. Általában a rokon értelmű szavakat az egyszavas szavakra korlátozza le, és nem ad hozzá hosszabb variációs lehetőséget. A másik módszer akkor kerül alkalmazásra, amikor több mint egy szó felel meg a kritériumnak. Ekkor aszerint választja a szinonimát, hogy melyiket használják leggyakrabban. Ezt a kérdéstesztet lehet úgy tekinteni, mint hozzárendelő vagy megértő eljárást. Ha a szövegrész mellett ott van a szinonima és van utalás rá, akkor ez a kérdéstípus hozzárendelős eljárásba tartozik. Ekkor már meglévő szóhoz rendeli hozzá a szót. Az *antonim* kérdésektől azt várják el, hogy a szót párosítsa össze az ellentétes szavával. A WordNet kétféle kapcsolatot kínál erre, a direkt és az indirekt kapcsolatot. A direkt ellentét az adott szónak az ellentéte, míg az indirekt ellentét az adott szó ellentétének a szinonimái. A szinonim és az antonim kérdéseknél a feladat kiválasztja, hogy melyik a leghasonlóbb és melyik a legellentétesebb jelentésű rokon értelmű és ellentétes értelmű szó. A *hiperném* és *hiponém* kérdéstípusok hasonló szerkezetűek. A hiperném a fogalmak egy általános meghatározás osztályába sorolását jelenti. A *feleletválasztós* kérdés típusnál adott a fő kérdés és ezt követi számos válasz, amelyekből csak egy a megfelelő.
Mind a hat kérdést számos formában lehet előállítani, de az elsődleges mindegyik típusnál a szóbank használata. A szóbank rendelkezik egy listával, amely a válaszokat tartalmazza, ezt követi a kérdés, ahová beillesztésre kerül az adott szó.

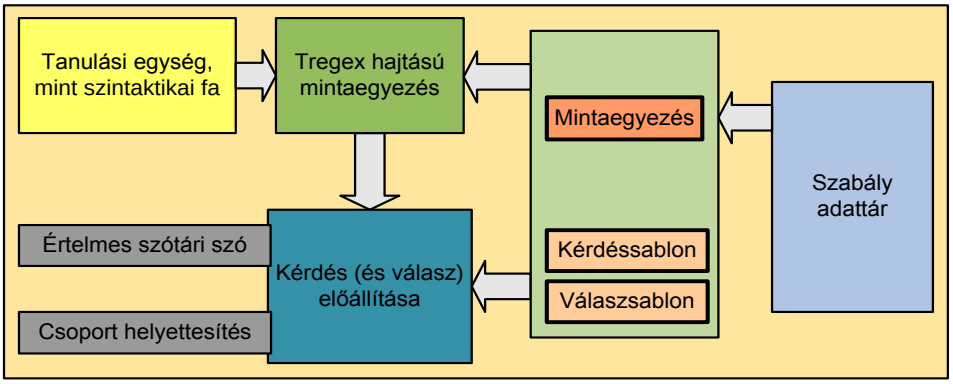
4.1.3. Kérdésgenerálási architektúrák

Az automatizált kérdésgeneráló rendszerek fejlesztésével foglalkozó kutatások eredményei 2008 után már komplex architektúrák formájában jelentek meg. Nielsen és szerzőtársai [Nielsen, 2008] megalkottak egy olyan modellt, amelyben a 4.1.1. fejezetben leírt adatokat és műveleteket felhasználták a rendszer elkészítéséhez. Ezek ismeretében a kérdések automatikus előállítása a következő lépésekből áll. Elsőként a forrásdokumentumból ki kell választani azt a tartalmat, amire szeretnénk, hogy az előállítandó kérdés vonatkozzon. Ezt követően az elvárt válasz ismeretében ki kell választani a kérdés típusát. Végül elő kell állítani azt a kérdést, amely megfelel a megadott tartalom, illetve kérdéstípus paramétereinek.
A szöveges dokumentum alapján végzett kérdésgenerálás általános modellje a 4.2. ábrán látható [Nielsen, 2008].



4.2. ábra: Szöveges dokumentum alapján végzett kérdésgenerálás általános modellje

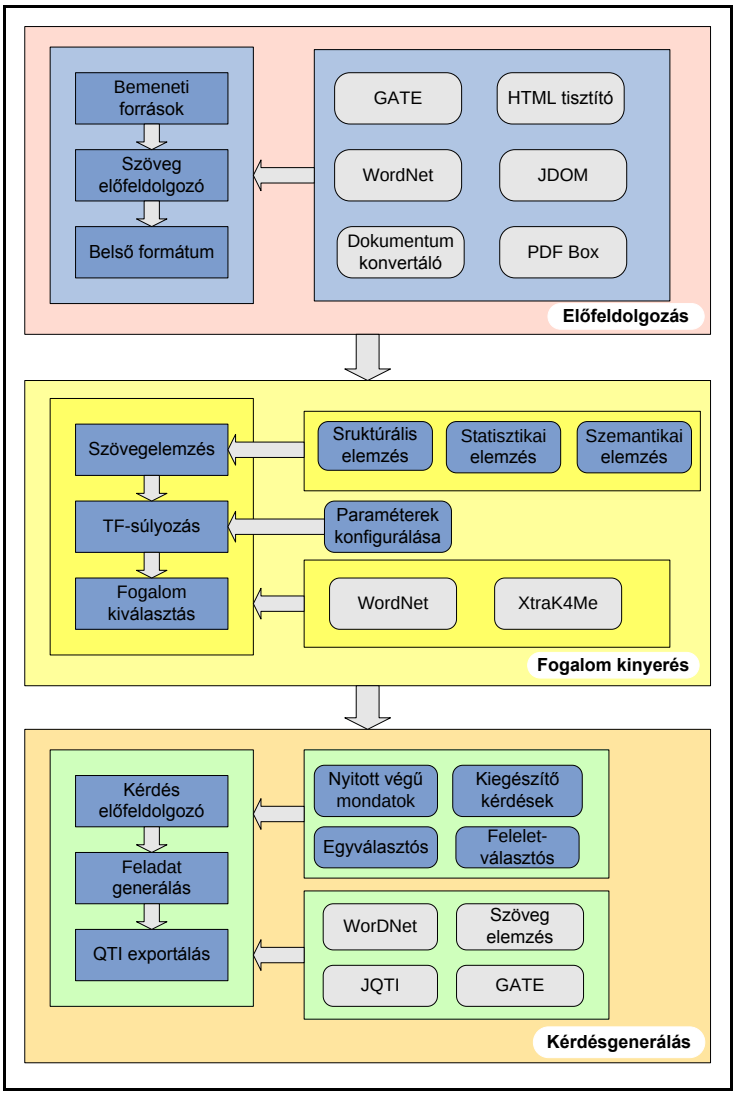
Az OpenLearn olyan nyílt keretrendszer, amely XML formátumot használ az oktatási anyag tárolásához. A CEIST módszer [Piwek, 2009], amit az OpenLearn nyílt keretrendszeréhez fejlesztettek ki, rendelkezik egy előfeldolgozó fázissal, ahol a dokumentum tartalmát tiszta szövegformátumból átkonvertálják egy szövegelemző fastruktúrába a nyelvtani elemző modul segítségével. Az egység fő motorja egy mintaegyeztető algoritmus, amit ahhoz használnak, hogy megtalálják az előzőleg egyeztetett mintákat a fában. A fában megtalált mondatokhoz konverziós szabály sémáját felhasználva generálnak kérdéseket, melyet a 4.3. ábra szemléltet [Piwek, 2009].



4.3. ábra: Ceist architektúra

Gütl és munkatársai 2008-ban elkészítettek egy automatizált kérdésgeneráló rendszert (Automatic Question Generation, AQC) [Gütl, 2008]. A kifejlesztett prototípus tapasztalatain és eredményein alapulva 2010-ben továbbfejlesztették a rendszert (Enhanced Automatic Question Creator, EAQC). A továbbfejlesztett rendszer több különböző nyelv tanulási tananyagából is támogatja automatikus tesztkérdések készítését. A rugalmas tervezés lehetővé tette, hogy önállóan használható legyen az eszköz és adaptálni lehessen speciális tanulási környezetbe.

A 4.4. ábrán az EAQC modell jól illusztrálja az alapvető fogalmi egységeket és a már meglévő eszközöket [Gütl, 2011].



4.4. ábra: EAQC modell

Az EAQC rendszer három fő modulból áll:

- Az előfeldolgozási modul feladata a különböző bemeneti fájlformátumok átalakítása, nyelvfelismerés és átalakítás belső XML sémává, amely tartalmazza az összes szükséges adatot a további feldolgozáshoz. A jelenlegi modell az angol [WordNet, 2010] és a német nyelvet [GermaNet, 2009] támogatja, de lehetőség van más nyelvek és eszközök beépítésére is a modulba.
- A fogalmak kinyerésére szolgáló modul strukturális, statisztikai és szemantikai elemzést végez, és kinyeri a legmegfelelőbb fogalmakat a dokumentumból.
- A kérdésgeneráló modul meghatározza a releváns mondatokat, majd hozzáfűzi a korábban kiválasztott mondatokhoz. Ebben a modulban történik a kérdéstípusok kiválasztása: nyitott végű mondatok, egyválasztós, többválasztós, feleletválasztós kérdések.

4.1.4. A mintarendszer architektúrájával szembeni követelmények

A Ph.D. munkámban kidolgozott kérdésgeneráló mintarendszer esetén, olyan architektúrát terveztem és valósítottam meg, amely működőképes külső szemantikai adatbázis nélkül is. Ennek oka, hogy a WordNet jellegű adatforrás nem állt rendelkezésre a kiválasztott területen és nyelvben. Emiatt a fogalomszótár egy belső, saját fejlesztésű modullal valósítottam meg.

Az AQG rendszerekkel szemben támasztott főbb követelmények:

- *Magyar nyelvű támogatás*
  Az automatizált kérdésgenerálás kutatása még új területnek számít nem csak magyar, hanem külföldi vonatkozásban is. Ezért a kísérleti jelleggel elkészített mintarendszerek döntően angol nyelvi szövegek alapján képesek dolgozni. Mivel munkám célja elsősorban a magyar nyelvű szövegértés automatizált mérése területén jelentkező igények kielégítése volt, ezért szükségessé vált az erre specializált mintarendszer kifejlesztése.
- *Ingyenes nyelvtani elemző használata*
  Munkám eredményességét nagymértékben befolyásolja a szavak nyelvtani elemzésének pontossága. Ezért ezt a feladatot sok éves nyelvészeti ismeretek alapján kifejlesztett ingyenes szoftverrel végeztem el. A Szószablya [Németh, 2003] fő előnye a nagy tudásbázis, az eredmények exportálhatósága, valamint az ingyenes használhatóság.
- *Annotációk támogatása*
  Mivel a legtöbb dokumentum nem elhanyagolható részét olyan mondatok képezik, melyek nem kapcsolódnak szorosan a dokumentum által reprezentált téma lényegéhez, ezért szükségessé vált ezeknek a mondatoknak a kizárására az automatizált kérdésgenerálás folyamatából. A mondatok annotálása révén lehetőséget biztosítottam nem csupán a témához nem kapcsolódó mondatok kizárására, hanem a különböző annotációk révén a mondattípusok közötti választásra is.
- *Feleletválasztós és kiegészítő kérdések generálása*
  Az elkészített alkalmazással lehetőséget biztosítottam a feleletválasztós, valamint a kiegészítő kérdések automatizált előállítására. A kidolgozott koncepció alkalmas a kérdésként kiemelésre kerülő szó helyettesítésére szolgáló válaszalternatívák automatizált előállítására is. Ezáltal megvalósítottam, hogy a válaszalternatívák megjelenítése esetén feleletválasztós, míg elhagyása esetén kiegészítő kérdéssort generáljon a rendszer.
- *Nyílt moduláris rendszer*
  A munkám során kidolgozott mintarendszert a modularitás elve alapján építettem fel. Ennek eredményeképpen lehetőség van a már elkészített modulok egymástól független továbbfejlesztésére, valamint a rendszer új modulokkal való bővítésére.
- *Automatizált gépi tanulás*
  A kérdések és a válaszalternatívák generálása nem előre létrehozott tudásbázis alapján történik, hanem a mintarendszer dinamikusan alkalmazkodik az elemzés alatt álló dokumentum egyedi jellegéhez és nyelvezetéhez. Ennek érdekében a kérdésként kiemelésre kerülő szavak meghatározása a dokumentum elemzésére kidolgozott neurális hálózat alapján kerül meghatározásra. Hasonlóképpen a válaszalternatívák is az aktuális dokumentum szókészletéből kerülnek ki a klaszterezéssel nyert távolságinformációk alapján.
- *Elektronikus és nyomtatható feladatlapok készítésének lehetősége*
  A modell alapján elkészített számítógépes alkalmazással lehetővé tettem a tesztlapok elektronikus, illetve nyomtatható változatainak előállítását. Ezek révén lehetővé vált a létrehozott kérdéssorok alkalmazása a számítógépi infrastruktúrával nem rendelkező körülmények között is. Az internetes támogatással rendelkező számítógépes

munkaállomásokról viszont lehetőség van a kitöltött tesztlapok elektronikus levélben való elküldésére is.

- *Együttműködési képesség a létező e-learning keretrendszerrel*
  A MOODLE keretrendszer lehetőséget biztosít a kérdések importálására többféle formátumba. Ezek közül az első a GIFT formátum (General Import Format Technology), melynek létrehozásához olyan eszközre van szükség, mely segítségével a Word dokumentumból GIFT formátumú kérdéseket lehet készíteni. Másik lehetőség a MOODLE XML konverter, amely lehetővé teszi a tesztek létrehozását MOODLE XML formátumba. A konvertálás után kapott XML állomány importálható a kérdésadatbázisba. Ezen feladatok megvalósítása további kutatási feladatot jelent számomra.

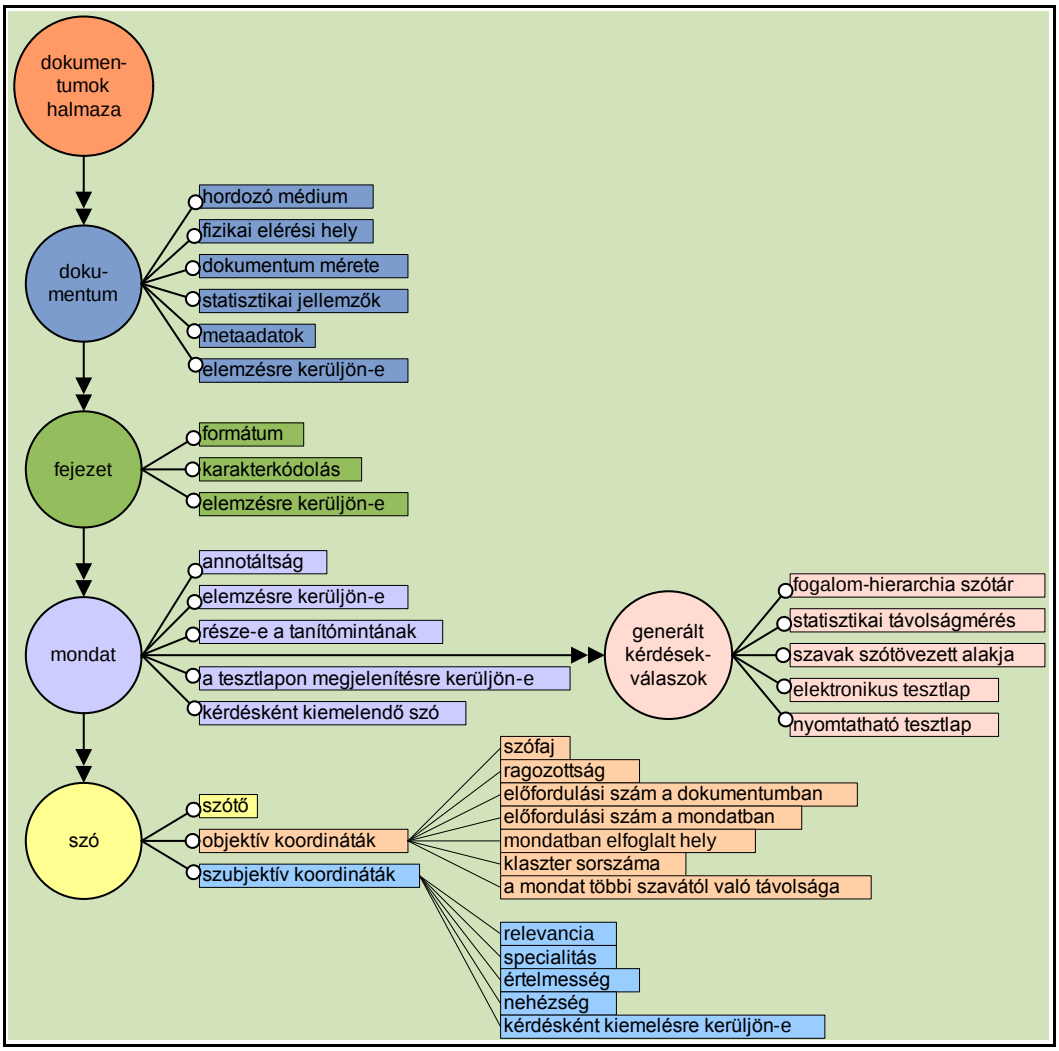
Fő nehézségek, speciális problémák:

- *Nyelvi elemzés*
  A magyar nyelv automatizált elemzése a nyelv összetett szabályrendszere miatt csak nehezen megvalósítható. Az automatizált nyelvtani elemző létrehozása túlmutat Ph.D. munkám keretein, ezért fel kellett tárnom a témában több éves tapasztalattal rendelkező nagy szaktudású csoportok által elért eredményeket. Kutatásaim során megismertem a 4.2.2. fejezetben bemutatott szótövezőket, illetve a Humor SDK mintarendszert [Krauth, 2007]. Ezek közül a Szószablyára esett a választásom, mivel a többi rendszerrel szemben ez biztosította a legpontosabb elemzési lehetőséget a különböző tématerületekhez tartozó dokumentumok esetén.
- *Kérdésként kiemelésre kerülő szó automatizált meghatározása*
  A tesztlapon megjelenítésre kerülő mondatoknak a kérdésként kiemelésre kerülő szava a 6. fejezetben ismertetésre kerülő osztályozó algoritmus eredményei alapján kerül meghatározásra. Az algoritmus a kérdésekre szánt mondatok szavainak objektív koordinátái ismeretében meghatározza a szavak szubjektív koordinátáit. A szubjektív koordináták közül a „*kérdésként kiemelésre kerüljön-e*” névvel ellátott bináris értékkészlettel definiált koordináta befolyásolja legnagyobb mértékben az adott szó kérdésként való kiemelését. Abban az esetben, ha ez a koordináta nem jelöli ki egyértelműen a kérdésre használandó szót (mert a mondat egyetlen szavát sem jelöli ki, vagy egynél több szót is kijelöl), akkor az algoritmus a többi szubjektív koordináta értékét is figyelembe veszi a választáshoz. A mintarendszerrel végzett tesztek eredményei igazolták, hogy azok a szavak a leginkább alkalmasak a hallgatók tudásának mérésére, melyeknek a „*nehézség mértéke*” nevű szubjektív koordinátája az átlagostól nehezebb értéket képvisel.
- *A kérdésre adható válaszok automatizált meghatározása*
  A feleletválasztós kérdések automatizált előállításához a mondatból kérdésként kiemelésre kerülő szó meghatározásán túl a kérdésre adható lehetséges válaszok automatizált előállítására is szükség van. A válaszalternatívák automatizált előállításának algoritmusa a 7.1.2. fejezetben kerül ismertetésre. Az algoritmus lényege, hogy minden kérdésként kiemelésre kerülő szóhoz öt lehetséges választ hoz létre. Ezek között a válaszok között minden esetben szerepel a kérdésként kiemelt szó. Ez képezi a feladat tökéletes megoldását. A hallgatók tudásszintjének pontosabb mérése érdekében a többi alternatíva a kiemelt szótól egyenlő lépésközzel növekvő távolságban kerül kiválasztásra. A szavak egymáshoz viszonyított távolságát az 5. fejezetben ismertetésre kerülő klaszterezési algoritmusokkal határoztam meg. Ez alapján a hallgatók válaszának a helyes megoldástól való távolsága használható tudásuk mérésére. Annak érdekében, hogy a válaszalternatívák fogalmi szinten is illeszkedjenek a kiemelt szóhoz a szótávolságon kívül kidolgoztam egy fogalomhierarchiára épülő keresési algoritmust is. Ez az algoritmus a szavak egymáshoz

viszonyított távolságát a szavak kategóriaszavához, illetve szófajhoz tartozása alapján határozza meg. A két módszer együttes alkalmazásával megvalósítottam, hogy a kidolgozott algoritmus által előállított válaszok jól közelítsék az emberi intelligenciával előállítottakat.

4.2. A kidolgozott kérdésgenerálás rendszertervi elemei

A kérdésgenerálás automatizálása területén végzett munkám során kidolgoztam több, egymással kapcsolatban álló modulból álló rendszert. A rendszer minden modulja jól definiálható feladatot végez el, mely során a bemenetén kapott adatok feldolgozásával újabb adatokat állít elő a további modulok számára. Az elkészített rendszermodell funkcionális rendszertervét a 4.1. ábra szemlélteti. A rendszerterv úgy épül fel, hogy minden modulja a többitől függetlenül elemezhető és továbbfejleszthető legyen.
Kidolgozásra került a dokumentumrendszer szerkezeti modellje is, amely a rendszer bemeneti és kimeneti adatai a 4.5. ábrán bemutatott gráf struktúrájú adatbázisban kerültek tárolásra.



4.5. ábra: A kidolgozott dokumentumrendszer szerkezeti modellje

Ez a struktúra lényegében egy fa, amelyben a hierarchia az adatok tartalmazási relációit reprezentálja. Az adattípusokat jellemző attribútumok az adattípusokat szimbolizáló körök mellett kerültek felsorolásra. Az információs modell az alábbi struktúraelemekből épül fel: dokumentumok halmaza, dokumentum, fejezet, mondat, generált kérdések és szó.

A megfelelő struktúra kialakítása esetében ismerni kell azokat a jellemzőket, amelyeket a szöveges dokumentumok tartalmazhatnak. Az elemezhető attribútumok csoportosíthatók formai elemekre, metaadatokra és tartalmi elemekre. A statisztikai jellemzők határvonalat jelentenek a formai és tartalmi elemzés között. A szövegek karaktersorozatokra bonthatók, így leírhatóvá válnak olyan statisztikai jellemzőkkel, mint az egyes különálló dokumentumok hossza, az azokban előforduló szavak hossza vagy a szavak előfordulási gyakoriságai. A tartalmi elemek csoportjába tartozik a dokumentum témája, információtartalma, stílusa és motívumai.
A kidolgozott dokumentumrendszer szerkezeti modelljében (4.5 ábra) szereplő adattípusok és attribútumok jellemzői a következők:

Dokumentum

- *hordozó médium*: digitális adathordozón tárolt szöveges dokumentum,
- *fizikai elérési hely*: az adott dokumentum elérési helye,
- *dokumentum mérete*: a dokumentum által elfoglalt hely bájtban kifejezett mennyisége,
- *statisztikai jellemzők*: szavak, karakterek eloszlása, magánhangzók és mássalhangzók száma,
- *metaadatok*: fizikai metaadatok, amelyet az adathordozón való tároláskor kap,
- *elemzésre kerüljön-e:* a dokumentumok halmazából az adott dokumentum elemzésre kerüljön-e a kérdésgenerálás során.

Fejezet

- *formátum*: a tárolási formátum az alkalmazott szoftver lehetőségeitől és beállításaitól függ,
- *karakterkódolás*: a dokumentum nyelvével függ össze (magyar nyelvű szövegek a közép-európai vagy az UTF-8 kódolást használnak),
- *elemzésre kerüljön-e*: a dokumentum fejezetei közül az adott fejezet elemzésre kerüljön-e a kérdésgenerálás során.

Mondat

- *annotáltság*: a mondat tudományterülethez tartozását leíró kód,
- *elemzésre kerüljön-e*: a fejezet mondatai közül az adott mondat elemzésre kerüljön-e a kérdésgenerálás során,
- *része-e a tanítómintának*: a szakértői döntés alapján a mondat szavai szerepeljenek-e az osztályozás tanítómintájában,
- *a tesztlapon megjelenjen-e*: azt jelöli, hogy az adott mondat megjelenjen-e az elektronikus vagy nyomtatható tesztlapon,
- *kérdésként kiemelendő szó*: a mondat kérdésként kiemelésre javasolt szava.

Szó

- *szótő*: a szó szótöve;
- *objektív koordináták*: a szó objektív módszerekkel meghatározható koordinátái:
  - *szófaj*: a szó szófaja (meghatározását a szakértő, ennek hiányában a program végzi el),
  - *ragozottság*: a szó ragozottsága,
  - *előfordulási szám a dokumentumban*: a szó dokumentumban való előfordulási száma,
  - *előfordulási szám a mondatban*: előfordulási szám az adott szót tartalmazó mondatban,
  - *mondatban elfoglalt hely*: hely az adott szót tartalmazó mondatban,
  - *klaszterek sorszáma*: a szót tartalmazó klaszter sorszáma,
  - *a mondat többi szavától való távolsága*: távolságátlaga az adott szót tartalmazó mondatnak azon szavaitól, melyek nem kerültek kizárásra a mondatból;
- *szubjektív koordináták*: a szónak az osztályozó algoritmussal meghatározásra kerülő koordinátái:
  - *relevancia*: a szó abszolút relevanciájának mértéke,
  - *specialitás*: a szó abszolút specialitásának mértéke,

- *értelmesség*: a szó abszolút értelmességének mértéke,
- *nehézség*: a szó abszolút nehézségének mértéke,
- *kérdésként kiemelésre kerüljön-e*: az osztályozó algoritmus alapján az adott szó kérdésként kiemelésre kerüljön-e.

Generált kérdések és válaszalternatívák

- *fogalomhierarchia szótár*: a szavak közötti kapcsolatokat kategóriaszó-szófaj-szó háromszintű hierarchiában tároló szótár,
- *statisztikai távolságmérés*: a szavak klaszterezésénél a szavak közötti távolságok definíciója (szógyakoriság illetve vektortér),
- *szavak szótövezett alakja*: válaszalternatívák szótövezett alakja,
- *elektronikus tesztlap*: kérdések és válaszok számítógépes környezetben való megjelenítése,
- *nyomtatható tesztlap*: tesztlap nyomtatható formátumban való megjelenítése.

4.2.1. Szöveges dokumentumok előfeldolgozását végző modul

Elektronikus szöveges dokumentumok első és egyik legfontosabb lépése az előfeldolgozás, melynek során a dokumentumokat az adott feladatnak megfelelő és a szövegek sajátosságainak tárolására alkalmas modell szerinti alakra hoztam. Az előfeldolgozás feladata a további elemzés hatékonyságának megteremtése, amely egységesítési, formalizálási és normalizálási lépéseket tartalmazhat. Szöveges dokumentumok esetén az előfeldolgozás szempontjából legfontosabb jellemző a formátum, a karakterkódolás és az annotáció.

Az annotáció nyelvi jelenségek szintaktikai és/vagy szemantikai jelölése a szövegben. A gyakorlatban szintaktikai és szemantikai annotációs sémákat különböztetünk meg. A szintaktika annotáció kétféleképpen valósulhat meg [Roberts & Atwell, 2000]:

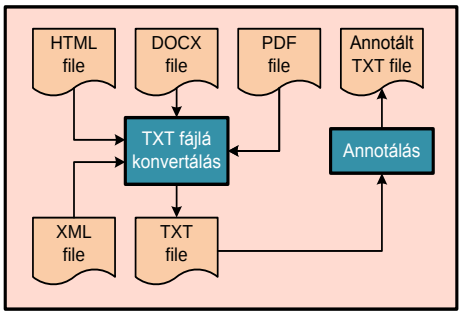
- megadható minden szóhoz, hogy milyen mondatrész szerepét tölti be (Part-Of-Speech tagging),
- minden szó esetén meghatározható a főigétől való függése (dependency-based tagging).

A szemantikai kódolás megvalósítására a szakirodalom szintén két módszert említ [Reeve & Han, 2005]:

- minden szóhoz hozzárendelhető a mondatban betöltött szemantikai szerepe,
- a szavak rögzített szakterületen való elhelyezésének megadása.

A dokumentum tartalmazhat olyan mondatokat is a kérdésgenerálás szempontjából, melyek nem jellemzik a dokumentum lényegét. Ezek a mondatok kizárásra kerültek a kérdések előállításának folyamatából. A megoldást a dokumentum mondatainak annotálásával értem el. Az annotálás révén megvalósíthatóvá vált, hogy a dokumentum csak bizonyos kritériumoknak megfelelő mondatai kerüljenek elemzésre. A kidolgozott algoritmus alkalmas több annotációt tartalmazó szűrőfeltétel megfogalmazására, illetve ez alapján való keresésre. Ezáltal, megvalósítható a dokumentum akár összes annotált mondattípusának kérdésként való kiválaszthatósága is.

A megvalósított modell előfeldolgozó modulja alkalmas DOC, DOCX, HTML, PDF és XML formátumban kódolt dokumentumoknak, bemeneti adatként való kezelésére. Az előfeldolgozó modul funkcionális rendszerterve a 4.6. ábrán látható.



4.6. ábra: Az előfeldolgozó modul funkcionális rendszerterve

A modul működéséhez szükséges adatok

Adat leírása	**Adatot előállító modul neve**
A kérdések előállítására szolgáló HTML, DOCX, PDF vagy XML formátumban kódolt szöveges dokumentum.	Web-es adatbázis

4.2. táblázat: Az előfeldolgozó modul működéséhez szükséges adatok

A modul által szolgáltatott adatok

Adat leírása	**Adatot feldolgozó modulok neve**
Annotációkkal ellátott mondatokból álló ASCII formátumban kódolt szöveges dokumentum.	- Szótövezést végző modul - Klaszterezendő szavak előállítását végző modul - Klaszterezést végző modul - Kérdésgenerálást végző modul

4.3. táblázat: Az előfeldolgozó modul által szolgáltatott adatok

A modul feladata a különböző formátumú dokumentumok egységes formára hozatala, amelynek során a formázási, strukturális és szerkesztési adatok mellőzésével pusztán a szöveges információk kerülnek kinyerésre. A konverzió célja egy egyszerűen kezelhető szöveges formátum elérése, amelynek kimenete ASCII (American Standard Code for Information Interchange) kódolású szöveg.

A konvertálásnál figyelmet kell fordítani a szöveges karakterkódolás helyes kezelésére. A dokumentum leírására használt kódkészlet a dokumentum nyelvével függ össze. A szótag vagy szójelölő írásmódot alkalmazó nyelvek teljes kódkészletét csak két bájton lehet tárolni. Erre fejlesztették ki az unicode kódtáblát, amely a nagy ábécéjű nyelvek összes karakterét képes kódolni. Az unicode-ra épül az UTF-8 kódolás, mely a karakterek unicode szabványon alapuló kódolásakor változó hosszúságú bináris egységeket használ. Az egynyelvű dokumentumgyűjtemény esetén gyakran előfordul, hogy az egyes dokumentumok kódolása eltérő. Magyar nyelvű szövegek esetén elsősorban közép-európai (ISO-8859-2) vagy UTF-8 kódolást használnak. Léteznek olyan dokumentumok is, amelyek nyugat-európai kódolásúak, ezért az ő és ű betűk helytelenül jelennek meg. Az előfeldolgozó modul feladatai közé tartozik a karakterkódolások egységesítése is [Tikk, 2006]. Mivel a felsorolt formátumokat kezelő alkalmazások közvetlenül képesek a dokumentumokat ASCII kódolású szöveges formátumúra konvertálni, ezért külön konvertáló program készítésére nem volt szükség.

Az előfeldolgozást végző modul kimenetén keletkezik az annotált dokumentumfájl, amely egyrészt a klaszterező modul, másrészt a szótövező modul bemeneteként szolgál.

Az előfeldolgozást végző modul a 4.5. ábrán látható dokumentumrendszer modellben beállítja az összes dokumentum „*elemzésre kerüljön-e*” attribútumának értékét a kérdésgenerálás igényei szerint. Hasonlóképpen ez a modul határozza meg az elemzésre kerülő dokumentumok fejezeteinek „*elemzésre kerüljön-e*” attribútumához tartozó értéket is. Ezt követően a modul

megváltoztatja az elemzésre kerülő fejezetek „*formátum*” attribútumát ASCII szöveges formátumra. Ezen kívül beállítja az elemzésre kerülő fejezetek mondatainak „*annotáltság*” attribútumának értékét a mondat típusa szerint. Végül az annotációval ellátott mondatoknak kitölti az „*elemzésre kerüljön-e*” attribútumát aszerint, hogy az adott mondat szerepeljen-e az aktuális kérdésgenerálási folyamatban.

4.2.2. Szótövezést végző modul

A szótövezés olyan szavak szótőre redukálását jelenti, amelyek valamilyen jelentésmódosító ragot, toldalékot, prefixet vagy suffixet kaptak. Az alkalmazásorientált számítógépes nyelvészeti szakirodalom szótövezésnek nevezi azt az eljárást, amikor az adott szó szótövének a meghatározása a cél.
A szótövezést több, lényegileg különböző megközelítéssel lehet megvalósítani:

- Algoritmikusan, nyelvspecifikus átírószabályok implementálásával. Ezek a módszerek a leggyorsabbak. Az erre épülő algoritmusok egymillió szót néhány másodperc alatt feldolgoznak. Angol nyelvre a legelterjedtebb Porter algoritmusa, melynek működési elvét több nyelvre adaptálták [Porter, 2001].
- A szavakat és szótövüket, szótöveiket tartalmazó szótár alkalmazásával. Csak azokra a szavakra működnek, amelyek szerepelnek a szótárban [Tomlinson, 2004].

Porter kifejlesztett egy általános, szótövezőket leíró nyelvet is, a *Snowball*-t [Porter, 2001], amely már nem csak ún. szuffixeket képes levágni, hanem prefixeket is. Ennek segítségével 14 európai nyelvhez készítettek szótövezőt, köztük magyarhoz is. A Snowball-nyelvre [Snowball, 2005] adaptált *Tordai-féle magyar szótövező* az inflexiós toldalékok levágását végzi el, azaz csak a ragokat és jeleket vágja le, a képzőket nem [Tordai, 2005]. A kutatók négy különböző erősségű szótövezőt készítettek, amelyeket az annotált Szeged Korpuszon [Csendes, 2003] tanítottak be.

1. A *gyenge1* szótövező csak a leggyakoribb főnévi esetet, többes számot és birtokost kezel.
2. A *gyenge2* az összes főnévi esetet, többes számot és birtokost kezeli. Mindkét szótövező figyelembe veszi a szótőjelölt hosszát és, hogy tartalmaz-e érvényes mássalhangzó-magánhangzó kombinációt.
3. A *közepes* változat 12 gyakori főnévi esetet kezel, a birtokos és birtokok, valamint a személyek számát is figyelembe véve. Ezen kívül kezeli a leggyakoribb igealakokat (idő, szám, személy), a melléknevek fokozását, valamint a számneveknél a törtszámnév és sorszámnév toldalékait.
4. Az *erős* szótövező a legtöbb inflexiós toldalékot és az összes igealakot figyelembe veszi.

A főnevek esetén az algoritmus kilenc lépésben határozza meg a szótövet. Az egyes verziók abban különböznek egymástól, hogy mely lépéseket alkalmazzák és azon belül milyen toldalékcsoportokat kezelnek.
Magyar nyelvre elsőként a Morphologic Kft. készített szótövező eljárást. A hazai nyelvtechnológiai piacon a Szószablya [Németh, 2003] projekt keretében a BME Média Oktatási és Kutató Központ (MOKK) által kifejlesztett HUNMORPH programcsomaghoz kapcsolódó HUNSTEM szabály alapú szótövező terjedt el, amely ingyenesen hozzáférhető.

Szótövezés során az alábbi tipikus hibák léphetnek fel:
Alultövezés, ha két szóhoz, amelyek jelentése ekvivalens a feldolgozás szempontjából az algoritmus különböző tövet rendel. Például *szabványosságot* → *szabványos*, *szabványt* → *szabvány*.

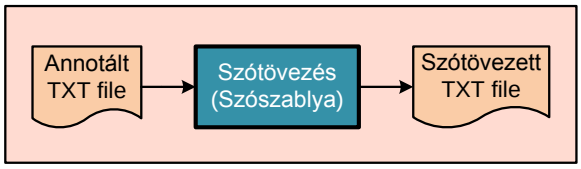
Túltövezés, akkor fordul elő, ha két szóhoz, amelyek jelentése különböző, az algoritmus ugyanazt a tövet rendeli. Például *házak → ház, házas → ház.*
Félreelemzés, olyan végződést vág le az algoritmus, ami valójában a tő része. Például *nemzet → nemz, német → ném.*
A szótövezők mérésére leggyakrabban a Paice által bevezetett hibaszámláláson alapuló alul- és túltövezési indexeket használják [Paice, 1990].
Alultövezési index (Under-stemming Index, UI)

$$UI = 1 - (n_s / n_{all})$$

Ahol: n_s sikeresen közös tőre redukált szópárok, n_{all} közös tővel rendelkező szópárok.
Túltövezési index (Over-stemming Index, OI)

$$OI = 1 - (n_s / n_{all})$$

Ahol: n_s sikeresen eltérő tőre redukált szópárok, n_{all} eltérő tővel rendelkező szópárok.
A dokumentumok szavainak szótövét és szófaját a Szószablya szótövező alkalmazás segítségével állítottam elő. A Szószablya projekt segítségével magyar szótári szavak különböző alakjainak gyakoriságát és megjelenési formáit vizsgálhatjuk egy internetes szövegekből épített adatbázisra épülő komplex keresőfelület segítségével. Az elemzett szavakról a következő jellemzők kerülnek előállításra pontosvesszőkkel szeparált szöveges fájlban: szóalak, szótő, gyakoriság, szótagszám, elemzés, szófaj. A szótövezést végző modul funkcionális rendszertervét a 4.7. ábra szemlélteti.



4.7. ábra: A szótövezést végző modul funkcionális rendszerterve

A modul működéséhez szükséges adatok

Adat leírása	**Adatot előállító modul neve**
Annotációkkal ellátott mondatokból álló ASCII formátumban kódolt szöveges dokumentum.	Annotálást végző modul

4.4. táblázat: A szótövezést végző modul működéséhez szükséges adatok

A modul által szolgáltatott adatok

Adat leírása	**Adatot feldolgozó modulok neve**
A dokumentum szavait, valamint a szótövező alkalmazással hozzájuk meghatározott objektív jellemzőket tartalmazó szöveges fájl.	- Klaszterezendő szavak előállítását végző modul - Klaszterezést végző modul

4.5. táblázat: A szótövezést végző modul által szolgáltatott adatok

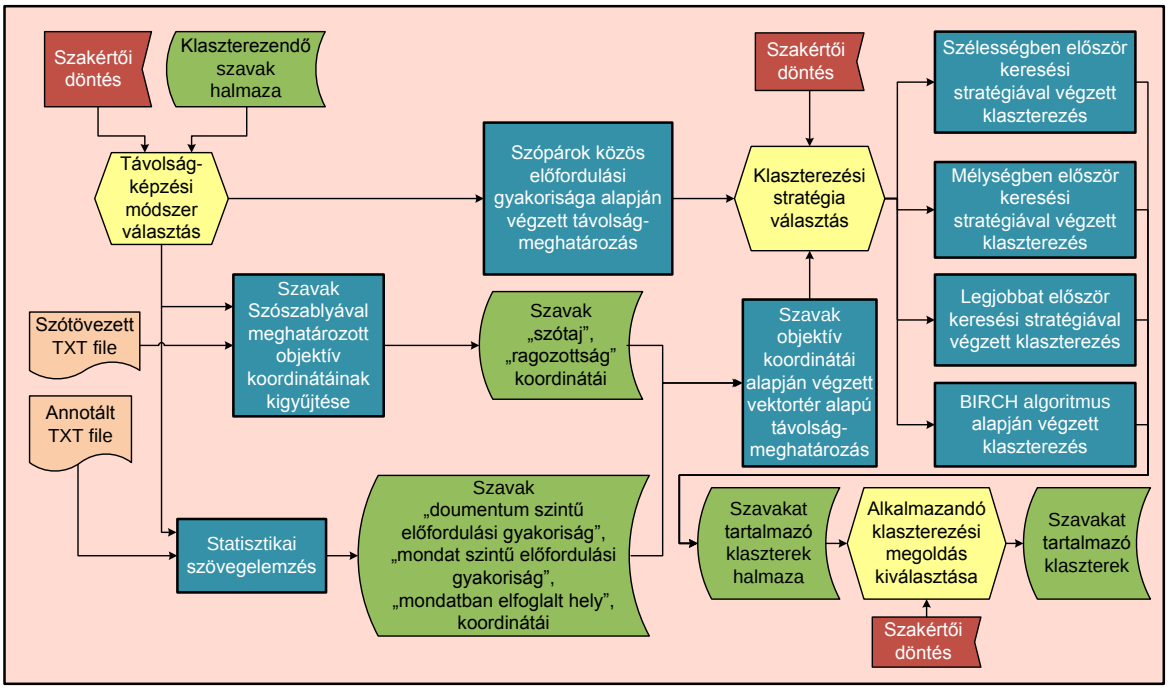
A szótövezést végző modul a 4.5. ábrán látható dokumentumrendszerből sorra veszi azokat a mondatokat, melyek „*elemzésre kerüljön-e*” attribútuma igaz értéket tartalmaz és meghatározza ezen mondatok szavainak a „*szótő*”, „*szófaj*”, „*ragozottság*”, „*előfordulási szám a dokumentumban*”, „*előfordulási szám a mondatban*”, valamint a „*mondatban elfoglalt hely*” attribútumához tartozó értéket.

4.2.3. A klaszterezést végző modul

A klaszterezés célja, az objektumokból olyan elkülönült csoportokat alkotni, hogy az egy csoportba kerülők minél hasonlóbbak, az eltérő csoportokban lévők, pedig minél különbözőbbek legyenek [Jain, 1988].

A klaszterezés alapvető fontosságú feladat az automatizált kérdésgenerálás során. Egy n szóból álló objektum esetén a szavak egymáshoz viszonyított távolságának meghatározása n^2-tel arányos tárkapacitást és számítási időt igényel. Az azonos klaszterbe kerülő szavakat az algoritmus a klaszterezés kritériumainak értelmében egymással megegyezőeknek tekinti és ezáltal a közöttük lévő távolságot elhanyagolhatónak veszi. A távolságok meghatározása a szavak helyett a jóval kevesebb számú klaszterek között kerül csak meghatározásra. Ez lényegesen csökkenti mind a tárigényt, mind pedig a számítási igény által jelentett költséget. Minél távolabbi szavak kerülhetnek egy klaszterbe annál kevesebb számú klaszter is elegendő a dokumentum szavainak csoportosításához, ám annál nagyobb lesz az egy klaszterben lévő szavak maximális távolságával jellemezhető elhanyagolásból származó hiba.

A feladatom elvégzése során a szavak klaszterezésének a költségtakarékosságon kívül van egy másik szintén fontos szerepe is. Az egy klaszterben lévő szavak ugyanis azok közös előfordulási gyakorisága vagy vektortér alapú távolsága szempontjából kváziszinonimáknak tekinthetőek. Ezek bár nem feltétlenül azonosak a szavakról nyilvántartott szinonimákkal, ám igaz rájuk, hogy a közös előfordulási gyakoriságuk vagy objektív jellemzőik szempontjából közel vannak egymáshoz. Az egy klaszterben lévő szavaknak ezt a tulajdonságát felhasználtam a kérdésként kiemelésre kerülő szavakhoz lehetséges, válaszként felkínált alternatívák automatikus előállítása során is. A klaszterezést végző modul funkcionális rendszertervét a 4.8. ábra mutatja.



4.8. ábra: A klaszterezést végző modul funkcionális rendszerterve

A modul működéséhez szükséges adatok

Adat leírása	Adatot előállító modul neve
A dokumentum szavait, valamint a szótövező alkalmazással hozzájuk meghatározott objektív jellemzőket tartalmazó szöveges fájl.	Szótövező modul
A dokumentum mondatait annotáltan tartalmazó szöveges fájl.	Annotáló modul
A dokumentum szavai közül a klaszterezendőket tartalmazó halmaz.	Klaszterezendő szavak előállítását végző modul
Távolságképzési módszer.	Szakértői döntés
Klaszterezési stratégia.	Szakértői döntés
Alkalmazandó klaszterezési megoldás.	Szakértői döntés

4.6. táblázat: A klaszterezést végző modul működéséhez szükséges adatok

A modul által szolgáltatott adatok

Adat leírása	Adatot feldolgozó modulok neve
A dokumentum minden klaszterezendő szavát egy klaszterhez rendelő struktúrákat tartalmazó halmaz.	- Válaszalternatíva szótárat készítő modul, - Osztályozandó attribútumok előállítását végző modul

4.7. táblázat: A klaszterezést végző modul által szolgáltatott adatok

A modul működése

A klaszterezést végző modul működéséhez, bemeneti információként várja a klaszterezendő szavak halmazát. Ezt a halmazt az előző modulok állítják elő, a kérdésgenerálásra alkalmazott dokumentumból kiszűrve az annotációval nem rendelkező mondatokat, valamint a stop szavak listáján szereplő szavakat. A megmaradt szavak rendezetlen halmaza adja a klaszterezést végző modul egyik bemenetét. Amikor ez rendelkezésre áll, akkor szakértői vélemény alapján kerül kiválasztásra az alkalmazandó szótávolság-képzési módszer.

Az egyik távolságképzési módszer a klaszterezendő szavak együttes előfordulásának gyakorisága, ahol két szó távolsága annál kisebb, minél több olyan mondat található a dokumentumban, amelyben mindkét szó szerepel. Mivel ezek a távolságértékek különböző méretű és stílusú dokumentumok esetén jelentősen eltérhetnek egymástól, ezért normalizálásukra az $S_{i,j} = f_{i,j} / \max(f_i, f_j)$ összefüggést alkalmaztam, ahol $S_{i,j}$ az *i.* és *j.* szó távolságviszonyát reprezentáló (0,…,1) intervallumba eső számérték (a nagyobb érték a szavak közötti kisebb távolságot jelöli); $f_{i,j}$ a dokumentum azon mondatainak száma, melyekben az *i.* és a *j.* szó is szerepel; f_i a dokumentum azon mondatainak a száma melyekben az *i.* szó szerepel; f_j a dokumentum azon mondatainak a száma melyekben a *j.* szó szerepel.

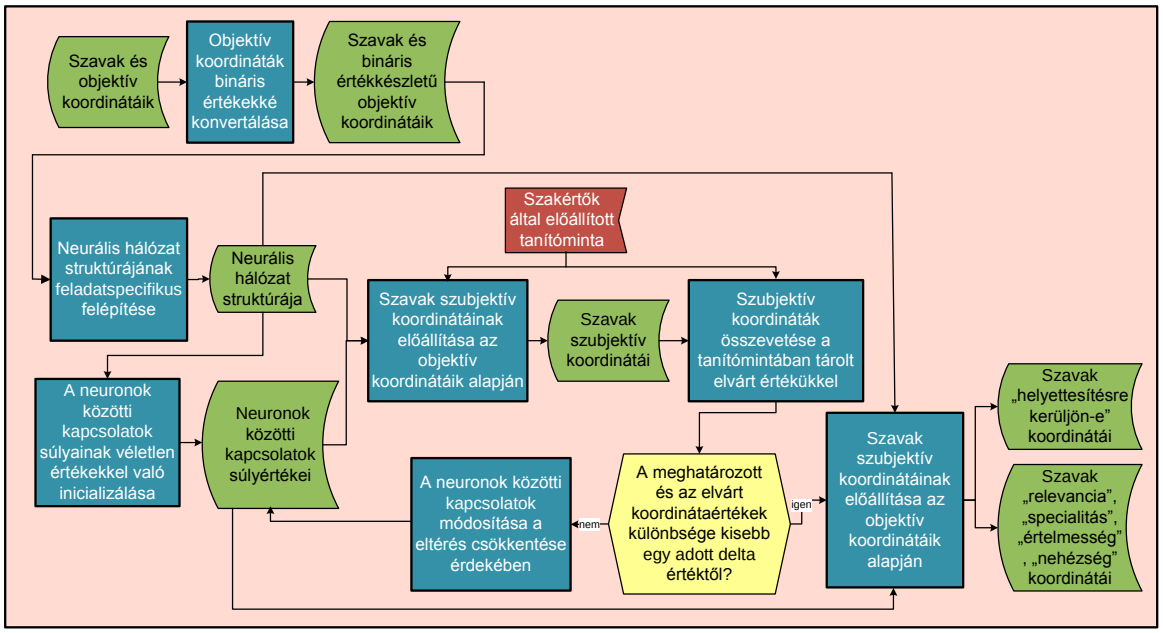
A másik távolságképzési módszer értelmében, minden szót többdimenziós objektív térben elhelyezkedő pont reprezentál. Ebben a reprezentációs modellben, a szavak távolsága, a pontok vektortér alapú módszerrel számítható távolságával határozható meg. Abban az esetben, ha a szakértő ezt a távolságképzési módszert választja, a klaszterezés megkezdése előtt elő kell állítani a klaszterezendő szavakhoz tartozó objektív koordinátákat. A szavaknak a nyelvészeti, valamint statisztikai módszerekkel objektíven mérhető tulajdonságait a szavak *objektív koordinátáknak* nevezzük.

Munkám során a következő objektív koordinátákat használtam a klaszterezés elvégzéséhez: *szófaj*, *ragozottság*, *előfordulási szám a dokumentumban*, *előfordulási szám a mondatban*, *mondatban elfoglalt hely*. Ezeket az információkat egyrészt a *Szószablya* keretrendszerrel, másrészt saját fejlesztésű statisztikai szövegelemző algoritmussal állítottam elő. A szavak közötti távolságok meghatározását követően a klaszterezést irányító szakértőnek választania kell a modul által kínált klaszterezési stratégiák közül. Jelenleg, a megvalósításra került alkalmazásban négy klaszterezési stratégiát dolgoztam ki, melyek mindegyike a HAC (Hierarchical Agglomerative Clustering)

klaszterezési koncepciónak egy-egy lehetséges megvalósítási alternatíváját reprezentálja. A szakirodalomban jól ismert szélességi, mélységi, valamint legjobbat először kereséseken kívül, a klaszterezés folyamatát implementáltam egy kevésbé ismert BIRCH (Balanced Iterative Reducing and Clustering using Hierarchies) algoritmussal is, amely kifejezetten sok klaszterezendő elemet (szót) tartalmazó feladat esetén nyújt, más módszerektől jóval hatékonyabb eredményt. Mivel mind a négy módszer több lehetséges klaszterezési eredményre is vezethet, ezért ezek közül az alkalmazásra kerülőt a szakértő határozza meg, a modul által felkínált lehetséges alternatívák közül való választással. A klaszterezés eredményét olyan csoportosítás képezi, melyben minden klaszterezendő szó eleme egy klaszternek. Ezeket az adatokat a modul, belső adattárolási struktúrában szolgáltatja a következő modulok számára.
A klaszterezést végző modul a 4.5. ábrán látható dokumentumrendszerből sorra veszi azokat a mondatokat, melyek „*elemzésre kerüljön-e*” attribútuma igaz értéket tartalmaz és a megadott távolságképzési módszer, valamint klaszterezési stratégia szerint meghatározza ezen mondatok minden szavának „*klaszter sorszáma*”, valamint „*a mondat többi szavától való távolsága*” attribútumainak értékét.

4.2.4. Az osztályozást végző modul

Az osztályozási feladatot neurális hálózat alkalmazásával végeztem el, melyet mind strukturális, mind pedig működési szempontból a konkrét feladathoz igazítottam. Az osztályozást végző modul funkcionális rendszertervét a 4.9. ábra mutatja.



4.9. ábra: Az osztályozást végző modul funkcionális rendszerterve

A modul működéséhez szükséges adatok

Adat leírása	Adatot előállító
Az osztályozandó szavakat és a hozzájuk tartozó hét objektív koordinátát tartalmazó halmaz.	Osztályozandó attribútumok előállítását végző modul
Szakértők által előállított tanítóminta, amely tartalmazza a neurális hálózat tanítására szolgáló szavakat a hozzájuk tartozó objektív és szubjektív koordinátáikkal együtt.	Szakértő

4.8. táblázat: Az osztályozást végző modul működéséhez szükséges adatok

A modul által szolgáltatott adatok

Adat leírása	**Adatot feldolgozó modulok neve**
Az osztályozásra került szavakat és a hozzájuk tartozó *„kérdésként kiemelésre kerüljön-e”* szubjektív koordinátákat tartalmazó halmaz.	Kérdésgeneráló modul
Az osztályozásra került szavakat és a hozzájuk tartozó *relevancia, specialitás, értelmesség, nehézség* szubjektív koordinátákat tartalmazó halmaz.	Kérdésgeneráló modul

4.9. táblázat: Az osztályozást végző modul által szolgáltatott adatok

A modul működése

Az osztályozást végző modul bemeneti információként kapja az előző modulok által előállított szavakat és a hozzájuk tartozó objektív koordinátákat. A szavak objektív koordinátái részben megegyeznek a klaszterezést végző modulban előállított objektív koordinátákkal, ám az osztályozás megkezdése előtt ezek a koordináták kiegészülnek a klaszterezés eredményének ismeretével nyerhető újabb információkkal. Ezek az új objektív dimenziók: *a szót tartalmazó klaszter sorszáma*, valamint *a szó átlagos távolsága a szót tartalmazó mondat többi szavától*. Ezekkel együtt az osztályozás bemenetén minden szó, hétdimenziós objektív térben kerül elhelyezésre. A megvalósításra kerülő neurális hálózatban minden neuront kétértékű aktivációs függvénnyel modelleztem. A szakirodalomban ismertek ugyan többértékű aktivációs függvényt használó neurális hálózatok is, ám a feladatom során a szavak jól elkülönülő halmazokba sorolására volt szükség. A neurális hálózat struktúrája az objektív koordinátáknak az ismeretét követően kerül felépítésre. Ennek oka, hogy minden osztályozási feladat, egyedileg felépített neurális hálózatot igényel. Mivel a legtöbb objektív koordináta értékkészlete, csak a klaszterezést követően válik ismertté, ezért a bemeneti értékek fogadására és továbbítására alkalmas neuronok száma is, csak a pontos számosság ismeretében határozható meg.

A feladat megoldására, háromrétegű előrecsatolt hálózatot alkalmaztam, ahol a rejtett rétegben lévő neuronok számát azonosnak választottam, a kimeneti rétegben lévő neuronok számával. Az osztályozás megkezdése előtt a szakértőnek lehetősége nyílik a neurális háló tanítására, az általa kidolgozott tanítóminta szerint [3]. A tanítóminta rendezett formában tartalmazza tetszőleges számú szó objektív, valamint szubjektív koordinátáit. A szubjektív koordináták a szavaknak azokat az egzaktul nem meghatározható jellemzőit jelölik, melyek az embernek az adott szóra vonatkozó megítélését fejezik ki. A kidolgozott mintarendszerben a következő szubjektív koordináták kerültek bevezetésre: *nehézség*, *relevancia*, *specialitás*, *értelmesség*.

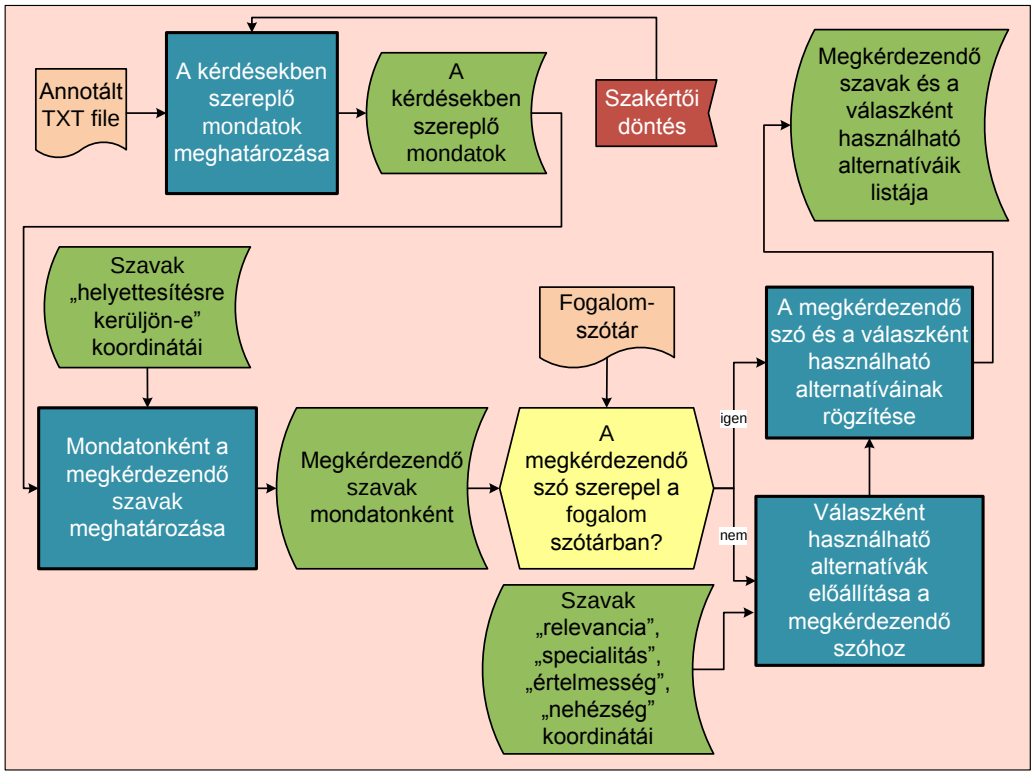
A tanulás első lépéseként a neurális hálózat súlyértékei, véletlenszerűen kerülnek meghatározásra a súlyok számára definiált intervallumon, egyenletes eloszlás szerint. Ezt követően a szakértőtől kapott objektív koordináták alapján, a neurális hálózattal meghatározásra kerülnek a szavak szubjektív koordinátái. A szubjektív koordináták meghatározását követően, a tanító algoritmus kiértékeli a válaszok jóságát, a szakértőtől kapott szubjektív koordináták alapján. Amennyiben az eltérés nagyobb, egy előre definiált értéktől, az algoritmus módosítja a neuronok közötti kapcsolatok súlyértékeit, olyan irányban, hogy ezáltal a szakértőtől kapott szubjektív koordinátáktól való eltérés csökkenjen. Az új súlyértékek alapján, ismételten meghatározásra kerülnek a tanítómintában szereplő szavak szubjektív koordinátái.

A tanulási folyamat addig tart, amíg a neurális hálózat által meghatározott szubjektív koordináták kellően közel nem kerülnek a szakértő által definiált értékeikhez [14]. A tanulást követően a konkrét dokumentumból valóban osztályozandó szavak objektív koordinátái kerülnek rákapcsolásra, a neurális hálózat bemenetére, melyekre adott válasz fogja jelenteni, a szavak szubjektív koordinátáit. Mivel a szubjektív koordináták jelentős részének értékkészlete is kettőnél több értéket tartalmaz, ezért a kimeneten is szükség van értékek közötti konverzióra. Ennek iránya

ellentétes, a bemeneti illesztésnél alkalmazottal. Itt, a neurális hálózat által szolgáltatott bináris jelek kerülnek többértékűvé alakításra. A szubjektív koordináták közül a „*kérdésként kiemelésre kerüljön-e”* névvel ellátott koordináta értéke szolgál annak eldöntésére, hogy a vizsgált szó kérdésként kiemelésre kerüljön-e a mondatból. Ez a koordináta azonban csak irányadó szerepet tölt be. A pontos döntés meghozatalát, a kérdésgenerálást ellátó modul végzi. A mondaton belül ugyanis lehet több szó is, melynek kiemelését ez a koordináta előirányozza, míg lehet olyan mondat is, mely egyik szava sem felel meg ennek az osztályozás által szolgáltatott feltételnek. A *relevancia*, *specialitás*, *értelmesség* valamint *nehézség* névvel ellátott szubjektív koordináták arra használhatóak, hogy a kérdésre adható lehetséges válaszok ebből a szubjektív szempontból is minél közelebb legyenek a mondatból kiemelt valóban helyes válaszhoz.
Az osztályozást végző modul a 4.5. ábrán látható dokumentumrendszerből sorra veszi azokat a mondatokat, melyek „*elemzésre kerüljön-e*” attribútuma igaz értéket tartalmaz. Ezen mondatok esetén – szakértői döntések alapján – beállítja a mondatok „*része-e a tanítómintának*” attribútumához tartozó értéket. Az osztályozást végző modul egyik bemenetét a tanítómintába kerülő mondatok szavainak „objektív koordináták” valamint „szubjektív koordináták” nevű halmazokba tartozó attribútumai adják. A másik bemenetet a tanítómintába nem kerülő, elemzendő mondatok szavainak „objektív koordináták” nevű halmazba tartozó attribútumai reprezentálják. A modul ezen információk alapján előállítja a tanítómintába nem kerülő elemzendő mondatok szavainak a „szubjektív koordináták” nevű halmazba tartozó attribútumainak az értékét.

4.2.5. A kérdésgenerálást végző modul

A kérdésgenerálást végző modul funkcionális rendszertervét a 4.10. ábra szemlélteti.



4.10. ábra: A kérdésgenerálást végző modul funkcionális rendszerterve

Az eddig ismertetett modulok feladata a kérdésgeneráláshoz szükséges információk előállítása. Ezek az információk vezérlik a kérdésgenerálást végző modul működését. Ebben a modulban történik a kérdésekben használandó mondatok, valamint a mondatokból, kérdésként kiemelésre kerülő szavak meghatározása. Ezen felül ez a modul állítja elő a feltett kérdésekre adható lehetséges válaszokat is. A modul ezeket az információkat szolgáltatja, az eredményeket interaktív, illetve nyomtatható változatban megjelenítő modul számára.

A modul működéséhez szükséges adatok

Adat leírása	**Adatot előállító modul neve**
Az osztályozásra került szavakat és a hozzájuk tartozó *„kérdésként kiemelésre kerüljön-e”* szubjektív koordinátákat tartalmazó halmaz.	Kérdésgeneráló modul
Az osztályozásra került szavakat és a hozzájuk tartozó *relevancia, specialitás, értelmesség, nehézség* szubjektív koordinátákat tartalmazó halmaz.	Kérdésgeneráló modul
A kérdésként kiemelhető szavakat és a rájuk adható lehetséges válaszokat strukturált formában tartalmazó fájl.	Válaszalternatíva szótárat készítő modul
A dokumentum mondatait annotáltan tartalmazó szöveges fájl.	Annotáló modul
A kérdésekben szereplő mondatok típusainak halmaza.	Szakértői döntés

4.10. táblázat: A kérdésgenerálást végző modul működéséhez szükséges adatok

A modul által szolgáltatott adatok

Adat leírása	**Adatot feldolgozó modulok neve**
Az aktuális paraméterek alapján előállított tesztben a kiválasztott mondatokból megkérdezésre kerülő szavakat és a rájuk adható lehetséges válaszokat strukturáltan tartalmazó lista.	Tesztek elektronikus és nyomtatható változatainak elkészítését végző modul

4.11. táblázat: A kérdésgenerálást végző modul által szolgáltatott adatok

A modul működése

A kérdésgenerálás első lépéseként a dokumentumnak, a kérdésként szereplő mondatait kell meghatározni. Ehhez a modulnak, bemeneti információként meg kell kapnia a dokumentum mondatait annotáltan tartalmazó szöveges fájlt. Ezt követően, szakértői döntés alapján, határozhatók meg azok a mondattípusok, melyekből kérdéseket lehet előállítani. Minden lényeges mondattípus külön annotációval lett ellátva a dokumentumban. Ezáltal könnyen meghatározhatóak a szakértő által definiált szűrőfeltételeknek megfelelők. A mondatok típusán kívül, ezen a ponton dönthető el a kérdésként szereplő mondatok száma is. Ez szintén szakértői vélemény alapján történik. Ezeknek az információknak a megváltoztatásával, lehetőség nyílik ugyan abból a dokumentumból, több különböző típusú és mennyiségű kérdés feltételére is.

A mondatok meghatározását követően a kérdésként kiemelhető szavak kiválasztása következik. Ehhez iránymutatásként szolgálnak a szavaknak az osztályozási modultól kapott *„kérdésként kiemelésre kerüljön-e”* nevű szubjektív koordinátái. Mivel ezek a koordináták minden szót egyedileg jellemeznek, ezért előfordulhat, hogy egy mondaton belül több szó is megjelölhető kérdésként kiemelhetőnek. Ebben az esetben, a modul a lehetséges alternatívákat a neurális hálózat kimeneti neuronjainak bemeneti függvénye szerint rangsorolja. Azok a szavak, amelyeknél ez az érték magasabb, nagyobb valószínűséggel kerülnek kérdésként kiemelésre. A kérdésgenerálás utolsó lépéseként, a megkérdezésre kiválasztott szóhoz, meg kell határozni a válaszként adható lehetséges alternatívákat. Az alternatíváknak olyanoknak kell lenniük, hogy egyértelműen mérhető legyen általuk, a válaszadó személy tudásszintje. Szerepelni kell közöttük a kérdésként kiemelt szónak, mint tökéletes megoldásnak, ám szerepelnie kell közöttük ettől a szótól objektív és szubjektív szempontból egyre távolabb lévő lehetőségeknek is. Ezek a helyes megoldástól egyre távolodó alternatívák alkalmasak a válaszadó tudásszintjének pontozására. Az alternatívák közötti

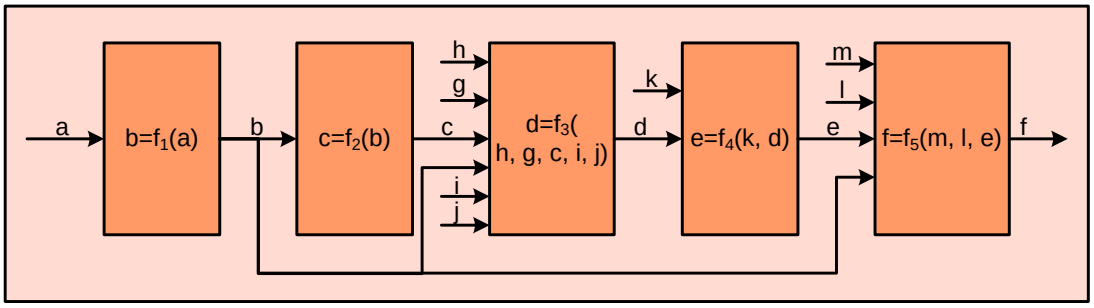
távolságképzést több szempont alapján valósítottam meg. Egyrészt felhasználtam a klaszterezést végző modulban kiválasztott távolságképzési módszert (szógyakoriság vagy vektortér), valamint építettem a fogalomhierarchiával reprezentált szó-kategóriaszó kapcsolat által meghatározható távolságra is. Ezáltal a válaszként automatikusan felkínált alternatívák segítségével pontosan meghatározható a válaszadó személy felkészültsége, mind a konkrét dokumentum ismerete (klaszterezésnél használt távolságképzés), mind pedig a fogalmak jelentése (fogalomhierarchia alapján számított távolságképzés) szempontjából.
Az osztályozást végző modul a 4.5. ábrán látható dokumentumrendszerből sorra veszi azokat a mondatokat, melyek „*elemzésre kerüljön-e*" attribútuma igaz értéket tartalmaz. Ezeknek a mondatoknak meghatározza a „*tesztlapon megjelenítésre kerüljön-e*" attribútumaihoz tartozó értékét a szakértő által meghatározott darabszám figyelembevételével. Ezt követően a tesztlapon megjelenítendő mondatokra a „*szubjektív koordináták*" nevű halmazba tartozó attribútumok, valamint az osztályozás során megtanult hipotézisfüggvény alapján meghatározza a „*szubjektív koordináták*" nevű halmazba tartozó attribútumok értékeit. Az objektív és szubjektív koordináták ismerete alapján a tesztlapon megjelenítendő minden mondatnak meghatározza a „*kérdésként kiemelendő szó*" attribútum értékét. A kérdésekre adható válaszalternatívák a „*fogalomhierarchia"* és a *„statisztikai távolságmérés"* alapján valósul meg, amely „*elektronikus"* és „*nyomtatható"* feladatlap formában jelenik meg a felhasználó előtt.

4.3. Az automatizált kérdésgenerálás rendszertervének összefoglalása

A szöveges dokumentumokon végzett automatizált kérdések, valamint válaszok előállításához több egymással adatkapcsolatban álló alrendszer (rendszermodell) kidolgozására volt szükség. Ezeknek az alrendszereknek a megtervezését, a moduláris rendszerek tervezési szabályai alapján végeztem el. Annak érdekében, hogy a rendszer moduljai egymástól függetlenül elemezhetőek, valamint továbbfejleszthetőek legyenek, minden modul számára interfészeket dolgoztam ki, melyekben definiáltam a modul működéséhez igényelt, illetve a modul által szolgáltatott adatokat. Jelen fejezetben bemutattam a kidolgozott rendszer moduljait, valamint a közöttük lévő kapcsolatokat jelentő adatáramlásokat. A rendszer főbb moduljait részleteiben is kifejtettem. Ezek alapján jól követhető, hogy milyen adatokra és tevékenységekre van szükség a bemenetként kapott szöveges dokumentum alapján végzett automatizált kérdés-, valamint válaszgenerálás megvalósításához.
A 4.11. ábra transzformációs függvények láncolataként ábrázolja a kérdésgenerálás bemenetén kapott adatoknak elvárt kimeneti struktúrákra való alakításának folyamatát.

Ahol: a, A kérdések előállítására szolgáló HTML, DOCX, PDF vagy XML formátumban kódolt szöveges dokumentum;
b, Annotációkkal ellátott mondatokból álló ASCII formátumban kódolt szöveges dokumentum;
c, A dokumentum szavait, valamint a *Szószablya* alkalmazásával hozzájuk meghatározott objektív jellemzőket tartalmazó szöveges fájl;
d, A dokumentum minden klaszterezendő szavát egy klaszterhez rendelő struktúrákat tartalmazó halmaz;
e, Az osztályozott szavakat és a hozzájuk tartozó szubjektív koordinátákat tartalmazó halmaz;
f, Az aktuális paraméterek alapján előállított tesztben a kiválasztott mondatokból megkérdezett szavakat és a rájuk adható lehetséges válaszokat strukturáltan tartalmazó lista;
g, A dokumentum szavai közül a klaszterezendőeket tartalmazó halmaz;
h, Távolságképzési módszer;
i, Klaszterezési stratégia;
j, Alkalmazandó klaszterezési megoldás;

k, Szakértők által előállított tanítóminta;
l, A kérdésként kiemelhető szavakat és a rájuk adható lehetséges válaszokat strukturált formában tartalmazó fájl;
m, A kérdésekben szereplő mondatok típusainak halmaza;
f_1, Szöveges dokumentumok előfeldolgozási folyamatát leíró transzformációs függvény;
f_2, Szöveges dokumentumok szótövezési folyamatát leíró transzformációs függvény;
f_3, Szavak klaszterezési folyamatát leíró transzformációs függvény;
f_4, Szavak osztályozási folyamatát leíró transzformációs függvény;
f_5, A kérdések előállításának folyamatát leíró transzformációs függvény;



4.11. ábra: A kidolgozott AQG rendszerben kezelt adatstruktúrák transzformációs lánca

A modell helyességének igazolására a transzformációs függvényeket Java nyelven implementáltam. Az automatizáltan létrehozott kérdéssort különböző korú, nemű és eltérő ismeretbázissal rendelkező személyekkel töltettem ki. A tesztekre adott válaszokat összevetve a manuálisan előállított kérdéssorok eredményeivel bebizonyítottam, hogy a kidolgozott modell alkalmazható a manuálisan jóval nagyobb költséggel előállítható kérdéssorok kiváltására.

5. fejezet

Dokumentumok szavainak távolság alapú klaszterezése

Az automatikus kérdésgenerálás megvalósíthatóságának fontos feltételét képezi, a dokumentum szavainak megfelelő klaszterekbe szervezése. Ennek elsődleges célja, a dokumentumkészlet szavaiból olyan csoportok képzése, melyben szereplő szavak, bizonyos előre definiált tulajdonságaik alapján hasonlítanak egymáshoz. A szavak közötti tartalom alapú hasonlóságok helyes feltárása megteremti az alapokat arra, hogy tudásintenzív módszereket alkalmazó algoritmusokkal, automatikusan meghatározhatóak legyenek a kérdésként kiemelendő szavak, valamint a helyettesítésükre felkínálandó alternatívák. A szavak klaszterekbe rendezése során, két lényegesen eltérő koncepció lett kidolgozva a távolságok definiálására. Az egyik alapján a szavak távolsága a dokumentumok mondataiban való közös előfordulásuk gyakorisága szerint kerül kiszámításra. Ennek értelmében, két szó annál közelebb van egymáshoz, minél több olyan mondat található a dokumentumokban, melyekben a két szó közösen szerepel. A másik koncepció szerint a szavak távolsága, a szavak objektív módszerekkel meghatározható koordinátái által definiált többdimenziós térben való helyeinek vektortér alapú távolsága alapján került meghatározásra. Mindkét koncepció több különböző algoritmussal is implementálásra került az eredmények összehasonlíthatósága valamint előnyös, illetve hátrányos tulajdonságaik feltárása érdekében.

5.1. A klaszterezés irodalmi áttekintése

A klaszterezés, egy adathalmaz pontjainak, rekordjainak, hasonlóság alapján való csoportosítását jelenti. A klaszterezés feladatával eleinte a statisztikában kezdtek foglalkozni, majd az adatbányászat keretében, az igen nagyméretű adathalmazok klaszterezésének kérdései kerültek az előtérbe. Így a klaszterezés az adatbányászat egyik legrégebbi és talán leggyakrabban alkalmazott része. A klaszterezés, illetve osztályozás során az adatpontokat diszjunkt csoportokba, klaszterekbe, illetve osztályokba soroljuk, azaz particionáljuk az adathalmaz elemeit.
A klaszterezés célja, a dokumentumokból olyan elkülönült csoportokat alkotni, hogy az egy csoportba kerülők minél hasonlóbbak, az eltérő csoportokban lévők, pedig minél különbözőek legyenek [7].
A klaszterezés formális felírása:

$$O = \{o\}, \qquad d = O\ x\ O \rightarrow \mathbb{R}^+$$
$$P_o: \{\{O_i, \dots, O_j\} \mid,\ O_i \cap O_j = \emptyset,\ \cup O_i = O\}$$
$$avg\ \{d\ (o_i, o_j) \mid \exists_k:\ o_i, o_j \in O_k\} < avg\ \{d(o_i, o_j) \mid \nexists_k: o_i, o_j \in O_k)\} \qquad (5.1)$$

ahol: O az o objektumok halmaza, d távolságfüggvény, P_o a O-nak egyfajta klaszterekre osztása.
Helyes klaszterezés esetén a klaszteren belüli átlagos távolság kisebb, mint a különböző klaszterekhez tartozó elemek közötti átlagos távolság.
A feladat nehézségei:

- az objektumok reprezentálása,
- a hasonlóság mérésére különböző módszerek léteznek,
- a klaszterhatárok kialakítása nem egyértelmű,
- nincs egyértelmű mérőszám a csoportképzés jóságának mérésére,
- nagyméretű feladatok kezelésének hatékonysága alacsony.

A klaszterezés folyamat során a megfelelő csoportok kialakítása nem egyértelmű feladat, mert az emberek gyakran más-más szempontokat vesznek figyelembe a csoportosításnál. Ugyanazt az adathalmazt, alkalmazástól és szokásoktól függően, eltérően klasztereznek az emberek. Klaszterezéskor, az adathalmaz mellett meg kell adnunk, hogy miként definiáljuk az elemek hasonlóságát, továbbá, hogy mi alapján csoportosítjuk az elemeket. Objektív definíciót kell adnunk az elemek hasonlóságának mértékére és a klaszterezés minőségére. Amennyiben megfelelő matematikai modellbe ágyaztuk a feladatot, lehetőség nyílik olyan algoritmusok megkeresésére, amelyek jól és gyorsan oldják meg a feladatot. A klaszterezés, az adatpontok hasonlósági viszonyaiból indul ki, ezért az első fontos lépés a hasonlósági függvény kiválasztása. Nagy adathalmazok klaszterezése során gyakran találkozunk azzal a problémával, hogy az adatpontok távolságának, illetve hasonlóságának kiszámítása nem oldható meg elég gyorsan ahhoz, hogy az adott klaszterező algoritmus elfogadható futási időben véget érjen. Számos esetben e problémának, az adatok sok attribútummal való jellemzése az oka. A hasonlóság vagy távolság hatékony kiszámításának problémáját megoldhatja az adatok dimenziójának csökkentése, amely során az adathalmazt olyan formába alakítják, amelyen már hatékonyan tudják a kívánt klaszterező algoritmust futtatni [1].

5.1.1. A hasonlóság és távolság tulajdonságai

Legyen adott n elem (objektum, egyed, megfigyelés). Tetszőleges két elem (x, y) között értelmezzük a *hasonlóságukat*. Gyakran a hasonlóság helyett annak inverzét, a *különbözőség*et (távolság) használjuk $(d(x, y))$. A $d(x, y)$-től elvárjuk azt, hogy [Bodon, 2006]:

- nem negatív: $d(x, y) \geq 0$,
- reflexív: $d(x, x) = 0$,
- szimmetrikus: $d(x, y) = d(y, x)$ legyen,
- és teljesüljön a háromszög-egyenlőtlenség: $d(x, z) \leq d(x, y) + d(y, z)$.

A klaszterezés legáltalánosabb esetében minden egyes elempár távolsága előre adott. Az adatokat ekkor, egy ún. távolságmátrixszal reprezentáljuk:

$$\begin{bmatrix} 0 & d(1,2) & d(1,3) & \dots & d(1,n) \\ & 0 & d(2,3) & \dots & d(2,n) \\ & & 0 & \cdots & d(3,n) \\ & & & \ddots & \vdots \\ & & & & 0 \end{bmatrix}$$

ahol $d(i, j)$ adja meg az i-edik és a j-edik elem különbözőségét.
A gyakorlatban az n adatpont legtöbbször attribútumokkal van leírva, és a hasonlóságokat az attribútumok értékeiből valamilyen függvény segítségével számolhatjuk ki. Ha megadjuk a függvényt, akkor felírhatjuk a megfelelő hasonlóságmátrixot.

5.1.2. Klaszterezési módszerek

A legszélesebb körben használt szabványos módszerek a hierarchikus [Jain, 1999], particionáló [Day, 1992], hibrid [Murty, 1980], inkrementális, illetve nem inkrementális, monothetikus kontra polythetikus [Salton, 1991] és fuzzy [Ruspini, 1969] módszerek. Mindegyik módszernek megvan a maga tudományos megalapozottsága, hatékonyságuk és bonyolultságuk azonban különböző. A módszerek alapvetően abban a tulajdonságukban különböznek, hogy végeredményként *bináris*

vagy *fuzzy* kimenetet adnak. Bináris (binary clustering) esetben egy adott szöveg csak egy klaszter tagja lehet, míg Fuzzy kimenet esetén tagsági függvényekről beszélünk, ami minden szövegnek minden klaszterre vonatkozóan 0 és 1 közötti értékben fejezik ki egy adott klaszterhez való tartozásának az értékét. A felsorolt klaszterezési módszerek közül munkám során a hierarchikus módszereket használtam, ezért ezzel részletesen foglalkozom.
A *hierarchikus módszerek* onnan kapták a nevüket, hogy az elemeket hierarchikus adatszerkezetbe (fába, dendogramba, taxonómiába) rendezik el. Az adatpontok a fa leveleiben találhatók. A fa minden belső pontja egy klaszternek felel meg, amely azokat a pontokat tartalmazza, melyek a fában alatta találhatók. Két fő fajta hierarchikus eljárást különböztetünk meg: a *lentről építkezőt*, más néven *egyesítőt* és a *fentről építkezőt*, avagy az *osztót*. A lentről felfelé építkező felhalmozó eljárásoknál kezdetben minden elem különálló klaszter, majd az eljárás a legközelebbi klasztereket egyesíti, a hierarchiában egy szinttel feljebb újabb klasztert alakítva ki. A fentről lefelé építkező lebontó módszerek fordítva működnek: egyetlen, minden adatpontot tartalmazó klaszterből indulnak ki, amit kisebb klaszterekre particionálnak, majd ezeket is tovább bontják. A hierarchikus algoritmusok kulcslépése az egyesítendő vagy osztandó klaszterek kiválasztása. Miután az *egyesítés (osztás)* megtörténik, minden további műveletet az új klasztereken végzünk el.
A HAC [Manning, 2008] algoritmus esetén, minden lépésben két legközelebbi klasztert von egybe. A leállás feltétele a minimális klaszterszám vagy a maximális összevonási távolság.
Az algoritmus lépései:

- minden elem egy önálló klaszter,
- a két legközelebbi klaszter meghatározása,
- a két legközelebbi klaszter összevonása egybe,
- a távolságok aktualizálása,
- a fenti eljárás folytatása, amíg a leállási feltétel ezt megengedi.

Legismertebb HAC klaszterezési eljárások: legkisebb, legnagyobb és átlagos távolságon alapuló eljárás, Ward módszer, a BIRCH algoritmus, a CURE (Clustering Using REpresentatives) algoritmus, a Chameleon algoritmus.
A HAC módszer az egyik legszélesebb körben használt és legegyszerűbb klaszterezési algoritmus.
A klaszterek közötti távolságot $(d(C_{i,}C_j))$ a következő módokon értelmezhetjük:

- minimális távolság: $d_{min}(C_{i,}C_j) = \min_{p\in C_{i,}q\in C_j} d\,(p,q),$ (5.2)

- maximális távolság: $d_{max}(C_{i,}C_j) = \max_{p\in C_{i,}q\in C_j} d\,(p,q),$ (5.3)

- átlagos távolság: $d_{avg}(C_{i,}C_j) = \frac{1}{|C_i|\ |C_j|} \sum_{p\in C_i} \quad \sum_{q\in C_j} d\,(p,q),$ (5.4)

- egyesített klaszter átmérője: $d_D(C_{i,}C_j) = D(C_i \cup C_j).$ (5.5)

5.1.3. Jelentés alapú klaszterezés

A szövegek automatikus feldolgozásánál alapvető elem a szavak tartalom alapú hasonlóságának ismerete. Ezen tezaurusz automatikus előállításának legfontosabb eszköze a szavak tartalom, jelentés alapú klaszterezése. A szavak jelentése szorosan kötődik a hozzá kapcsolódó kontextushoz [Dash, 2005], mely szerint a jelentés a használati kontextusok együttese, illetve a jelentés a kontextus alapján válik egyértelművé. A kontextus a szó előfordulási környezetét jelenti. A kontextus és jelentés kapcsolatát többek között [Dash, 2005a] vizsgálta részletesen, melyben négyféle alapkontextus kerül megkülönböztetésre:

- lokális,
- mondat szintű,
- témakör szintű,
- globális.

A lokális kontextus alapú jelentésvizsgálatnál elegendő a szó szomszédságában előforduló szavakat vizsgálni. A mondat szintű elemzésnél a mondat teljes szókészletét figyelembe kell venni. A témakörnél rendszerint a cikk, a fejezet, a dokumentum szókincsét használják fel. A globális értelmezés ezen lokális értelmezések integrációjából épül fel.

A kísérletek során megfigyelhető, hogy legtöbb esetben a helyi szövegösszefüggésből nyert információ elegendő a kulcsszavak aktuális összefüggő jelentésének megértéséhez. Megfigyelhető az is a gépi fordításban, hogy ha lehetséges a szó jelentését kivonni a helyi szövegkörnyezetből, akkor néhány fordítási probléma megszüntethető [Dash, 2007]. A vizsgálatok során több speciális problémával szembesülhetünk. Egyik nehéz feladat a szavakból képzett kifejezések használata. Egy szólánc egyes esetekben önálló jelentést fog kapni, mely lényegesen eltérhet az egyes alkotó elemek jelentéseinek összegétől.

Ezen probléma feldolgozását tovább nehezíti az a tény, amire többek között [Dolores, 2007] eredményei is rávilágítanak: a szavak jelentését nem a szavak elemzésén keresztül közvetlenül kapjuk meg, előbb a teljesre (a mondatra, a témakörre) pontosítunk, teszünk fel hipotézist, majd ez alapján határozzuk meg a részek, az alkotó szavak jelentését. A jelentés és kontextus szoros kapcsolatából következően, a szavak jelentésének automatikus meghatározásánál az egyik alapvető módszer a szavak kontextusának vizsgálata. A módszer alapötlete, hogy a szavak valamely kontextusát vizsgálva a hasonló kontextusú szavak kerülnek egy csoportba.

[Pedersen & Bruce, 1997] továbbá [Pedersen & Bruce, 1998] feltételeznek egy hasonlóságon (különbözőségen) alapuló diszkriminatív megközelítést, amely kiszámítja a hasonlóságot (különbséget) a célszó minden egyes előforduló párja között. Ez az információ bejegyzésre kerül a hasonlósági (különbségi) mátrixban, amelynek sorai/oszlopai reprezentálják a célszó diszkriminálandó előfordulásait. A mátrixcella bejegyzési mutatják annak a fokát, amennyire a megfelelő sor és oszlop által reprezentált előfordulási pár hasonló (eltérő). A hasonlóságot (különbséget) az előfordulások elsőrendű kontextusvektorokból számítják, amelyek minden egyes előfordulást egy vektor jellemzőiként mutatnak, amely közvetlenül a célszó közelében van abban az előfordulásban.

[Schutze, 1998] bevezeti a másodrendű kontextusvektorokat, amelyek előfordulást reprezentálnak, átlagolva a tartalmi szavak jellemző vektorait, amelyek a célszó szövegkörnyezetében vannak abban az előfordulásban. Ezek a másodrendű kontextusvektorok lesznek bemenetei a klaszterező algoritmusnak, amely az adott szövegkörnyezetben klaszterez a vektortérben, ahelyett, hogy hasonlósági mátrixrendszert építene fel.

[Purandare & Petersen, 2004] munkájában objektumtérként vektorteret vesznek, ahol megkülönböztetnek elsőrendű és másodrendű kontextusvektorokat. Az elsőrendű kontextusvektorok közvetlenül mutatják milyen jellemzőkből épül fel a kontextus. Minden

kísérletben a célszó kontextusa 20, mindkét oldalról környező tartalmi szóra korlátozódik. Ez igaz akkor is, amikor az oktató adatok halmazából választunk jellemző vonásokat vagy akkor is, amikor tesztelőfordulásokat konvertálunk vektorokká a klaszterezéshez. A tesztelőfordulás reprezentálható egy másodrendű vektorral megkeresvén az elsőrendű kontextusvektorok átlagát, amelyek megfeleltetésben állnak azokkal a szavakkal, amelyek a célszó körül fordulnak elő. Ezért a másodrendű kontextusreprezentáció a jellemző szavak elsőrendű kontextusvektorain múlik. A másodrendű kísérletek a jellemzők két különböző típusát használják, ezek az előfordulások és bigrammok, melyeket úgy definiálnak, mint ahogy az elsőrendű kísérletekben.
A szókategóriák felfedezése fontos lépés a teljesen automatizált szövegtartalom-feldolgozó rendszerekben. A szókategóriák tekinthetők hasonló nyelvtani jellemzőkkel rendelkező szavakat tartalmazó klasztereknek. [Kovács, 2006] munkájában, a kifejezés kontextusa olyan mondatok halmazaként van definiálva, mely a kifejezést tartalmazza. Az oktató halmaz mondatok halmazaként adott. A vizsgálatban a mondat szavak halmazaként tekinthető. Ezért a szavak helyzete figyelmen kívül marad. Az oktató halmaz formális definíciója a következőképpen van megadva:

- w = a szó,
- W = w szavak halmaza,
- s = W részhalmaza, egy mondat,
- S = mondatok halmaza, oktató halmaz.

Egy mondat jelöli a kapcsolat-előfordulást a tartalmazott szavak között. Egy szó előfordulhat számos mondatban. Ez a halmaz a szó kontextusaként definiált:

$$C(w) = \{s\ w | s\ \in S, w\ \in s\}.$$

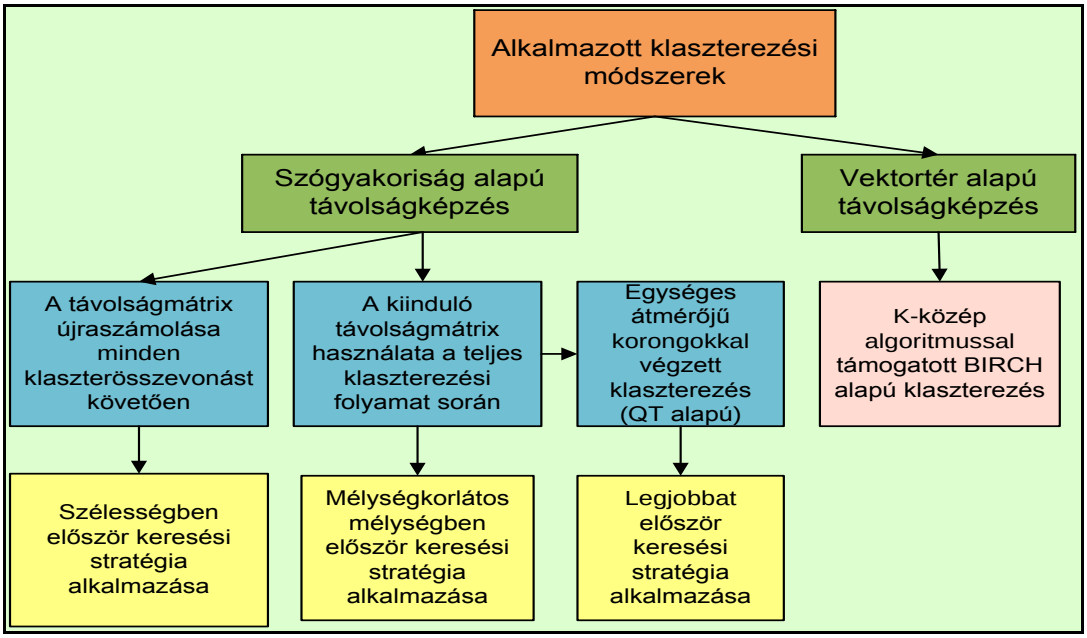
Erre a fogalomra alapozva, két szó hasonlósága a megfelelő kontextushalmazok hasonlóságával definiálható. Az efféle hasonlósági érték kiszámításához két halmaz közötti hasonlóság, majd halmazokon belüli két halmaz hasonlóságát kell definiálni. A szokásos megközelítésben, két halmaz közötti hasonlóság a következő módon definiált [Kovács, 2006]:

$$d(s_1, s_2) = 1 - \frac{|s_1\ \cap\ s_2|}{|s_1\ \cup\ s_2|}.$$

A szavak közötti kontextustávolság-mátrix generálása után a szavakat klaszterezni kell e távolságértékekre alapozva. A klaszterező algoritmusok közül a hierarchikus egyesítő klaszterezési módszert használták a szerzők.
[Shin, 2003] a dolgozatban jelentések és kifejezések kapcsolódását vizsgálják. Ez a kutatás alkalmazza a k-közép algoritmust a szókapcsolatok automatikus klaszterezéséhez. Ez a módszer osztályozza a központi szó kontextuális szavait k klaszterbe. A szókapcsolatban lévő szavaknak is vannak szókapcsolatai. A szókapcsolathoz tartozó célszót hívják „központi szónak” és a szókapcsolatban lévő szóra úgy utalnak, mint „kontextuális szó”. Ha a szókapcsolatminta a kontextuális szavak között hasonló, az azt jelenti, hogy a kontextuális szavakat használják a hasonlósági kontextusban – ahol a centrális szóhoz hasonló értelemben használják és kapcsolódnak – a mondatban. Ha a kontextuális szavakat a szókapcsolatokban előforduló hasonlóságok alapján klaszterezik, a többjelentésű központi szavakhoz tartozó kontextuális szavak a központi szó értelmezései alapján osztályozhatók.

5.2. A szavak klaszterezésére kidolgozott módszerek és algoritmusok

A dokumentumok szavainak klaszterezésére két lényegesen eltérő koncepció került kidolgozásra (5.1. ábra). A szógyakoriság alapú távolságképzés alkalmazásával a szavak mondatokban való közös előfordulási gyakoriságának, míg a vektortér alapú távolságképzés során a szavak objektíven mérhető tulajdonságai közötti hasonlóságnak a figyelembe vételére van lehetőség.
Az 5.2.1., illetve 5.2.2. pontokban a két koncepció, illetve a megvalósításukat végző algoritmusok kerülnek ismertetésre.



5.1. ábra: Alkalmazott klaszterezési módszerek

5.2.1. Szavak közös előfordulási gyakoriságára épülő távolság-meghatározás

A szavak közös előfordulási gyakoriságára épülő távolság-meghatározási koncepció értelmében két szó távolsága azon mondatok számával definiálható, melyekben mindkét szó egyszerre szerepel. Az így meghatározott távolságadatok négyzetes mátrixszal (távolságmátrix) írhatóak le. A mátrixnak mind a sorai, mind pedig az oszlopai a dokumentum szavaival kerülnek indexelésre. A mátrix i. sorában és j. oszlopában szereplő érték az (5.6) összefüggés alapján került meghatározásra:

$$S_{i,j} = f_{i,j} \,/\, \max(f_i,\ f_j) \qquad d = 1 - S_{i,j} \tag{5.6}$$

ahol: S_{ij} az i. és j. szó távolságviszonyát reprezentáló [0, 1] intervallumba eső számérték (a nagyobb érték a szavak közötti kisebb távolságot jelöli); f_{ij} a dokumentum azon mondatainak száma, melyekben az i. és a j. szó is szerepel; f_i a dokumentum azon mondatainak a száma, melyekben az i. szó szerepel; f_j a dokumentum azon mondatainak a száma, melyekben a j. szó szerepel.
A klaszterezés a szavaknak az (5.6) összefüggés által definiált távolsága alapján történik.

A távolságszámítás klaszterekre való kiterjesztése

Az általam elkészített szoftverben a HAC algoritmus általános módszerének [Bodon, 2010] két feladatspecifikus továbbfejlesztése került megvalósításra. Az első módszer a klaszterezendő szavaknak azt a tulajdonságát használja ki, hogy ezek kapcsolata a távolságon kívül a mondathoz tartozásukkal is jellemezhető. Ez teszi őket különbözővé a mélyebb kapcsolat nélkül – pusztán a távolságuk alapján – klaszterezhető pontoktól. Ez a módszer az (5.6) összefüggés használatát szavak helyett klaszterekre terjeszti ki. Ennek értelmében f_i jelenti azon mondatok számát, melyben az i. klaszterhez tartozó minden szó szerepel; f_j jelenti azon mondatok számát, melyben a j. klaszterhez tartozó minden szó szerepel; f_{ij}, pedig azon mondatok számát, melyben mind az i. mint pedig a j. klaszterhez tartozó minden szó szerepel.
Az (5.6) összefüggés következetes alkalmazásával minden klaszterbe csak olyan szavak kerülhetnek, melyek a dokumentum legalább egy mondatában együtt szerepelnek. Ez a koncepció, a szavak mondatokhoz tartozásában rejlő tulajdonságának megőrzése mellett, korlátot szab a klaszterekbe tehető szavak maximális számára is. A módszer hátránya, hogy minden klaszter-összevonási lépést követően szükség van a teljes távolságmátrix újraszámolására. Az (5.6) összefüggés klaszterekre való kiterjesztésével ugyanis a távolságmátrix már nem szavak, hanem klaszterek távolságát tárolja, a klaszterek száma és tartalma pedig minden összevonást követően megváltozik. A költségfüggvény egy lépésre és a teljes lépésszámra a következőképpen határozható meg:

$$O(N-d),$$
$$O(N*d*N) = O\,(N^2*d). \qquad (5.7)$$

A klaszterezési folyamat minden lépésében azok a klaszterek kerülnek összevonásra, melyek távolsága az (5.6) összefüggés alapján a legkisebb. Minden olyan eset, amikor egy klaszter több más klaszterhez viszonyított távolsága egyformán a legkisebb, a keresési folyamat elágazását eredményezi. Ezáltal ugyanannak a feladatnak több megoldása is lehetséges. A megoldások megtalálása fa-bejárási algoritmusok alkalmazását igényli. Mivel a jelenlegi feladatban egy megoldás jósága kizárólag a szubjektív térrel való korrelációjától függ, ezért a klaszterezési folyamat során kapott közbenső állapotok jósága nem mérhető közvetlenül. Az algoritmus sebességének növelése érdekében, a hagyományos „szélességi keresés" (breadth-first-search) [Stuart, 2005] algoritmus mellett, a klaszterezést megvalósítottam a „mélységkorlátos mélységben először" (depth limited depth-first-search) [Stuart, 2005] algoritmussal is.
A „szélességi és mélységkorlátos mélységben először" algoritmus célfüggvénye a legelső olyan állapot megtalálása, melyben a dokumentum minden klaszterezendő szava, eleme pontosan egy klaszternek, valamint egyetlen klaszterben sincs két olyan szó, melyek távolsága meghaladja a paraméterként megadott értéket. A kidolgozott két algoritmus abban különbözik egymástól, hogy a kezdeti klaszterezetlen állapotból kiindulva milyen stratégia szerint határozzák meg a következő klaszterezési lépést. Eszerint a szélességi keresés minden lépésben előállítja az összes olyan állapotot, melyek a kiinduló állapotból az adott lépés számával azonos számú szó klaszterbe szervezésével előáll. Ezzel szemben, a „mélységkorlátos mélységben először" keresés az éppen meghatározott állapotot klaszterezi tovább, az előre definiált lépésszám vagy a teljes klaszterezettség eléréséig.
A mélységi keresés alkalmazása esetén minden klaszterösszevonást követően, b (elágazási tényező) számú új csomópont születik. A maximális lépésszámot igénylő megoldás mélységét m-el jelölve, a keresés tárigénye $O(b*m)$ lesz, azaz lineáris a hely komplexitásával. A keresés tárigénye nagyon kicsi, mivel csak a gyökércsomóponttól a levélcsomópontig vezető utat kell

tárolnia, kiegészítve az út minden egyes csomópontja melletti kifejtetlen csomópontokkal. A keresés időigénye $O(b^m)$, ami megegyezik a szélességi keresés legrosszabb esetben vett időigényével [Stuart, 2000]. A mélységkorlátozott keresés szinte megegyezik a mélységi kereséssel, de egy bizonyos mélység után nem terjeszti ki a csomópontokat, ezáltal kényszerítve ki a visszalépést. A mélységi keresőhöz hasonlóan időigénye $O(b^k)$, tárigénye $O(b * k)$, ahol k a mélységkorlátot jelöli. A mélységkorláttal történő keresésnél kulcsfontosságú egy jó korlát megválasztása. Az iteratívan mélyülő keresés lényege, hogy ötvözi a szélességi és mélységi keresés előnyös tulajdonságait, azaz addig próbálja az összes lehetséges mélységkorlátot, míg megoldást nem talál. A kiterjesztés a szélességi keresésnek megfelelően történik, azzal a különbséggel, hogy itt a csomópontot többször is kiterjeszthet, s így a keresés teljes és optimális, de a mélységi kereséshez hasonlóan memóriaigénye viszonylag kicsi. A mélységkorlátozott keresésnél a kiterjesztések száma:

$$1 + b + b^2 + b^3 + \ldots + b^k = O(b^{k+1})$$

míg, az iteratívan mélyülő keresésnél ez

$$(k + 1)1 + (k)b + (k - 1)\ b^2 + \ldots + 2b^{k-1} + 1b^k = O(b^k).$$

Például a $b = 10$ elágazási tényező mellett az iteratívan mélyülő keresés 1-es mélységtől k mélységig csak körülbelül 10%-kal terjeszt ki több csomópontot, mint az egyszerű szélességi keresés vagy a k mélységkorláttal rendelkező mélységkorlátozott keresés. Az iteratívan mélyülő keresés időigénye $O(b^k)$, tárigénye $O(b * k)$ [Cawsey, 2002].
A megvalósításra került klaszterező algoritmusokban használt fogalmakat az 5.1. táblázat foglalja össze.

Definiálandó fogalom	**A fogalom definíciója**
szó	A magyar helyesírás szabályai szerint egy értelmes szó.
mondat	*Szavak* listája (1 vagy több *szó*).
dokumentum	*Mondat*ok listája (1 vagy több *mondat*).
klaszter	*Szavak* halmaza (1 vagy több *szó*).
állapot	*Klaszter*ek halmaza, ahol a *dokumentum* minden *szava* pontosan egy *klaszter* eleme.
két klaszter összevonása	A *klaszter*ek összevonásának feltétele, hogy mindkét *klaszter* ugyanabban az *állapot*ban legyen. Két *klaszter* összevonása során mindkét *klaszter* megszűnik és létrejön helyettük egy új *klaszter*, melynek *szavai* az összevont *klaszter*ek szavainak uniójával adható meg.
klaszter előfordulási gyakorisága	A *dokumentum* azon *mondat*ainak a száma, melyek a *klaszter*ben szereplő minden *szó*t tartalmaznak.
két klaszter közös előfordulási gyakorisága	A *dokumentum* azon *mondat*ainak a száma, melyek mindkét *klaszter*ben szereplő minden *szó*t tartalmaznak.
két klaszter távolsága	Két klaszter távolsága egy osztással határozható meg, melyben a számláló a két klaszter közös előfordulási gyakorisága, a nevező pedig a két klaszter előfordulási gyakoriságainak maximuma.
kiinduló állapot	Olyan *állapot*, ahol minden *klaszter* pontosan egy *szó*t tartalmaz.
végállapot	Olyan *állapot*, melyben nem található két olyan *klaszter*, melyek távolsága kisebb egy előre megadott értéktől.
aktuális állapot	Az elemzés alatt álló állapot.
gyermek állapot	Egy *állapot gyermek állapot*a az az *állapot,* amely az adott *állapot* két *klaszter*ének összevonásával kapható.

5.1. táblázat: A megvalósításra került klaszterező algoritmusokban használt fogalmak

A szélességi kereséssel megvalósított klaszterezés lépéseit az 5.1. algoritmus mutatja.
Szélességi kereséssel megvalósított HAC algoritmus folyamatábrája az *A.1* Függelékben található.

Lépés-szám	Cselekvés
1.	A *kiinduló állapot* előállítása. (A *dokumentum* összes különböző *szavának* kigyűjtése és az így kapott *szavak* mindegyikének külön *klaszter*be tétele.)
2.	Ha a *kiinduló állapot* teljesíti a *végállapot* feltételét, akkor a *kiinduló állapot* felvétele a megoldások listájára! Lépj a 19. lépésre!
3.	A *kiinduló állapot* felvétele az elemzendő állapotok listájára.
4.	Az *aktuális állapot* legyen az elemzendő állapotok listájának legelső eleme.
5.	Az *aktuális állapot klaszter*einek minden párosítását elvégezve a két legközelebbi *klaszter* távolságának meghatározása.
6.	Az *aktuális állapot klaszter*einek minden párosítását elvégezve összevonandónak nyilvánítani azokat a *klaszter*párokat, melyek távolsága azonos a legközelebbi *klaszter*ek távolságával.
7.	Összevonni minden olyan összevonandónak nyilvánított *klaszter*párt, melyek összevonása nem hiúsítja meg más összevonandónak nyilvánított *klaszter*pár összevonását. (Példa: jelölje A, B, C, D, E, F az *aktuális állapot* hat *klaszter*ét. Jelölje továbbá A-B, A-D, D-F, C-E az *aktuális állapot* három összevonandónak nyilvánított *klaszter*párát. Ekkor az A-B *klaszter*ek összevonása meghiúsítja az A-D *klaszter*ek összevonását, Az A-D *klaszter*ek összevonása meghiúsítja az A-B, illetve a D-F *klaszter*ek összevonását, A C-E *klaszter*ek összevonása viszont nem hiúsít meg egyetlen összevonandó *klaszter*pár összevonását sem. Az ilyen *klaszter*párok összevonását kell elvégezni ebben a lépésben.)
8.	Ha maradtak a 7. lépésben összevonásra nem kerülő összevonandónak nyilvánított *klaszter*párok akkor lépj a 12. lépésre!
9.	Ha az *aktuális állapot* nem teljesíti a *végállapot* feltételét, akkor lépj az 5. lépésre!
10.	Ha az *aktuális állapot* nem szerepel a megoldások listáján, akkor az *aktuális állapot* felvétele a megoldások listájára!
11.	Ha a megoldások listáján lévő *állapot*ok száma elérte a beállított maximális értéket, akkor lépj a 19. lépésre, egyébként lépj a 17. lépésre!
12.	A 7. lépésben összevonásra nem kerülő összevonandónak nyilvánított *klaszter*párok összevonása oly módon, hogy minden összevonandónak nyilvánított *klaszter*pár számára egy új *állapot* kerül létrehozásra, melyben az *aktuális állapot klaszter*eiből csak az *új állapot* létrehozását kiváltó *klaszter*pár kerül összevonásra.
13.	A 12. lépésben létrehozott minden új *állapot* felvétele az *aktuális állapot gyermekállapot*ait tároló listára.
14.	Az *aktuális állapot gyermekállapot*ai közül azokat, amelyek teljesítik a *végállapot* feltételét és nem szerepelnek a megoldások listáján, vedd fel a megoldások listájára.
15.	Ha a megoldások listáján lévő állapotok száma elérte a beállított maximális értéket, akkor lépj a 19. lépésre!
16.	Az aktuális állapot gyermekállapotai közül azokat, amelyek nem teljesítik a *végállapot* feltételét és nem szerepelnek az elemzendő állapotok listáján, vedd fel az elemzendő állapotok listájára.
17.	Az *aktuális állapot* törlése az elemzendő állapotok listájáról.
18.	Ha az elemzendő állapotok listája nem üres, akkor lépj a 4. lépésre!
19.	Vége.

5.1. algoritmus: A szélességi kereséssel megvalósított klaszterezés algoritmusa

A mélységi kereséssel megvalósított klaszterezés lépéseit az 5.2. algoritmus mutatja.
Mélységi kereséssel megvalósított HAC algoritmus folyamatábrája az *A.2* Függelékben található.

Lépés-szám	Cselekvés
1.	A *kiinduló állapot* előállítása. (A *dokumentum* összes különböző *szavának* kigyűjtése és az így kapott *szavak* mindegyikének külön *klaszter*be tétele.)
2.	A *kiinduló állapot* felvétele az elemzendő állapotok listájára.
3.	Az *aktuális állapot* legyen az elemzendő állapotok listájának utolsó eleme.
4.	Az *aktuális állapot klaszter*einek minden párosítását elvégezve a két legközelebbi *klaszter* távolságának meghatározása.
5.	Ha az *aktuális állapot* nem teljesíti a *végállapot* feltételét és az elemzendő állapotok listáján lévő állapotok száma kevesebb, mint a keresés mélységkorlátja akkor lépj a 13. lépésre!
6.	Ha az *aktuális állapot* nem teljesíti a *végállapot* feltételét, akkor lépj a 9. lépésre!
7.	Ha az *aktuális állapot* nem szerepel a megoldások listáján, akkor az *aktuális állapot* felvétele a megoldások listájára!
8.	Ha a megoldások listáján lévő *állapot*ok száma elérte a megadott maximális megoldások darabszámát, akkor lépj a 18. lépésre!
9.	Az elemzendő állapotok listáján lévő utolsó *állapot* törlése a listáról.
10.	Ha az elemzendő állapotok listája nem tartalmaz több *állapot*ot, akkor lépj a 18. lépésre.
11.	Ha az elemzendő állapotok listájának utolsó *állapota* nem tartalmaz olyan *gyermekállapot*ot, amin még nem került elvégzésre a végállapot vizsgálat akkor lépj a 9. lépésre!
12.	Az elemzendő állapotok listáján lévő utolsó *állapot* azon *gyermekállapot*ai közül, amelyeken még nem lett elvégezve a végállapot vizsgálat a soron következő felvétele az elemzendő állapotok listájának végére. Lépj a 3. lépésre.
13.	Az *aktuális állapot klaszter*einek minden párosítását elvégezve összevonandónak nyilvánítani azokat a *klaszter*eket, melyek távolsága azonos a legközelebbi *klaszter*ek távolságával.
14.	Összevonni minden olyan összevonandónak nyilvánított *klaszter*párt, melyek összevonása nem hiúsítja meg más összevonandónak nyilvánított *klaszter*pár összevonását. (Példa: jelölje A, B, C, D, E, F az aktuális állapot hat *klaszter*ét. Jelölje továbbá A-B, A-D, D-F, C-E az aktuális állapot három összevonandónak nyilvánított *klaszter*párát. Ekkor az A-B *klaszter*ek összevonása meghiúsítja az A-D *klaszter*ek összevonását, Az A-D *klaszter*ek összevonása meghiúsítja az A-B, illetve a D-F *klaszter*ek összevonását, A C-E *klaszter*ek összevonása viszont nem hiúsít meg egyetlen összevonandó *klaszter*pár összevonását sem. Az ilyen *klaszter*párok összevonását kell elvégezni ebben a lépésben.)
15.	A 14. lépésben összevonásra nem kerülő összevonandónak nyilvánított *klaszter*párok összevonása oly módon, hogy minden összevonandónak nyilvánított *klaszter*pár számára egy új *állapot* kerül létrehozásra, melyben az *aktuális állapot klaszter*eiből csak az új *állapot* létrehozását kiváltó *klaszter*pár kerül összevonásra. Ebben a lépésben létrehozott minden új *állapot* felvétele az *aktuális állapot gyermekállapot*ainak listájára.
16.	Az *aktuális állapot* legelső *gyermekállapot*ának felvétele az elemzendő állapotok listájának végére.
17.	Lépj a 3. lépésre!

5.2. algoritmus: A mélységi kereséssel megvalósított klaszterezés algoritmusa

A korongokkal végzett lefedés módszere

Mivel a távolságmátrix minden klaszterösszevonás utáni újraszámolásának költsége már száz körüli mondatszám esetén is meghaladja a rendelkezésre álló számítási kapacitást, ezért szükség volt olyan algoritmus kidolgozására [1], amely kizárólag a kiinduló távolságmátrix alapján végzi el a klaszterezéshez szükséges összes számítást. A távolságmátrix folyamatos újraszámolásának mellőzésével jelentősen csökkenteni lehet a klaszterezési folyamatnak mind a számítási idő, mind pedig a memóriafoglalási igényét. Pusztán a klaszterezés kezdetén létrehozott távolságmátrix használatával azonban elveszik az az információ mellyel biztosítani lehet, hogy csak olyan szavak

kerüljenek egy klaszterbe, melyek a dokumentum legalább egy mondatában együtt szerepelnek. A szélességi és mélységi keresés algoritmusban, erre az információra épült a klaszterek távolságának meghatározásán túl, a klaszterek méretének korlátok között tartása is. Mivel a HAC algoritmus szabályainak értelmében a kiinduló állapotban minden pont (szó) külön klaszterben foglal helyet, ezért a kiinduló állapotban létrehozott távolságmátrixból nem deríthető ki kettőnél több szó esetén, hogy van-e olyan mondat, melyben azok együtt szerepelnek [7]. A kiinduló távolságmátrixra épülő klaszterezés esetén a klaszterek méretének korlátok között tartására egy ismert módszer, a klaszterek legtávolabbi szavai közötti távolság alapján végezni klaszterezést. Az elkészített algoritmus ennek a módszernek a QTC (Quality Treshold Clustering) algoritmus feladatspecifikus adaptálásával valósítottam meg. Az algoritmus a klaszterek átmérőjét a következőképpen definiálja [1]:

$$d = max_{i,j}\left\{\sqrt{\left(x_i - x_j\right)^2}\right\} \tag{5.8}$$

ahol: d a klaszter átmérő, x_i az i-dik pont összes objektív koordinátája, x_j a j-dik pont összes objektív koordinátája.

A klaszterezés során minden klaszter, egy előre definiált átmérőjű koronggal került modellezésre [1]. Ez hasonlít a legtávolabbi szavak távolságára épülő klaszterezéshez, mivel a korongok átmérőjével korlát szabható a klaszterbe tehető legtávolabbi pontok (szavak) távolságára. Ezen túlmenően azonban ez az algoritmus lehetőséget nyit a pontok több klaszterhez tartozásának kezelésére is. Ez a többletinformáció felhasználható arra, hogy a több klaszterhez is rendelhető szavak klaszterezése, az aktuális igényeknek megfelelően, dinamikusan változtatható legyen. A klaszterezést végző optimáló algoritmus célfüggvénye, a dokumentum összes klaszterezendő szavának minimális számú klaszterbe gyűjtése. Korlátfeltételként jelentkezik a klaszterek számára előre definiált egységes átmérő betartása. A klaszterezési folyamat kiinduló állapotában – az általános HAC algoritmus szabályainak értelmében – minden szó egy külön klaszterbe kerül. Az algoritmus során a klasztereket reprezentáló korongok átmérője folyamatosan növelésre kerül, ezáltal a klaszterek egymás után kebelezik be a szavakat. Ha a folyamat során egy klaszter minden szava eleme az adott klaszteren kívül más klaszternek is, akkor az adott klaszter megszüntetésre kerül.

Az algoritmus a „legjobbat legelőször” (Best-First-Search) [Stuart, 2005] stratégia alkalmazásával került megvalósításra. A folyamat során az algoritmus mindig annak a klaszternek növeli meg az átmérőjét, amelyik a növelést követően több klaszter megszűnését eredményezi. Ennek megállapításához egy teszt kerül végrehajtásra, melyben előállításra kerülnek a klaszterezési folyamat aktuális állapotában szereplő klaszterek összes lehetséges összevonásával létrejövő állapotokat. Az összevonhatóság feltétele, hogy a klaszterek átmérője ne haladhatja meg az előzőleg meghatározott értéket. Az így kapott állapotok, reprezentálják az aktuális állapot lehetséges következő állapotait. Ezen állapotok közül, a legkevesebb klasztert tartalmazót eredményező klaszternövelési lépés kerül végrehajtásra. Több egyformán minimális számú klasztert tartalmazó lehetőség esetén a legelsőként megtalált kerül megvalósításra. A folyamat leállásának feltétele, hogy minden klaszter átmérője megegyezzen az előre meghatározott értékkel. Eredményként minden klaszter tartalmaz egy vagy több olyan szót, amelyet a többi klaszter nem, ám tartalmazhat olyan szavakat is, melyeket más klaszterek is tartalmaznak. A „legjobbat először” keresési stratégiával megvalósított klaszterezés folyamatát az 5.3. algoritmus mutatja. A „legjobbat először” keresési stratégiával megvalósított klaszterezés főbb paramétereit és eredményeit az *A.3* Függelék szemlélteti (futási idő, klaszterezendő szavak száma, első megoldás klasztereinek száma, ε=0.25 átmérőjű klasztereiben lévő szavak listája, ε=0.25 átmérőjű klasztereiben lévő legtávolabbi szavak távolsága, ε=0.25 átmérőjű klasztereiben lévő szavak átlagos száma).

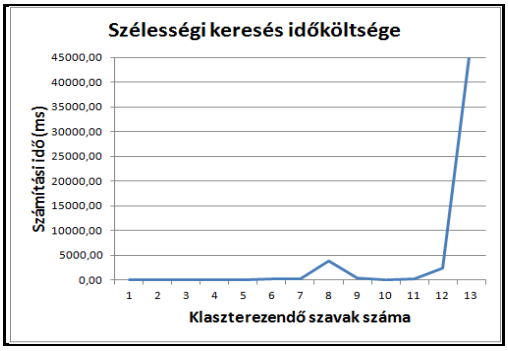
Lépés-szám	Cselekvés
1.	A *kiinduló állapot* előállítása. (A *dokumentum* összes különböző *szavának* kigyűjtése és az így kapott *szavak* mindegyikének külön *klaszter*be tétele.)
2.	Minden klaszter összevonási lehetőségének megvizsgálása az összes többi klaszterrel. (Két klaszter összevonható, ha legtávolabbi szavaik távolsága nem nagyobb a megengedett korongátmérőtől.)
3.	A 2. pontban összevonhatónak minősített klaszterpárok egy listában való összegyűjtése.
4.	Ha a lista nem tartalmaz legalább egy klaszterpárt, akkor menj a 7. lépésre!
5.	A listán szereplő minden klaszterpár esetén annak a megállapítása, hogy összevonásuk hány klaszter megszűnését eredményezné. (Minden klaszterpár összevonásával megszűnik a klaszterpár mindkét eleme, létrejön egy új klaszter, mely egyben tartalmazza a klaszterpár mindkét elemét valamint megszűnik minden olyan klaszter, melynek minden elemét tartalmazza legalább egy másik klaszter.)
6.	A listán szereplő klaszterpárok közül annak az egynek az összevonása, amely a legtöbb klaszter megszűnését eredményezi. Több azonosan érték esetén a legelőször megtalált ilyen klaszterpár összevonása. Lépj a 2. lépésre!
7.	Vége.

5.3. algoritmus: A „legjobbat először keresési” stratégiával megvalósított klaszterezés algoritmusa

A „legjobbat először keresés” egy heurisztikus kiértékelő függvénnyel becsli az n állapotból a cél elérésének költségét és abba az állapotba lép, amelyikre ez a költség a legkisebb. A keresés tár – és időköltsége $O(b^m)$, ahol m a keresési tér maximális mélysége. Jól megválasztott heurisztikus függvénnyel a komplexitás azonban jelentősen csökkenthető. A csökkenés mértéke az adott problémától és a heurisztikus csökkenés minőségétől függ [Stuart, 2005].

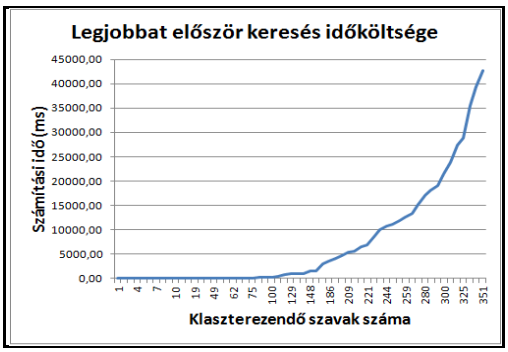
Az implementált algoritmusok eredményeinek összehasonlítása

A megvalósított klaszterező algoritmusokban a feladat egy lehetséges megoldását jelenti minden olyan állapot, melyben az előfeldolgozó modultól kapott összes szó eleme egy vagy több klaszternek, és az előzőleg beállított méretkorlát miatt egyetlen klaszter sem képes több szó tárolására. Ugyanannak a klaszterezési feladatnak több megoldása is lehetséges, melyek paramétereikben (klaszterek száma; a klaszteren belül a legtávolabbi szavak távolsága; a dokumentum azon mondatainak a száma melyben a klaszterekben lévő szavak együttesen szerepelnek stb.) különböznek egymástól. Bár a megvalósított algoritmusok alkalmasak a feladat összes lehetséges megoldásának feltárására, a disszertációban – terjedelmi okok miatt – csak az első megoldás paraméterei kerültek összehasonlításra.
Az $S_{i,j} = f_{i,j} \,/\, \max(f_i,\, f_j)$ képlet klaszterekre való kiterjesztésével a megoldás megtalálásának ideje – a távolságmátrix folyamatos újraszámolása miatt – négyzetesen függ a klaszterezendő szavak számától. Az így kapott érték hatványkitevőjébe kerül a keresési fa elágazási tényezője. Ez a szám a dokumentum mondatainak jellegétől (szóismétlés) függ és előre nehezen becsülhető. Az 5.2. ábrák a megvalósított algoritmusok futási időigényének összehasonlítását mutatják, a klaszterezendő szavak számának függvényében. Látható, hogy a megoldás megtalálásának időigénye csak részben függ a klaszterezendő szavak számától. Mivel a megoldások több klaszterösszevonás eredményeképpen jönnek létre, jól látható, hogy a legelső megoldás megtalálásához a szélességi keresésre épülő stratégiának szinte az egész fát be kell járnia. A diagram jól mutatja a távolságmátrix folyamatos újraszámolása által igényelt időtöbbletet is, a legjobbat először stratégiára épülő korongokkal való lefedés módszerével szemben [1].

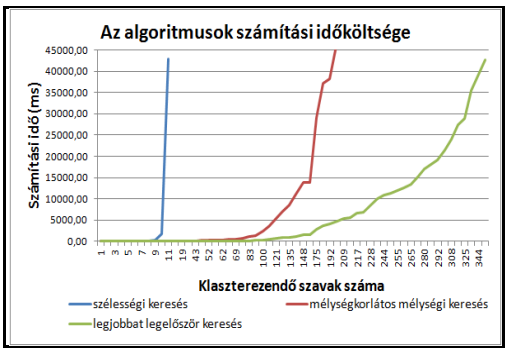


5.2.a

5.2.b



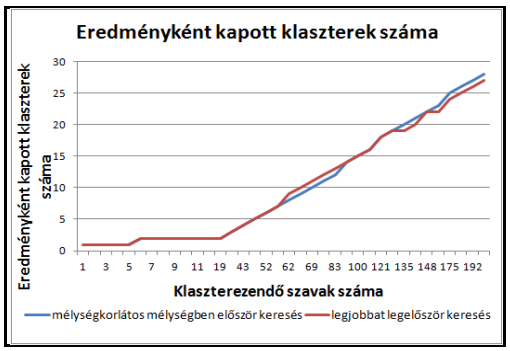
5.2.c



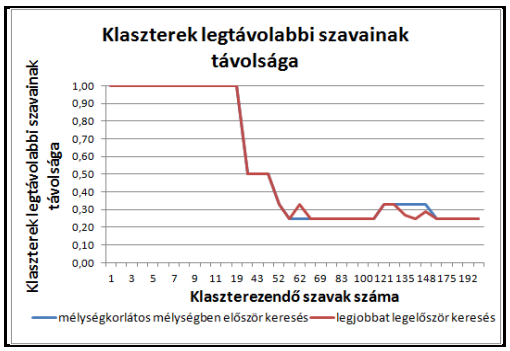
5.2.d

5.2. ábra: A megvalósított algoritmusok időigénye

Az 5.3. ábra a távolságmátrix képletének klaszterekre való kiterjesztésével, illetve a korongokkal való lefedéssel végzett klaszterezési módszerek eredményeit mutatja. A megoldásokban szereplő klaszterek száma az 5.3.a ábrán, a legtávolabbi szavakat tartalmazó klaszterben a legtávolabbi szavak távolsága pedig az 5.3.b ábrán látható. A legtávolabbi, még összevonandó klaszterek, illetve szavak távolsága mindkét módszer esetén a [0, 1] intervallumra normált távolságadatokat követően ε =0.25-re lett beállítva.



5.3.a



5.3.b

5.3. ábra: A klaszterezendő módszerek által kapott megoldások összehasonlítása

A bemutatott klaszterezési módszerek tesztelésével nyert információk alapján az AQG mintarendszer klaszterezési feladatát a szógyakoriságra épülő távolságképzési koncepcióval valósítottam meg. Ehhez az egységes átmérőjű korongokkal végzett klaszterezés feladatspecifikus továbbfejlesztését alkalmaztam a legjobbat legelőször keresési stratégia használatával.

5.2.2. Szavak objektív koordinátái alapján végzett vektortér alapú távolság-meghatározás

Az alkalmazott távolság-meghatározási koncepció értelmében a dokumentum szavai egy többdimenziós térben helyezkednek el, melynek dimenzióit a szavak objektív módon mérhető tulajdonságai alkotják. A koncepció alapján a dokumentum minden szava egy ponttal reprezentálható ebben a többdimenziós térben. A klaszterezési folyamatban a szavak elhelyezése a következő objektív koordináták alapján történt:

- szófaj (szótövező programmal meghatározva),
- ragozottság (szótövező programmal meghatározva),
- előfordulási száma a dokumentumban (saját szoftverrel meghatározva),
- előfordulási szám az adott szót tartalmazó mondatban (saját szoftverrel meghatározva),
- helye az adott szót tartalmazó mondatban (saját szoftverrel meghatározva),
- távolságátlaga az adott szót tartalmazó mondat többi szavától (saját szoftverrel meghatározva).

A vektortér alapú távolságra épülő klaszterezéssel az volt a célom, hogy az előző pontban ismertetett módszer mellett olyan klaszterezési lehetőséget is biztosítsak, amely nagyméretű dokumentumok esetén is elfogadható időn belül képes elvégezni a szavak klaszterezését. Erre a feladatra a szakirodalomban leginkább elfogadott módszert a BIRCH algoritmus szolgáltatta.

A BIRCH algoritmus célja [8], hogy a rá épülő klaszterezési módszerek számára gyors és, memóriatakarékos előklaszterezést végezzen. A folyamat során a klaszterezendő pontok a távolságuk alapján halmazokba kerülnek. Ezek a halmazok általában nem teljesítik a klaszterezéssel szemben elvárt speciális szempontokat, ezért szükség van a további klaszterezésükre, melyet már feladatspecifikus algoritmusok végeznek. A BIRCH algoritmus jelentősége abban áll, hogy a feladatspecifikus klaszterező algoritmusok számára minden halmaz pusztán egyetlen ponttal (ez általában a halmaz középpontja) kerül reprezentálásra. Ennek alkalmazhatóságához a BIRCH algoritmus biztosítja, hogy az általa előállított halmazokban a pontok távolsága kisebb egy előre megadott küszöbértéknél. A küszöbérték a BIRCH algoritmus leglényegesebb paramétere. Értékének növelésével növelhető a halmazban lévő pontok száma, mely a halmazok számának csökkenését eredményezi. A küszöbérték növelésével elérhető, hogy a halmazok száma elég kicsi legyen ahhoz, hogy a feladatspecifikus algoritmus már képes legyen elvégezni rajtuk a kívánt klaszterezést. A küszöbérték csökkentésével biztosítható, hogy a klaszterezendő halmazokban lévő pontok kellően közel legyenek a halmazt reprezentáló középponthoz. Ezáltal növelhető a klaszterezés jósága. Látható tehát, hogy a küszöbérték helyes meghatározása optimálási feladat.

A BIRCH algoritmus koncepciójának leglényegesebb strukturális eleme a CF (Clustering Feature) nevű adathármas. Ez teszi lehetővé, hogy a pontok csoportokba sorolása ne igényelje a már besorolt pontok elemzését. Az ebből adódó sebességnövekedésből származik a BIRCH algoritmus hatékonysága. A CF ereje abban áll, hogy az általa reprezentált halmaz elemeinek egyenkénti vizsgálata nélkül, pusztán egyetlen aritmetikai művelettel lehetővé teszi a halmaz feladat szempontjából lényeges tulajdonságainak kiszámítását. A CF struktúra elemeit az (5.9) összefüggés szemlélteti. A klaszterezendő pontok dimenzióját d jelöli [Zhang, 1996]:

$$CF = \{N, \overrightarrow{LS}, SS\} \tag{5.9}$$

Ahol: N az adott CF által reprezentált halmazban lévő pontok száma,
$\overrightarrow{LS}$ az adott CF által reprezentált halmazban lévő pontok helyvektorának összege,
SS az adott CF által reprezentált halmazban lévő pontok d-dimenziós négyzetes összege.

Ez alapján a halmaz középpontja az (5.10), két halmaz távolsága az (5.11), míg két halmaz összevonásával keletkező halmaz jellemzői az (5.12) összefüggés alapján határozhatóak meg.

$$\overrightarrow{XO} = \frac{\overrightarrow{LS}}{N} \tag{5.10}$$

Ahol: $\overrightarrow{XO}$ a halmaz középpontját reprezentáló d-dimenziós térbeli pont helyvektora.

$$D = \sqrt{\left(\overrightarrow{XO_1} - \overrightarrow{XO_2}\right)^2} \tag{5.11}$$

Ahol: D a két halmaz távolsága,
$\overrightarrow{XO_1}$ az egyik halmaz középpontját reprezentáló d-dimenziós térbeli pont helyvektora,
$\overrightarrow{XO_2}$ a másik halmaz középpontját reprezentáló d-dimenziós térbeli pont helyvektora.

$$CF_e = \left\{N_1 + N_2, \overrightarrow{LS_1} + \overrightarrow{LS_2}, SS_1 + SS_2\right\}. \tag{5.12}$$

A BIRCH algoritmus a klaszterezendő pontokat egy kiegyenlített B+ fában helyezi el. A fa két leglényegesebb strukturális építőeleme a levél-csomópont és a nemlevél-csomópont. Mindkét csomóponttípus közös jellemzője, hogy pontok halmazát reprezentálják, melyet a CF adathármas alkalmazásával tartanak nyilván. Mindkét csomóponttípus tartalmaz továbbá egy pointert annak a csomópontnak a nyilvántartására, mely az adott csomópontot magában foglalja. Ez az adott csomópont szülő-csomópontja. Minden levél-csomópont pontokat (illetve pontok halmazát) tárol. Ezek száma nincs korlátozva, viszont korlátozva van a levél-csomópontban lévő legtávolabbi pontok távolsága. Ennek a korlátnak a teljesülése a küszöbérték-feltétellel ellenőrizhető. Betartása a csomópont által reprezentált halmaz CF-ének ismeretében az (5.11) összefüggés alapján könnyen ellenőrizhető. Minden levél-csomópont tartalmaz továbbá egy pointert a fában őt követő, valamint egy pointert a fában őt megelőző levélcsomópont gyors eléréséhez. Ezek a pointerek a BIRCH algoritmust követő feladatspecifikus klaszterezés során nyújtanak gyors hozzáférést a BIRCH algoritmus által szolgáltatott halmazokhoz. Minden nemlevél-csomópont egymással megegyező típusú további csomópontokat (levél, illetve nemlevél-csomópontok) fog össze. Ezek az adott nemlevél-csomópont gyermek-csomópontjai. Ezáltal minden nemlevél-csomópont halmazok halmazát reprezentálja. Egy adott nemlevél-csomópont által reprezentált halmaz CF értéke az (5.12) összefüggés alapján számítható.
A BIRCH algoritmus első lépéseként létrehozásra kerül egy nemlevél-csomópont, mely a fa gyökér-csomópontját reprezentálja. A klaszterezendő pontok egymást követően kerülnek a fa gyökér-csomópontjához ahonnan a megfelelő alacsonyabb szintek felé továbbítódnak mindaddig, amíg egy levél-csomópontba nem kerülnek. Ha egy levél-csomópontba pont érkezik, akkor az (5.11) összefüggés alapján meghatározásra kerül, hogy a pont befogadását követően az adott levél-csomópont teljesíti-e a küszöbérték-feltételt. Ha igen, akkor az érkező pontot elfogadja az adott levélcsomópont. Ha a feltétel nem teljesül, akkor az adott levél-csomópontban lévő pontok két halmazra bomlanak. Ehhez meghatározásra kerül a halmaz két legtávolabbi pontja, melyek a két halmaz magpontjait fogják reprezentálni. Létrehozásra kerül egy új levél-csomópont, melyben elhelyezésre kerül az egyik magpont. A másik magpont az eredeti halmazban marad. Az eredeti halmaz összes többi pontja abba a levél-csomópontba kerül, amelyik magpontjához közelebb van. A folyamat eredményeként az új levél-csomópont továbbításra kerül az aktuális levél-csomópont szülőjéhez, aki elhelyezi azt a gyermek-csomópontjai között. Ha egy nemlevél-csomóponthoz pont érkezik és az adott nemlevél-csomóponthoz nem tartozik gyermek-csomópont, akkor létrehozásra kerül egy levél-csomópont az adott nemlevél-csomópont gyermek-csomópontjaként és elhelyezésre kerül benne az érkező pont. Ha azonban az adott nemlevél-csomópont tartalmaz

gyermek-csomópontot, akkor az érkező pont ahhoz a gyermek-csomóponthoz kerül továbbításra, amelyikhez a CF érték alapján a legközelebb van. Ha egy nemlevél-csomóponthoz gyermek-csomópontként elhelyezendő csomópont érkezik, akkor az adott nemlevél-csomópont elhelyezi azt a gyermek-csomópontjai között. Ha az elhelyezést követően az adott nemlevél-csomópont gyermek-csomópontjainak száma meghaladja a BIRCH algoritmus paramétereként megadott maximális elágazási tényezőt, akkor az adott nemlevél-csomópont által reprezentált halmaz két részre szakad. Ehhez meghatározásra kerül az adott nemlevél-csomópont két egymástól legtávolabb lévő gyermek-csomópontja. Ezek lesznek az új halmazok magjai.
Az egyik mag elhelyezésre kerül egy új nemlevél-csomópontban, a másik marad eredeti helyén. Ezt követően az adott nemlevél-csomópont összes többi gyermek-csomópontja abba a nemlevél-csomópontba kerül, amelynek magjához közelebb van. A folyamat végén létrejön egy új nemlevél-csomópont, amely továbbításra kerül az adott nemlevél-csomópont szülőjéhez, aki elhelyezi azt a gyermek-csomópontjai között. Abban az esetben, ha a szétszakadó csomópont a fa gyökér-csomópontja, akkor létrehozásra kerül egy új nemlevél-csomópont. Ebben elhelyezésre kerül a fa gyökér-csomópontja valamint a halmaz szétszakadásával létrejött nemlevél-csomópont és a továbbiakban ez az új nemlevél-csomópont reprezentálja a fa gyökér-csomópontját.
Egy pontnak a fában való elhelyezéséhez a fa gyökér-csomópontjától a megfelelő levél-csomópontig $1 + log_B M$ számú csomóponton kell áthaladni, ahol B a nemlevél-csomópontok maximális gyermek-csomópontjainak számát, M pedig a fa maximális méretét jelöli. Ehhez minden nemlevél-csomópontnál maximálisan B számú gyermek-csomópont megvizsgálására van szükség a helyes út meghatározásához. A vizsgálatok időköltsége a tér dimenziójának d függvénye. Ezekből következik, hogy egy pont elhelyezésének időköltsége [Zhang, 1996]:

$$O\big(d * N * B * (1 + log_B M)\big). \tag{5.13}$$

Abban az esetben, amikor a halmazok szétbontására van szükség maximálisan $ES - 1$ elem/ csomópont kerül áthelyezésre, ahol ES jelenti a szétbontandó halmaz elemeinek/gyermek-csomópontjainak számát. Ezáltal egy halmaz szétbontásának maximális időköltsége [Zhang, 1996]:

$$O\big(d * (ES - 1) * B * (1 + log_B M)\big). \tag{5.14}$$

A fa átszerkesztéseinek a száma a küszöbérték függvénye. Minél kisebb a küszöbérték annál gyakrabban van szükség a halmazok szétbontására. A fa ismételt megszerkesztéseinek a száma a $log_2 \frac{N}{N_T}$ összefüggéssel közelíthető, ahol N_T az adott küszöbérték esetén az egy levél-csomópontban lévő pontok átlagos számát jelöli. Ezek alapján a BIRCH algoritmus számítási idejének költségigénye az

$$O\left(d * N * B * (1 + log_B M) + log_2 \frac{N}{N_T} * d * (ES - 1) * B * (1 + log_B M)\right) \tag{5.15}$$

összefüggés alapján számítható [Zhang, 1996].

A továbbfejlesztett BIRCH algoritmus kvázioptimális paramétereinek meghatározása

A BIRCH algoritmus alapú klaszterezéssel végzett tesztek eredményei rámutattak arra, hogy a folyamat időigénye jelentősen függ az algoritmus küszöbérték, valamint maximális elágazási tényező paramétereitől [8]. A paraméterek megváltoztatásával ugyanazon bemeneti adathalmaz

esetén lényegesen eltérő számítási költségigények tapasztalhatók. Ezáltal felmerült az igény annak a küszöbértéknek, illetve maximális elágazási tényezőnek a megtalálására, melyek alkalmazása esetén a klaszterezési folyamat időszükséglete minimális értéken tartható. A BIRCH algoritmusra épülő klaszterező szoftverrel végzett tesztek kimutatták, hogy a feladat költségigénye legnagyobb mértékben az algoritmus küszöbérték-paraméterétől függ. A költséget kisebb mértékben ugyan, de szintén jelentősen befolyásoló tényező a CF fa maximális elágazási tényezője.
Az elvégzett tesztekből kiderült, hogy a küszöbérték paraméter optimális értékétől negatív irányba való eltéréssel exponenciálisan megnő a BIRCH algoritmus által eredményezett halmazok száma. Ennek oka, hogy a kisebb küszöbérték minden halmaz esetén a klaszterezendő pontok nagyobb közelségét követeli meg. Ez a követelmény pedig csak a halmazok számának növelésével teljesíthető. Az exponenciálisan növekedő halmazszám szintén exponenciális költségnövekedést eredményez a BIRCH algoritmus előklaszterezésére épülő k-közép algoritmusnál is. A küszöbérték paraméter optimális értékétől pozitív irányba való eltérésével megnő a halmazokba kerülő pontok száma melyek a halmazokat reprezentáló levél-csomópontok szétbontásakor igényelnek folyamatosan növekvő többletköltséget.
Ezzel egyidejűleg, az algoritmus maximális elágazási tényező paraméterének optimális értékétől negatív irányba való eltérésével, exponenciálisan nő a fa mélysége. A levél-csomópontok számának állandó értéke mellett ez, a nemlevél-csomópontok számának lényeges növekedését eredményezi. A megnövekedett számú csomópontok adminisztrálása (létrehozás, szétbontás, memóriafoglalás stb.) növekvő költséget jelent, melynek mértéke a paraméter optimális értékétől való távolság függvényében nő. A maximális elágazási tényezőnek az optimális értékétől pozitív irányba való eltérés a fa minden nemlevél-csomópontja esetén növekvő számú gyermek-csomópont megjelenését eredményezi. Nagyobb számú gyermek-csomópont esetén minden új pont elhelyezéséhez minden nemlevél-csomópontnál arányosan nagyobb számú alternatíva közül kell kiválasztani azt a gyermek-csomópontot, amely CF értéke alapján a legközelebb van az elhelyezendő ponthoz. Ez a szintenként arányosan növekvő számítási igény a teljes fára nézve a mélységek hatványával arányos költséget igényel minden új pont elhelyezésénél.
A tesztek eredményeinek elemzése alapján nyilvánvalóvá vált, hogy a klaszterezési folyamat hatékonyságának maximálása érdekében az említett két paraméter optimális értékének ismerete kulcsfontosságú. A paraméterek optimális értékei azonban feladatspecifikusak. Mivel nagymértékben függnek a klaszterezendő pontok számától és elhelyezkedésétől, ezért nem határozható meg olyan konkrét érték, mely minden feladat esetén minimumközeli költséget eredményez. Ezáltal a paraméterek optimális értékét, minden konkrét feladat esetén egyedileg kell meghatározni a minimális költségű klaszterezés eléréséhez. Az optimális értékeik meghatározása ugyanannak a klaszterezési feladatnak az egymást követő többszöri elvégzését igényli a paraméterek különböző értékei esetén. Ezáltal a paraméterek optimális értékeinek meghatározáshoz szükséges költség többszöröse a klaszterezés egyszeri elvégzésével járó költségnek. Az optimális értékek feltárása, tehát csak azokban az esetekben térül meg, amikor a klaszterezendő objektumok főbb tulajdonságai szempontjából (objektumok száma, fókuszpontok száma, fókuszpontok távolsága, objektumok fókuszpontok körüli szórása) azonos minták klaszterezésére van szüksége.
A BIRCH algoritmus említett két paraméteréhez tartozó optimális értékek meghatározása a lokális kereső algoritmusok alkalmazásával lehetséges. A lokális kereső algoritmusok legismertebb képviselői: „hegymászó keresés” (hill-climbing search) [Bart Selman, 2006], „szimulált lehűtés” (simulated annealing) [Kirkpatrick, 1983], „lokális nyaláb keresés” (local beam search), „genetikus algoritmus” (genetic algorithm) [Álmos, 2003]. A feladat megvalósíthatóságának bizonyítása, valamint az optimálás által szolgáltatott eredmények első közelítéses elemezhetősége céljából a legegyszerűbbnek számító „hegymászó keresési algoritmus” alkalmazását választottam.

Amennyiben a paraméterek pontosabb értékeinek a meghatározása a cél, ez a további algoritmusokkal biztosítható. A megvalósításra került algoritmus meghatározza:

- a BIRCH alapú klaszterezés küszöbérték, valamint maximális elágazási tényező paramétereinek a költségigény minimálása szempontjából optimális (lokális optimum) értékeit,
- azoknak az azonos tulajdonsággal rendelkező klaszterezendő mintáknak a minimális számát, melyek esetén a paraméterek optimális értékének meghatározási költsége már megtérül,
- a paraméterekre kapott lokális optimum értékek távolságát a globális optimumtól.

Az optimálás során a költség (jele: C) a küszöbérték (jele: T), valamint a maximális elágazási tényező (jele: B) függvényében kerül vizsgálatra $C = f(T, B)$. Ezáltal az optimálási tér egy háromdimenziós koordináta-rendszerben ábrázolható, ahol a költsége a T, B tengelyek által definiált síkkal párhuzamos tengely mentén kerül feltüntetésre. Az optimálás megkezdése előtt meg kell határozni a függvény értelmezési tartományát. Ez a vizsgálatra kerülő T, illetve B paraméterek minimális és maximális értékeinek megadásával, valamint az így definiált intervallumban vizsgálni kívánt értékek számának megadásával történik. Ezek alapján a szoftver meghatározza az egyes paraméterek mentén vizsgálható értékeket a közöttük lévő távolság kiszámítása révén:

$$\varepsilon = \left(\frac{maximális\ érték - minimális\ érték}{vizsgálni\ kívánt\ értékek\ száma - 1}\right). \tag{5.16}$$

A BIRCH algoritmus paramétereinek optimálására alkalmazott „hegymászó keresési algoritmus” formális leírását az 5.4. algoritmus mutatja.

Lépés-szám	Cselekvés
1.	Legyen *aktuális-pont* egy véletlenszerűen megválasztott (T, B) koordinátákkal adott pont.
2.	Az *aktuális-pont*hoz tartozó költség meghatározása a klaszterezési folyamatnak az *aktuális-pont* által reprezentált paraméterekkel történő elvégzésével.
3.	Az *aktuális-pont* szomszédainak meghatározása az *aktuális-pont* koordinátáinak ε-al való növelésével, illetve csökkentésével.
4.	Az *aktuális-pont* szomszédaihoz tartozó költségek meghatározása a klaszterezési folyamatnak az *aktuális-pont* minden szomszédjára az adott szomszéd által reprezentált paraméterekkel történő elvégzésével.
5.	Ha az *aktuális-pont* legkisebb költségű szomszédjának költsége nem kisebb az *aktuális-pont* költségénél, akkor lépj a 7. lépésre!
6.	Az *aktuális-pont* legyen a legkisebb költségű szomszédja által reprezentált pont. Lépj a 3. lépésre!
7.	A T, illetve B paraméterek optimális értékei az *aktuális-pont* koordinátái által adottak.

5.4. algoritmus: „Hegymászó keresési algoritmus” formális leírása

Az algoritmus alapján látható, hogy az optimális paraméterértékek meghatározásához ugyanannak a klaszterezési feladatnak az ismételt elvégzésére van szükség különböző paraméterértékek mellett. Minimális esetben (amikor az *aktuális-pont* sarokpont, melynek ezáltal csak 3 szomszédja van, valamint az optimum éppen az *aktuális-pont* vagy valamelyik szomszédja) ez a klaszterezési feladat négyszeri elvégzését igényli. Ebből látható, hogy a klaszterezés költségének megtérülése csak relatív nagyszámú azonos tulajdonságokkal rendelkező minták klaszterezése esetén várható. Az optimálás nélküli klaszterezés költségét S-el, az optimálással kapott klaszterezés költségét O_L-el az optimum megtalálási költségét, pedig M-el jelölve az i. klaszterezést követően az optimálás nélküli klaszterezés költsége: $i * S$, valamint az optimálással kapott T, B paraméterértékekkel

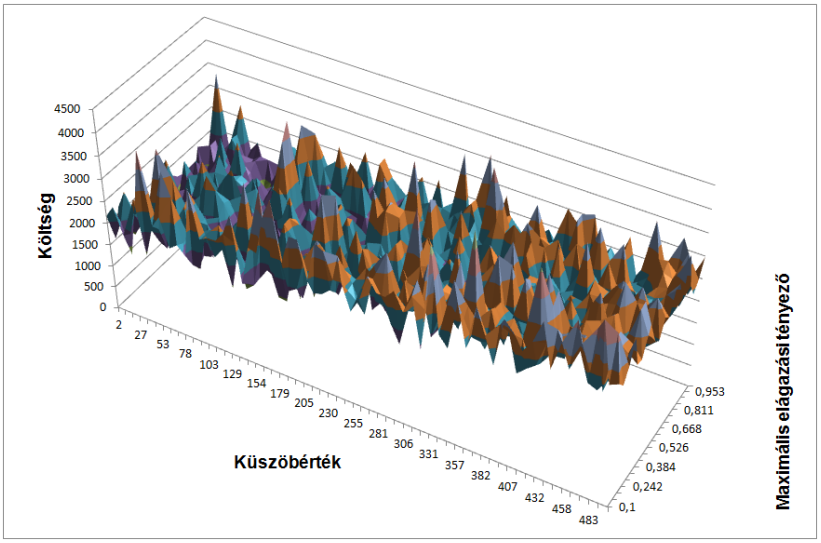
végzett optimálás költsége: $i * O_L + M$. Az összefüggések alapján implementált algoritmus alkalmas annak a klaszterezendő mintaszámnak a meghatározására (jele: i_T), amely esetén az optimum megtalálásának költsége már megtérül $i_T > \frac{M}{S - O_L}$. A megtérülés mértékét P-vel jelölve:

$$P = j * S - (j * O_L + M) \text{ ahol, } j \geq i_T. \tag{5.17}$$

Mivel az optimálás nélküli klaszterezés költsége jelentősen függ a T, B paraméterektől, ezért az eltérő megválasztásuk esetén kapott i_T értékek jelentősen különbözhetnek egymástól. Amennyiben az optimum megtalálásán túl a feladat részét képezi az i_T értékének pontos meghatározása is az elkészített szoftver az optimálás nélküli klaszterezés költségét az átlagköltséggel helyettesíti. Az átlagköltség meghatározásához az értelmezési tartomány minden pontjára meghatározásra kerül a klaszterezés költsége. Mivel az átlagköltség meghatározását egy teljes leszámlálásra épülő algoritmus végzi, ezért a folyamat elvégzése csak az i_T érték pontosítása esetén érdemes. Ehhez, a teljes leszámlálásával végzett állapottér feltárással határozható meg az implementált „hegymászó keresés algoritmussal" kapott lokális optimum távolsága (jele: d) a keresett globális optimumtól (jele: O_G)

$$d = O_L - O_G. \tag{5.18}$$

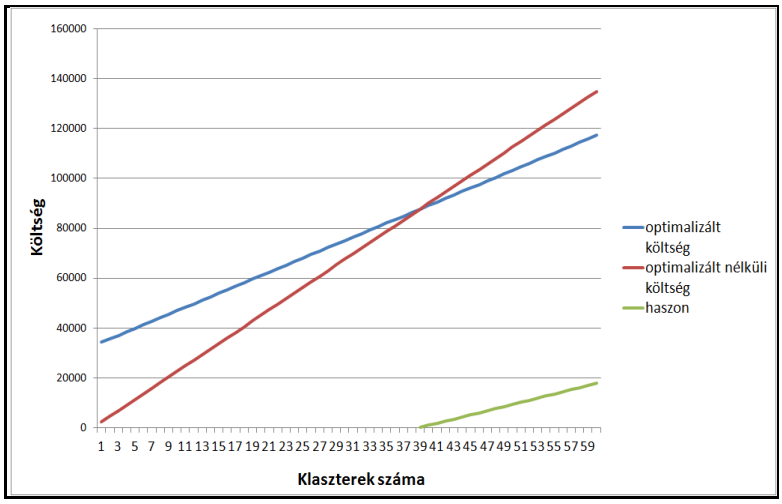
A BIRCH algoritmus optimális paraméterértékeinek meghatározását végző szoftvermodul eredményeinek bemutatásához a Magyar Elektronikus Könyvtár rendszeréből ingyenesen letölthető Kovács Gábor, Informatikai ismeretek című könyvének 10.000 szót tartalmazó egységét választottam ki klaszterezésre. A klaszterezés költségét leíró függvény értelmezési tartományának küszöbérték tengelyén a [0.1, 1] intervallumból egymástól egyenlő távolságra lévő 20 darab értéket választottam ki. Az értelmezési tartomány maximális elágazási tényező tengelyén pedig a [2, 500] intervallumból egymástól egyenlő távolságra lévő 60 darab értéket választottam ki. A költség értéke csak az értelmezési tartományban definiált helyeken kerülhet meghatározásra. Az értelmezési tartomány minden pontján elvégzett költségmeghatározást követően kapott optimálási tér az 5.4. ábrán látható.



5.4. ábra: Az értelmezési tartományán minden pontján kiszámított költség értéke

A „hegymászó keresési algoritmussal" meghatározásra került, hogy a klaszterezés minimális költsége a 0.195 távolságegységnyi értékű küszöbérték, valamint a 85 darab értékű elágazási tényező esetén érhető el. Ezen paraméterek által eredményezett (lokális) optimális költség 1409 ms. Az optimális megoldás megtalálásának időszükséglete 32786 ms.

Az optimálási tér teljes feltárásával pontosan meghatározásra került, hogy a globális minimumhoz tartozó költség 1075 ms, az átlagos klaszterezési költség pedig 2250.53 ms.
Ezek alapján a szoftver meghatározta, hogy az optimálással kapott paraméterek alkalmazása esetén az optimum megtalálásának költsége a 38. klaszterezést követően térül meg: $\frac{32786}{(2250.53-1409)} = 38.96$. Az eredmény ellenőrizhető, mivel a 38. klaszterezést követően az optimálás nélküli átlagos költséggel számított összköltség $38 * 2250.53 = 85520.20$ ms, míg az optimálással kapott paraméterértékek alkalmazása esetén $38 * 1409 + 32786 = 86328$. A 39. klaszterezést követően viszont az optimálás nélküli összköltség 87770.74 ms, míg az optimálás esetén 87737 ms. Az optimálással, illetve az optimálás nélkül kapott költségeknek a klaszterezés számától való függését az 5.5. ábra szemlélteti.



5.5. ábra: Az optimálással, illetve az optimálás nélkül kapott költségek

A teljes optimálási tér feltárását követően a szoftver meghatározta a „hegymászó keresés algoritmussal" kapott lokális optimum távolságát a paraméterek globális optimumától:

$$1409\ ms - \ 1075\ ms = 334\ ms.$$

Ezzel igazoltam, hogy a BIRCH algoritmus két leglényegesebb paraméteréhez tartozó érték megfelelő beállításával jelentősen csökkenthető az algoritmus költségigénye. Az alkalmazott „hegymászó keresési algoritmus" által szolgáltatott eredményekkel igazoltam a feladat megoldhatóságát. Meghatároztam továbbá az optimálási folyamat költségigényének megtérüléséhez szükséges klaszterezések számát. Egy egyszerű példán végzett teljes leszámlálásra épülő algoritmus adatai megmutatták, hogy az optimálás célfüggvényének értékei között sok lokális szélsőérték található (5.5. ábra). Ezek a lokális szélsőértékek jelentősen rontják a „hegymászó keresés algoritmus" hatékonyságát a költségfüggvény globális minimumának megtalálásában. Hasonló tulajdonságokkal rendelkező kellően nagyszámú mintahalmaz klaszterezése esetén az alkalmazott optimumtól való kis eltérések is jelentős költségkülönbséget eredményeznek a feladat egészére vonatkozóan. Ezekben az esetekben megnő a globális optimumhoz lehető legközelebb eső megoldás megtalálásának fontossága. Ennek érdekében új koncepcióra épülő algoritmust fejlesztettem ki, amely ötvözi a „hegymászó keresés" és a „genetikus algoritmus" előnyös tulajdonságait.
Az új módszer a hegymászó keresési algoritmussal megegyezően diszkretizálja a valós értékkel jellemezhető paramétereket. Ezáltal a költségfüggvény kiértékelésére csak előre meghatározott diszkrét pontokban van lehetőség. Ezek a pontok képezik a költségfüggvény értelmezési tartományát. Az értelmezési tartomány határai, valamint a pontok közötti távolság az optimálást

megelőzően állíthatóak be. Mivel a feladat költségfüggvényének értelmezési tartománya két dimenziós, ezért a diszkretizálást követően a vizsgálható pontok egy rács rácspontjaiban helyezkednek el. Az új fejlesztésű optimum-kereső algoritmus a genetikus algoritmushoz [1] hasonlóan több optimum-esélyes jelölttel (egyeddel) dolgozik. Minden egyedet egy pont reprezentál a költségfüggvény diszkretizált értelmezési tartományában. Az egyidejűleg létező egyedek csoportja alkotja az algoritmus által kezelt populációt. A kiinduló populáció minden egyede véletlenszerűen meghatározott koordinátákkal kerül létrehozásra. Az egyedszám paraméteresen definiálható, mely alapértelmezett értékeként 5 egyedet alkalmaztam. Ezt követően a populáció szaporítása történik a továbbiakban ismertetésre kerülő keresztezési és mutációs módszerekkel. A szaporítás eredményeként kapott populáció egyedszáma szintén paraméteresen állítható. Az alapértelmezett értékként itt 10 egyedet alkalmaztam. A szaporítást követően szelektálással meghatároztam és töröltem a nagyobb költségű egyedeket a kiinduló populációszám ismételt eléréséig.

Minden létrehozott egyed egy-egy „hegymászó keresési" folyamat kiinduló pontját képezi. A keresések elvégzését követően a populáció minden egyedének koordinátái módosulnak a keresés által szolgáltatott lokális minimum koordinátáira. Ezáltal a populáció minden egyede egy lokális minimumba kerül. Az így kapott populáció egyedeinek költsége alapján meghatározható a populációra jellemző átlagos költség. Ez a költség képezi alapját az optimum-kereső algoritmus leállítási feltételének. Ha az n. populáció átlagos költsége már meghaladja az $n-1$. populáció átlagos költségét, akkor az algoritmus nem képez új populációt és az $n-1$. populáció legjobb egyedét tekinti az optimálás eredményeként szolgáló legjobb megoldásnak. Amennyiben az n. populáció az első populáció, vagy az átlagos költsége kisebb vagy egyenlő az $n-1$. populáció átlagos költségétől, akkor az n. populáció egyedei alapján új $n+1$. populáció kerül létrehozásra. Az új populációba változtatás nélkül átkerül az n. populációnak az optimálás megkezdése előtt beállított számú legjobb egyede. Az $n+1$. populáció egyedszámának a beállított értékre való felnövelése, az áthelyezett egyedek szaporításával történik.

A generáláshoz véletlenszerűen választásra kerül egy egyed a csökkentett számú populációból, melyen elvégzésre kerül a következő négy lehetőség egyike:

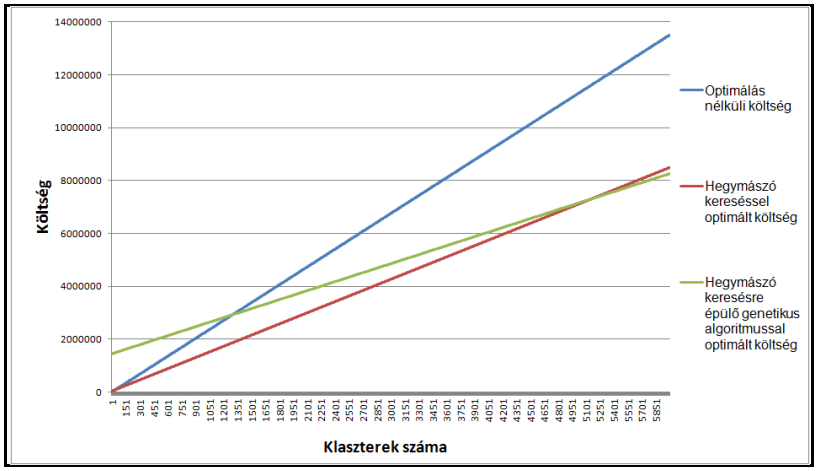
- első típusú keresztezés a csökkentett egyedszámú populáció legjobb egyedével (A kiválasztott egyed koordinátáit (t_i, b_i)-vel a legjobb egyed koordinátáit (t_j, b_j)-vel jelölve a létrehozásra kerülő új egyed koordinátái (t_i, b_j). Amennyiben a kiválasztott egyed a legjobb egyed ez a szaporítási lehetőség nem alkalmazható.)
- második típusú keresztezés a csökkentett egyedszámú populáció legjobb egyedével (A kiválasztott egyed koordinátáit (t_i, b_i)-vel a legjobb egyed koordinátáit (t_j, b_j)-vel jelölve a létrehozásra kerülő új egyed koordinátái (t_j, b_i). Amennyiben a kiválasztott egyed a legjobb egyed ez a szaporítási lehetőség nem alkalmazható.)
- eltolás típusú mutáció végzése a csökkentett egyedszámú populáció kiválasztott egyedével (A kiválasztott egyed koordinátáit (t_i, b_i)-vel jelölve az új egyed koordinátái $(t_i + \varepsilon_1, b_i + \varepsilon_2)$. Ahol ε_1 a [-1, 0, +1] rácsponttávolság egyikét jelenti a küszöbérték paraméter diszkretizált tartományában, az ε_2 pedig a [-1, 0, +1] rácsponttávolság egyikét jelent a maximális elágazási tényező paraméter diszkretizált tartományában. Az $\varepsilon_1 = \varepsilon_2 = 0$ érték valamint a mutációt követően a költségfüggvény értelmezési tartományából való kilépés esetén új ε_1, illetve ε_2 értékek kerülnek meghatározásra.)
- a kiválasztott egyed koordinátáitól független új egyed elhelyezése a költségfüggvény diszkretizált értelmezési tartományában véletlenszerűen.

Az implementálásra került „hegymászó keresésre” épülő „genetikus algoritmus” formális leírását az 5.5. algoritmus tartalmazza.

Lépés-szám	Cselekvés
1.	Hozz létre a populáció megadott egyedszámának megfelelő számú egyedet és helyezd el ezeket véletlenszerűen a költségfüggvény értelmezési tartományát adó rácspontokban!
2.	A populáció minden egyedére végezd el a „hegymászó keresést”. Az eredményként kapott pontok lesznek a populáció új egyedei.
3.	Ha nem ez az első populáció és a populáció egyedeinek átlagos költsége nagyobb az előző populáció egyedeinek átlagos költségétől, akkor lépj a 7. lépésre!
4.	A populációból a megadott számú legjobb egyedén kívül távolíts el minden egyedet.
5.	Növeld vissza a populáció egyedszámát az eredeti értékre a megmaradt egyedek szaporításával (keresztezések, mutációk)!
6.	Lépj a 2. lépésre!
7.	A T, illetve B paraméterek optimális értékei az előző populáció legalacsonyabb költséget reprezentáló egyedének koordinátái által adottak.

5.5. algoritmus: A hegymászó keresésre épülő genetikus algoritmus formális leírása

A „hegymászó keresésre” épülő genetikus algoritmussal meghatározásra került, hogy a klaszterezés minimális költsége a 0.858 távolságegységnyi értékű küszöbérték, valamint a 36 darab értékű maximális elágazási tényező esetén érhető el. Ezen paraméterek által eredményezett optimális költség 1135 ms. Az optimális megoldás megtalálásának időszükséglete 1.446.685 ms. Ezek alapján a szoftver az előző részben ismertetett módszerrel meghatározta, hogy az optimálással kapott paraméterek alkalmazása esetén az optimum megtalálásának költsége az 1298. klaszterezést követően térül meg az optimálás nélküli állapothoz képest. Ez jóval nagyobb, mint a normál hegymászó keresés alkalmazásával elért 38 klaszterezés. Ennek oka, hogy a most ismertetett módszer időigénye jóval nagyobb a hegymászó kereséstől, ezért az emiatt jelentkező kezdeti hátrány ledolgozásához jóval több klaszterezésre van szükség. Cserébe azonban az új módszer által eredményezett optimum a hegymászó kereséstől kapott optimumhoz képest $1409 - 1135 = 274$ ms.-al közelebb van a teljes leszámlálással feltárt globális optimumhoz. Meghatározásra került, hogy az új módszer alkalmazásával az 5161. klaszterezést követően alacsonyabb költség érhető el, mint a kizárólag hegymászó keresési algoritmus által szolgáltatott optimum használatával.



5.6. ábra: A vizsgált klaszterezési lehetőségek költségigényének összehasonlítása

A BIRCH algoritmus alapú klaszterezés gyakorlati alkalmazása

A BIRCH algoritmussal megvalósított klaszterezés gyakorlati alkalmazásához különböző tudományterülethez tartozó elektronikus könyvek szolgáltatták a bemeneti adatokat. A tesztelést a következő dokumentumok alapján végeztem el:

1. természettudomány – informatika (Kovács Gábor: Informatikai ismeretek),
2. társadalomtudomány – filozófia (Karikó Sándor: Az alkalmazott filozófia esélyei),
3. szépirodalom – népköltészet (Merényi László: Dunamelléki eredeti népmesék).

A BIRCH algoritmus valóságos szövegeken való tesztelésének célja, hogy kimutatásra kerüljön a kapott klaszthereknek az adott témakörhöz való illeszkedési tényezője. A tesztek egységesítése érdekében minden vizsgált könyvből 10.000 szót tartalmazó fejezetrész lett kiválasztva a klaszterezéshez.
Először elvégzésre került a BIRCH algoritmus alapú előklaszterezés, majd az ezáltal kapott ponthalmazok finomklaszterezésre kerültek a k-közép algoritmussal. Az eredményként kapott klaszthereknek a vizsgált témakörhöz való tartozásának elbírálásához először meghatározásra került a témakör középpontját jelölő virtuális pont. Ez a pont a klaszterek középpontjainak átlagolásával került meghatározásra. Ezt követően minden klaszter esetén kiszámításra került az adott klaszter középpontjának a virtuális ponttól való távolsága. Az 5.2. táblázatban a távolságadatokból számított átlagos, illetve maximális távolságértékek szerepelnek.

A könyv sorszáma	**Klaszthereknek a középponttól való átlagos távolsága** (távolságegység)	**Klaszthereknek a középponttól való maximális távolsága** (távolságegység)
1.	58.63	793.56
2.	148.00	706.75
3.	125.88	842.21

5.2. táblázat: Átlagos, illetve maximális távolságértékek

A táblázatban szereplő számértékek a pontok ismertetett hét dimenziós koordinátáiból számított távolságok alapján kerültek meghatározásra, ezért mértékegységük távolságegység.
A vizsgálat eredményeiből az derült ki, hogy az informatika témaköréhez tartozó szöveges dokumentum szavai jóval szorosabban illeszkednek a témakörükhöz, mint bármely más vizsgált témakör szavai. Ez nem meglepő, hiszen az egyértelműség érdekében az informatika világában kell leginkább kerülni a változatos szóhasználatot.

5.3. A klaszterezés eredményeinek összefoglalása

A modulnak fontos feladata a nagyszámú szóból álló dokumentumok vizsgálandó egységeinek lecsökkentése a dokumentum bizonyos szempontból egymáshoz közeli szavainak klaszterekbe rendezésével. Ezáltal a legkisebb vizsgálható egységet, a szavak helyett, a klaszterek reprezentálják, amely lényeges erőforrás-megtakarítást jelent. A klaszterezést elvégeztem a szavak közös előfordulási gyakoriságára épülő, valamint a szavak objektív koordinátái által definiált többdimenziós térben vizsgált távolságra épülő stratégiák alapján is. Részletesen bemutattam a különböző keresési stratégiákkal megvalósított algoritmusokat és azok folyamatábráit. Elemeztem az algoritmusok futási időköltségét.
Kidolgoztam a BIRCH algoritmus költségigényeit: egy pont elhelyezésének költségigényét, egy halmaz szétbontásának maximális költségigényét és a BIRCH algoritmus számítási idejének költségét. Meghatároztam a BIRCH algoritmus paramétereinek (küszöbérték, maximális elágazási tényező) optimalizálás nélkül és optimalizálással kapott költségeit.

6. fejezet

Dokumentumok szavainak osztályozása

A dokumentumok szavainak klaszterezése révén előálltak mindazok az objektív koordináták, melyek leírják a szavak elhelyezkedését a definiált hétdimenziós objektív térben. Munkám további részét egy olyan szabályrendszer feltárása képezte, mely képes az objektív térben adott szavakat (mint pontokat) az objektív koordináták alapján automatikusan elhelyezni az igényeink szerint definiált szubjektív térben. A szubjektív tér dimenzióit úgy határoztam meg, hogy minél jobban kiemeljék a dokumentum szavainak emberi értelmezés szerinti tulajdonságait (relevancia, specialitás, értelmesség, nehézség), valamint megalapozott támogatást nyújtsanak annak eldöntésére, hogy a dokumentum mondataiból mely szavak kerüljenek kiemelésre kérdésgenerálás céljából. Az osztályozási feladat magját előrecsatolt háromrétegű neurális hálóval valósítottam meg – amely, egy osztályozási feladatot jelent – melyet mind strukturális, mind pedig működési szempontok alapján illesztettem a konkrét feladat speciális tulajdonságaihoz. A hálózat betanítását felügyelt tanítási módszerrel végeztem el. Ennek értelmében létrehoztam egy tanítómintát, mely a benne szereplő szavak objektív koordinátáihoz tartalmazza a szakértő által javasolt szubjektív koordinátákat. A tanítást követően a neurális hálózat képessé vált a vizsgált dokumentumok szavait automatikusan elhelyezni a szubjektív térben.

6.1. Az osztályozás elvi és módszertani áttekintése

A számítógépi intelligencia problémaköreinek egyik fő eleme az osztályozás feladata, melynek során az objektumokat előre definiált osztályba soroljuk. Az osztályozás feladatánál feltesszük, hogy az objektumok egy magadott tulajdonsághalmazzal jellemezhetőek és az objektum osztálykódja az objektum tulajdonságaitól függ. Az osztályozás célja [3] az ismeretlen kapcsolatleíró $f: O \rightarrow 2^C$ függvény mellett egy olyan

$$g : O \rightarrow 2^C$$

osztályozó függvény meghatározása, melyre

$$E(f, g, S) \rightarrow min$$

teljesül, ahol az $E(.,.)$ függvény a hibafüggvényt, S a tanítóhalmazt és C az osztálykódok halmazát jelöli. Az így kapott g függvény felhasználható az f függvény közelítésére a teljes O halmazon. A hibát egy adott objektumnál valós értékű osztályozó függvények mellett rendszerint az eltérésnégyzettel mérjük [3]:

$$E(f, g) = \sum_{o \in S} (f(o) - g(o))^2.$$

Az osztályozó függvény statisztikai alapú meghatározása felügyelt tanítással történik. A felügyelt tanítás során a tanítóhalmaz

$$T = \{(o, f(o)) \mid o \in S\}$$

alakú, tehát a mintában minden objektumhoz ismertek az odatartozó kategóriaértékek. Az ismeretlen g osztályozó függvény meghatározására alapvetően kétféle módszer terjedt el:

- particionáló módszer, melyben az objektumok vektorterét különböző kategóriákhoz rendeljük;
- osztályvalószínűségi függvény alapú, amikor egy $P(c|o)$ valószínűségi értéket határozzunk meg minden érintett kategóriára.

A szövegosztályozás bármely területen való sikeres alkalmazásának előfeltétele a megfelelő mennyiségű és minőségű tanítókörnyezet megteremtése. A tanítóobjektumok a hozzájuk rendelt osztályokat jól jellemző mintaadatok. Az osztályozó tanulóalgoritmusa a tanítóobjektumok alapján megtanulja az egyes osztályok jellegzetességeit, ebből modellt készít, amellyel becslés adható az ismeretlen kategóriájú objektum címkéjére. A tanulásnak ezt a fajtáját felügyelt tanulás (supervised learning) néven ismeri az irodalom. A szakértő az általa megadott tanítóadatokon keresztül „felügyeli" a tanulás folyamatát.
Ez a fajta tanulási módszer hatékonyan megvalósítható a feladathoz tervezett neurális hálózatok alkalmazásával. Az ismertetésre kerülő osztályozási folyamat során a bemeneti adatokat az objektum szavainak objektív koordinátái alkotják, melyek különböző források alapján egzaktul meghatározhatóak. Az osztályozás kimenetén a szavak szubjektív koordinátái állnak.
A gyakorlati alkalmazástól függően különböző kényszerfeltételek írhatóak elő az osztályozó-függvényre vonatkozóan. Az egyik lehetőség a objektumhoz rendelt kategóriák számának korlátozása. Ettől függően az osztályozási feladatoknak két alaptípusát különböztetjük meg:

- egycímkés (single-label) esetnek nevezzük azt, amikor minden objektumot pontosan egy kategóriához kell hozzárendelni,
- többcímkés (multi-label) esetről beszélünk, amikor az objektumok 0-tól C-ig tetszőleges számú kategóriába besorolhatóak.

Az egycímkés osztályozásnak speciális esete a bináris osztályozás, amikor két kategóriába (c, illetve $\bar{c}$) kell besorolni az objektum szavait. Ha a kategóriák függetlenek egymástól, akkor a bináris osztályozókból egyszerűen készíthető többcímkés osztályozó. Ehhez elég a $c_1, \dots, c_{|C|}$ kategóriákra vonatkozó feladatot felbontani $|C|$ számú független c_j, $\overline{c_j}$ bináris osztályozási feladatra ($c_j \epsilon C$). Fordítva ugyanez nem alkalmazható. A többcímkés osztályozó nem alakítható át automatikusan binárisra vagy egycímkésre. Ugyanis, ha az előbbi egynél több kategóriát rendel egy objektumhoz, akkor nem feltétlenül egyértelmű, hogy ezek közül hogyan kell a legmegfelelőbbet kiválasztani. Ha egyet sem rendel, akkor különösen nehéz a legkevésbé rossz kategória meghatározása [Fabrizio, 2002].
A célul tűzött feladat megoldása során bináris osztályozóval valósítottam meg a mondatok szavainak kérdésként való kiemelését végző algoritmust, míg többcímkés módszerrel a többértékű szubjektív koordinátákra történő leképzés.
Az osztályozás témakörébe tartozó feladatok megoldása során tehát olyan leképző függvény megtalálása a cél, amely a rendelkezésre álló lehetőségek alapján minimális hibával képes a bemenetként kapott szavaknak a kimeneti kategóriákba sorolására. Abban az esetben, ha a konkrét feladat során rendelkezésre állnak háttér információk a leképző függvény megalkotásához, akkor ezek alkalmazásával heurisztikus függvények készíthetőek a leképzéshez.

Osztályozók problémái

Az osztályozási eljárás során a tanuló algoritmusok a paramétereket olyan szűk intervallumon, határokon belül tanulják meg, hogy ugyanabból a populációból származó más adatokon nem képes elérni az elvárt érteket. Ez az általános jellemzője a *validációs* módszereknek. Az adatok egyik részét a paraméterek megtanulására használják – ez a tanuló minta, a másikat, pedig a tesztmintára.

Ilyen módon úgy lehet a tanuló mechanizmusok paramétereit beállítani, hogy azok elkerülik a túltanítást, és így megbízhatóbb osztályozási pontosságot kapunk. Több fajtája ismeretes.
Egyszerű esetben csak egyszer végezzük el a tesztelést az eredeti adatok megadott hányadán. Ez a módszer akkor lehet megfelelő, amennyiben elegendő tanítóminta áll rendelkezésre. Így gyakran osztják két egyenlő részre az alapadatokat.
A keresztvalidálás során több lépésben történik a modell ellenőrzése. Általánosan k-szoros „kereszt validálásnak" (k-fold cross-validation) [Mitchell, 1997] hívják, melyben a k határozza meg, hogy hány lépésben végezzük el a folyamatot. Így, ha például k=10, akkor 10-szer validálunk olyan módon, hogy minden esetben más adatok alkotják a tesztmintát. Ezt célszerű úgy végezni, hogy az egyes osztályok az elemeik számával arányosan vegyenek részt a hallgatók- és így a tanítómintában is. Minden egyes ciklusban kiszámítjuk a modellünk pontosságát, és a legvégén az átlagot használják fel a végső pontosság megállapításához. A módszer előnye, hogy a minta nagyságától függően választhatjuk meg a ciklusok számát, így kisebb minták esetén is jól működik.
A „hagyj ki egyet" (leave-one-out) módszer az előbbi módszer speciális esete, mikor a k megegyezik a minták számával. Ebben az esetben minden egyes ciklusban csak egyetlen minta szolgál tesztelésre. Jellemzően kis mintaszám esetén érdemes használni [Zsebók, 2012].
Dokumentumok osztályozására a legelterjedtebben alkalmazott módszereket a döntési fa, a legközelebbi szomszéd (k-NN), a Bayes háló, a szupportvektor gép, a neurális háló, valamint a CPN (Counter-Propagation Network) háló algoritmusainak feladatspecifikus továbbfejlesztései alkotják.

Döntési fa alapú osztályozók

A döntési fán alapuló szövegosztályozó [Guangzuo, 2004] olyan fa, amelyben a közbenső csomópontok attribútumokat reprezentálnak, a csomópontokból kiinduló ágak ellenőrzési feltételeket írnak elő az adott attribútumra vonatkozóan a tesztobjektumra, végül a levelek kategóriákkal vannak címkézve. A d tesztobjektum osztályozása a döntési fa csomópontjaihoz tartozó attribútumok d-beli súlyának vizsgálata alapján rekurzív módon történik, az objektumhoz végül a levél kategóriacímkéjét rendeljük hozzá. A fa leveleiben a döntés eredményei, az osztály-hovatartozások állnak. Az egyes döntésekben az objektumok attribútumai játszanak szerepet. A döntési fa egyes csomópontjainál több gyerekág is létezhet. A döntés fentről lefelé halad, mindig egy ágon.
A döntési fa használatának nagy előnye könnyű értelmezhetőségében és interpretálhatóságában rejlik. Szintén fontos lehet néhány alkalmazásban, hogy már minimális mintaszámra is működik, habár a megbízható osztályozáshoz természetesen nagyszámú mintára van szükség.
A legtöbb osztályozó standard döntési fa tanulóalgoritmust használ, mint az ID3 (Interative Dichotomizer 3) és továbbfejlesztései: a C4.5 és a C5.0 ill. a CART (Classification and Regression Trees).

ID3 entrópia alapú osztályozó

Az ID3 [Bodon, 2010] az egyik legősibb és legismertebb osztályozó algoritmus. A teszt attribútum kiválasztásához az információnyereség elvét alkalmazza (entrópia csökkenés), vagyis azt az attribútumot választja, amely a legnagyobb információnyereséggel bír.
Ha Y egy l lehetséges értéke $p_i(i = 1, \dots, l)$ valószínűséggel felvehető valószínűségi változó, akkor Y Shannon - féle entrópiáján a

$$H(Y) = H(p_{1,\dots,}p_k) = -\sum_{j=1}^{l} p_j \, log_2 p_j$$

számot értjük. Az entrópia az információelmélet központi fogalma, és az Y változó értékével kapcsolatos bizonytalanságunkat fejezi ki.

Tegyük fel, hogy lehetőségünk volt egy X változót megfigyelni, ennek értéke tapasztalataink szerint x_i. Ekkor az Y -nal kapcsolatos bizonytalanságunk nagysága:

$$H(Y|X = x_i) = -\sum_{j=1}^{k} \mathbb{P}(Y = y_j|X = x_i) \; log_2 \mathbb{P}\big(Y = y_j\big|X = x_i\big).$$

Azaz, a várható bizonytalanságunk abban az esetben, ha lehetőségünk van X-et megfigyelni:

$$H(Y|X) = \sum_{i=1} \mathbb{P}(X = x_i) \; H(Y|X = x_i).$$

Eszerint X megfigyeléséből adódó információnyereség:

$$I(Y, X) = H(Y) - H(Y|X)$$

azaz, ennyi információt hordoz X az Y -ról (ennyivel csökken az entrópia).

Az algoritmus minden attribútum esetén kiszámítja az entrópia csökkenését, majd végül azt a változót választja ki, melyre ez az érték a legnagyobb volt, azaz amelyre $I(Y, X)$ maximális (vagyis $H(Y|X)$ minimális). Ezen attribútum szerint ágazik szét az ID3 algoritmus annyi felé, ahány értéke van a kiválasztott attribútumnak.

A túltanulás elkerülése végett, egyrészt meg lehet adni az elérni kívánt mélységet, illetve fametszést (tisztítást) lehet alkalmazni, melyet bizonyos ágak törlésével lehet elérni. Ezekre azért van szükség, mert bizonyos illeszkedésen túl már csak a tanulómintákban jelenlévő sajátosságokat tanulja meg a modell, és emiatt kevésbé hatékonyan működik a tesztadatokra. Egy optimális értéket kell megtalálni, hiszen a fa túlzott redukálása szintén rontja az osztályozás pontosságát.

Legközelebbi szomszédokon alapuló osztályozó

A legközelebbi szomszédokon alapuló osztályozó lokálisan [Tikk, 2007], az adott tesztobjektumhoz hasonló tanítóobjektumok címkéje alapján dolgozik. A módszer legfontosabb paramétere, hogy a döntés meghozatalánál hány hasonló tanítóobjektumot vesz figyelembe. Ezt a paramétert *k*-val jelöljük, az osztályozót általánosan, pedig $k-$NN (nearest neighbor) osztályozónak nevezzük.

A $k-$NN osztályozó működése:

- bejövő objektumhoz a hozzá legközelebb álló *k* darab tanítóobjektum meghatározása,
- ezen objektumok osztálykódjait képezzük,
- ez lesz a hozzárendelt objektum.

A módszer előnye; egyszerű és gyors, hátránya; nem érzékeli a megbízhatóságot és a közelítés mértékét.

Naív Bayes osztályozó

Valószínűségelméleti megközelítésben a Φ osztályozó létrehozásának feladatát a $P\big(c_j\big|d_i\big)$ feltételes valószínűségi értékekre vonatkozó becslésként fogalmazható meg. Ez az érték megadja, hogy milyen valószínűséggel tartozik a d_i objektum a c_j kategóriába. A feltételes valószínűségre a becslés a Bayes-tétel alapján [Tikk, 2007]:

$$P(c_j|d_i) = \frac{P(c_j)P(d_i|c_j)}{P(d_i)}.$$

A d_i objektumot ahhoz a c kategóriához rendeljük, amelyikre a $P(c_j|d_i)$ értéke maximális:

$$c_{MAP} = \underset{j=1,\ldots,|C|}{\arg\max}\ P(c_j|d_i) = \underset{j}{\arg\max}\ \frac{P(c_j)P(d_i|c_j)}{P(d_i)} = \underset{j}{\arg\max}\ P(c_j)P(d_i|c_j).$$

Az alsó index a becslés maximumára utal, azaz a valószínűséget a rendelkezésre álló megfigyelések figyelembe vételével határozhatjuk meg. A $P(d_i)$ érték a számlálóból elhagyható, mert ez minden osztály esetén ugyanaz lesz.

Szupportvektor-gépek

Az SVM (Support Vector Machine) alapverziója a lineáris osztályozók [Boyer, 2009] családjába tartozik, és bináris osztályozási problémák megoldására alkalmas. A korábban említett lineáris osztályozókhoz képest az a fő ismérve, hogy nemcsak egy olyan hipersíkot keres, amely elválasztja a pozitív és negatív tanítómintákat, hanem ezek közül a legjobbat, vagyis, amelyik a két osztály mintái között épp középen fekszik.

Osztályozásra Joachims [Joachims, 1999] alkalmazta elsőként a nem szeparábilis esetre vonatkozó módosítást. Előnyei közé tartozik, hogy az SVM használata mellett általában nincs szükség a problématér redukciójára, mivel egyrészt a módszer elég robusztus a túltanulásra, másrészt jól skálázható, azaz nem érzékeny a magas dimenziószámra.

Az SVM hatékonyságának kulcsa, hogy jó általánosító képességgel bír, azaz ismeretlen adatokra is pontos becslést képes adni. A tanítóadatok tulajdonságai és a módszer tanítóadatokon való viselkedése alapján elméleti felső hibakorlát adható az új adatokra vonatkozó hibára.

Neurális hálózat alapú osztályozók

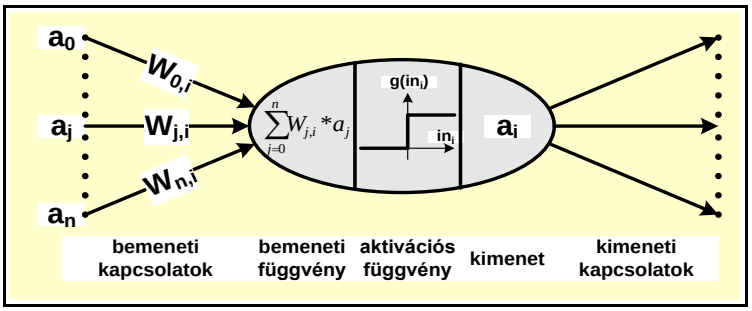
Munkám során az osztályozást végző modellt úgy terveztem, hogy képes legyen bármilyen tématerülethez tartozó szöveg kezelésére. Ilyen általános esetekre az egyik legelterjedtebben alkalmazott módszer a neurális hálóval végzett tanulás, melynek elvét McCulloch és Pitts [McCulloch és Pitts, 1943] publikálta 1943-ban. A neurális hálók irányított kapcsolatokkal összekötött egységekből (neuron) állnak. A j jelű neuronból az i jelű felé vezető kapcsolat hivatott a j jelű neuron aktivációs állapotát (jele: a_j) az i jelű neuronhoz továbbítani. Minden egyes kapcsolat rendelkezik a hozzá társított $W_{j,i}$ numerikus súllyal, ami meghatározza a kapcsolat erősségét és előjelét. Minden egyes neuron először a bemeneteinek egy súlyozott összegét számítja ki a (6.1) összefüggés alapján:

$$in_i = \sum_{j=0}^{n} W_{j,i} * a_j. \tag{6.1}$$

Az i-edik neuron aktivációs állapota a bemeneteinek súlyozott összegére alkalmazott g jelű aktivációs függvénnyel kerül meghatározásra a (6.2) összefüggés szerint [McCulloch és Pitts, 1943]:

$$a_i = g(in_i) = g\left(\sum_{j=0}^{n} W_{j,i} * a_j\right). \tag{6.2}$$

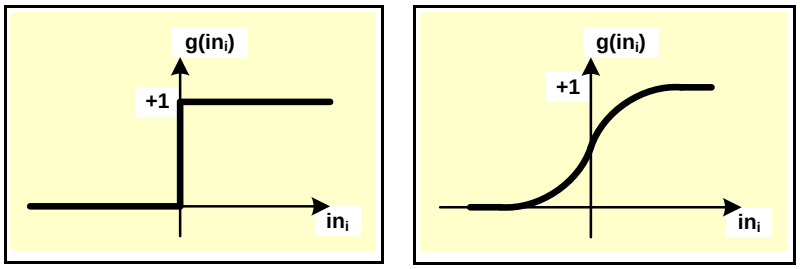
A neuron matematikai modelljét a 6.1. ábra szemlélteti.



6.1. ábra: A neuron matematikai modellje

Az aktivációs függvénnyel szemben követelmény, hogy a neuron legyen aktív (+1 körüli érték), ha a „helyes" bemeneteket kapja és „inaktív" (0 körüli érték) ha „helytelen" bemeneteket kap. Ezen felül az aktivációs függvénynek nemlineárisnak kell lennie, mivel lineáris függvény esetén az egész neurális háló egy egyszerű lineáris függvénnyé fajul [Dagan, 1997]. A 6.2. ábra két lehetséges aktivációs függvényt mutat be. Az 6.2.a ábrán a küszöbérték függvény, a 6.2.b ábrán, pedig a szigmoid függvény látható.

A munkám során a küszöbérték függvényt alkalmaztam, de a későbbi tesztelések megkönnyítése végett lehetőséget biztosítottam ennek tetszőleges cserélhetőségére is. Az 6.2. ábrán [McCulloch & Pitts, 1943] bemutatott mindkét függvény küszöbpontja az $x = 0$ értéknél van. A küszöbpont módosításával lehetőség nyílik a neuron aktiválódásához szükséges bemeneti összeg értékének eltolására pozitív, illetve negatív irányban. Módosításával elérhető, hogy alacsonyabb (negatív irány) vagy magasabb (pozitív irány) értékű bemeneti összeg hatására aktiválódjon a neuron. A megoldott feladat során ezt az értéket minden neuron esetén $x = 0$-ra állítottam be ám lehetőséget hagytam a későbbi módosíthatóságára.



6.2.a 6.2.b

6.2. ábra: Aktivációs függvények a neuron kimenetének meghatározásához

Az információk áramlási iránya alapján a neurális hálózatok két fő csoportba sorolhatóak. Ezek az előrecsatolt hálók, (feed-forward network) illetve a visszacsatolt hálók (recurrent network) [Ramasundaram, 2010]. Az előrecsatolt hálók a pillanatnyi bemenet függvényét reprezentálják, azaz a súlyokon kívül nincs semmilyen más belső állapotuk. A visszacsatolt hálók kimenetei visszacsatolásra kerülnek a bemeneteire. Ennek eredményeként a háló aktivációs szintje dinamikus rendszert alkotva elérhet stabil állapotba, de akár oszcillációba vagy kaotikus viselkedésbe is kerülhet. Ezen felül a visszacsatolt neurális hálózatoknak az aktuálisan bemenetre adott válasza a kezdeti állapotától függ, amely a korábbi bemenetektől függhet. Ebből a tulajdonságból kifolyóan a visszacsatolt hálózatok rövid távú memóriát is biztosítanak [Robert Hecht-Nielsen, 1989].

Munkám során az előrecsatolt hálózati struktúrát alkalmaztam, mivel a memória által nyújtott többletinformáció a konkrét feladatunk esetén nem volt hasznosítható. Az előrecsatolt neurális hálózatok megkülönböztethetőek egymástól a rétegek száma, az egyes rétegekben lévő neuronok

száma, valamint a neuronok közötti kapcsolatok alapján is. Általánosságban a neurális hálózat három különböző feladatot ellátó rétegből épül fel. Ezek a bemeneti réteg, rejtett réteg és kimeneti réteg. A bemeneti réteg végzi a bemenetről kapott jelek továbbítását a rejtett réteg felé. A kimeneti réteg szolgáltatja a hálózatnak az aktuális bemeneti jelekre adott válaszát. Bizonyos egyszerű feladatra tervezett neurális hálózatok nem rendelkeznek rejtett réteggel. Az ilyen hálózatokat perceptronhálózatnak nevezi az irodalom. Egy vagy több rejtett réteg hozzáadásának az az előnye, hogy kiterjeszti a hálózat által reprezentálható hipotézisek terét. Egyetlen megfelelően nagy neuronszámú rejtett réteggel a bemenetek tetszőleges folytonos függvénye tetszőleges pontossággal reprezentálható. Két rejtett réteg alkalmazása esetén nemfolytonos függvények is reprezentálhatóak. Az optimális működéshez a rejtett rétegben szükséges neuronok számának meghatározása még napjainkban sem megoldott probléma. Túl sok rejtett neuron használata esetén a hálózat képes lesz memorizálni a teljes tanítómintát, ezáltal nem fejlődik ki az általánosító képessége, mellyel alkalmassá válik a korábban még nem látott mintához a helyes válasz hozzárendelésére.
A rejtett réteggel nem rendelkező neurális hálózatok tanulásának folyamatát a 6.1. algoritmus szemlélteti [Stuart, 2005].

függvény PERCEPTRON-TANULÁS (*példák*, *háló*) **eredmény** egy *perceptronhipotézis*
bemenetek:*példák*, egy mintahalmaz, mindegyikhez $x = x_1, \dots, x_n$ bemenet és y kimenet tartozik
háló, egy $W_j, j = 0 \dots n$ súlyokkal és g aktivációs függvénnyel rendelkező perceptron
ismételd
minden p-re $p \in példák$
$in \leftarrow \sum_{j=0}^{n} W_j * x_j[p]$
$Err \leftarrow y[p] - g(in)$
$W_j \leftarrow W_j + \alpha * Err \times g'(in) \times x_j[p]$
amíg a definiált leállási feltétel nem teljesül
eredmény NEURÁLIS-HÁLÓ-HIPOTÉZIS(*háló*)

6.1. algoritmus: A perceptron tanulási algoritmusa

Err jelöli a perceptron kimenetén tapasztalt jelnek az elvárt értéktől való különbségét. Az egy vagy több rejtett réteggel rendelkező háló tanuló algoritmusa hasonló az 6.1. algoritmuséhoz. Ebben az esetben azonban több kimenet is lehetséges, így a kimeneti érték nem skalár, hanem egy $h_W(x)$ vektor, valamint minden példához egy y kívánt kimeneti vektor tartozik. A háló kimeneti rétegében a hiba az $y - h_W$ összefüggéssel számítható. Ezzel szemben a rejtett rétegben a hiba meghatározása már jóval összetettebb. A tanítómintákból ugyanis nem derül ki, hogy miknek kell lenniük a rejtett neuronok aktiváltsági állapotainak. Ezáltal a kívánt értéktől való eltérés is nehezen határozható meg. A feladat megoldása a kimeneti hibajelnek a rejtett réteg(ek) felé való visszaterjesztésével (back-propagation) érhető el. Jelölje Err_i az $y - h_W$ hibavektor i-edik komponensét. Jelölje továbbá Δ_i a módosított hibát, $\Delta_i = Err_i \times g'(in_i)$. Ezek alkalmazásával a súlyfrissítési szabály [Stuart, 2005]:

$$W_{i,j} \leftarrow W_{i,j} + \alpha \times a_j \times \Delta_i. \tag{6.3}$$

A bemeneti neuronok és a rejtett neuronok közötti összeköttetések frissítésének megvalósításához a kimeneti csomópontok hibájához hasonló mennyiséget kell definiálni. Ezen a ponton kell elvégezni a hibajel visszaterjesztését. Ennek értelmében megállapítható, hogy a j jelű rejtett csomópont valamilyen arányban „felelős” minden egyes vele összeköttetésben lévő kimeneti csomópont Δ_i hibájáért. Így a Δ_i értéket a rejtett csomópont és a kimeneti csomópont közötti összeköttetés erőssége alapján osztjuk és visszaterjesztjük, hogy megkapjuk a rejtett réteg Δ_j értékét. A Δ értékek terjesztési szabálya a következő [Stuart, 2005]:

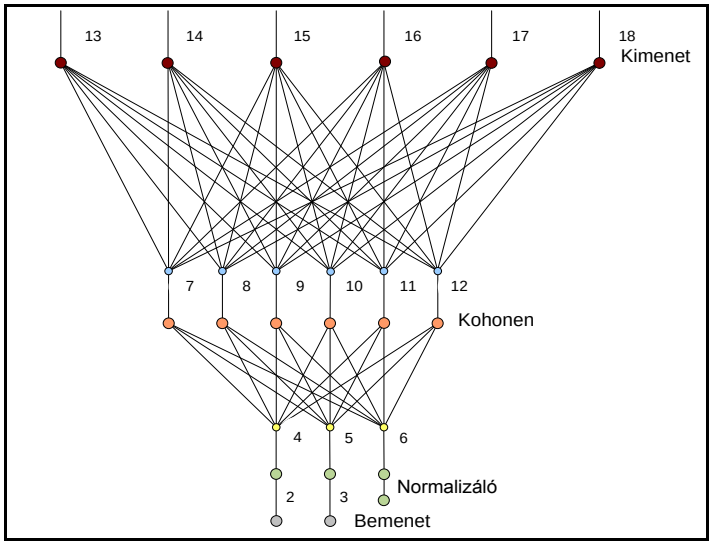
$$\Delta_j = g'(in_j) * \sum_i W_{j,i} * \Delta_i. \tag{6.4}$$

Ezek alapján a bemenetek és a rejtett réteg közötti súlyok frissítési szabálya szinte azonos a kimeneti réteg frissítési szabályával [Stuart Russel, 2005]:

$$W_{k,j} \leftarrow W_{k,j} + \alpha \times a_k \times \Delta_j. \tag{6.5}$$

CPN hálózat alapuló osztályozó

A CPN hálózat alapú osztályozó [Hecht-Nielsen, 1987], [Dudás, 2006] lényegében a neurális hálózatok egy feladatspecifikus alkalmazásai. A CPN hálózat bemeneti rétege itt is az osztályozandó objektumok leírását tartalmazza (például az objektumok többdimenziós térben elfoglalt helyének leírására szolgáló koordináták alapján). A rejtett rétegben (Kohonen-réteg) lévő neuronok ebben az esetben a klasztereket reprezentálják (SOM). A kimeneti réteg neuronjai pedig azokat az osztályokat írják le, amelyekbe a bemeneten lévő objektumok besorolásra kerülnek. A kimeneti réteget a CPN hálózatok esetén Grossberg-rétegnek is nevezik. A 6.3. ábra egy CPN hálózat alapú osztályozó modelljét szemlélteti [Hecht-Nielsen, 1989].



6.3. ábra: CPN hálózat alapú osztályozó

A hálózat működése hasonló a neurális hálónál ismertetettel. A különbség abban jelentkezik, hogy a Kohonen-rétegben egy kompetitív réteg van, egyetlen győztes neuronnal. A győztes áll legközelebb a beérkező objektumhoz. A neuronok bemeneti függvénye a (6.6) összefüggés alapján kerül meghatározásra:

$$s = \sum w_i * s_i. \tag{6.6}$$

Az s_i jelöli a bejövő jelet (a bemeneti neuron koordinátáit), w_i pedig az él súlyát. Az s érték megfelel a w és az s vektorok skaláris szorzatának, ami a közbezárt szög koszinuszával arányos. A tanulási folyamat során a győztes neuron mindig közelebb kerül a bemenetre kapcsolt objektumhoz:

$$w_{t+1} = w_t + \alpha(bemenet - w_t) \tag{6.7}$$

ahol, az α paraméterrel a tanulási ráta állítására van lehetőség.

6.2. Az osztályozást megvalósító algoritmus

6.2.1. Az algoritmus bemeneteinek és kimeneteinek ismertetése

A megvalósításra került osztályozó algoritmus a dokumentum szavainak objektív koordinátái, illetve a tanítóminta alapján állítja elő szavak szubjektív koordinátáit. Az objektív koordináták, megválasztásánál a szavak olyan tulajdonságainak a megtalálása volt a célom, melyekkel jól jellemezhetőek a vizsgált szavak, ugyanakkor meghatározásuk nem ró lényeges számítási többletet az osztályozást végző programmodulra. Ezek alapján minden szó objektív vetülete hét koordinátával került definiálásra. Az alkalmazott objektív koordinátákat és lehetséges értékeiket a 6.1. táblázat foglalja össze.

Sorszám	Név	Értékkészlet
1	szófaj	-1 : a szótövező program nem ismerte fel a szót, vagy több lehetséges szófajt is kínált fel hozzá és ezek közül a felhasználó meg nem választott 0 : a szótövező program által javasolt szófajt az osztályozó algoritmus nem ismerte fel 1 : NOUN - főnév 2 : ART - névelő 3 : ADV - határozószó 4 : NUM - számnév 5 : VERB - ige 6 : ADJ - melléknév 7 : CONJ - kötőszó 8 : DET - határozott névelő 9 : PREV - igekötő 10 : UTT-INT mondatszó/indulatszó [Csendes D., 2004]
2	ragozottság	-1 : a szótövező program nem ismerte fel a szót, vagy több lehetséges szófajt is kínált fel hozzá és ezek közül a felhasználó meg nem választott 0 : nem ragozott 1 : ragozott
3	előfordulási száma a dokumentumban	[1, előfordulási szám] egész szám
4	előfordulási szám az adott szót tartalmazó mondatban	[1, előfordulási szám] egész szám
5	hely az adott szót tartalmazó mondatban	[1, az adott szót tartalmazó mondat azon szavainak száma, melyek nem kerültek kizárásra az elemzésből] egész szám
6	az adott szót tartalmazó klaszter sorszáma	-1 : a dokumentum szavainak klaszterezése még nem készült el, vagy a több lehetséges klaszterezési eredmény közül még nem lett kiválasztva az alkalmazandó klaszter [1, klaszterek száma] egész szám
7	távolságátlaga az adott szót tartalmazó mondatnak azon szavaitól, melyek nem kerültek kizárásra az elemzésből	[0.0, 1.0] normalizált valós szám

6.1. táblázat: Az alkalmazott objektív koordináták és lehetséges értékeik

Az objektív koordinátákat összefoglaló táblázat alapján látható, hogy a 3., illetve 6. koordináta kivételével a vizsgált dokumentum azonos szavai is különböző objektív értékekkel kerülhetnek leírásra. Ebből az következik, hogy az osztályozás bemenetén nem szavaknak, hanem koordinátáknak kell állniuk. Az elemzésben résztvevő szavak objektív koordinátái négy forrás bevonása alapján kerültek meghatározásra:

- a szótövező program interneten ingyenesen elérhető alkalmazással,
- szakértők véleményének interaktív kinyerésével,
- a dokumentum szavainak klaszterezését végző saját fejlesztésű alkalmazással,
- a dokumentum egyszerű statisztikai elemzésével nyerhető adatok alapján.

Az 1. objektív koordináta meghatározását a szótövező program, illetve a szakértő végzi. A 2. koordináta megállapítása a szótövező program feladata az 1. koordináta értékének figyelembevétele alapján. A 3., 4., illetve 5. koordináták értékei a szöveg egyszerű statisztikai elemzésével egyértelműen meghatározhatóak. A 6. koordinátát a klaszterező modul szolgáltatja. A 7. koordináta értéke pedig a klaszterező modul eredményeinek ismeretében statisztikai elemzéseket követően határozható meg.
Az osztályozást végző algoritmus kimenetét a szavaknak az emberi megítéléstől függő (szubjektív) koordinátái alkotják. Az osztályozás eredményeként szolgáló szubjektív koordinátákat és lehetséges értékeiket a 6.2. táblázat foglalja össze.

Sorszám	Név	Értékkészlet
1	relevancia mértéke	[1, 5] egész szám
2	specialitás mértéke	[1, 5] egész szám
3	értelmesség mértéke	[1, 5] egész szám
4	nehézség mértéke	[1, 5] egész szám
5	kérdésként kiemelésre kerüljön-e	igen nem

6.2. táblázat: A szubjektív koordináták és lehetséges értékeik

6.2.2. Az alkalmazott neurális hálózat struktúrája

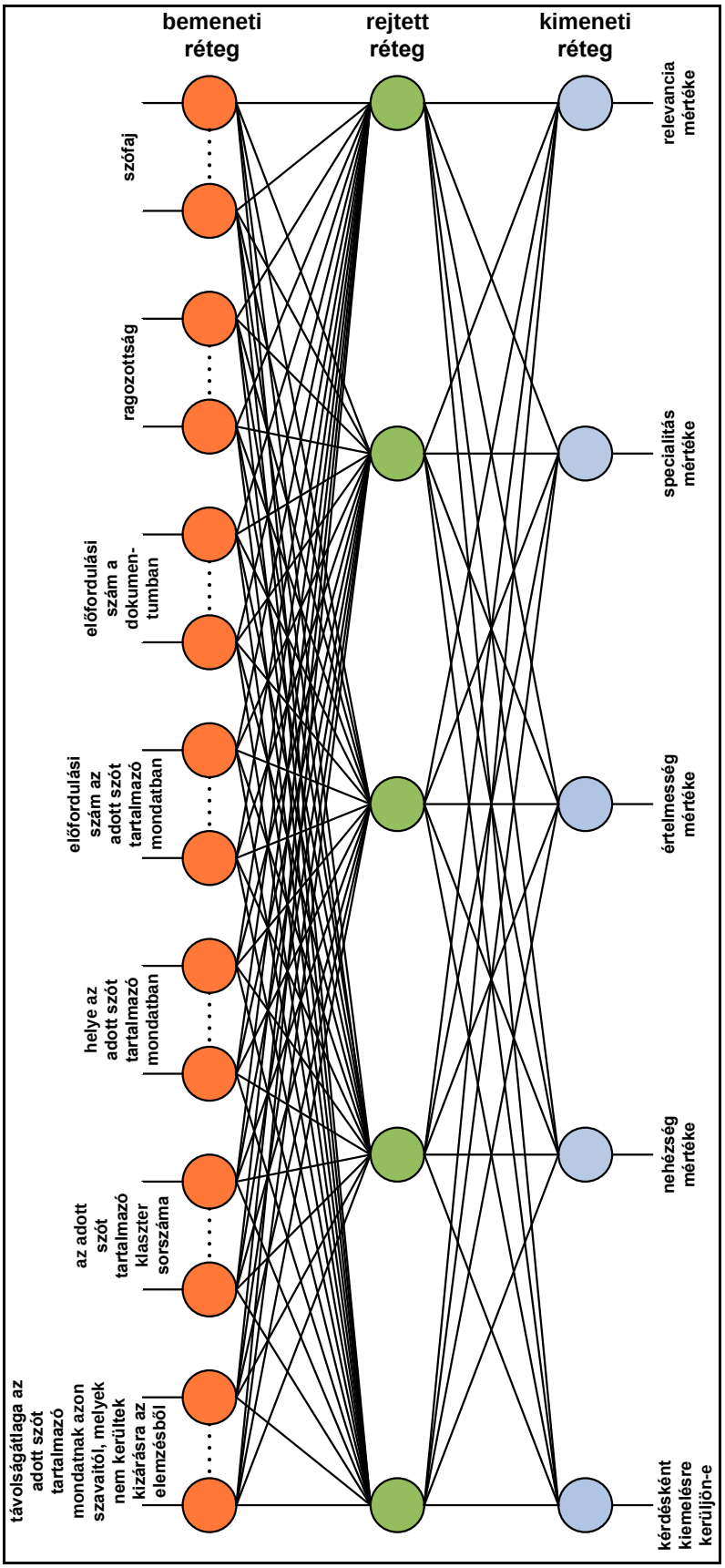
Az osztályozás egy előrecsatolt háromrétegű (egy belső réteg) neurális háló alkalmazásával került megvalósításra. A neuronok közötti kapcsolatok súlyértékei [-1.0, 1.0] tartományba tartozó valós számok. A háló neuronjainak aktiváltsági állapota eltolás nélküli küszöbérték függvénnyel lett reprezentálva, mely a bemeneti jeleket súlyozottan összegző bemeneti függvény értékét a [0, 1] egész intervallumra képezi le az 6.8. összefüggés alapján:

$$a_i = 1, ha \sum_{j=0}^{n} W_{i,j} * a_j > 0.0$$
$$a_i = 0, egyébként. \tag{6.8}$$

A háromrétegű neurális hálózat első rétegét a bemeneti neuronok alkotják, melyeknek feladata, hogy az objektív koordináták értékeit a hálózat belső (rejtett) rétegében lévő neuronok felé továbbítsák. A kimeneti neuronok az osztályozandó szavak szubjektív koordinátáira vannak leképezve. A kimenetek értékeinek könnyebb szeparálhatósága érdekében minden kimeneti értékhez külön neuront hoztam létre. Mivel a szubjektív koordináták értékei minden vizsgált dokumentum esetén állandóak, ezért a kimeneti neuronok száma minden dokumentum esetén: $4 * 5 + 1 * 2 = 22$ darab. A szubjektív koordináták minden vizsgált dokumentum esetén ezzel az értékkészlettel kerültek modellezésre. A kimeneti neuronok definiálják a szubjektív koordináták

értékeit minden szó osztályozása esetén. Abban az esetben, ha egy szubjektív koordináta több különböző értékét definiáló kimeneti neuron is aktív állapotba kerül, akkor az algoritmus azt választja ki közülük, amelyiknek a bemeneti függvénye által szolgáltatott értéke a legnagyobb. Ezáltal a neuronok kétértékű (aktív/nem aktív) állapotán kívül kihasználhatóak a többértékű bemeneti függvény által nyújtott lehetőségek is.
Hasonlóan, ahogy az alkalmazott neurális hálózat kimeneti neuronjai a szubjektív koordináták különböző értékeit reprezentálják úgy a bemeneti neuronok az objektív tér koordinátáinak különböző értékeihez vannak hozzárendelve. Az objektív koordináták értékeit összefoglaló 6.1. táblázat adatai alapján látható, hogy az egyes koordináták által felvehető értékek erősen függnek az elemzés alatt álló dokumentum jellegétől. A felvehető értékek közül a legnagyobb szórást általában a 6. objektív koordináta mutatja, mely az adott szót tartalmazó klaszter sorszámát hivatott tárolni. A koordináta által felvehető értékek halmaza jelentősen függ attól, hogy hány klaszterbe sikerül besorolni az elemzés alatt álló dokumentum szavait. Jelentős szórást mutathat a 3. objektív koordináta lehetséges értékkészlete is, amely azt tartalmazza, hogy a dokumentum egyes szavai hány alkalommal fordultak elő a vizsgált dokumentumban. Sajnos egyik objektív koordináta esetén sem határozható meg az általa felvehető értékek halmaza a konkrét dokumentum ismeretét megelőzően. Ezáltal minden előre definiált értékkészlet esetén fennáll az a veszély, hogy az aktuálisan vizsgálandó dokumentum valamelyik objektív koordinátája nagyobb értéken szerepel, mint a hozzá előre meghatározott maximális érték. Ilyen eset áll elő akkor, ha a neurális háló bemenetét képező objektív koordináták közül egynél vagy akár többnél is nagyobb értékkészlet kezelésére van szükség az adott feladat során, mint amilyenre a hálózat a korábbi tanulások alkalmával fel lett készítve. Ezekben az esetekben az előre definiált bemeneti neuron számú hálózat nem képes a feladat kezelésére.
A bemeneti neuronok számának túl nagyra választása sem vezet eredményre, mivel ha az aktuális dokumentum esetén az adott objektív koordináta ettől jóval kevesebb értéket vehet csak fel, akkor az előre létrehozott nagyszámú neuron közül a többség sosem fog aktívvá válni. Ez azon kívül, hogy feleslegesen növeli a memóriafoglalást és a számításhoz szükséges időt a neurális háló tanulási képességét is rontja. A feladat megoldása érdekében a kifejlesztésre került algoritmus csak azt követően építi fel a neurális háló bemeneti rétegét, miután befejeződött az aktuálisan vizsgált dokumentum szavaihoz tartozó objektív koordináták meghatározása. Csak ekkor válik ismerté ugyanis az egyes objektív koordináták értékkészlete. Ezáltal került megvalósításra, hogy az algoritmus az objektív koordináták minden értékéhez külön neuront vehessen fel a bemeneti rétegben.
A neurális hálózatok építésének egyik kulcskérdését jelenti a belső rétegben lévő neuronok számának meghatározása. Az optimális értéktől alacsonyabb számú neuron alkalmazása esetén a hálózat nem lesz képes a feladat megtanulásához szükséges mennyiségű információ tárolására. A kevés számú belső neuron hatása leglátványosabban abban jelentkezik, hogy a neurális háló betanultsági szintje $\left(\frac{helyesen\ osztályozott\ szavak\ száma}{osztályozott\ szavak\ száma}\right)$ nem képes elérni az elvárt értéket. Az optimális értéktől több belső neuron használata esetén a neurális hálózat általánosító képessége csökken. A csökkenés mértéke egyenesen arányos az optimális értéken felül használt belső neuronok számával. Az általánosítási képesség csökkenésével a neurális hálózat csökkent mértékben képes a tanításra használt mintákban rejlő szabályok feltárására. Ehelyett az eredményei egyre inkább a konkrét minták betanulásából származnak.
Jelen feladat esetében a belső rétegben helyezendő neuronok számának pontos meghatározása továbbfejlesztési lehetőségként jelentkezik. Első közelítésként számukat azonosnak vettem a kimeneti rétegben lévő neuronok számával. Eredményként a vizsgált dokumentumok szinte mindegyikénél sikerült 0.01 pontossággal megközelíteni a tökéletes betanultsági szintnek számító

1.0 értéket. Ez tehát azt támasztja alá, hogy a közbenső rétegben lévő neuronok száma nem kevesebb az optimális értéktől. Ezek alapján az alkalmazásra került neurális hálózat struktúráját a 6.4. ábra szemlélteti.



6.4. ábra: Az osztályozási feladatban alkalmazásra került neurális hálózat alapstruktúrája

6.2.3. Az alkalmazott neurális hálózat tanulási folyamata

A neurális hálózat tanítása a felügyelt tanítás szabályai alapján valósítottam meg [Fajszi, 2010]. Ez alapján az elemzendő dokumentum klaszterezését, valamint a neurális hálózat automatikus felépítését követően a tanításba bevont szakértőnek lehetősége nyílik a dokumentum mondatai közül kiválasztani azokat, amelyek a tanulásra szolgáló mintát fogják alkotni. A megvalósított modell jelenlegi állapotában nem állít fel korlátot a tanításra választott mondatok számára vagy egyéb jellemzőjére vonatkozóan. Ez teljes mértékben a szakértőre van bízva. Annak érdekében azonban, hogy a hálózat a tanító mintához nem tartozó mondatok esetén is elfogadható pontossággal tudja elvégezni a szavak osztályozását, érdemes betartani azt az általánosan elfogadott szabályt, mely szerint a dokumentum mondatainak közelítőleg a fele kerüljön a tanításra szolgáló mintába.
Az osztályozásra kidolgozott modell implementálása során megvalósítottam, hogy az alkalmazott neurális hálózat elmenthető legyen. Ezáltal egy újabb dokumentum esetén az elmentett adatok alkalmazása erős támogatást nyújt az aktuális feladat pontosabb elvégzéséhez. Javasolt koncepció tehát az első néhány dokumentum összes mondatának tanító mintaként való kezelése. Ezáltal ezeket a dokumentumokat a hálózat csak tanulásra használja. A módszerrel elérhető, hogy az újonnan kapott dokumentumok esetén a neurális hálózat már tanítás nélkül is a kívánt betanultsági szinttel rendelkezzen az osztályozási feladat elvárt pontosságú elvégzéséhez. A tanulásra szánt mondatok kijelölését követően a szakértőnek definiálnia kell a kijelölt mondatok összes szavának szubjektív koordinátáit. Más szavakkal a szakértőnek el kell helyeznie a tanításra szánt mondatok minden szavát az ötdimenziós szubjektív térben. Ezt követően indítható el a neurális hálózat tanulási folyamata, mely ettől a ponttól kezdve már nem igényel emberi beavatkozást.
A tanulás első lépéseként az előállított hálózat neuronjai közötti kapcsolatok súlyértékei véletlenszerűen beállításra kerülnek a [-1.0, 1.0] intervallumból. A véletlen számokat generáló algoritmus az adott intervallumon egyenletes eloszlás szerint állítja elő a véletlen számokat.
A súlyok kezdeti értékének beállítását követően az algoritmus sorra veszi a tanítóminta szavait és egymás után a hálózat bemenetére kapcsolja azokat. Ezt egy speciális illesztőmodulon keresztül végzi el, mely a szó minden objektív koordinátájának értékéhez meghatározza a neurális hálózatnak azt a neuronját a dinamikusan felépített bemeneti rétegben, amely az adott koordináta értékét hivatott reprezentálni. Az illesztés eredményeként az adott szónak az objektív térben való elhelyezkedését megadó koordinátákat reprezentáló bemeneti neuronok aktív állapotba kerülnek. Aktív állapotukról, kapcsolataik révén értesítik a rejtett rétegben lévő minden neuront. Ezt követően, a rejtett rétegben lévő neuronok meghatározzák saját aktiváltsági állapotukat a bemeneti rétegtől kapott jelek alapján, a (6.8) összefüggés alapján. A (6.8) összefüggést a bemeneti és a rejtett réteg közötti neuronokra alkalmazva a_i jelenti az i jelű rejtett rétegbeli neuron aktiváltsági állapotát, a_j pedig a j jelű bemeneti rétegbeli neuronét. A rejtett rétegben lévő neuronok aktiváltsági állapotának beállítását követően kerül sor a hálózat kimeneti rétegében lévő neuronok aktiváltsági állapotának meghatározására. Ehhez szintén a (6.8) összefüggés kerül alkalmazásra, de ebben az esetben a_i az i jelű külső rétegbeli neuron aktiváltsági állapotát, a_j pedig a j jelű rejtett rétegbeli neuronét jelöli. Összefoglalóan az i jelű – kimeneti rétegben lévő – neuron aktiváltsági állapota a bemeneti rétegben lévő neuronok aktiváltsági állapotának függvényében a (6.9) összefüggés alapján határozható meg [3]:

$$a'_i = g\left(\sum_{j=0}^{n} W_{j,i} * g\left(\sum_{k=0}^{m} W_{k,j} * a_k\right)\right) \tag{6.9}$$

Ahol: a_k: a k jelű bemeneti rétegbeli neuron aktiváltsági állapota,
a'_i: az i jelű kimeneti rétegbeli neuron aktiváltsági állapota,
m: a bemeneti rétegben lévő neuronok száma,
n: a rejtett rétegben lévő neuronok száma,
$W_{k,j}$: a k jelű bemeneti rétegbeli neuron és a j jelű rejtett rétegbeli neuron közötti kapcsolat erőssége,
$W_{j,i}$: a j jelű rejtett rétegbeli neuron és az i jelű kimeneti rétegbeli neuron közötti kapcsolat erőssége,
g: aktivációs függvény, mely 1-es értéket reprezentál, ha a paramétereként kapott összeg pozitív, 0-át egyébként.

A kimeneti rétegben lévő neuronok aktiváltsági állapotai írják le a neurális hálónak a bemenetként kapott szó objektív koordinátáira adott válaszát a szubjektív térben. Más szavakkal: a kimeneti rétegben lévő neuronok aktiváltsági állapotai jelölik ki azokat a szubjektív kategóriákat, melyekbe a bemenetként kapott szó tartozik. A válasz előállítását követően – a felügyelt tanítás szabályai szerint – következik a válasz helyességének megállapítása. Ehhez a bemeneti neuronoknál ismertetett módszerhez hasonlóan egy speciális illesztőmodullal meghatározásra kerülnek az egyes kimeneti neuronok elvárt aktivációs állapotai. Erre az illesztő modulra azért van szükség, mert a kimenetre csatolt szubjektív koordináták többértékűek, ám a 6.4. ábrán bemutatott módon felépített neurális hálóban minden kimeneti értékhez külön neuron került felvételre. A kimeneti illesztő modul végzi el a szubjektív tér koordinátáinak minden értékéhez a megfelelő kimeneti neuron hozzárendelését. Mivel a kimeneti neuronok bináris értékeket reprezentálnak, ezért aktiváltsági állapotuk helyességének elbírálása csupán négy lehetséges kimenetelhez vezethet. Ezeket a helyesség elbírálása szempontjából lehetséges kimeneteleket a 6.3. táblázat foglalja össze.

Tapasztalt aktiváltsági állapot	Elvárt aktiváltsági állapot	Végrehajtandó cselekvés
0	0	nincs
0	1	erősítés
1	0	gyengítés
1	1	nincs

6.3. táblázat: A kimeneti rétegében lévő neuronok aktiváltsági állapotának helyesség-elbírálása

A táblázat adatai alapján jól látható, hogy minden kimeneti neuron aktiváltsági állapota kétféleképpen térhet el az elvárt értéktől. Ennek megfelelően két esetet különböztetünk meg:

1. amikor az adott kimeneti neuron nem kerül aktiválásra, holott a tanítóminta aktuális szava esetén aktív állapoton kellene lennie,
2. amikor az adott kimeneti neuron aktiválásra kerül, holott a tanítóminta aktuális szava esetén inaktív állapoton kellene lennie.

Ezek alapján a hibás működés javítására szolgáló beavatkozás is kétféle lehet.
Az 1. esetben azt kell elérni, hogy az adott kimeneti neuron bemenetére kapcsolt neuronok nagyobb arányban legyenek aktívak, valamint azok a neuronok amelyek aktívak, nagyobb súllyal továbbítsák ezt az állapotukat az adott kimeneti neuron felé.
A 2. esetben éppen az ellenkező irányú beavatkozásra van szükség. El kell érni, hogy az adott kimeneti neuron bemenetére kapcsolt neuronok kisebb arányban legyenek aktívak, valamint azok a neuronok amelyek aktívak, kisebb súllyal továbbítsák ezt az állapotukat az adott kimeneti neuron

felé. A neurális háló tanulási folyamata a $w_{t+1} = w_t + \alpha(bemenet - w_t)$ képlet (6.7) alapján a következőképpen módosul.

függvény NEURÁLIS_HÁLÓ_TANULÁS (*tanítóminta*, *háló, elvárt_pontosság, lépésszámkorlát, súlyegység*)
eredmény *hipotézisfüggvény*

bemenetek: *tanítóminta:* szavak halmaza, melyben minden szó esetén adottak az objektív és a szubjektív koordináták
háló: az adott feladathoz létrehozott struktúrájú neurális háló, [-1.0, 1.0] tartományból véletlenszerűen feltöltött súlyértékekkel és eltolás nélküli küszöbérték aktivációs függvénnyel
elvárt_pontosság: a *tanítóminta*ban szereplő szavak *hipotézisfüggvény* által meghatározott szubjektív koordinátáinak megengedett maximális eltérése a tanítómintában tárolt értéküktől
lépésszámkorlát: a *tanítóminta* szavainak maximális ismétlési száma
súlyegység: a neuronok közötti kapcsolatok súlyértékén egy beavatkozással végezhető módosítás mértéke

eredmény: *hipotézisfüggvény:* a *tanítóminta*ban szereplő szavak objektív és szubjektív koordinátái közötti kapcsolatot a *pontosság* és a *lépésszámkorlát* által meghatározott feltételek szerint közelítő függvény

- **ismételd**
  - *összesített_hiba* = 0
  - *lépésszám* = 0
  - **ismételd**
    minden t-re $t \in$ tanítóminta
    - állítsd be a *háló* bemeneti rétegében lévő neuronok aktiváltsági állapotát a t objektív koordinátái alapján
    - a *hipotézisfüggvény* alkalmazásával állítsd be a *háló* kimeneti rétegében lévő neuronok aktiváltsági állapotát $a_i = g\left(\sum_{j=0}^{n} W_{j,i} * g\left(\sum_{k=0}^{m} W_{k,j} * a_k\right)\right)$
    - a t szubjektív koordinátái alapján határozd meg a *háló* kimeneti rétegében lévő neuronok *elvárt_aktiváltsági állapot*-át
    - *aktuális_hiba* = azoknak a kimeneti neuronoknak a száma, amelyeknek a hipotézisfüggvény által megállapított aktiváltsági állapota eltér a t szubjektív koordinátái alapján elvárttól
    - *összesített_hiba = összesített_hiba + aktuális_hiba*
    - **ismételd**
      minden c-re $c \in$ kimeneti rétegben lévő neuronok
      - **eljárás** SÚLYMÓDOSÍTÁS (c, *elvárt_aktiváltsági_állapot*, *súlyegység*)
- *hiba = összesített_hiba* / a *tanítóminta* elemeinek száma / a *háló* kimeneti rétegében lévő neuronok száma
- *lépésszám = lépésszám* + 1
- **amíg** *hiba* > 1 - *elvárt_pontosság* **és** *lépésszám* < *lépésszámkorlát*

6.2. algoritmus: A neurális háló tanulását megvalósító függvény algoritmusa

A neurális háló súlymódosítást végző eljárását a 6.3. algoritmus mutatja.

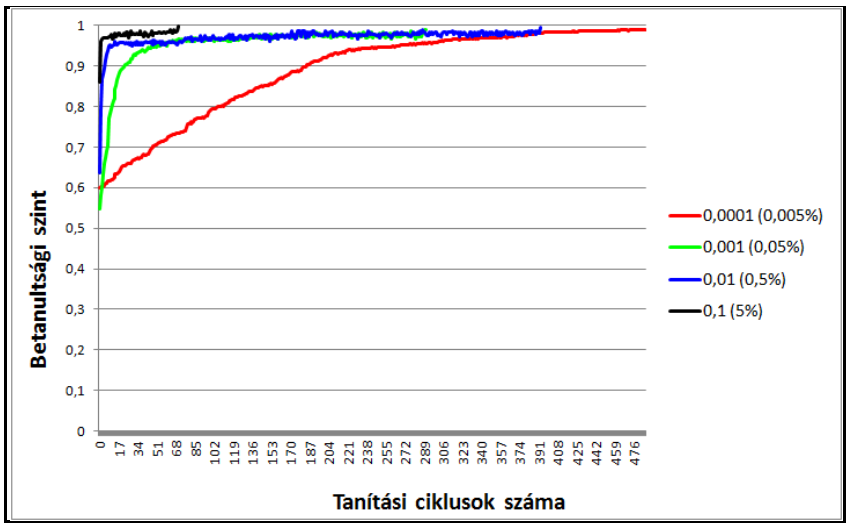
eljárás SÚLYMÓDOSÍTÁS (*neuron*, *aktiváltság*, *súlyegység*)

bemenetek: *neuron:* a neurális hálónak a tanítás alatt álló neuronja
aktiváltság: a *neuron* elvárt aktiváltsági állapota
súlyegység: a neuronok közötti kapcsolatok súlyértékén egy beavatkozással végezhető módosítás mértéke

- **ha** *neuron* aktiváltsági állapota azonos *aktiváltság*-al, **akkor vége**
- **ismételd**
  minden c-re $c \in$ *neuron* bemenetéhez kapcsolódó neuron
  - **ha** *neuron* aktív állapotban van, **akkor**
    - **ha** c aktív állapotban van, **akkor** *súlyegység*-el csökkentsd c és *neuron* közötti kapcsolat súlyát
    - **ha** c és *neuron* közötti kapcsolat súlya pozitív, **akkor** SÚLYMÓDOSÍTÁS (c, *inaktív*, *súlyegység*)
    - **egyébként** SÚLYMÓDOSÍTÁS (c, *aktív*, *súlyegység*)
  - **egyébként**
    - **ha** c aktív állapotban van, **akkor** *súlyegység*-el növeld c és *neuron* közötti kapcsolat súlyát
    - **ha** c és *neuron* közötti kapcsolat súlya pozitív, **akkor** SÚLYMÓDOSÍTÁS (c, *aktív*, *súlyegység*)
    - **egyébként** SÚLYMÓDOSÍTÁS (c, *inaktív*, *súlyegység*)

6.3. algoritmus: A neurális háló súlymódosítást végző eljárásának algoritmusa

A neurális háló aktuális betanultsági szintje a *helyesen osztályozott szavak száma/ osztályozott szavak száma* összefüggéssel határoztam meg. Ez egy [0.0, 1.0] intervallumba eső valós szám.
Az algoritmusok alapján jól követhető a neurális hálózat tanulásának folyamata. A tanulás akkor fejeződik be, ha a tanítómintában szereplő szavak osztályozásának átlagos hibája már nem haladja meg a megengedett mértéket, vagy a tanítóminta szavainak ismételt tanulása már eléri az engedélyezett ismétlési számot. Ez utóbbi feltételre azért van szükség, mivel bizonyos paraméterek esetén bekövetkezhet a hiba értékének oszcillálása. Az oszcilláció hatására a hiba egy adott érték körül ingadozik és ezáltal nem képes a megkívánt szint alá csökkenni. A tanítóminta ismétlésének korlátozása nélkül ilyen esetekben a rendszer sosem lenne képes kijutni a tanulás állapotából. Az algoritmus tesztelése során az elvárt pontosság értékét 0.99-re állítottam be. Ez azt jelenti, hogy a rendszer 99%-os biztonsággal jól osztályozza a tanítómintában lévő szavakat. Az elvárt pontosság növelésével exponenciálisan nő a megkívánt betanultsági szint elérésének időszükséglete, valamint gyakrabban kerül a rendszer az oszcillációba. A 6.5. ábra a tanulás *súlyegység* paraméterének különböző értékei esetén a neurális háló betanultsági szintjében tapasztalt változást mutatja a tanítóminta ismétlési számának függvényében [9].



6.5. ábra: A tanulás súlyegység paraméterének vizsgálata különböző értékek esetén

A tesztek eredménye alapján elmondható, hogy minél kisebb a súlyegység, annál többször kell megismételni a tanítóminta betanítását. Ennek oka, hogy kisebb súlyegység esetén a nem megfelelő aktiváltsági állapotnak egy lépésben való korrigálására csak kisebb mértékben van lehetőség. A 0.0001 súlyegységű tanulás esetén például $\frac{2}{0.0001} - 1 = 19999$ alkalommal kell módosítani egy adott súly értékét ahhoz, hogy az eljuttatható legyen a [-1.0, 1.0] intervallum ellentétes végpontjába. Ez a példa természetesen a legrosszabb esetet mutatja, amikor a súlyok kezdeti értékét előállító véletlenszám-generátor az intervallum egyik végpontját reprezentáló értékkel inicializálja az adott súlyt, míg annak elvárt értéke éppen az intervallum másik végpontjában lenne. Általános esetben elegendő a példában kapott érték felével számolni. Ez azt jelenti, hogy a kezdeti súly pontosan fél intervallumnyi távolságra van az ideális értékétől.
A 6.5. ábrán szemléltetett tesztek alapján megfigyelhető az is, hogy kisebb súlyegység esetén a tanulás sokkal pontosabban tart a célállapot felé. Ezt a kisebb ingadozások alapján lehet megállapítani. Ennek oka, hogy nagyobb értékmódosítás esetén nagyobb annak a veszélye, hogy a kapcsolat súlyát nem lehet kellő pontossággal beállítani az ideális értékére. Egy bizonyos tanítási lépést követően előáll az az állapot, hogy a súly értékének egységnyi növelésével kapott érték már

meghaladja az ideális állapotát, majd csökkentésével az ideális állapot alatt marad. Abban az esetben, ha az ingadozás két végpontjában felvett súlyérték közül az ideális állapothoz közelebbinek a távolsága is nagyobb az elvárt pontosságtól, akkor az adott súly értéke az alkalmazott súlyegység esetén sosem állítható be kellő pontossággal. Ekkor az adott súly a tanulás további lépéseiben végig oszcillációt fog mutatni. Ha ez a jelenség a neurális háló jelentős számú kapcsolatára kiterjed, akkor az a tudásszint értékében is állandó oszcillációt fog mutatni. Ennek elkerülése érdekében a súlyegység értékét érdemes a tanulásra szánt idő figyelembevételével minél kisebb értékre állítani. A neurális háló a betanulását követően a tanítómintában szereplő szavak esetén is megőrizte a paraméterként meghatározott elvárt pontosságot.

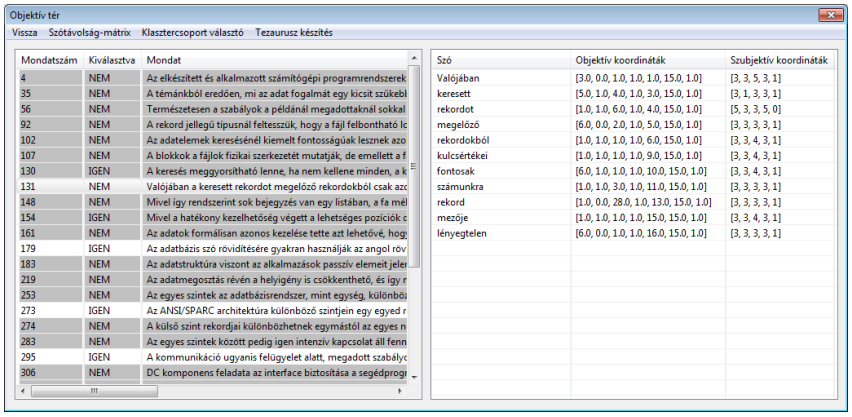
6.2.4. A tanulás eredményeinek újrahasznosítása

Annak érdekében, hogy a neurális háló által megtanult hipotézisfüggvény a későbbi dokumentumok számára is alkalmazható legyen megvalósítottam a szükséges információk elmentését [3]. Statikus szerkezetű neurális hálók esetén ez a hálózat neuronjai közötti kapcsolatok súlyértékeinek elmentését, majd szükség esetén a visszatöltésüket jelenti. Az általam alkalmazott neurális háló struktúrája azonban a 6.2.2. pontban ismertetett okok miatt mindig dinamikusan alkalmazkodik az éppen beolvasott dokumentum szerkezetéhez. Ebből kifolyóan, mivel a bemenetek száma jelentősen függ az éppen elemzés alatt álló dokumentum jellegétől, ezért csak korlátozott mértékben van lehetőség az elmentett súlyok értékeinek egy új dokumentum esetén való alkalmazására. Ha ugyanis az aktuális dokumentumban egy adott objektív koordináta értékeinek reprezentálására több bemeneti neuronra van szükség, mint az elmentett hálóban, akkor a többlet információ tárolására nem lesz elegendő neuron. Az ellenkező esetben, amikor az aktuális dokumentumban egy adott objektív koordináta értékeinek reprezentálására lényegesen kevesebb bemeneti neuron is elegendő lenne, mint amennyit az elmentett hálózat létrehozott ennek a koordinátának a kezelésére, akkor a felesleges neuronok sosem fognak aktív állapotba kerülni. Ha a tanulás során nem aktiválódó (felesleges) neuronok száma jelentős, akkor az jelentősen befolyásolhatja tanulást. Bizonyos megszorításokkal azonban elérhető, hogy az elemzés alatt álló dokumentumok kódolásához szükséges bemeneti neuronok száma ne térjen el jelentős mértékben egymástól. Ennek érdekében a főbb szempontok a következőek:

- a neurális háló korábbi tanulásának adatai csak a korábbi dokumentummal megegyező stratégiával klaszterezett dokumentumok esetén használhatóak (A stratégiák egységesítése jelenleg továbbfejlesztés tárgyát képezi. Megvalósítását a tanulás eredményeként elmentett mondatoknak tanítómintaként való alkalmazásában látom.),
- a dokumentum klaszterezése során a klaszterek maximális átmérőjét definiáló paraméter értékének azonosnak kell lennie az alkalmazni kívánt neurális hálóhoz tartozó dokumentum klaszterezésénél beállítottal,
- a dokumentum szavainak száma nem térhet el jelentősen (a tesztek alapján 20%-tól nagyobb mértékben) az alkalmazni kívánt neurális hálóhoz tartozó dokumentum szavainak számától,
- a dokumentum tématerületének meg kell egyeznie az alkalmazni kívánt neurális hálóhoz tartozó dokumentum tématerületével.

A vizsgálatok során azt tapasztaltam, hogy ezekben az esetekben, a dokumentumok objektív koordinátáinak értékkészlete nem különbözik olyan mértékben egymástól, ami az osztályozás eredményét kimutathatóan befolyásolná. Ezáltal a fenti szabályoknak megfelelő dokumentumok alapján készített neurális hálózat a megszerzett tudásával együtt alkalmazható az új dokumentumok ismételt tanítás nélküli klaszterezésére is.

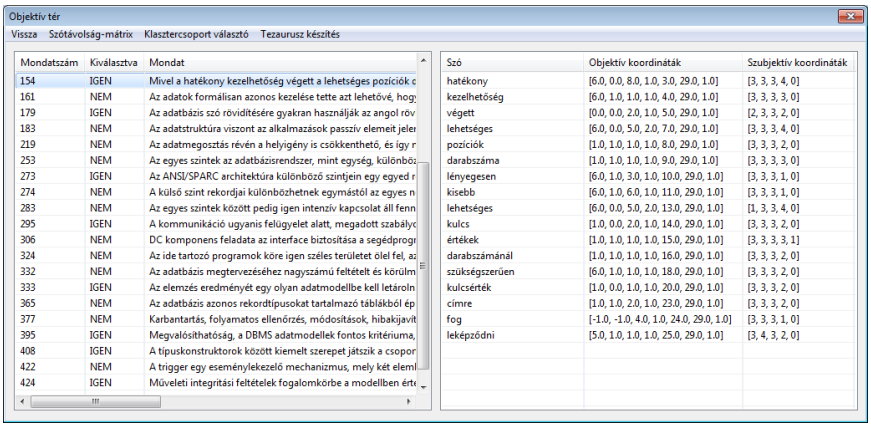
Az ismertetett módszer kiegészítéseként implementálásra került egy másik tudás-újrahasznosítási stratégia is [9]. Ebben nem a betanult neurális hálózat adatait, hanem az osztályozás eredményéül szolgáló mondatokat menti el a rendszer. A megvalósított módszer lényege, hogy az osztályozott mondatok az osztályozás eredményéül szolgáló szubjektív koordinátáikkal együtt egy adatbázisban kerülnek eltárolásra. Az új dokumentumok osztályozása előtt a rendszer összeveti az aktuális dokumentum mondatait az adatbázisban eltároltakkal. Egyezőek találása esetén automatikusan kitölti az aktuális dokumentumban megtalált mondat szubjektív koordinátáit az adatbázisban tárolt párja alapján. Ezt követően a dokumentumban lévő mondatot kiveszi az osztályozandóak közül. Ezeket a mondatokat az implementált alkalmazásban kiszürkített háttérrel jelöltem meg 6.6. ábrán. Így a tanítást már csak egy csökkentett mondatszámú dokumentumon kell elvégezni, mely által csökken az osztályozás időigénye is. Kellő számú dokumentum betanítását követően az azonos tématerülethez tartozó dokumentumokból az adatbázis már jelentős számú mondatot képes felkínálni.



6.6. ábra: A korábbi tanulás eredményeként osztályozott mondatok kivétele

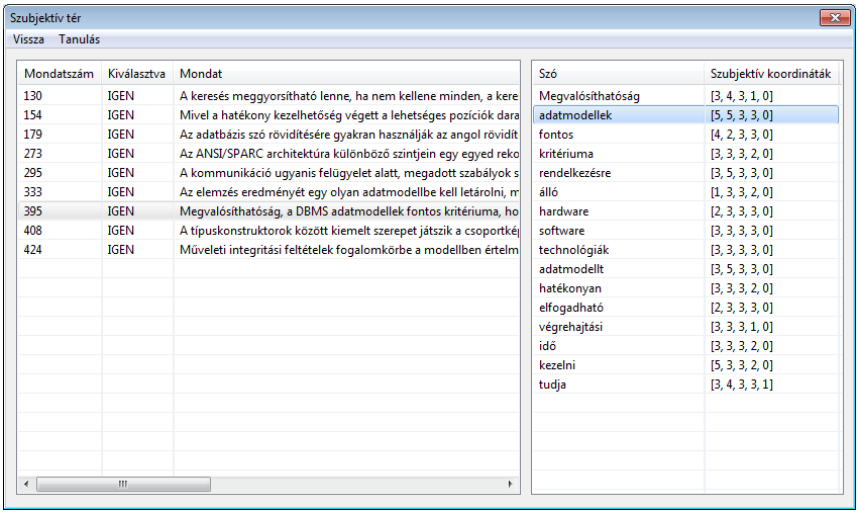
6.3. Az elkészített alkalmazás bemutatása

A 6.2. pontokban ismertetett osztályozási feladatot megvalósító modell alapján elkészítettem egy Java nyelven implementált alkalmazást, mely grafikus interaktív felhasználói felülete révén támogatást nyújt mind a szakértő bevonásával végzett tanítás, mind pedig az osztályozás eredményeinek demonstrálása területén. A 6.7. ábrán látható képernyőn a szakértő kiválaszthatja azokat a mondatokat, amelyek a neurális háló tanítására szolgáló tanítómintát fogják képezni.



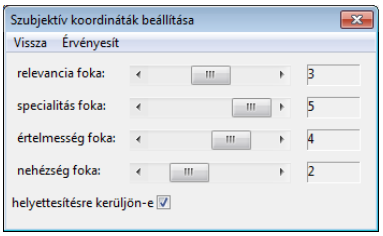
6.7. ábra: A neurális háló tanítására szolgáló tanítóminta mondatainak kiválasztása

A grafikus felhasználói felületen, a tanítómintában szereplő mondatok szavaihoz tartozó szubjektív koordináták beállításához a szakértő a ***Szubjektív tér*** menüpontot választja, ahol már csak a tanítómintába szánt mondatok jelennek meg (6.8. ábra).



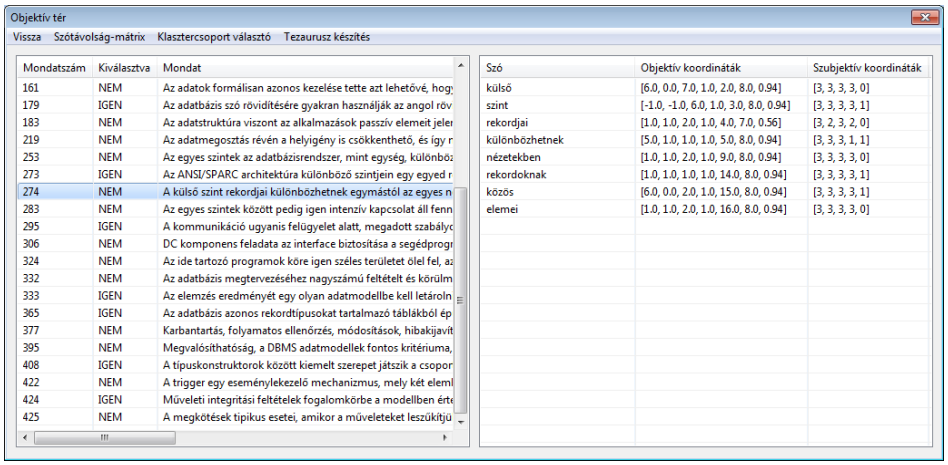
6.8. ábra: A tanítóminta mondatainak kiválasztása

A tanítóminta mondatainak beállítani kívánt szavain jobb egérgombbal kattintva megjelenik a 6.9. ábrán látható dialógusablak, melyben a szakértő megadhatja a kiválasztott szónak a szubjektív térben való elhelyezkedését leíró koordinátákat. A tanítóminta szavainak a szubjektív térben való elhelyezését követően a neurális háló tanítása a ***Tanítás*** menüpont kiválasztásával indítható el [3].



6.9. ábra: A tanítóminta kiválasztott szavának elhelyezése a szubjektív térben

A tanulás befejezésével a neurális háló, a megtanult hipotézisfüggvény alkalmazásával automatikusan kitölti a tanítómintába nem tartozó mondatok szavainak szubjektív koordinátáit 6.10. ábra.



6.10. ábra: A tanítómintába nem tartozó mondatok koordinátáinak automatikus meghatározása

6.4. Az osztályozás hatékonyságának mérésére szolgáló módszerek

Az osztályozó rendszer hatékonyságát, tesztdokumentum alapján határoztam meg. A dokumentum első része (tanítóminta) szolgál az osztályozó rendszer tanítására, a második résszel (tesztadatok), pedig az osztályozó hatékonyságának mérését végeztem el. Az osztályozás hatékonyságának meghatározására több módszer ismert a szakirodalomban. A fejezet további részében ezek közül a legfontosabbak kerülnek elemzésre.
Minden bináris állítás aszerint, hogy az osztályozó algoritmus igaznak, vagy hamisnak minősítette a következő négy kategória valamelyikébe kerül besorolásra [Fawcett, 2003]:

- igaznak minősített igaz állítás (TP: true positive),
- igaznak minősített hamis állítás (FP: false positive),
- hamisnak minősített igaz állítás (FN: false negative),
- hamisnak minősített hamis állítás (TN: true negative).

Az egyes kategóriákat strukturált elrendezésben a 6.4. táblázat mutatja.

	igaz állítások	hamis állítások
igaznak minősített	TP	FP
hamisnak minősített	FN	TN

6.4. táblázat: Bináris állítások helyességének mutatói

A hatékonyság mérésére szolgáló minta minden szavának a megfelelő kategóriába besorolását követően, meghatározható a rendszer *érzékenysége* (R) és *pontossága* (P) a következő egyenlőségek alapján:

$$R = \frac{TP}{TP+FN} \qquad P = \frac{TP}{TP+FP} \tag{6.10}$$

Az érzékenység, az igaznak minősített valóban igaz állítások arányát mutatja meg az összes igaz állítás számához képest. Az érzékenység a rendszernek csak a teljességét jellemzi, a precizitását nem. Ez abban mutatkozik meg, hogy az érzékenység értéke maximálható pusztán a hamisnak minősített igaz állítások számának minimálása révén. Ezáltal az érzékenység maximális értéke biztosítható egy olyan algoritmussal, amely minden állítást igaznak minősít, függetlenül annak valóban igaz voltától.
Ezzel ellentétben a pontosság, a valóban igaz állítások arányát, az igaznak minősített állítások számához képest reprezentálja. A pontosság, ellentétben az érzékenységgel, a rendszer precizitását jellemzi figyelmen kívül hagyva annak teljességét. Ez abban mutatkozik meg, hogy az érzékenység mértéke maximálható pusztán az igaznak minősített hamis állítások számának minimálása révén. Ezáltal a pontosság maximális értéke biztosítható egy olyan algoritmussal, amely csak azokat az állításokat minősíti igaznak, amelyek igazságáról teljesen meg van győződve. A pontosság szempontjából lényegtelen, hogy az algoritmus hány valójában igaz állítást minősít hamisnak. Egy jó osztályozó algoritmus egyszerre törekszik mind az érzékenység, mind pedig a hatékonyság értékének maximálására [Tikk, 2007].
Az érzékenység és a pontosság értékeinek adott cél szerinti összehangolására szolgáló módszer az *F-mérték* [Rosell, 2004]. Ezzel a hatékonysági mutatóval az említett két tényező súlyozott mértékben reprezentálható. Ezáltal ötvözhetőek az előző két módszer előnyös tulajdonságai. Az F-mérték meghatározása a (6.11) képlet alapján történik [Rijsbergen, 1979]:

$$F = \frac{1}{\alpha * \frac{1}{P} + (1-\alpha)1/R} = \frac{(\beta^2+1)PR}{\beta^2 P + R} \tag{6.11}$$

ahol az α, illetve β paraméter értéke a két mérték súlyát határozza meg. A gyakorlatban általában a kiegyensúlyozott F-mértéket használják, amikor mindkét tényező egyenlő súllyal szerepel, ekkor $\beta = 1$, illetve $\alpha = 1/2$.

Jelen fejezetben terjedelmi okok miatt csak ennek a három hatékonysági mutatónak a hatásait vizsgálom. A munkám során azonban meghatározásra kerültek még az osztályozás hatékonyságát mutató következő paraméterek is, melyek a disszertáció *B.1* Függelékben megtekinthetőek [Tikk, 2007]:

- szabatosság: $A = TP + \frac{TN}{TP+FP+TN+FN}$;
- hamis pozitív arány: $FPR = \frac{FP}{FP+TN}$;
- specifikusság: $SPC = 1 - FPR$;
- negatív prediktív érték: $NPV = \frac{TN}{TN+FN}$;
- hibás találatok aránya: $FDR = \frac{FP}{FP+TP}$.

6.4.1. A hatékonyság mérésének eredményei

Az osztályozás hatékonyságának mérése során a szakértő meghatározza a kijelölt szövegből tanításra használandó mondatokat. Ezek a mondatok alkotják a tanítómintát, a szöveg többi mondata pedig a tesztmintát. A tanítóminta mondatainak számát célszerű a teljes szöveg mondatszámához képest 50% körüli értékre választani. Ilyen arány esetén érhető el ugyanis a legjobb hatásfokú tanulás. A szakértő a tanítómintában lévő összes mondat esetén megadja a mondat szavainak szubjektív koordinátáit. Ezt követően a neurális hálózat súlyértékeit véletlenszerűen választott kezdőértékről az algoritmus folyamatosan növeli, illetve csökkenti annak értekében, hogy a bemenetre kapcsolt szavak hatására a kimeneten jelentkező érték minél jobban megközelítse a szakértő által meghatározott értékeket. A beállított mértékű betanultsági szint elérésekor a tanulás befejeződik. Ezt követően a neurális hálózat a tesztminta elemeire hajtja végre a tanítómintánál megtanult átviteli formulát. Ezzel meghatározva a szakértő által nem definiált szubjektív koordinátákat is. Ezt követően meghatározásra kerülnek az osztályozás hatékonyságának mérésére definiált mutatók értékei, melyeket az algoritmus az *„osztalyozas.txt”* szöveges fájlban tesz elérhetővé a szoftver kezelői számára. A fájlban a hatékonyság mérésére szolgáló mutatók mind az öt szubjektív koordinátára külön-külön, valamint az egész neurális hálózatra vonatkozóan egységesen is eltárolásra kerültek. Az öt különböző értékkel rendelkező szubjektív koordináták bináris kódolása révén minden érték egy külön állítást reprezentál. Ezáltal minden szubjektív koordináta esetén az osztályozott állítások száma az *osztályozandó szavak száma * a szubjektív koordináta lehetséges értékei* összefüggés alapján számítható.

Az osztályozó jóságának meghatározására három tesztet készítettem el, amelyek alapjául az *Adatbázisrendszerek* című tantárgy tananyaga szolgált. A tesztlapok 30, 40 és 50 tesztkérdést tartalmaztak. A 6.5. táblázat az algoritmus eredményeinek kiértékelésével meghatározott értékeit tartalmazza.

Tesztek	Állításosztályok				
	TP	FP	TN	FN	∑
Teszt1 (30)	775	10	2676	15	3476
Teszt2 (40)	1056	72	3736	64	4928
Teszt3 (50)	1050	167	3964	165	5346

6.5. táblázat: Az osztályozás pontosságának alap-mutató értékei

Az osztályozás pontosságának számított főbb mutató értékeit a 6.6. táblázat foglalja össze.

Tesztek	Osztályozó algoritmus		
	Érzékenység (R) %	Pontosság (P) %	F-mérték %
Teszt1 (30)	98,10	98,72	98,41
Teszt2 (40)	94,28	93,61	93,95
Teszt3 (50)	86,41	86,27	86,34

6.6. táblázat: Az osztályozás pontosságának számított mutató értékei

A legjobb eredményeket a 30 kérdést tartalmazó feladatlap tesztelése során kaptam. A 6.6. táblázatban jól látható, hogy 30 kérdést tartalmazó feladatlap esetén az *érzékenység* értéke 98,1%. Ez azt jelenti, hogy a rendszer az igaz állításokat az esetek mindössze 1.9%-ában minősített hamisnak. A *pontosság* értéke is ebben az esetben a legnagyobb 98,72%. Ez azt jelenti, hogy a rendszer által igaznak minősített állítások az esetek mindössze 1.28%-ában voltak valójában hamisak. Az *F-mérték* értéke az *érzékenység* és a *pontosság* súlyozott értékét mutatja $\alpha = 0,5$ érték esetén, amely ebben az esetben 98,41%.

6.5. Az osztályozás eredményeinek összefoglalása

A dokumentum szavainak osztályozását végző algoritmus eredményeként megvalósításra került a szavak szubjektív vetületét reprezentáló koordináták automatizált feltárása, valamint megalapozott iránymutatást értem el a szöveg mondataiból kérdésként kiemelendő szó meghatározása területén. Ezek képezik alapját a kérdésként kiemelt szó lehetséges helyettesítésére felkínálandó szavak automatikus generálásának, valamint a kérdésre adott válasz értékelésének. Az osztályozás hatékonyságának mérésére szolgáló módszerek közül meghatároztam a főbb mutató értékeket: érzékenység, pontosság és F-mérték.

7. fejezet

Válaszgeneráló mintarendszer

7.1. A kérdésekre adható válaszlehetőségek automatizált előállítása

Az automatizált kérdésgeneráló rendszer feladata, hogy meghatározza a feltett kérdésekre adható lehetséges válaszokat. A kérdéseket megválaszoló személynek ezek közül a lehetséges válaszok közül kell kiválasztania azt az egyet, amelyiket leginkább alkalmasnak találja a mondatból kivett szó helyére. A válaszok előállításának automatizálása során azt az irányelvet tartottam szem előtt, hogy egzakt módszerekkel mérhető legyen az általuk képviselt válasz jósága. A válasz jósága mérhető a referenciaszó és a válaszként megadott szó szemantikai távolságával:

$$d_{sz} = d(\omega_r, \omega_e)$$

ahol: d_{sz} a szemantikai távolság, ω_r a referenciaszó jósága, ω_e a válaszlehetőségek jósága.
Ehhez olyan szavak megtalálása volt a feladat, melyeknek a kivett szótól való távolsága egyenlő lépésközökkel folyamatosan növekszik. Ezáltal lehetőség nyílt arra, hogy:

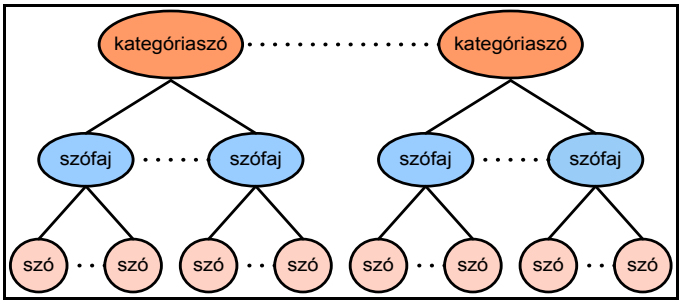
- jól mérhető módon megállapítható legyen a válaszként visszatételre javasolt szó távolsága a mondatból kivett szótól,
- a lehetséges válaszok egyenletesen fedjék le a forrásként szolgáló dokumentum szavai által meghatározott teret.

A válasz jósága annál magasabb, minél közelebbi szót választ az automatikusan előállított válaszlehetőségek közül a mondatból kérdésként kiemelésre került szóhoz. A távolság definíciója függ attól, hogy a szakértő melyik koncepciót választotta az 5. fejezetben ismertetett klaszterezési alternatívák közül. A szélességi, a mélységi valamint a legjobbat először keresési algoritmusokkal implementált $\frac{f_{i,j}}{max(f_i,f_j)}$ képlet alapján számított távolság esetén egy szó annál közelebb van a mondatból kivett szóhoz, minél több olyan mondat van a forrásként szolgáló dokumentumban, amelyben mindketten szerepelnek. Az elvégzett tesztek eredményei azt mutatták, hogy a pusztán statisztikai alapokon nyugvó távolságmérési módszer nem minden esetben tükrözi a szavak szemantikai értelemben vett távolságát [Dudás, 2011]. Másik probléma, hogy az algoritmus által felkínált lehetséges válaszalternatívák közül a forrásként szolgáló dokumentum mélyebb ismerete nélkül is a legtöbb kérdésre viszonylag könnyen meghatározható volt a helyes válasz pusztán a magyar nyelvtan szabályainak ismerete alapján. Ennek oka, hogy a válaszként javasolt szavak esetén nem volt kritérium a kérdésként kivett szóval megegyező szófaj, illetve ragozottság biztosítása. Ennek kapcsán a válaszlehetőségeket előállító modell továbbfejlesztése három ponton valósul meg.

1. minden válaszalternatíva szótövezett alakban jelenjen meg a válaszadó előtt,
2. az alkalmazott statisztikai távolságmérés mellett a kérdésként kiemelésre került szóval fogalmi szinten azonos szó is szerepeljen a lehetséges válaszalternatívák között,
3. a fogalmi szinten azonos szó szófaja egyezzen meg a kérdésként kiemelt szó szófajával.

Az 1. pont megvalósítása révén biztosíthatóvá vált, hogy pusztán a magyar nyelvtan szabályaira való támaszkodás ne legyen elegendő a helyes válasz megtalálásához. A 2. és 3. pontok azonban egy teljesen új megközelítési koncepciót nyitnak a szavak távolságának mérésére. Ezáltal a távolságok egy fa struktúrával váltak modellezhetővé, ahol a fa levél-csomópontjai reprezentálják a lehetséges válaszok előállításához használható szavakat. A fa többi csomópontja definiálja azokat a hierarchikus kategóriákat, melyekbe a levél-csomópontokban lévő szavak besorolásra kerülnek.

Ennek a hierarchikus koncepciónak az alapján két szó távolsága egyenesen arányos azoknak a kategóriáknak a számával, amennyit felfelé kell haladni a fában az egyik szótól a másik szóig való eljutáshoz. Munkám során a szavak közötti távolságnak ezt a reprezentálási formáját *fogalomhierarchiának* neveztem el. A megvalósításra került modellben három szintű fogalomhierarchiát építettem fel. A hierarchia legalsó szintjén helyezkednek el a kérdések előállításához használt dokumentumnak az elemzésből kizárásra nem kerülő egymástól különböző szavai szótövezett alakban. A szavak feletti első kategorizálási rétegben a szavak szófaj szerinti besorolása valósul meg. Ez a réteg gondoskodik arról, hogy az azonos szülő-csomóponthoz tartozó szavak szófaja megegyezzen. A fogalomhierarchia legfelső szintjén az azonos szófajhoz tartozó szavak az ősszavuk alapján kerülnek tovább kategorizálásra. A szavak kategória szavai határozza meg azt a fogalmi tématerületet, amelyhez a fa levél-csomópontjaiban lévő szavak szemantikai szempontból tartoznak. Kategóriaszó lehet például: természettudomány, társadalomtudomány, nyelvtudomány stb.
A kidolgozott háromszintű fogalomhierarchia felépítését a 7.1. ábra mutatja.



7.1. ábra: A háromszintű fogalomhierarchia felépítése

A fogalomhierarchia alkalmazása révén, a kérdésként kivételre kerülő szavakhoz a modell nem pusztán a klaszterezési algoritmusokkal kapott távolságadatok alapján, hanem a fogalomhierarchia alapján mérhető távolságok alapján is képes a válaszlehetőségek automatikus előállítására. Ezzel a módszerrel lényegesen pontosabban határozható meg a válaszadó személynek a választott dokumentumra épülő tudása, mivel a fogalmi egyezőségre épülő alternatívák megjelenése jelentősen megnehezíti a helyes válasz megtalálását.

7.1.1. A fogalomhierarchiára épülő szótár felépítése

Mivel a nyelvtani elemzést végző szótövező alkalmazás nem képes a szavakhoz tartozó fő kategóriák meghatározására, ezért a fogalomhierarchia legfelső szintjén lévő kategóriaszó megadását manuálisan végeztem el. Ehhez a fogalomhierarchia felépítéséhez szükséges adatokat Excel táblázatban gyűjtöttem össze. Ennek a táblázatnak a *fogalomtábla* elnevezést adtam. A fogalomtábla első oszlopában a fogalomhierarchia levél-csomópontjait képező szavak kerültek felvételre. Ezekkel a szavakkal szemben semmilyen megkötöttséget nem támasztottam. Ennek oka, hogy ezáltal könnyen megvalósíthatóvá vált a kérdések előállítására használható különböző dokumentumok szavaival való automatikus feltöltés. Ezen kívül lehetővé tette, hogy a fogalomtáblában szereplő szavak függetlenek legyenek a kérdések előállítására használt dokumentumok szavaitól. Ezáltal ugyanaz a fogalomtábla használhatóvá vált több egymástól független dokumentum esetén is. A fogalomtábla második oszlopában kerültek eltárolásra az első oszlopban szereplő szavak kategóriaszavai. Egyelőre nem ismert olyan algoritmus, amely – akár adatbázisokból való kikereséssel – képes lenne magyar szavak kategóriaszavainak előállítására. Ezért a táblázat második oszlopának feltöltését manuálisan végeztem el. A feltöltés során

igyekeztem közérthető kifejezéseket használni a kategóriaszavak jelölésére, megadván annak a tudományterületnek a nevét, amihez az adott szó tartozik.
A kategóriaszavak pontos megnevezésére azonban a modell elvárt működése szempontjából nem volt szükség. A kategóriaszó jelölésére alkalmazott kóddal szemben lényegében csak annyi a megkötés, hogy azonos legyen minden olyan szó esetén, melyek ugyanahhoz a kategóriaszóhoz tartoznak, és különböző legyen azon szavak esetén, melyek eltérő kategóriaszóhoz tartoznak. A fogalomtábla harmadik oszlopa tartalmazza a szavak szófaját. Ez képezi a háromszintű fogalomhierarchia középső rétegét. A harmadik oszlop kitöltése nem kötelező, mivel ezt az információt az alkalmazott szótövező alkalmazásból is ki lehet nyerni. Megadásával azonban jelentősen kiterjeszthető a fogalomhierarchia használhatósága olyan dokumentumok esetén is, amelyek szavai nem tartoznak az általános szókincshez. Az ilyen szavak esetén ugyanis a szótövezést végző alkalmazás nem képes a szófaj meghatározására. Ezenfelül a harmadik oszlop kitöltése abban az esetben is elősegíti a fogalomhierarchia használhatóságát, amikor a szótövező alkalmazás több egyformán esélyes szófajt is talál bizonyos szavakhoz. Ilyen esetekben az adott szóhoz a fogalomtábla harmadik oszlopában megadott szófaj kerül felhasználásra, a szótövező alkalmazás által javasolt legelső szófaj-alternatíva helyett. Bár a fogalomhierarchia alapjául szolgáló fogalomtábla általánosan használható bármilyen dokumentumra épülő kérdésgenerálás során a fogalomhierarchiát minden dokumentum esetén egyedileg kell felépíteni. Ennek oka, hogy a szótövező alkalmazással feltárt minden információt a szakértőnek módjában áll megváltoztatni. Ezáltal az adott dokumentum elemzése már nemcsak az objektíven előre definiálható fogalomtábla és a szótövező alkalmazás révén valósul meg, hanem figyelembe kell venni a szakértő szubjektív véleményét is. Ez pedig dokumentumonként (kisebb mértékben szakértőnként) eltérő lehet.
A fogalomhierarchia automatikus felépítését a 7.1. algoritmus mutatja.

Lépés-szám	**Cselekvés**
1.	Jelölje s a $szavak$ lista legelső szavát.
2.	Jelölje f az s szó szófaját ($f \in szófajok$), valamint jelölje t az s szó szótövét ($s \in szótövek$).
3.	Ha a $fogalomtábla$-nak létezik olyan sora, amelynek első oszlopában lévő szó azonos s-el vagy t-vel, akkor jelölje i a $fogalomtábla$-nak ezt a sorát. Egyébként i értéke ismeretlen.
4.	Ha i értéke ismeretlen, akkor menj a 17. lépésre!
5.	Jelölje *ő* a $fogalomtábla$ i-vel jelölt sorában tárolt kategóriaszót.
6.	Ha a $fogalomhierarchia$-ba szerepel az *ő*-vel jelölt kategóriaszó, akkor menj a 8. lépésre!
7.	Vedd fel a $fogalomhierarchia$-ba az *ő*-vel jelölt kategóriaszót!
8.	Ha f-et nem a szakértő adta meg akkor menj a 10. lépésre!
9.	Ha a $fogalomhierarchia$-ban az *ő*-vel jelölt szóból még nem nyílik f-et reprezentáló ág, akkor hozz létre a $fogalomhierarchia$-ban az *ő*-vel jelölt szóhoz egy f-et reprezentáló ágat!
10.	Ha a $fogalomtábla$ i-vel jelölt sorában nincs megadva a szófaj, akkor menj a 12. lépésre!
11.	Ha a $fogalomhierarchia$-ban az *ő*-vel jelölt kategóriaszóból még nem nyílik olyan ág, ami a fogalomtábla i-vel jelölt sorának lévő szófajt reprezentálja, akkor hozz létre a $fogalomhierarchia$-ban az *ő*-vel jelölt kategóriaszóhoz egy olyan ágat, ami a $fogalomtábla$ i-vel jelölt sorában lévő szófajt reprezentálja!
12.	Ha f több szófaj alternatívát is tartalmaz, akkor menj a 14-re!
13.	Ha a $fogalomhierarchia$-ban az ő-vel jelölt kategóriaszóból még nem nyílik f-et reprezentáló ág akkor hozz létre a $fogalomhierarchia$-ban az ő-vel jelölt főkategóriához egy f-et reprezentáló ágat! Menj a 15-re!
14.	Ha a $fogalomhierarchia$-ban az ő-vel jelölt kategóriaszóból még nem nyílik az f-el jelölt szófaj-alternatívák közül a legelsőt reprezentáló ág, akkor hozd létre a $fogalomhierarchia$-ban az *ő*-vel jelölt kategóriaszóhoz az f-el jelölt szófaj-alternatívák közül a legelsőt reprezentáló ágat!

Lépés-szám	Cselekvés
15.	Jelölje á a $fogalomhierarchia$-ban az ő-vel jelölt kategóriaszóból nyíló f-et reprezentáló ágat.
16.	Ha az á-val jelölt ágból nem nyílik a t-vel jelölt szótövet reprezentáló csomópont, akkor hozz létre az á-val jelölt ágból nyíló t-vel jelölt szótövet reprezentáló csomópontot!
17.	Ha a $szavak$ listának már minden eleme megvizsgálásra került, akkor menj a 19. lépésre!
18.	Jelölje s a $szavak$ lista soron következő szavát! Menj a 2. lépésre!
19.	Vége.

7.1. algoritmus: A fogalomhierarchia felépítését végző algoritmus

A 7.1. algoritmus bemenetei és kimenete:

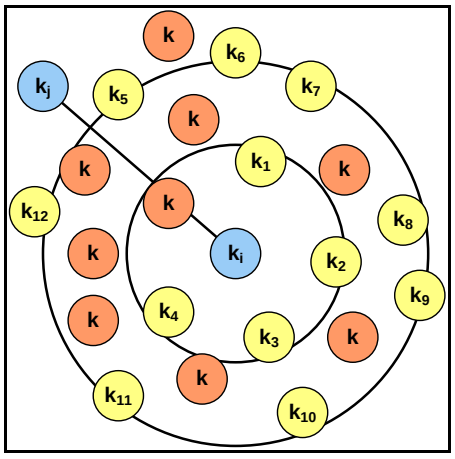
- A $szavak$ halmaza, amely tartalmazza a kérdések előállítására szolgáló dokumentum összes olyan szavát, melyek nem kerültek kizárásra az elemzésből.
- A $szófajok$ halmaza, valamint a $szótövek$ halmaza, amelyek tartalmazzák a $szavak$ halmaz elemeinek szófaját, illetve szótövét. Ezeknek a halmazoknak az elemei a szótövező alkalmazás, illetve a szakértő döntései alapján kerültek meghatározásra, illetve hozzárendelésre a $szavak$ halmaz elemeihez.
- A $fogalomtábla$, mely az aktuálisan vizsgált dokumentumtól függetlenül kezelhető és szó – kategóriaszó – szófaj struktúrában tartalmazza a szavakat a hozzájuk meghatározott kategóriaszóval valamint szófajjal.
- Az algoritmus kimenete a $fogalomhierarchia$, mely a kérdések előállításához tartalmazza a fogalmak háromszintű fa modelljét.

A 7.1. algoritmus alapján látható, hogy ha a kérdések előállítására használt dokumentum soron következő szava nem található meg a fogalomtáblában, akkor az a szó nem kerül felvételre a háromrétegű fogalomhierarchiába sem. Ezen kívül érdemes megfigyelni azt is, hogy ha a soron következő szó szófaját a szakértő adta meg, akkor függetlenül a szótövező alkalmazás, illetve a fogalomtábla javaslataitól a soron következő szó a szakértő által javasolt szófajhoz kerül besorolásra a fogalomhierarchiában. Prioritás szerint ezt követi a fogalomtáblában szereplő szófaj. Ezt csak akkor veszi figyelembe az algoritmus, ha a szakértő nem határozta meg a soron következő szó szófaját. A szótövező alkalmazás által javasolt szófajt csak abban az esetben vizsgálja meg az algoritmus, ha a fogalomtáblában sem talált a soron következő szóhoz szófajt. Több lehetséges szófaj-alternatíva közül minden esetben a legelső kerül alkalmazásra. Nem azért mert ez a legjobb, hanem azért, mert jelenleg nem áll rendelkezésre olyan módszer, ami több lehetséges szófaj esetén a véletlenszerű választástól pontosabban lenne képes kiválasztani a megfelelőt. Abban az esetben, ha a soron következő szó szófaját a szótövező alkalmazás sem ismerte fel, akkor az adott szó hiába található meg a fogalomtáblában nem kerül rögzítésre a fogalomhierarchiában. Az algoritmusból az is látható, hogy a fogalomhierarchiában nem a dokumentum szavai, hanem ezeknek a szavaknak a szótöve kerül mindig eltárolásra.

7.1.2. Válaszlehetőségek automatikus előállítása

A kérdések előállítására választott dokumentum szavainak klaszterezését, valamint a háromrétegű fogalomhierarchia felépítését követően minden információ rendelkezésre áll a mondatokból kérdésként kiemelésre került szavak helyére tehető válaszalternatívák automatikus előállítására. Ennek első lépéseként az algoritmus meghatározza a kérdésként kivett szó szófaját, valamint szótövét. Ezeket az információkat elsősorban a szakértő véleménye alapján határozza meg. Csak ha a szakértő nem nyilatkozott az adott szóval kapcsolatban, akkor vizsgálja a fogalomtábla, valamint a szótövező alkalmazás által kínált lehetőségeket.

A kidolgozott modellben minden kérdésként kivett szó helyére öt lehetséges alternatívát kínál fel az algoritmus. A válaszadó személynek ezek közül az alternatívák közül kell kiválasztania azt az egyet, amelyiket a legmegfelelőbbnek tartja a kivett szó helyettesítésére. A kérdésre adható legjobb választ természetesen az a szó képezi, ami a mondatból kivételre került. Ezért ennek a szónak mindenképpen szerepelnie kell az automatikusan előállított öt lehetőség között. Abban az esetben, ha a mondatból kivett szó szótöve szerepel a fogalomhierarchiában, valamint van ezzel a szóval azonos szinten lévő más szó is ezen a hierarchiaszinten, akkor az algoritmus véletlenszerűen választ egyet ezek közül. Ha sikerül a fogalomhierarchiában ilyen szót találni, azzal biztosítható, hogy a lehetséges válaszok között legyen legalább egy olyan, amely ugyanahhoz a fogalmi szinthez tartozik, mint a mondatból kivett szó. Ezen felül ez az alternatíva teljesíti a válasz-alternatívákkal szemben támasztott 1. illetve 3. pontokban rögzített kritériumokat is, mivel a kérdésből kivett szóval azonos szófajhoz tartozik, valamint szótövezett alakban szerepel. A fennmaradt három (illetve, ha nem sikerült a fogalomhierarchiában megfelelő alternatívát találni, akkor négy) további alternatíva előállítása a dokumentum szavainak klaszterezéséhez alkalmazott távolság alapján történik. Bár két szó távolsága a klaszterezés elvégzése nélkül is meghatározható, a klaszterek képzésével a feladat elvégzése szempontjából további két lényeges tulajdonság kezelésére nyílik lehetőség. Egyrészt biztosítható általa, hogy egy adott szóhoz megfelelő távolságra lévő másik szó megtalálásához ne kelljen az adott szó összes többi szóhoz való távolságát megvizsgálni. Ehelyett a szavak az őket tartalmazó klaszterekkel kerülnek helyettesítésre és a távolságok pusztán a klaszterek között kerülnek meghatározásra. Mivel a szavak száma nagyságrendekkel nagyobb a klaszterek számánál, így az adott szóhoz kívánt távolságra lévő másik szó meghatározásának költsége is nagyságrendileg csökkenthető, a szavanként való távolságmeghatározás költségéhez képest. A klaszterezésnek, a feladatom szempontjából másik fontos tulajdonsága, hogy az egy klaszteren belüli szavak egymás kváziszinonimáinak tekinthetőek. Ezt a szinonimasági viszonyt a szavak távolságával definiáltam.
A még meghatározandó alternatívák a következő módon kerülnek felvételre. Az algoritmus meghatározza azt a klasztert, amely a mondatból kivett szót tartalmazó klasztertől a legtávolabb helyezkedik el. Ennek a klaszternek a kivett szót tartalmazó klasztertől való távolságát annyi egyenlő részre osztja fel, ahány alternatíva meghatározására még szükség van (három vagy négy). Ezzel kerül meghatározásra az a távolságegység, amilyen lépésekkel halad az algoritmus a kivett szót tartalmazó klasztertől a hozzá legtávolabbi klaszterig. Ezt követően az algoritmus sorra veszi azokat a klasztereket, amelyek a kivett szót tartalmazó klasztertől 0, 1, ..., *n* távolságegységnyi távolságra vannak, *n* jelöli a még meghatározandó válaszalternatívák számát. Minden vizsgálat során a kivett szót tartalmazó klasztertől adott távolságra lévő klaszterek egy kör kerületén helyezkednek el. A kör középpontját a kivett szót tartalmazó klaszter jelöli ki, sugara pedig azonos a vizsgált távolsággal. Abban az esetben, ha az így kapott kör kerületét több klaszter is metszi, akkor az algoritmus azt választja ki közülük, melynek középpontja (a klaszterezés során definiált távolságképzési kritérium szerint) a legközelebb esik a kör kerületéhez. Ha a módszer alkalmazásával sikerül olyan klasztert találni, amely metszi az aktuális távolsággal definiált sugarú kör kerületét, akkor az algoritmus véletlenszerűen választ egy szót a megtalált klaszterből. Minden ilyen kiválasztott szó egy újabb válaszalternatívát jelent. Amennyiben az adott kör kerületén egyetlen klaszter sem található akkor az algoritmus dinamikusan növeli, illetve csökkenti a kör sugarát mindaddig, amíg megfelelő szót nem talál. A hiányzó válaszalternatívák meghatározásához a 7.2. ábrán látható módon az algoritmus sorra megvizsgálja a kérdésként kivett szót tartalmazó klasztertől 0, 1, ..., *n* távolságra lévő klasztereket.



7.2. ábra: A hiányzó válaszalternatívák meghatározása a szavak távolsága alapján

A mondatokból kérdésként kiemelt szavak helyére automatikusan előállított válaszalternatívák meghatározását végző algoritmus a fogalomhierarchia, illetve a klaszterek távolságai alapján öt szót állít elő minden kiemelt szó lehetséges helyettesítésére. Ezeknek az alternatíváknak az előállítását a 7.2. algoritmus foglalja össze.

Alternatíva sorszáma	**Alternatíva előállítása**
1.	A kérdésként kivett szó szótövezett alakja.
2.	A kérdésként kivett szóval azonos kategóriaszóhoz és szófajhoz tartozó szó szótövezett alakja (ha nincs ilyen szó, akkor ez a hely a soron következő alternatívával kerül feltöltésre).
3.	A kérdésként kivett szóval azonos klaszterben lévő másik szó (ha nincs ilyen szó, akkor ez a hely a soron következő alternatívával kerül feltöltésre).
4.	A kérdésként kivett szót tartalmazó klasztertől legtávolabb lévő klaszter távolságát a még hiányzó válaszalternatíváknak megfelelő számú egyenlő részre osztva a soron következő távolságra lévő klaszterből véletlenszerűen választott szó szótövezett alakja (ha nincs ilyen szó, akkor az elvárt távolság értéke dinamikusan növelésre, illetve csökkentésre kerül mindaddig, amíg a megfelelő szó megtalálásra nem kerül).

7.**2. algoritmus:** A válaszalternatívák automatikus előállításának folyamata

Az így előállított válaszalternatívákat az algoritmus összekevert sorrendben tárja a kérdések megválaszolója elé.

7.2. A válaszgenerálás területén elért eredményeim összefoglalása

Az 5. fejezetben bemutatott klaszterezési algoritmusok kidolgozása azt a célt szolgálta, hogy megvalósítható legyen általuk a jelen fejezetben ismertetésre került a kérdésekre adható lehetséges válaszok automatikus előállításának folyamata. A klaszterezésből nyert távolság, a távolság alapú szinonima információk, valamint a háromrétegű fogalomhierarchia alapján előállítottam a kérdésként kivett szavakra adható öt lehetséges válaszalternatívát. A lehetséges válaszok előállítását aszerint végeztem, hogy egzaktul mérhető legyen a helyes választól való távolságuk, a válaszok jóságának pontozhatósága érdekében. Az algoritmus továbbfejlesztésével megoldottam, hogy a fogalmi szinten azonos szó szófaja egyezzen meg a kérdésként kiemelt szó szófajával, valamint a szavak ragozott alakja helyett azok szótövezett alakban jelenjenek meg a válaszadó előtt. Eredményként sikerült olyan intelligens alkalmazást készítenem, amely révén automatizálható a válaszok előállításának folyamata.

8. fejezet

A kérdésgeneráló mintarendszer implementálása és értékelése

Az ismertetett módszerek kidolgozásának és algoritmusok formájában való implementálásának célja az volt, hogy rendelkezésre álljanak azok az információk, melyek alapját képezik a munkám fő célkitűzésének számító AQG rendszer elkészítésének. A kidolgozott és implementált módszerek révén szöveges dokumentumokhoz elvégezhető a szöveg szavainak paraméterként adott szabályok szerinti klaszterekbe szervezése. A klaszterezés eredményeként a klaszterekbe került szavak teljesítik az előre definiált távolságképzési kritériumot. Azaz, minden klaszter esetén a klaszter bármely két szavának távolsága kisebb egy paraméterként megadott ε értéktől.
A mondatokból kérdésként kiemelendő szó a 6. fejezetben ismertetett osztályozási módszerrel kerül meghatározásra. A bemutatásra került neurális hálózat „*kérdésként kiemelésre kerüljön-e*" elnevezésű kimenete bináris értékkészlettel rendelkezik. A mondatok azon szavai esetén, melyeket az osztályozó modell kérdésként kiemelhetőnek javasolja, ez a szubjektív koordináta igaz értéken szerepel. A mondat többi szava esetén ez a koordináta hamis értéket reprezentál. Az osztályozást végző algoritmust úgy valósítottam meg, hogy ezen kérdésként kiemelhetőséget meghatározó szubjektív koordináta esetén a bináris kimeneti értéket előállító neuron bemeneti függvénye által szolgáltatott értéke is kinyerhető legyen. Ezáltal lehetőség nyílik egy adott mondat esetén több kiemelhetőnek minősített szó esetén ezeknek a szavaknak rangsorba állítására és a rangsor szerinti legnyomatékosabban javasolt szó kiválasztására.
Jelen fejezet bemutatja, hogy az ismertetett módszerekkel nyert információk alapján milyen elvek és módszerek kerültek alkalmazásra a kiválasztott mondatokból kérdésként kiemelendő szó automatikus meghatározásához. A fejezetben elemzem az automatizált kérdésgeneráló mintarendszer gyakorlati működését és értékelem az eredményeket.

8.1. Kérdések automatizált előállítása

Az automatizált kérdés- illetve válaszgenerálás működését befolyásoló főbb paraméterek:

- ε: egy klaszteren belül a szavak maximális távolsága,
- d: két szó távolságát leíró függvény (eltérő a szógyakorisággal),
- N: a klaszterezendő szavak száma,
- A: az elemzésre kerülő mondatok annotációja,
- M: a dokumentum annotált mondatai,
- SZ_e: az elemzésre kerülő szavak halmaza,
- SZ_t: a neurális hálózat tanítására használt szavak halmaza ($SZ_t \epsilon SZ_e$),
- K_o: az elemzésre kerülő szavak objektív koordinátái,
- K_{sz}: az elemzésre kerülő szavak szubjektív koordinátái,
- α: a neurális hálózat élein egy lépésben végezhető súlymódosítási mértéke,
- m: a bemeneti rétegben lévő neuronok száma,
- n: a rejtett rétegben lévő neuronok száma,
- k: a kimeneti rétegben lévő neuronok száma,
- g: a neurális hálózat neuronjainak aktiváltsági állapotát leíró függvény,
- p: a neurális hálózat tanulásának leállási feltételét leíró függvény,
- $FHSZ$: fogalomhierarchia szótár.

A bemutatott módszerek kidolgozása során olyan algoritmusok elkészítése volt a célom, melyek képesek több ezer mondatból álló dokumentumok esetén a beállított paramétereknek megfelelő klaszterezési, illetve osztályozási feladatok elvégzésére. Tapasztalataim alapján ezzel a szövegmennyiséggel jellemezhető leginkább az a tudásanyag, amelyből kérdéseket kívánunk feltenni. Azonban sem a kérdések megválaszolására fordítható idő, sem pedig a dokumentum jellege nem teszi lehetővé, hogy a dokumentum minden mondatára kérdések generálódjanak. Ezért szükség volt ésszerű elvek alapján korlátozni a feltehető kérdések számát. Erre a feladatra első közelítésben megoldást jelentett egy újabb paraméter definiálása, mellyel megadható, hogy a teljes beolvasott dokumentumból hány mondat kerüljön véletlenszerűen kiválasztásra. Ezt követően a kérdések csak ezekre a véletlenszerűen kiválasztott mondatokra kerültek előállításra. Az első közelítéses módszer finomítása révén megvalósítottam, hogy a kérdések előállítását irányító szakértő egy külön paraméterben azt is megadhassa, milyen típusú mondatok kerülhessenek a kérdésgeneráló modulhoz.

A kidolgozott algoritmus gyakorlati alkalmazásának szinte minden területén elvárt a kevert mondattípusú dokumentumoknak az ilyen irányú szelektálhatósága. A kívánt megoldást a dokumentum mondatainak *annotálásával* értem el. Ennek értelmében a dokumentumnak minden olyan mondata, amely alkalmas lehet kérdésként való kiválasztásra annotálásra került egy mondat típusát leíró speciális karaktersorozattal. További lehetőségként megvalósításra került, hogy a kérdésként kiválasztható mondatok típusának beállítása során ne csak egy konkrét mondattípus kiválasztására legyen lehetőség. Az algoritmus alkalmas több annotációt tartalmazó szűrőfeltétel megfogalmazására, illetve ez alapján való keresésre. Ezáltal megvalósítható a dokumentum akár összes annotált mondatának kérdésként való kiválaszthatósága is. Az annotációk alkalmazásával a dokumentumnak azok a mondatai, melyek nem kerültek annotálásra teljes mértékben kizárhatóak a kérdésekből.

A dokumentum mondatainak annotálását követően előáll az a dokumentumfájl, ami már feldolgozható a munkám során elkészített algoritmussal. Ez adja a kérdésgeneráló algoritmus egyik bemenetét. Az elvárt működéshez szükség van azonban arra a fájlra is, ami a dokumentum minden szavára – azok előfordulásának sorrendjében – definiálja a szavak szófaját, valamint szótövét. Ezekre az információkra a mondatokból kérdésként kiemelt szavakra adható válaszlehetőségek automatikus előállításakor van szükség. Ezeket az információkat a 4. fejezetben ismertetett ingyenesen elérhető szótövező alkalmazás segítségével állítottam elő. Ennek a szótövező alkalmazásnak az eredményei egy pontosvesszőkkel szeparált szöveges fájlban érhetőek el, mely az elkészített algoritmus másik bemenetét képezi. Ez reprezentálja a szótövezett fájlt. A dokumentumfájl, valamint a szótövezett fájl megadását követően az algoritmus sorra veszi a két fájl szavait és amennyiben egyező szavakat talál, azokat összekapcsolja. Mivel a szótövezett fájlban minden szó külön sorban szerepel, ezért a szöveg szavakra való szeparálása könnyen elvégezhető. A dokumentumfájl esetén azonban erre a feladatra szószeparáló jelek előfordulását keresi a program a szövegben. A dokumentum szavai ezen jelek mentén kerülnek elválasztásra egymástól. A dokumentumfájlnak szavakra való bontását követően mindkét fájl szavak sorozataként kezelhető. A két listán párhuzamosan haladva kell megtalálnia az algoritmusnak a dokumentum fájlban lévő szó párját a szótövezett fájlban. Ennek megvalósításához a következő két nehézséget kellett megoldanom:

1. Mivel a szótövezést végző alkalmazás általános témájú szövegek elemzésére lett kifejlesztve, ezért a speciális tématerületeket jellemző szavak (pld. *entrópia*, *determinisztikus*, *adatbányászat* stb.) felismerésére nem alkalmas. Az ilyen szavakat a szószablya kihagyja az elkészített szótövezett fájlból, és semmilyen módon nem jelzi hiányukat. Ezáltal a kihagyott szó helyére a szótövezett fájlban a dokumentum következő szava kerül. Ez a jelenség megfelelő algoritmusok kifejlesztését igényelte.

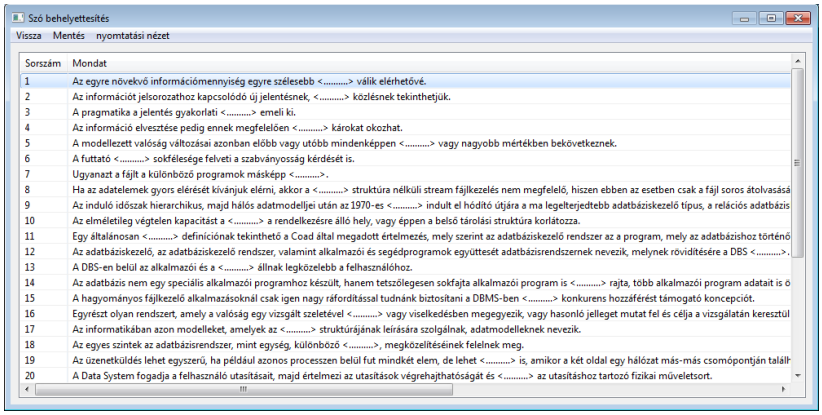
2. Léteznek olyan szavak, melyek szófaja nem határozható meg pusztán az adott szó ismeretében. Az ilyen szavak szófajának meghatározásához a teljes mondat nyelvtani elemzésére van szükség, mivel szófajuk függ a mondatkörnyezettől, amiben előfordulnak. Ilyen szavak például: *bizonyos*, *véletlen*, *vár* stb. Az alkalmazott szótövező szoftver azonban nem képes a mondatok szemantikai elemzésére. Ehelyett a szavak lehetséges szófajait egymást követően sorolja fel és az adatok értelmezőjére bízza a helyes szófaj kiválasztását. Ezáltal a szótövezett fájl azon helyén, ahol a dokumentum következő szavának kellene állnia, még az előző szó egy lehetséges új alternatívája szerepel. Ez is egy olyan feladat, ahol megfelelő speciális algoritmus alkalmazása nélkül a fájlok szinkron beolvasása elcsúszhat egymáshoz képest. Az olyan szavak esetén, melyekre a szótövező alkalmazás több lehetséges szófajt is megállapít, a kidolgozott algoritmus összegyűjti ezeket a lehetséges szófaj-alternatívákat és jelzi a szakértő felé a választás lehetőségét.

Az ismertetett módszer alkalmazásával a dokumentum minden mondata beolvasásra kerül. Azok is, melyek a megadott annotációs feltételnek nem felelnek meg. Erre azért van szükség, mert csak a két fájl szavainak párhuzamos megfeleltetésével biztosítható a két beolvasás szinkronizálása. A dokumentum mondatainak beolvasását követően az algoritmus sorra megvizsgálja a beolvasott mondatokat és ellenőrzi, hogy teljesítik-e az annotációkkal definiált szűrési kritériumot. Azokat a mondatokat, melyek teljesítik a szűrőfeltételt tartalmazó annotációk valamelyikét, felfűzi a kérdésként alkalmazható mondatok listájára. Ezt követően eltávolítja belőlük az annotációt, hogy a tesztlapon való megjelenítésükkor ez az információ már ne jelenjen meg. Az így kapott – szűrésnek megfelelt – mondatok listájából véletlenszerűen kerül kiválasztásra az algoritmus paramétereként definiált számú mondat. A tesztlapon megjelenő kérdések ezekből kerülnek előállításra. A kiválasztott mondatok kérdésként kiemelendő szava a korábbi fejezetben ismertetett osztályozó algoritmus által kerül meghatározásra. Az osztályozó algoritmus a kérdésekre kiválasztott mondatok szavainak objektív koordinátái ismeretében meghatározza a szavak szubjektív koordinátáit. A szubjektív koordináták közül a „*kérdésként kiemelésre kerüljön-e*” névvel ellátott bináris értékkészlettel definiált koordináta határozza meg, hogy a mondat adott szava kerülhet-e kérdésként kiemelésre vagy nem. Abban az esetben, ha ez a koordináta az adott kérdésre kiválasztott mondat egyik szavát sem javasolja kiemelésre, akkor az algoritmus az annotációs feltételnek megfelelő mondatok listájából véletlenszerűen választ egy másik mondatot helyette. Ha azonban ez a szubjektív koordináta az adott mondat több szavát is kiválaszthatónak irányozza elő, akkor a kivételre kerülő szó az osztályozást végző neurális hálózatnak ezt a szubjektív koordinátát definiáló kimeneti neuronjának bemeneti függvénye alapján kerül meghatározásra.

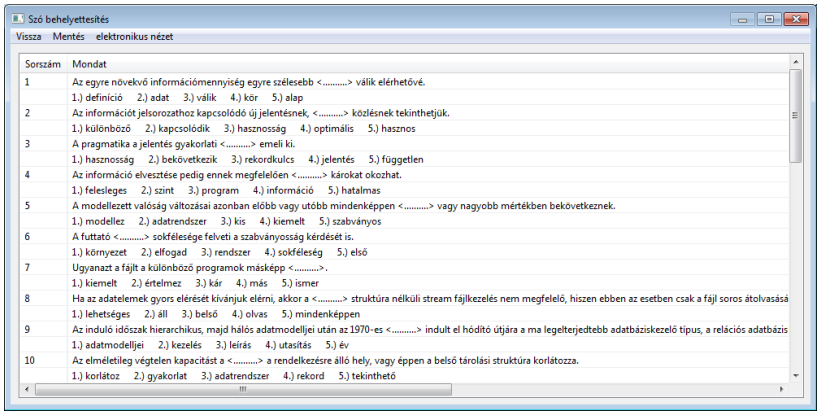
8.2. A kidolgozott modell alapján implementált program bemutatása

A kidolgozott modell helyességét Java nyelven implementált szoftverrel igazoltam. Az elméleti modell működési folyamatának tesztelésén kívül a szoftvert alkalmassá tettem a kérdések elektronikus, illetve nyomtatható formában való reprezentálására is. Az elektronikus változat esetén mind az előállított kérdések, mind pedig a rájuk adható válaszok számítógépes környezetben jelennek meg a felhasználó előtt. Ebben a változatban a tesztet kitöltő felhasználó a megválaszolandó kérdést tartalmazó mondatra kattintva tudja előhívni a kérdésre adható lehetséges válaszokat tartalmazó menüt. A válasz megadását követően a kiválasztott alternatíva automatikusan behelyettesítésre kerül a mondatba. A teszt végeztével a kitöltött tesztlap fájl formátumban elmenthető, és a teszt elbírálója felé továbbítható. A tesztlap nyomtatható formában

való reprezentálásához a szoftver a kérdésként kivett szó helyét kipontozva jelzi a tesztlap kitöltője számára, valamint a lehetséges válaszokat a mondatok alatt tünteti fel egymás mellett felsorolt alakban. A tesztlap nyomtatott formátumának kitöltéséhez a tesztelő személynek a megfelelőnek ítélt szót aláhúzással, vagy a kipontozott részbe való beírásával kell jeleznie. A tesztlap kétféle megjelenítési formájára a 8.1. ábrák mutatnak egy-egy példát [9].



8.1.a. ábra: Elektronikus kitöltési forma



8.1.b. ábra: Nyomtatható kitöltési forma

8.1. ábra: Az automatikusan előállított tesztlap elektronikus és nyomtatható változata

8.3. A kérdésgeneráló mintarendszer értékelése

Az elkészült automatizált kérdésgeneráló mintarendszer tesztelésére és az eredmények értékelésére a 2011/12 tanév második félévében került sor a Miskolci Egyetem Comenius Főiskolai Karán. A teszt célja a kérdésgenerálás területén elért eredményeim gyakorlati alkalmazhatóságának igazolása volt. A vizsgálat kiterjedt az elkészített szoftverrel automatikusan előállított kérdések hallgatók által való elfogadhatóságának, illetve értelmezhetőségének elemzésére. Ezen felül célom volt a különböző kategóriákba sorolható hallgatók tudásszintjének felmérése és az eredményeik statisztikai elemzése is. A kérdésgeneráló mintarendszer jelenleg három mondattípus előállítására alkalmas (fogalom, definíció, kijelentő mondat), amelyek közül mindhárom típus szerepelt az elkészített feladatlapon, feleletválasztós kérdések formájában. A Ph.D. munkám során kidolgozott kérdésgenerálási koncepció gyakorlatban való alkalmazhatóságának vizsgálata érdekében, egy manuálisan előállított feladatsort is elkészítettem. Miután a tesztben résztvevő személyek, mindkét feladatsort kitöltötték lehetőségem nyílt az új koncepciónak a napjainkban alkalmazott manuális kérdésgenerálással való összehasonlítására.

8.3.1. Mintarendszer tesztelése a felsőoktatásban

Az automatizált kérdésgenerálás eredményeinek gyakorlati teszteléséhez használt tanagyagot a mérnök informatikus (B.Sc.) és a web-programozó (FSZ) szakon oktatott *Adatbázisrendszerek* című tantárgyhoz tartozó jegyzet képezte, amely a főiskolai kar honlapjáról szabadon letölthető.
A tesztelésben 45 fő vett részt, akik közül 40 főiskolás hallgató és 5 szakértő. A hallgatókat úgy választottam ki, hogy csak az 50% (20 fő) tanulta az említett tantárgyat, a másik 50% nem ismerte annak tartalmát. A szakértők mindegyike a tantárgy oktatói közül került ki. A tesztelést végző személyek az így definiált kritériumok alapján három kategóriába kerültek besorolásra:

- nem tanulta a tárgyat,
- tanulta a tárgyat,
- szakértő.

Akik nem tanulták a tantárgyat 18...20 év közöttiek, nem szerinti megoszlásuk alapján 3 férfi és 17 nő. Akik tanulták a tantárgyat 19...20 év közöttiek, nem szerinti megoszlásuk alapján 18 férfi és 2 nő. A szakértők csoportjának tagjai átlagosan 53 évesek és 3 férfi valamint 2 nő alkotta. A felmérésben két feladatlapot kellett minden tesztelő személynek kitöltenie. Az egyik a Ph.D. munkámban kidolgozott kérdésgeneráló mintarendszer által generált kérdéseket tartalmazta, míg a másik a tantárgyat oktató személyek által manuálisan összeállított kérdésekből állt. A tesztlap az *C.1* Függelékben megtalálható.
Mindkét esetben a feleletválasztós kérdésekként használt mondatok a tantárgy teljes írásos tananyagában annotált 1.000 mondat közül kerültek kiválasztásra véletlenszerűen. Mindkét tesztlap 30 kérdést tartalmazott, melyhez kérdésenként öt lehetséges választ társított a tárgy oktatója, illetve a számítógép. A lehetséges válaszok úgy lettek megtervezve, hogy a kérdés egyetlen helyes megoldásán kívül a többi alternatíva a helyes választól egyenletes lépésközzel fokozatosan távolodva helyezkedjen el. Természetesen az így előállított válaszalternatívák megjelenítési sorrendje véletlenszerűen lett megállapítva. A résztvevőknek a kérdésekhez tartozó válaszokból kellett kiválasztani azt, amelyiket leginkább alkalmasnak találták a mondatból kivett szó helyére. Nehezítette a választást, hogy a válaszlehetőségek ragozás nélkül jelentek meg a hallgatók előtt. Mindkét feladatlap kitöltésére 30-30 perc állt rendelkezésre. Az eredmények kiértékelése során minden helyes válaszhoz +1 pont, míg minden hibás válasz 0 pont lett megállapítva. A 8.1. táblázat a tesztelésre szolgáló feleletválasztós kérdésre mutat példát. A 8.1.a). ábrán az automatizált kérdésgenerátor által előállított kérdés, míg a 8.1.b). ábrán a tárgy oktatói által manuálisan előállított kérdés látható.

A számítástechnika <.....> egyik fontos jellemzője, hogy egyre több felhasználó egyre több, számítógépen tárolt adatot használ fel. 1.) felhasználó 2.) fejlődés 3.) adatbáziskezelő 4.) fájlszervezés 5.) túlcsordul **a)**
Adatbáziskezelő-rendszer az a programrendszer, melynek feladata az adatbázishoz történő hozzáférések biztosítása és az ..... belső karbantartási funkcióinak végrehajtása. A. kapcsolódik B. induló C. információ D. feladat E. adatbázis **b)**

8.1. táblázat: A tesztelésre szolgáló kérdések megjelenítési formátuma

A tantárgy oktatói által manuálisan összeállított kérdéssorral az volt a célom, hogy a kitöltésével kapott eredményeket össze tudjam hasonlítani az automatizált kérdésgenerálásra kidolgozott számítógépes alkalmazás által előállított kérdések eredményességével.
Hasonló megközelítés olvasható Canella, Ciancimino és Campos [Canella, 2010] cikkében, ahol a kiválasztott fogalmakat manuálisan minősítették a hallgatók. A fogalmak minősítése ötpontos Likert skála [Earl, 1998] segítségével történt. A Likert skálára jellemző, hogy azt kérdezi a válaszadótól, hogy egyetért-e vagy nem egy bizonyos állásponttal. Ebben öt kritériumot határoztak meg [Gütl, 2011]:

- helytállóság,
- nehézségi szint,
- szakterület,
- válaszhoz tartozik-e a fogalom,
- hiányos mondatokhoz tartozik-e a fogalom.

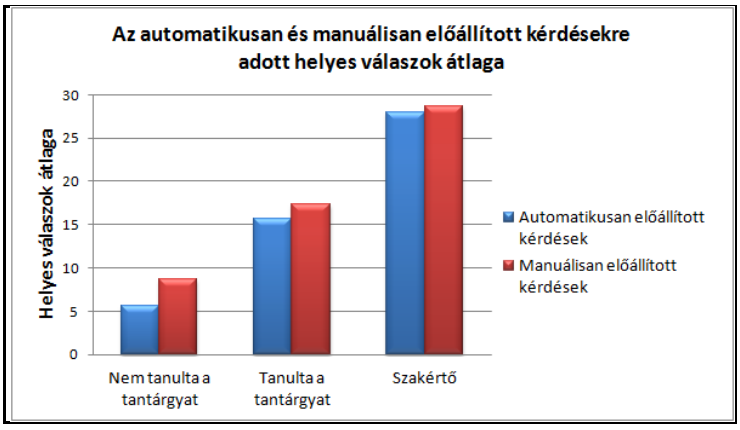
Az általam megtervezett kérdésgeneráló mintarendszerben a szubjektív koordinátákat a mondatok szavainak szemantikai jellemzői alapján határoztam meg. Ezek a következők: *relevancia*, *specialitás*, *értelmesség* és *nehézség*. Minden szubjektív koordináta öt lehetséges értékből álló intervallumon került definiálásra. Az intervallum értékei azt fejezik ki, hogy az adott szót a szakértő mennyire tartja az adott szubjektív koordinátához (mint kategóriához) tartozónak. Ez alapján minden szó mind a négy szubjektív koordinátája a következő öt lehetséges érték egyikével került leírásra:

- az adott szubjektív koordináta erős képviselője,
- az adott szubjektív koordináta képviselője,
- az adott szubjektív koordináta szempontjából irreleváns,
- az adott szubjektív koordináta ellenpárja,
- az adott szubjektív koordináta erős ellenpárja.

Az automatizált kérdésgeneráló alkalmazás a szavak klaszterezésével nyert távolságadatain túl az osztályozást végző modul által szolgáltatott szubjektív koordináták távolságait is felhasználja a kérdésekre adható lehetséges válaszok előállítására. Eredményként az egyes válaszalternatíváknak a helyes választól való folyamatos távolodása nemcsak az egzaktul mérhető adatok alapján, hanem az emberi szubjektív tényezők alapján is megvalósításra került. Ez teszi az egyetlen helyes válasz megtalálását valóban nehézzé.

8.3.2. A teszt eredményeinek kiértékelése

A tesztben résztvevő minden személy kitöltötte mind az automatizált kérdésgeneráló mintarendszer, mind pedig a tárgy oktatói által előállított feladatlapot. Ezt követően az eredmények kiértékelésre kerültek. A mintarendszerrel előállított feladatlapokon a 30 kérdésből 19 esetén lett *szakszavak* csoportjába tartozó szó kiemelve kérdésként. A többi 11 mondat esetén a kiemelt szó az *általános szavak* csoportjába tartozott. Ezzel szemben a manuálisan előállított feladatlapon a 30 kérdésből 25 esetén tartozott a kivett szó az *általános szavak* csoportjába, és csupán 5 esetben a *szakszavak* csoportjába. A helyes válaszok átlagát mindhárom csoportba tartozó hallgatók esetén a 8.2. ábra szemlélteti.



8.2. ábra: Az automatikusan és manuálisan előállított kérdések eredményei

A 8.2. ábra alapján jól látható, hogy a manuálisan készített kérdésekre a hallgatók mindhárom csoportja esetén több jó válasz született, mint a mintarendszer által generáltakra. Ebből látható, hogy az *általános szavak* területéhez kapcsolódó kérdésekre átlagosan jobban tudtak válaszolni a hallgatók. Ez a különbség legkisebb mértékben a tárgy szakértői esetén volt kimutatható, akik ismerték a mondatokban szereplő speciális fogalmak pontos tartalmát és fontosságát. A két teszt között a legnagyobb eltérés a tantárgyat nem ismerő hallgatók esetén volt tapasztalható. Az előzetes becsléseknek megfelelően a hallgatók ebben az esetben találták meg legnehezebben a *szakszavak* területéhez kapcsolódó kérdésekhez tartozó helyes választ. Ehhez képest az *általános szavak* területéhez tartozó kérdésekben viszonylag jól teljesítettek.
A 8.2. ábra pontos numerikus értékeit a 8.2. táblázat foglalja össze.

Csoportok\Kérdések	Automatikusan előállított kérdésekre adott helyes válaszok átlaga	Manuálisan előállított kérdésekre adott helyes válaszok átlaga
Nem tanulta a tantárgyat	18,83%	28,83%
Tanulta a tantárgyat	52,20%	57,67%
Szakértő	93,33%	95,33%

8.2. táblázat: Az automatikusan és manuálisan előállított kérdések eredményei

A következő elemzési lépésben annak a meghatározására törekedtem, hogy van-e kimutatható kapcsolat a szavak osztályozásakor definiált szubjektív koordináták és a helyes válaszok között. Ehhez először a hallgatóknak azt a csoportját vizsgáltam, akik nem tanulták a tantárgyat. A szubjektív koordináták hatásait a 8.3. ábrán látható diagrammal jelenítettem meg.

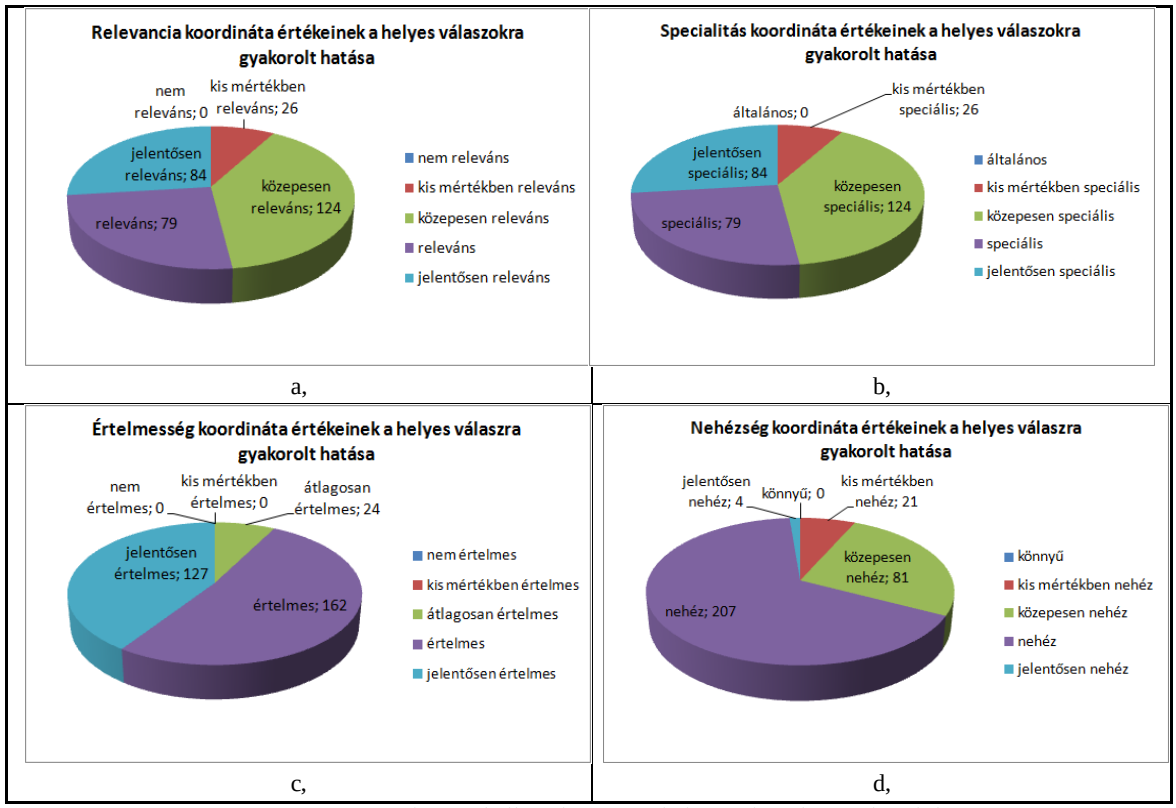
8.3. ábra: A tantárgyat nem tanuló hallgatók esetén a helyes válaszok száma

Az 8.3. ábra által reprezentált adatokat a 8.3. táblázat tartalmazza. A táblázat celláiban lévő bal oldali értékek a tesztelésre használt tantárgyat nem ismerő személyek helyes válaszainak számát, míg a jobb oldali érték a tantárgyat ismerőkét mutatja.

Koordináták\Értékek	a kategória erős ellenpárja	a kategória ellenpárja	a kategória szempontjából irreleváns	a kategória képviselője	a kategória erős képviselője
Relevancia	0 – 0	9 – 26	84 – 124	35 – 79	41 – 84
Specialitás	0 – 0	9 – 26	64 – 124	37 – 79	27 – 84
Értelmesség	0 – 0	0 – 0	28 – 24	92 – 162	53 – 127
Nehézség	0 – 0	5 – 21	41 – 81	125 – 207	2 – 4

8.3. táblázat: A szubjektív koordinátáknak a helyes válaszok számára gyakorolt hatása

A táblázat adataiból jól látható, hogy a szubjektív koordináták közül a *Nehézség* koordináta értékének a kategóriához tartozása esetén mutatható ki legszorosabb kapcsolat a helyes válaszok számával. Azokra a kérdésekre érkezett a legtöbb helyes válasz, amelyekből kivett szót a szakértő az átlagostól nehezebbnek értékelte.
Ezt követően a vizsgált szubjektív koordinátáknak a válaszok helyességére gyakorolt hatását a tantárgyat ismerő hallgatók körében vizsgáltam. A kapott adatokat a 8.4. ábrán látható diagramokkal jelenítettem meg.



8.4. ábra: A tantárgyat ismerő hallgatók esetén a helyes válaszokra gyakorolt hatása

A tantárgyat ismerő hallgatóknál a kapott adatok egyértelműen mutatják, hogy mind a négy szubjektív koordináta esetén az adott kategóriába az átlagostól jobban beleillő szavak eredményezték a legtöbb helyes választ. A tantárgyat nem ismerő hallgatóknál végzett hasonló kimutatással szemben a mostani adatok, közel dupla erősségű korrelációt mutatnak a kategóriához tartozást kifejező szubjektív koordináták és a helyes válaszok között. Ennek fő oka az, hogy a tantárgyat ismerők csoportja jobban kategorizált tudással rendelkezik a vizsgált tantárgy területén.

A legtöbb helyes választ itt is a *nehézség* nevű szubjektív koordináta eredményezte. A kimutatások alapján érdemes megfigyelni azt is, hogy ha az adott szubjektív koordinátához tartozás mértéke eléri a maximális értékét, akkor a helyes válaszok száma mindegyik koordináta esetén nulla körüli értékre csökken. Ennek oka, hogy a nagyon speciális, nagyon nehéz stb. szavak megtalálása még a tantárgyat ismerők számára is nehézséget okoz.
Az ismertetésre került kimutatásokon kívül a következő összefüggések is feltárásra kerültek:

- helyes válaszok száma hallgatókként,
- hallgatók elért pontszámának átlaga,
- hallgatók elért pontszámának szórása,
- hallgatók életkorának megoszlása,
- hallgatók nemének megoszlása,
- hallgatók iskolai végzettségének megoszlása.

Ezek bemutatása és elemzése azonban terjedelmi okokból nem képezi részét jelen fejezetnek. A disszertáció *C.1* Függelékben megtekinthetőek.

8.4. Az AQG mintarendszer implementálásának és értékelésének összefoglalása

Az annotációk alkalmazásával meghatároztam az akár több ezer mondatot is tartalmazó dokumentumból azokat a mondatokat, melyekből a kérdések előállításra kerülnek. Ezt követően az osztályozási módszerek eredményeire alapozva mondatonként meghatároztam a megkérdezésre kerülő szót. A Ph.D. munkám keretében kifejlesztett AQG mintarendszer gyakorlati alkalmazhatóságát igazolja a ME Comenius Főiskolai Karán végzett teszt. Ennek keretében kitöltésre kerültek az egyetem/főiskola *Adatbázisrendszerek* című tantárgyának anyagából a kérdésgenerátorral, illetve hagyományos kézi módszerrel előállított feladatlapok. A két eltérő koncepció alapján készített kérdések jobb összehasonlíthatósága érdekében a teszteket három különböző csoportba tartozó személyekkel töltettem ki. Az eredmények kiértékelésével azt tapasztaltam, hogy mindkét kérdés-előállítási típusra közel egyforma számú helyes válasz érkezett. Ez azt igazolja, hogy a kifejlesztett új koncepció alkalmazható a jóval időigényesebb manuális kérdés-előállítás helyett. Az új módszerben alkalmazott többféle hangolási lehetőséggel (klaszterezéssel kapott szótávolság, fogalomhierarchia, szubjektív koordináták stb.) pedig elérhető, hogy a kérdések még az ember által előállítottól is jobban tükrözzék a tesztelés alapjául szolgáló tananyag lényegét. Ennek hatását a tesztek eredményei is igazolták.
Az ember által előállított kérdésekre mindhárom kategóriához tartozó hallgatók esetén több helyes válasz érkezett. Ez a különbség azonban a téma szakértői esetén alig volt kimutatható, míg a témát nem ismerők felé haladva folyamatosan nőtt. Ez a tény egyértelműen azt mutatja, hogy az ember által készített kérdésekben a témát nem ismerők is képesek találni olyan stíluselemeket, amelyek révén tudásukat a valóságostól magasabbnak tűntethetik fel. Az új koncepció azonban jelentősen (34,69%) volt képes javítani a tudásmérés pontosságát. A gyakorlatban végzett tesztek alapján feltártam a szavak szubjektív koordinátáinak a helyes válaszok számára gyakorolt hatását is. Ez alapján azt tapasztaltam, hogy az átlagostól nehezebb tartalmú szöveg esetén a témát ismerő személyek jelentősen több helyes választ adtak az átlagostól nehezebbnek minősített szavakra, mint a témát nem ismerők. Ezáltal megállapítást nyert, hogy a téma szempontjából az átlagostól nehezebbnek minősített szavak a leginkább alkalmasak a tudás szintjének mérésére. Ez a felismerés egy újabb lehetőséget nyit a koncepció továbbfejlesztésére.

9. fejezet

Összefoglalás

Az értekezés olyan rendszermodellt mutat be, amely annotált magyar nyelvű szöveges dokumentumból – megadható mondattípusok alapján – feleletválasztós, kiegészítő kérdések automatikus előállítására alkalmas.
Ennek megfelelően a jelen kutatás során a következő főbb feladatokat valósítottam meg.

1. Az automatizált kérdésgenerálás megvalósítására olyan rendszermodellt dolgoztam ki, melyben több egymással adatkapcsolatban álló alrendszer együttes összehangolt működése biztosítja a feladat megoldását. A különböző feladatokat ellátó egységekhez (modulok) kimeneti, illetve bemeneti interfészeket definiáltam mely révén a rendszer moduljai egymástól függetlenül vizsgálhatókká, illetve fejleszthetőkké váltak (4. fejezet).
2. A dokumentum szavainak különböző koncepciókra épülő klaszterezése révén meghatároztam a szavak hasonlóságát leíró kritériumokat. A klaszterezést elvégeztem a szavak közös előfordulási gyakoriságára épülő, valamint a szavak objektív koordinátái által definiált többdimenziós térben mérhető vektortér alapú távolságok alapján is. A kidolgozott modellek helyességét igazoltam a szélességi, mélységi, legjobbat először keresés, valamint a BIRCH stratégiák feladatspecifikus implementációival (5. fejezet).
3. Kidolgoztam egy neurális hálózaton alapuló osztályozó algoritmust, ami képes feltárni a dokumentum szavainak objektív, valamint szubjektív koordinátái közötti kapcsolatot leíró szabályrendszert. A szavak szubjektív koordinátáinak automatizált meghatározásával előállítottam a kérdésként kiemelésre kerülő szó, valamint a kérdésre adható válaszok meghatározásához szükséges információkat (6. fejezet).
4. Az előállított információk alapján feladatspecifikus algoritmusokat fejlesztettem ki a kérdésekre adható lehetséges válaszok automatizált meghatározására. A kidolgozott algoritmusok saját implementálású fogalomhierarchiát, térfelosztást, valamint szakértői vélemények interaktív kinyerését alkalmazva valósítják meg a feladatot (7. fejezet).
5. A kérdések automatizált előállítása során annotációkra épülő mennyiségi és minőségi kritériumok alapján meghatároztam a kérdéssorra kerülő mondatokat. A kidolgozott rendszermodell gyakorlatban való alkalmazhatóságát különböző tudásszinttel rendelkező személyekkel teszteltem. Az eredmények azt igazolták, hogy a mintarendszerrel automatikus módon előállított tesztlapok a manuálisan előállítotthoz képest alig kimutatható különbséggel voltak képesek mérni a tesztben résztvevő személyek tudásszintjét. Ezáltal igazolást nyert, hogy a Ph.D. munkám során kifejlesztett kérdésgeneráló mintarendszer alkalmazható a jóval időigényesebb manuális kérdés-előállítás helyett (8. fejezet).

9.1. Új tudományos eredmények

Az új tudományos eredmények a következők alapján foglalhatók össze.

1. tézis: [3], [14]
Meghatároztam a rendszer adatmodelljének- és funkciómodelljének elemeit. Definiáltam az automatizált kérdésgenerálás megvalósításához szükséges feladatokat, a feladatokhoz kapcsolódó bemenő és kimenő adatokat. Korlátfeltételek meghatározásával kijelöltem a módszer

alkalmazhatósági területeit. Kidolgoztam a részfeladatok összehangolt ellátására szolgáló moduláris rendszertervet és egy dokumentum szerkezeti modellt. Bemutattam a kidolgozott modell elvi működőképességét.

2. tézis: [1], [7], [8]
Elemeztem a szavak távolságát mutató klaszterezési eljárások lehetőségeit és kidolgoztam egy alkalmas attribútum rendszert. Kidolgoztam és optimalizáltam egy paraméterezhető minőségi küszöbérték alapú klaszterező eljárást. Az automatizált kérdésgeneráló rendszerben a kijelölt klaszterek megfelelően reprezentálják a szavak szemantikai szerepkörét a vizsgált tartományon belül.

3. tézis:
Meghatároztam a kérdésként kiemelésre kerülő szavak kiválasztásához szükséges bemenő objektív koordinátákat. Kidolgoztam egy neurális hálózaton alapuló osztályozó algoritmust, amely képes feltárni a dokumentum szavainak objektív, valamint szubjektív koordinátái közötti kapcsolatot. A kidolgozott CPN adaptációs hálózat teljesíti az automatizált kérdésgeneráló rendszerben elvárt pontosságot.

4. tézis: [9], [14]
Kidolgoztam egy olyan algoritmust, amely az előálló objektív és szubjektív koordináták alapján meghatározza a kérdésként kiemelt szót és annak válaszalternatíváit. Az automatizált kérdésgenerálási módszer gyakorlati alkalmazhatóságának igazolására egy Java nyelven írt szoftvert készítettem. Az elemzések alapján megállapítható, hogy az automatizáltan előállított tesztek eredményei szoros korrelációban állnak a manuális tesztek eredményeivel.

9.2. További kutatási feladatok

- A nagyméretű dokumentumok klaszterezésének legelterjedtebben alkalmazott módszere a BIRCH algoritmus. Ennek feladatspecifikus adaptációját alkalmaztam a tudományos munkám során kidolgozott mintarendszerben. A BIRCH algoritmus alapú klaszterezéssel végzett tesztek eredményeinek alapos tanulmányozását követően felfedeztem néhány lehetőséget, melyekkel speciális esetekben az algoritmus hatékonysága javítható. Ezek közül a legígéretesebb irányokat az algoritmus főbb paramétereinek szoftveresen támogatott beállítása, illetve a nagy elágazási tényezőjű fában való keresés időigényének csökkentése jelentik. Ezekben a továbbfejlesztési témákban elért eredményeimet az értekezés *A.6* Függelék tartalmazza.

- A szakirodalomban mindmáig nem ismert egzakt módszer a többrétegű neurális hálózatok belső rétegeiben lévő neuronok számának meghatározására. Az alkalmazott neuronok száma a legtöbb esetben konkrét példák alapján való tesztelésekkel kerül meghatározásra. Az optimális számtól kevesebb belső neuron alkalmazása esetén a hálózat nem képes a bemenetek és kimenetek közötti kapcsolatot leíró átviteli függvény reprezentálására. Az optimális értéktől több belső neuron alkalmazásával viszont a hálózat a szabályrendszer feltárása helyett hajlamosabb a konkrét példák betanulására. Kutatási munkám során az alkalmazott háromrétegű neurális hálózat belső rétegében lévő neuronok számát azonosnak választottam a kimeneti rétegben lévő neuronokéval. A munkám speciális alkalmazási területe alapján nyerhető információk alapján lehetőség nyílik a belső rétegben lévő neuronok optimumközeli számának meghatározására. Ezt az osztályozandó dokumentum

által képviselt tudományterületnek, a dokumentum méretének, valamint a szavak objektív koordinátáinak alapján végzett tesztekkel kívánom meghatározni.

- Annak érdekében, hogy az osztályozásra alkalmazott neurális hálózat tanítását ne kelljen minden dokumentum esetén megismételni, megvalósítottam a hálózat betanult átviteli függvényének igény szerinti mentési és visszatöltési lehetőségét. Mivel azonban a neurális hálózat dinamikusan állítja be a bemeneti rétegében lévő neuronok számát a dokumentum klaszterezésével nyert információk alapján, ezért az elmentett információk csak ugyanolyan stratégiával és paraméterekkel klaszterezett dokumentumok esetén kompatibilisek egymással. A kidolgozott mintarendszer gyakorlati alkalmazhatóságának területe bővíthető lenne egy általános mentési és visszatöltési algoritmus kidolgozásával.

- A mintarendszer kifejlesztésével az volt a célom, hogy a feladatsorok idő és emberi erőforrásigényes manuális előállításának folyamatát automatizált módon valósítsam meg. Ehhez olyan kérdések és válaszalternatívák előállítására alkalmas algoritmusokat dolgoztam ki, melyekkel minél pontosabban mérhető a feladatlapot kitöltő személynek az adott témára vonatkozó tudásszintje. A gyakorlati tesztek alapján kimutattam a válasz-alternatívaként használt szavak szubjektív koordinátáinak a válasz helyességére gyakorolt hatását a tesztet kitöltő személyek előre ismert tudásszintjének függvényében. Ezáltal olyan információhoz jutottam, mely megmutatja, hogy a kérdezett témát különböző szinten ismerő személyek milyen szubjektív koordinátákkal rendelkező szavakat részesítenek előnyben a helyes válaszok kiválasztása során. Ezeket az információkat is használva lehetőség nyílik a már kidolgozott kérdés, illetve válaszalternatívákat előállító algoritmusok továbbfejlesztésére, melyekkel pontosabban mérhető a tesztben résztvevő személyek tudásszintje.

10. fejezet

Summary

The present dissertation is about a system model suitable for the automated generation of multiple choice open questions of predetermined sentence patterns based on annotated Hungarian text-documents.
According to the above, in the course of the present research I fulfilled the following tasks:

1. To generate multiple choice questions automatically, I have designed a system model in which a harmonized functioning of a database of a multiple subsystem supports the solution of the task. Output and input interfaces were defined to the modules assigned to fulfil different tasks, which granted the separate check and independent development of the modules (see Chapter 4).
2. Through the clustering of the different concepts represented by the words in the document the criteria describing the similarity of the words were defined. The clustering was carried out both on the basis of the common frequency of the words and their geometric distances measured in the multidimensional space defined by the subjective coordinates of the words. The validity of the model was proved by the width-first, depth-first, best-first search, as well as the task specific implementations of the BIRCH strategies (see Chapter 5).
3. A classification algorithm based on a neural network was designed which is capable of exploring the system of rules describing the subjective and objective connections between the coordinates of the words in the document. With the automated definition of the subjective coordinates of the words the information necessary to highlight the word answering the question and to determine the possible answers was produced (see Chapter 6).
4. On the basis of the information produced task-specific algorithms were developed to generate the answer options. The algorithms fulfil the task applying their own implemented concept hierarchy, subdivision of space and interactive extraction of professional opinions (see Chapter 7).
5. During the automated production of the questions the sentences of the protocol was determined on the basis of quantitative and qualitative criteria of annotations. The practical application of the system model was tested on humans with different educational background. The results proved that the test papers generated by the automated pattern system measured the intellectual capacities of the tested humans with little or no difference compared to the ones compiled manually. With the above it was proved that the pattern-question generating system developed in the course of my Ph.D. can efficiently substitute the time-consuming manual question generation (see Chapter 8).

10.1. The new scientific results

New scientific results can be summarized as follows.

Thesis 1: [3], [14]
The elements of the data structure and those of the function model were specified. I defined the tasks necessary to work out the automatic question generation, as well as the input and output data

related to the system components. The scope of application of the method was specified by determining the constraint conditions. A system-plan to serve the harmonized functioning of the subtasks and a documentary structural model were developed. The potential functioning of the model was demonstrated.

Thesis 2: [1], [7], [8]

The possibilities of clustering methods indicating the distances between the words were analysed and a suitable attribute-system was developed. A Quality Treshold (QT) method based on parameter-driven software was implemented and optimized. The assigned clusters well represent the semantic fields of the words in the Automatic Question Generation (AQG) system of the domain investigated.

Thesis 3: [3], [14]

The objective input coordinates necessary to select the assigned words in the question were defined. A classification algorithm based on a neural network to describe the connection between the objective and subjective coordinates of the words in the document was developed. The precision expected in the Automatic Question Generation (AQG) is achieved by the Counter-Propagation Network (CPN).

Thesis 4: [9], [14]

An algorithm to generate the answer alternatives on the basis of the objective and subjective coordinates of the words assigned as questions was developed. To prove the practical applicability of the automated question generating method, Java-language software was developed. On the basis of the analyses it can be concluded that the results gained from the automated tests closely correlate to the ones of the manual tests.

Irodalomjegyzék

[Álmos, 2002] Álmos, A., Dr. Horváth, G., Dr. Várkonyiné, Dr. Kóczy A. és Győri, S. (2002). Genetikus algoritmusok, Typotex, Budapest, 2002.

[Alshomrani, 2012] Alshomrani, S. (2012). Evaluation of Technical Factors in Distance Learning with respect to Open Source LMS, Asian Transactions on Computers (ATC ISSN: 2221-4275), Vol. 02, Issue 01.

[Atwell, 2000] Atwell, E., Demetriou, G., Hughes, J., Schiffrin, A., Souter, C, & Wilcock, S. (2000). A comparative evaluation of modern English corpus grammatical annotation schemes. ICAME Journal, Vol. 24, pp. 7-23.

[Babbie, 1998] Babbie, E. (1998), A társadalomtudományi kutatás gyakorlata, Balassi Kiadó és az ELTE Szociológiai Intézetének közös kiadása, Budapest, 1998. pp. 460 – 461.

[Barron, 1993] Barron, A. E. & Kysilka, M. L. (1993). The Effectiveness of Digital Audio in Computer-Based Traning, 1993, Vol. 25 Number 3.

[Benedek, 2003] Benedek, A. (2003). E-learning stratégiák In: Az eLearning szerepe a felnőttoktatásban és a képzésben (HARANGI L. – KELNER Gitta). Budapest, Magyar Pedagógiai Társaság Felnőttnevelési Szakosztály. 2003. p. 6-7.

[Bloom, 1956] Bloom, B. (1956). Taxonomy of Educational Objectives, Handbook 1: Cognitive Domain, Addison-Wesley Publ. 1956.

[Blumberg, 2003] Blumberg, R., Atre, S. (2003). The Problem with Unstructured Data, DM Review, February 2003.

[Blumberg, 2003b] Blumberg R, & Shaku A. (2003b). The problem with unstructured data, DM Review, 1 February.

[Bodon, 2006] Bodon, F. (2006). Adatbányászati algoritmusok, Free Software Foundation által kiadott GNU Free Documentation license 1.2-es, Budapest, 2006. március 07, pp. 145-146.

[Bodon, 2010] Bodon, F. (2010). Adatbányászati algoritmusok, Free Software Foundation által kiadott GNU Free Documentation license 1.2-es, Budapest, 2010. február 26, pp. 175-181.

[Bosak, 1999] Bosak, J. & Tim B. (1999). XML and the Second Generation Web, Scientific American, pp. 89-93.

[Boyer, 2009] Boyer, K. E., Lahti, W. J., Robert Phillips, Wallis, M. D., Vouk, M. A., & Lester, J. C. An Empirically Derived Question Taxonomy for Task-Oriented Tutorial Dialogue (2009). In Proceedings of the Second Workshop on Question Generation, Brighton, U.K., 2009, pp. 9-16.

[Brown, 2004] Brown, J., & Eskenazi, M. (2004). Retrieval of Authentic Documents for Reader-Specific Lexical Practice. In Proceedings of InSTIL/ICALL Symposium 2004. Venice, Italy.

[Brown, 2005] Brown, J. C., Frishkoff, G. A., & Eskenazi, M. (2005). Automatic Question Generation for Vocabulary Assessment. Proc. of HLT/EMNLP '05, 819-826.

[Canella, 2010] Canella, S., Ciancimino, E. & Campos, M. L. (2010) Mixed e-Assessment: an application of the student-generated question technique, Paper read at IEEE International Conference EDUCON 2010, Madrid, Spain, April.

[Catell, 1997] Catell, R. (1997). Object Database Standard, Morgan Kaufmann Publisher.

[Cawsey, 2002] Cawsey, A. (2002). Mesterséges Intelligencia Alapismeretek, Panem Könyvkiadó Kft. Budapest, 2002.

[Chen, 2006] Chen, C.Y., Liou, H.C. & Chang, J.S. (2006). FAST: An Automatic Generation System for Grammar Tests, Proceedings of the COLING/ACL on Interactive presentation sessions, pp. 1-4.

[Chu, 2005] William C. Chu, Hong-Xin Lin, Juei-Nan Chen & Xing-Yi Lin (2005). Context-Sensitive Content Representation for Mobile Learning, Department of Computer Science and Information Engineering, TungHai University.

[Collins, 2007] Collins, M., & Duffy, N. (2007). New Ranking Algorithms for Parsing and Tagging: Kernels over Discrete Structures, and the Voted Perceptron, Proc. of 40th Annual Meeting of the Association for Computational Linguistics, pp. 263-270.

[Coniam, 1997] Coniam, D. (1997). A Preliminary Inquiry into Using Corpus Word Frequency Data in the Automatic Generation of English Language Cloze Tests, CALICO Journal, 14 (2-4), pp. 15-33.

[Csendes, 2003] Csendes, D., Hatvani, Cs., Alexin, Z., Csirik, J., Gyimóthy, T., Prószéky, G. és Váradi, T. (2003). Kézzel annotált magyar nyelvi korpusz: a Szeged Korpusz. In: Alexin Zoltán; Csendes Dóra (szerk.) Az 1. Magyar Számítógépes Nyelvészeti Konferencia előadásai (MSZNY), 238–245. SZTE, Szeged (2003).

[Csendes, 2004] Csendes, D., Csirik, J. és Gyimóthy, T. (2004). The Szeged Corpus: A POS Tagged and Syntactically Annotated Hungarian Natural Language Corpus. In: Proceedings of the 5th International Workshop on Linguistically Interpreted Corpora (LINC 2004) at The 20th International Conference on Computational Linguistics (COLING 2004). Geneva, Switzerland, 23-29 August, pp. 19-23.

[Dagan, 1997] Dagan, I., Karov, Y. & Roth D. (1997). Mistake-driven learning in text categorization, In proc. of EMNLP-97, 2nd Conf. on Empirical Methods in Natural Language Processing, pp. 55-63, Providence, USA.

[Dash, 2005] Dash, N. S. (2005). The role of context in sense variation: introducing corpus linguistics in Indian contexts. Language In India**.** Vol. 5. No. 6. pp. 12-32.

[Date, 1995] Date, C. J. (1995). Introduction to Database Systems, Sixth Edition. Addison-Wesley. pp. 839.

[Day, 1992] Day, W. (1992). Complexity Theory: An introduction for of classification practitioners of classification, Clustering and Classification, World Scientific Publ., 1992.

[Dolores, 2007] Dolores, M. (2007). The Roles of Context in Word Meaning Construction: A Case Study, IJES, Vol. 7(1), 2007, pp. 169-173.

[Doyle, 1999] Doyle, B. (1999). The History of DITA, http://dita.xml.org/book/export/html/1047

[Dudás, 2006] Dudás, L. (2006). Mesterséges Intelligencia Módszerek 1.- 2. rész, Oktatási segédlet.
http://ait.iit.uni-miskolc.hu/~dudas/MIEAok/MIea11.PDF.

[Dudás, 2011] Dudás, L. (2011). Self-learning Semantic-distance-based Answering, Carpathian Journal of Electronic and Computer Engineering, ISSN 18474 – 9689, 4 (2011) pp. 35-40.

[Edmundson, 1963] Edmundson, H. P. (1963). A Statistician's View of Linguistic Models and Language Data Processing, In: Garvin P. L. (ed.), Natural Language and the Computer, McGraw Hill, New York, 1963.

[Fabrizio, 2002] Fabrizio, S. (2002). Machine Learning in Automated Text Categorization, ACM Computing Surveys, 34(1), pp. 1-47.

[Fajszi, 2010], Fajszi, B., Cser, L. és Fehér, T. (2010). Üzleti haszon az adatok mélyén, Alinea Kiadó – IOSYS Informatikai és Tanácsadó Zrt., 2010.

[Fan, 2006] Fan, W., Wallace, L., Rich, S. & Zhang, Z. (2006). Tapping into the Power of Text Mining, Communications of the ACM, 49(9): 76-82, 2006.

[Fawcett, 2003] Fawcett, T. (2003). ROC Graphs, Notes and Practical Considerations for Researchers, HP Labs Technical Report, HPL-2003-4, 2003.

[Fellbaum, 1998] Fellbaum, C. (1998). WordNet. An electronic lexical database. Ed. by Christiane Fellbaum, preface by George Miller. Cambridge, MA: MIT Pres.

[Fensel, 2001] Fensel, D., Harmelen, F., Horrocks, I. & McGuiness, D. (2001). OIL, An Ontology Infrastructure for the Semantic Web, IEEE Intelligent Sytems, pp. 38-45.

[Fogarasi, 2009] Fogarasi, I. (2009). Az e-learning technológiák nemzetközi piacának fejlődése és összefüggései a felsőoktatási implementációs tapasztalatokkal, stratégiákkal a 2001-2008 években, pp.1-113.

[Forgó, 2004] Forgó, S. Kis-Tóth L. és Hauser, Z. (2004). Tanulás tér- és időkorlátok nélkül. Iskolakultúra, 2004/12. sz. pp. 121-125.

[Forgó, 2005] Forgó, S. (2005). Az eLearning fogalma. In: Hutter Otttó – Magyar Gábor - Mlinarics József: E-LEARNING 2005 (eLearning kézikönyv), Műszaki könyvkiadó, 2005. pp. 14.

[Forgó, 2009] Forgó, S. (2009). Az új média és az elektronikus tanulás, In: Új Pedagógiai Szemle, 2009/8–9, pp. 91-97.

[Frank, 1999] Frank, E., Paynter, G.W., Witten, I.H., Gutwin, C. & Nevill-Manning, C.G. (1999). Domain-specific Keyphrase Extraction, Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence IJCAI, pp. 668-673.

[Funaoi, 2006] Funaoi, H., Akiyama, M., & Hirashima, T. (2006). Automatic Creation of Mis-choices and Comments for Multiple Choice Question Based on Problem Solving Model, ICCE 2006 Workshop Proc. of Problem-Authoring, -Generation and -Posing in a Computer-Based Learning Environment, pp. 49-54.

[Futó, 1999] Futó, I. (1999). Mesterséges Intelligencia. Aula Kiadó, Budapest, 1999.

[GermaNet, 2009] GermaNet (2009). GermaNet: Introduction, Eberhard Karls Universität Tübingen.
http://www.sfs.uni-tuebingen.de/GermaNet/

[Goble, 2005] Goble, C., Stevens, R. & Bechhofer, S. (2005). The Semantic Web and Knowledge Grids, Drug Discovery Today Technologies, pp. 225-233.

[Guangzuo, 2004] Guangzuo, C., Fei, C., Hu, C. & Shufang, L. (2004). OntoEdu: A Case Study of Ontology-based Education Grid System for E-Learning, Proc of GCCCE2004 International conference, Hong Kong, 2004.

[Gütl, 2008] Gütl, C. (2008). Automatic Limited-Choice and Completion Test Creation, Assessment and Feedback in modern Learning Processes, Paper read at LRN Conference 2008, Guatemala, February 12th – 16th.

[Gütl, 2010] Gütl, C., Lankmayr, K., & Weinhofer, J. (2010). Enhanced Approach of Automatic Creation of Test Items to Fos-ter Modern Learning Setting, in Proc. of the 9th European Conference on e-Learning, Porto, Portugal, 4-5 November 2010, pp. 225-234.

[Gütl, 2011] Gütl, C., Lankmayr, K., Weinhofer, J & Höfler, M. (2011), Enhanced Automatic Question Creator – EAQC: Concept, Development and Evaluation of an Automatic Test Item Creation Tool to Foster Modern e- Education, The Electronic Journal of e-Learning Volume 9 Issue 1 2011, (pp23-38), ISSN 1479-4403.

[Helic, 2000] Helic, D., Maurer, H. & Scherbakov, N. (2000). Web Based Training : What do We Expect from the System, Institute for Information Processing and Computer Supported New Media (IICM), Graz University of Technology.

[Holzinger, 2001] Holzinger, A. (2001). Multimedia Learning Systems based on IEEE Learning Object Metadata (LOM), Institute of Medical Informatics, Statistics and Documentation (IMI) Graz University, Austria.

[Hoshino, 2005] Hoshino, A., & Nakagawa, H. (2005). A Realtime Multiple-choice Question Generation for Language Testing: A Preliminary Study, Proc. of the 2nd Workshop on Building Educational Applications Using NLP, 17-20.

[Jain, 1988] Jain, A. K. & Dubes, R. C. (1988). Algorithms for Clustering Data, Prentice Hall Advanced Reference Series, 1988.

[Jain, 1999] Jain, A., Murty, M. & Flynn, P.J. (1999). Data Clustering: A review, ACM Computing Surveys, Vol. 31, No 3, 1999, pp. 264-321.

[Joachims, 1998] Joachims, T. (1998). Text Classifications with support vektors machines: Learning with many relevant features, In European Conference on Machine Learnings (ECML).

[Kirkpatrick, 1983] Kirkpatrick, C. D. & Gelatt, Jr., M. P. Vecchi (1983). Optimization by Simulated Annealing, Science, New Series, Vol. 220, No. 4598. (May 13, 1983), pp. 671-680.

[Kovács, 2006] Kovács, L. és Répási T. (2006). CLUSTERING BASED ON CONTEXT SIMILARITY, Proceedings of the TMCE 2006, April 18–22, 2006, Ljubljana, Slovenia.

[Kovács, 2009] Kovács, L., Barabás, P. és Répási, T. (2009). Ontology-based Semantic Models for Databases, Handbook of Research on Innovations in Database Technologies and Applications: Current and Future Trends, IGI Global Publisher, pp. 443-451.

[Kőfalvi, 2006] Kőfalvi, T. (2006). e-Tanítás. Információs és kommunikációs technológiák felhasználása az oktatásban. Nemzeti Tankönyvkiadó, Budapest.

[Krauth, 2007] Krauth, P. (2007). Számítógépes szövegelemzés, NHIT Információs Társadalom Technológiai Távlatai (IT3), Üzleti intelligencia, 3. kötet, pp. 177-179.

[Kudo, 2007] Kudo, T. (2007). CRF++: Yet Another CRF toolkit, http://crfpp.sourceforge.net/.

[Kupiec, 1995] Kupiec, J., Pederson, J. & Chen, F. (1995). A trainable document summarizer, Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval, pp. 68-73.

[Lafferty, 2001] Lafferty, J., McCallum, A., & Pereira. F. (2001). Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data, Proc. of ICML 2001, pp. 282-289.

[Lassila, (1999)] Lassila, O. & Swick, R. R. (1999). Resource Description Framework (RDF) Model and Syntax specification, W3C Recommodation.

[Lin, 2007] Lin, Y. C., Sung, L. C., & Chen, M. C. (2007). An Automatic Multiple-Choice Question Generation Scheme for English Adjective Understanding, ICCE 2007 Workshop Proc. of Modeling, Management and Generation of Problems / Questions in eLearning, pp. 137-142.

[LMS rendszerek, 2008] LMS rendszerek, (2008).
http://htmlinfo.hu/2008/07/14/nyilt-forraskodu-szoftverek-az-oktatasban/

[Luhn, 1957] Luhn, H. P. (1957). A statistical approach to mechanized encoding and searching of literary information, IBM J. Res. Dev. 1, 4 (October 1957), 309-317. DOI=10.1147/rd.14.0309. 1957.

[Manning, 2008] Manning, C., Raghavan, P., & Schütze, H. (2008). Introduction to Information Retrieval, Cambridge University Press, 2008.

[McCulloch, 1943] McCulloch, W. S. & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity, Bulletin of Mathematical Biophysics, 5:115-137.

[Miháltz, 2003] Miháltz, M. (2003). Magyar főnévi WordNet létrehozása automatikus módszerekkel. Első Magyar Számítógépes Nyelvészeti Konferencia (MSzNy-2003), Szeged, pp. 153-160.

[Miller, 1990] Miller, G., Beckwith, R., Fellbaum, C., Gross, D. & Miller, K. (1990). Five Papers on WordNet. CSL Report 43. Cognitive Science Laboratory. Princeton University.

[Miller, 1995] Miller, G. (1995). WordNet: a lexical database for English, Communication of the ACM, Vol. 38, 1995, pp. 39-41.

[Mitchell, 1997] Mitchell, T. M. (1997). Machine Learning, McGraw-Hill, Inc. New York, USA, 1997, ISBN:0070428077 9780070428072.

[Mitkov, 2003] Mitkov, R. & Ha, A. L. (2003). Computer-Aided Generation of Multiple-Choice Tests, Proceedings of the HLT-NAACL 2003 workshop on Building educational applications using natural language processing, pp. 17-22.

[Mitkov, 2003] Mitkov, R. & Ha, L. A. (2003). Computer-Aided Generation of Multiple-Choice Tests, Proc. of HLT-NAACL '03 Workshop: Building Educational Applications Using Natural Language Processing, pp. 17-22.

[Murty, 1980] Murty, M. & Krishna, G. (1980). A computati-onally efficient technique for data clustering, Pattern Recognition, Vol. 12., 1980, pp. 153-158.

[Nardi, 2002] Nardi, D. & Brachman R. J. (2002). An Introduction to Description Logics, In the Description Logic Handbook, edited by F. Baader, D. Calvanese, D.L. McGuinness, D. Nardi, P.F. Patel-Schneider, Cambridge University Press, pp. 5-44.

[Németh, 2003] Németh, L., (2003). A Szószablya fejlesztés, pp. 3-4.

[Nielsen, 2008] Nielsen, R. (2008). Question Generation: Proposed challenge tasks and their evaluation. Proceedings of the Workshop on the Question Generation Shared Task and Evaluation Challenge. NSF, Arlington.

[Papp, 2005] Papp, Gy. (2005). eLearning szabványok - Elemző tanulmány, In: Elearning szabványok. Javaslat az IHM által kiadott eLearning szabványajánlással kapcsolatos disszeminációra.

[Papp, 2005] Papp, Gy. és Vágvölgyi, Cs. (2005). Szabad forráskódú e-learning keretrendszerek összehasonlító elemzése, Kölcsey Ferenc Református Tanítóképző Főiskola, Debrecen, pp. 1-5.

[Pedersen and Bruce, 1997] Pedersen T. & Bruce, R. (1997). Distinguishing word senses in untagged text. In *Proceedings* of the Second Conference on Empirical Methods in Natural Language Processing, Providence, RI, August, pp. 197–207,.

[Pedersen and Bruce, 1998] Pedersen, T. & Bruce, R. (1998). Knowledge lean word sense disambiguation. In Proceedings of the Fifteenth National Conference on Artificial Intelligence, Madison, WI, July, pp. 800–805,.

[Piwek, 2008] Piwek, P., Prendinger, H., Hernault, H., & Ishuzuka, M. (2008). Generating questions: An inclusive characterization and a dialogue-based application. Workshop on the Question Generation Shared Task and Evaluation Challenge. NSF, Arlington.

[Piwek, 2009] Piwek, P., Mostow, J., Chali, Y., Forascu, C., Gates., D. (2009). The Question Generation Shared Task & Evaluation Challenge. In Workshop on the Question Generation Shared Task and Evaluation Challenge, Final Report, The University of Memphis: National Science Foundation.

[Porter, 2001] Porter, M.F. (2001). Snowball: A Language for Stemming Algorithms.

[Prószéky, 2008] Prószéky G. és Miháltz M. (2008). Magyar WordNet: Az első magyar lexikális szemantikai adatbázis. Magyar Terminológia, Vol. 1. Nr. 1., pp. 43-57.

[Purandare & Petersen 2004] Purandare & Petersen (2004). Word Sense Discrimination by Clustering Contexts in Vector and Similarity Spaces, Appears in the Proceedings of the Conference on Computational Natural Language Learning (CoNLL), pp. 41-48, May 6-7, 2004, Boston, MA.

[Ramasundaram, 2010] Ramasundaram, R. & Victor, S.P. (2010). Text Categorization by Backpropagation Network, International Journal of Computer Applications (0975 – 8887), Volume 8– No.6.

[Rijsbergen, 1979] Van Rijsbergen C.J. (1979). Information Retrieval: Butterworths, London, 2th Edition, 1979.

[Robert Hecht-Nielsen, 1987] Hecht-Nielsen, R. (1987). Counterpropagation Networks, Applied Optics, 26(23): 4979-4984, Dec. 1987.

[Robert Hecht-Nielsen, 1989] Hecht-Nielsen, Robert. (1989). Theory of the Backpropagation Neural Network, Washington, DC, USA.

[Rodney, 2008] Nielsen, R. (2008). Question Generation: Proposed challenge tasks and their evaluation. Proceedings of the Workshop on the Question Generation Shared Task and Evaluation Challenge. NSF, Arlington.

[Rosell, 2004] Rosell, M., Kann, V. & Litton, J. E. (2004). Comparing Comparisons: Document Clustering Evaluation Using Two Manual Classifications, Proceedings of the 3rd International Conference on Natural Language Processing, Hyderabad, India, pp. 207–216,.

[Rudin, 2003] Rudin, K. & Cressy, D. (2003). Will the Real Analytic Application Please Stand Up? DM Review, (13)3, pp. 30-34.

[Rus, 2007] Rus, V., Cai, Z. & Graesser, A. C. (2007). Experiments on Generating Questions About Facts, Computational Linguistics and Intelligent Text Processing, 8th International Conference, CICLing 2007, Mexiko City, Mexiko, February 18-24, 2007, Proceedings, pp. 444-455.

[Ruspini, 1969] Ruspini, E. (1969). A new approach to clustering, Information Control, Vol. 15., 1969, pp. 22-32.

[Russell Stuart, 2005] Stuart, R. (2005). Artificial Intelligence, A Modern Approach (2nd Edition), Rome: Pearson Italia, 2005.

[Salton, 1991] Salton, G. (1991). Developments in automatic text retrieval, Science, Vol. 253., 1991, pp. 974-980.

[Schluep, 2003] Schluep, S., Ravasio, P., Schär, S. G. (2003). Implementing Learning Content Management, Human-Computer Interaction – INTERACT'03.

[Schütze, 1998] Schütze H. (1998). Automatic word sense discrimination, Computational Linguistics, 24(1), pp. 97-123.

[Selman, 2006] Selman, B. & Gomes, C. P. (2006). Hill-climbing Search, Cornell University, Ithaca, New York, USA, pp. 333-336.

[Shin, 2003] Shin, Sa-Im & Choi, Key-Sun. (2003). Automatic Word Sense Clustering using Collocation for Sense Adaptation, Guseong-dong, Yuseong-gu, Daejeon, Republic of Korea.

[Sieber, T. 2006] Sieber, T. & Kammerer, M. (2006). Sind Metadaten bessere Daten?: Metadaten als Mittler zwischen Daten und Prozessen. Technische Kommunikation, 5(05/2006), pp. 56–58.

[Siemens, 2005] Siemens, G. (2005). Connectivism: A Learning Theory for the Digital Age, Instructional Technology and Distance Learning 2005, Vol. 2. No. 1. pp. 3-11.

[Sloman, 2005] Sloman, Q. Ni, M. (2005). An Ontology-enabled Service Oriented Architecture for Persative Computing, *Publ. of ITCC* Vol II, pp. 797-798.

[Snowball, 2005] Snowball (2005). The Snowball string processing language,
URL: http://snowball.tartarus.org/, 2005.

[Stuart, 2000] Stuart, J. R & Norvig, P. (2000). Mesterséges Intelligencia modern megközelítésben, Panem- Prentice-Hall, Budapest, 2000.

[Stuart, 2005] Stuart R. & Norvig, P. (2005). Mesterséges intelligencia modern megközelítésben, ISBN 963 545 411 2, Panem könyvkiadó, Budapest.

[Sumita, 2004] Sumita, E., Sugaya, F. & Yamamoto, S. (2004). Automatic Generation Method of a Fill-in-the-blank Question for Measuring English Proficiency, Technical report of IEICE, 104 (503), pp. 17-22.

[Tikk, 2002] Tikk, D. (2002). Szöveges dokumentumok osztályozása hierarchikus kategóriarendszerekben, In A Magyary Zoltán és az OTKA Posztdoktori Ösztöndíjasok Találkozója, Pécs, Hungary, November 14-15, pp. 24-27.

[Tikk, 2006] Tikk, D., Biró, Gy., Szidarovszky, F. P., Kardkovács, Zs. T., Héder M. és Lemák, G. (2006). Magyar internetes gazdasági tematikájú tartalmak keresése, In IV. Magyar Számítógépes Nyelvészeti Konferencia (MSZNY 2006), Szeged, pp. 3–14.

[Tikk, 2007] Tikk, D. (2007). Szövegbányászat, Typotex, Budapest, pp. 161 – 173.

[Timpf, 1996] Timpf, S. & Raubal, M. (1996). Experiences with Metadata. 7th Int. Symposium on Spatial Data Handling, SDH'96, Delft, The Netherlands (August 12-16, 1996), IGU, pp. 12B.31 - 12B.43.

[Tomlinson, 2004] Tomlinson, S. (2004). Finnish, Portuguese and Russian Retrieval with Hummingbird SearchServer at CLEF 2004, In Working Notes for the CLEF 2004 Workshop, Bath, UK, 2004, pp. 221-232.

[Tordai, 2005] Tordai, A. & Rijke, M. (2005). Hungarian Monolingual Retrieval at CLEF 2005, In Working Notes for the CLEF 2005 Workshop, Vienna, Austria, 2005.

[Tsuruoka, 2005] Tsuruoka, Y., & Tsujii, J. (2005). Bidirectional Inference with the Easiest-First Strategy for Tagging Sequence Data, Proc. of HLT/EMNLP, pp. 467-474.

[Vossen, 1999] Vossen, P. (1999). EuroWordNet General Document, Version 3. University of Amsterdam.

[Vas, 2007] Vas, R. F. (2007). Tudásfelmérést támogató oktatási ontológia szerepe és alkalmazási lehetőségei, BCE, Információrendszerek Tanszék.

[Walsh, 1999] Walsh, N. & Muellner, L., Stayton, B. (1999). DocBook, The Definitivee Guide, OReilly Publisher.

[Weaver, 2008] Weaver, D., Spratt, C. & Nair, C. S. (2008). Academic and student use of a learning management system: Implications for quality, Australasian Journal of Educational Technology, 2008, 24(1), pp. 30-41.

[WordNet, 2010] WordNet (2010). WordNet: A lexical database for English, Princeton University, http://wordnet.princeton.edu.

[Zhang, 1996] Zhang, T., Ramakrishnan, R. & Livny, M. (1996). BIRCH: An Efficient Data Clustering Method for Very Large Databases. Proc. of the 1996 ACMSIGMOD Int'l Conf. on Management of Data, Montreal, Canada, pp. 103–114.

[Zsebók, 2012] Zsebók, S. (2012). Gerincesek akusztikus vizsgálata, A gépi tanulás alkalmazásának lehetőségei, ELTE, Budapest, pp. 23-25.

Saját publikációk az értekezés témakörében

Idegen nyelvű folyóiratban közölt publikációk

[1] Bednarik, L. & Kovacs, L. (2012). Efficiency Analysis of Quality Threshold Clustering Algorithms, Production Systems and Information Engineering, University of Miskolc, Volume 6 (2012), ISSN 1785 – 1270, pp. 15-26.

[2] Kovacs, L. & Bednarik, L. (2011). Methodological Background of Course Material Ontology Models, NEWSLETTER, Precarpathian University PEDAGOGICS, Issue XXXVIII, Beregszász, Ukrajna, pp. 159-164.
Вестник Прикарпатского Университета 2011, Ивано-Франковск, Сборник научных трудов "Педагогика, Выпуск XXXVIII" Удк 11.1:37.013.2 страницы: 159-164.

Magyar nyelvű folyóiratban közölt publikációk

[3] Kovács, L. és Bednarik, L. (2012). Osztályozási feladatok a kérdésgenerálási mintarendszerben, A Gépipari Tudományos Egyesület Műszaki Folyóirata (GÉP), ISSN 0016 – 8572, LXIII. évfolyam, 2012/5, pp. 83-86.

[4] Bednarik, L. (2010**).** Digitális oktatási anyagok készítése és megjelenítése, Képzés és Gyakorlat Neveléstudományi szakfolyóirat, Célok és módszerek a tudásalapú társadalom nevelési intézményeiben**,** ISSN 1789 – 8587, Kaposvár, pp. 55-70.

[5] Bednarik, L. (2009). Mi a szövegbányászat?, 50 éves a sárospataki felsőfokú pedagógusképzés, ISSN 0230 – 0435, SPF 26, Sárospatak, pp. 131-135.

Idegen nyelvű konferencia kiadványban közölt publikációk

[6] Kovacs, L. & Bednarik, L. (2010), DITA Metadata and XSLT-FO in Document Management, XXIV**.** microCAD International Scientific Conference**,** ISBN 978-963-661-919-0**,** Miskolc, pp. 79-82.

[7] Kovacs, L. & Bednarik, L. (2011). Extension of HAC clustering method with quality threshold, 9th IEEE International Symposium on Intelligent Systems and Informatics (SISY 2011), Subotica, Serbia, 2011., 10.1109/SISY.2011.6034333, pp. 257 – 261.

[8] Kovacs, L. & Bednarik, L. (2011). Parameter Optimization for BIRCH Pre-Clustering Algorithm, 12th IEEE International Symposium on Computational Intelligence and Informatics (CINTI 2011), Budapest, Hungary, 10.1109/CINTI.2011.6108553, pp. 475 – 480.

[9] Kovacs, L. & Bednarik, L. (2012). Automated EA-type Question Generation from Annotated Texts, IEEE 7th International Symposium on Applied Computational Intelligence and Informatics (SACI 2012), Timisoara, Romania, 10.1109/SACI.2012.6250000, pp. 191 – 195.

Magyar nyelvű konferencia kiadványban megjelent publikációk

[10] Bednarik, L. (2010). Ontológiai technológiák és alkalmazásai, I. KÁRPÁT-MEDENCEI NEMZETKÖZI MÓDSZERTANI KONFERENCIA, Lehetőségek és alternatívák a Kárpát-medencében, ISBN 978-973-9541-13-9, Kaposvár, pp. 154-163.

[11] Bednarik, L. (2010**).** Dokumentumok meta-adatai és a DITA XML dokumentum-formátum**,** „Szellemi tőke, mint versenyelőny" konferencia**,** lektorált CD kiadvány, ISBN 978-963-216-270-6, Komárom, Szlovákia, pp. 212-227.

[12] Kovács, L. és Bednarik L. (2010), Digitális dokumentumok formátumai és az XSLT-FO, Multimédia az oktatásban konferencia, MTESZ Neumann János Számítógép-tudományi Társaság, In: Berke József (szerk), CD kiadvány, ISBN: 978 615 5036 04 0, Nyíregyháza, 2010.

[13] Bednarik, L. és Kovács, L. (2011). Tananyag-ontológia modell és a DITA keretrendszer, IV. MISKOLCI TANI-TANI KONFERENCIA, Pedagógia és társadalmi felelősségvállalás, ME BTK TANÁRKÉPZŐ INTÉZET, 2011.
http://www.tanarkepzo.hu/sites/default/files/

[14] Bednarik L. (2012). Automatizált kérdésgeneráló mintarendszer, Informatikai konferencia, ME Comenius Főiskolai Kar, Real Tudományok Intézete, Informatika, ME CFK Könyvtára, Konferencia-előadás kézirata, pp. 1-10.
http://www.ctif.hu/public/bednarikpublic.htm

[15] Bednarik, L. (2006). Az e-learning fázisainak Web-es megvalósítása az oktatásban, MAGYAR TUDOMÁNY NAPJA 2006 konferencia, ME Comenius Tanítóképző Főiskolai Kar, ME CFK Könyvtára, Konferencia-előadás kézirata, pp. 1-15.
http://www.ctif.hu/public/bednarikpublic.htm

Lektorált hazai kiadvány

[16] Bednarik, L. (2010). Oktatás- és információtechnika I., Miskolci Egyetem Comenius Tanítóképző Főiskolai Kar, ME CFK Könyvtára, Magyar Tudományos Művek Tára, pp. 1-56.

[17] Bednarik, L. (2004). Elektronikus tanulást segítő WEB-es mintarendszer kidolgozása, Miskolci Egyetem, Gépészmérnöki és Informatikai Kar, Általános Informatika Tanszék, Diplomamunka, Konzulens: Kovács., L., pp. 1-98.

Közlésre benyújtott, értékelés alatt álló publikációk

Kovacs, L., Gyöngyösi, E. & Bednarik, L. (2012). Development of classification module for automated question generation framework, Teaching Mathematics and Computer Science, Debrecen, Hungary, ISSN 1589 – 7389, *submitted.*

Kovacs, L. & Bednarik, L. (2012). Application of Semantic Clustering in Question Generation Engine, Transactions on Automatic Control and Computer Science, Scientific Bulletin of the "Politehnica" University of Timisoara, Romania, ISSN 1224-600X, *submitted*.

Ábrák jegyzéke

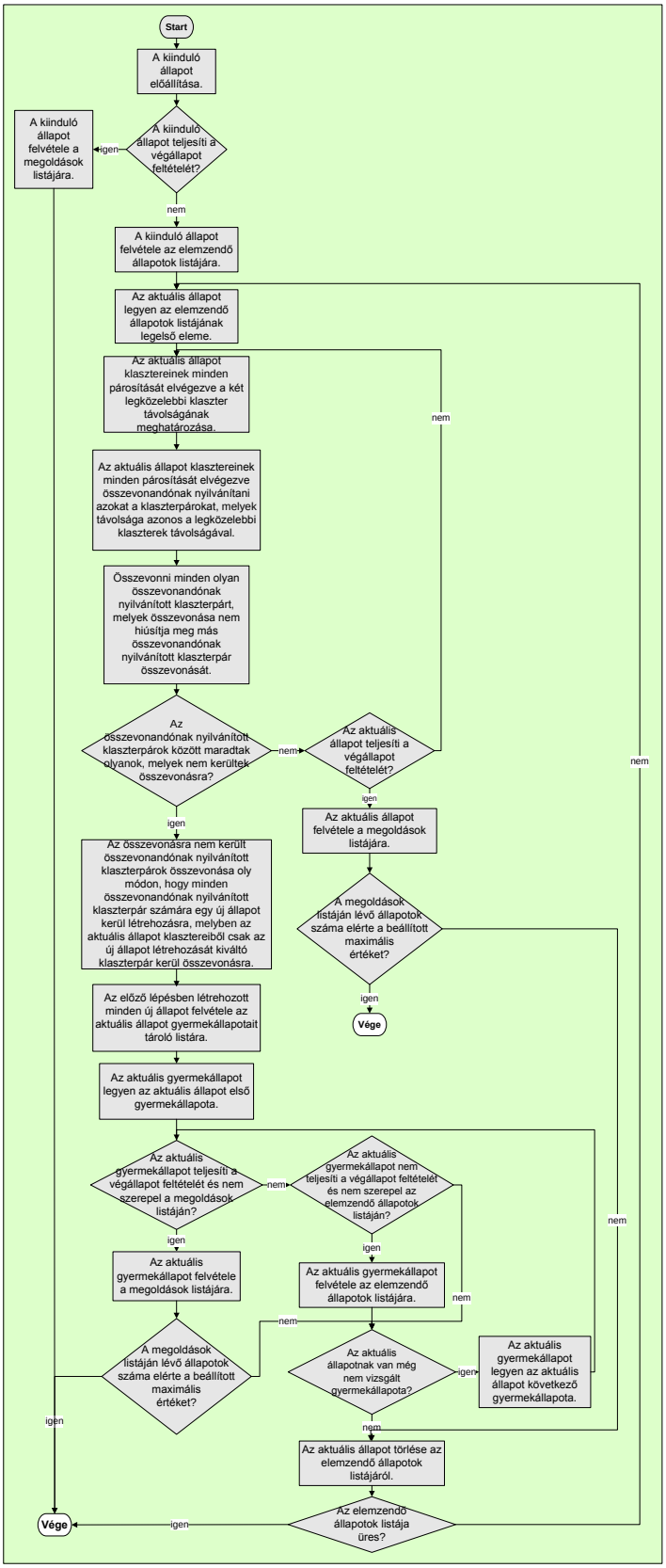
Táblázatok jegyzéke

Algoritmusok jegyzéke

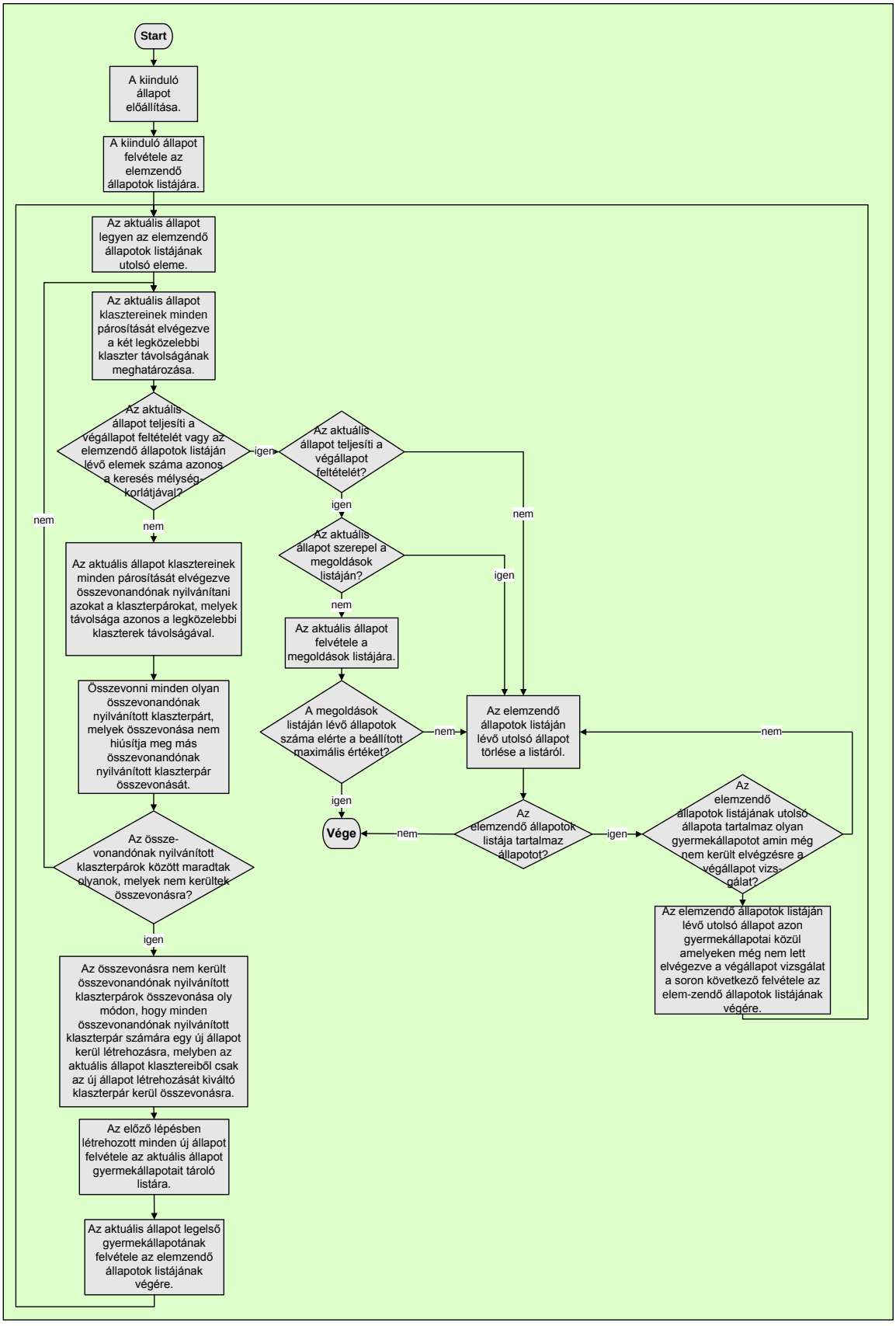
Függelék

A. Függelék

A.1. Szélességi kereséssel megvalósított HAC algoritmus folyamatábrája



A.2. Mélységi kereséssel megvalósított HAC algoritmus folyamatábrája



A.3. A BFS stratégiával megvalósított klaszterezés eredményei

A klaszterező algoritmus tesztelése során felhasznált hardver- és szoftverkörnyezet.

Dell Inspiron N5010 notebook paraméterei:

- gyártó: DELL,
- processzor: Intel Core i3 350M,
- memória: 4096MB,
- merevlemez: 320GB,
- optikai meghajtó: DVD-RW +/-,
- kijelző: 15.6”,
- grafikus vezérlő: ATI Radeon HD5470.

Szoftverkörnyezet:

- Windows7 Ultimate, 64 bit,
- Eclipse fejlesztőkörnyezet.

1. futtatás eredményei (30 kérdés esetén)

Klaszterezés futási idő: 10062ms.
A klaszterezendő szavak száma: 261
Az első megoldás klasztereinek száma: 30
Az első megoldás epszilon (0.25) átmérőjű klasztereiben lévő szavak: [
cluster:1 {rovására, energianövekedést, démon, gázmolekulákra, vonatkozó, birtokában, érhette, információ, }
cluster:2 {letárolva, adatok, lazább, szerkezetben, szigorúbb, finomabb, struktúrában, }
cluster:3 {leggyakoribb, művelet, felsoroltak, lekérdezés, }
cluster:4 {legközelebb, felhasználóhoz, DBS-en, alkalmazói, segédprogramok, állnak, }
cluster:5 {írni, programot, jelentette, megváltozott, fizikai, struktúra, }
cluster:6 {őket, komponensek, létrehozásának, idői, elkülönülnek, helyről, szerezzük, }
cluster:7 {alkalmazásokat, Elsőként, kiválasztása, történik, segítségével, létrehozni, épülő, adatbázist, }
cluster:8 {nézetet, tartalmazhat, adatbázisrendszerhez, alkalmazó, felhasználó, kapcsolódhat, }
cluster:9 {előforduláshoz, miközben, előfordulást, tárolhat, egyed, rekord, adatbázis, tartozik, }
cluster:10 {fejlesztőknek, készült, következő, csoportnak, alkalmazás, tervezőknek, segédprogramok, adatbázis, }
cluster:11 {viszonyítva, relációs, adatmodell, rugalmasabb, szerkezet, biztosít, elődeihez, }
cluster:12 {egyszerűségének, rugalmasságának, köszönhető, modell, elterjedése, }
cluster:13 {kód, hibamentességének, ellenőrzése, Tesztelés, elkészült, }
cluster:14 {haszon, elhanyagolhatóan, kicsi, elméletileg, hasznos, információ, alkalmazható, felhasználásához, nagyon, hosszú, idő, szükséges, elérhető, }
cluster:15 {marad, rendelő, számláját, kiegyenlítette, vehetjük, példaként, kereskedő, szállít, kért, áruból, mennyiség, raktáron, }
cluster:16 {születési, hely, fizetés, dolgozónál, ugyanaz, érték, dolgozók, adatait, leíró, rekordban, }

cluster:17 {egyértelmű, azonosítására, rekordkulcsnak, kulcsnak, nevezzük, tulajdonságot, tulajdonságcsoportot, egyedisége, alkalmas, egyed, rekord, }
cluster:18 {tőle, elemek, helyezkedhetnek, szekvencia, előnye, fájlt, végignézni, kulcsértékű, keresésekor, sorrendbe, rendezés, kulcstól, jobbra, szükséges, rekord, }
cluster:19 {hierarchikus, index, listák, kiemelkedő, szerepet, játszik, B+, fa, }
cluster:20 {látjuk, igaz, fog, adatbázist, jelenteni, lényegében, áttolja, definíciót, adatbáziskezelőre, tűnhet, adatbázis, }
cluster:21 {alkalmazásnak, szüksége, adatra, adat, fájlban, letárolásra, került, hagyományos, kezelésnél, }
cluster:22 {statikus, védelem, eszköze, mentés, dinamikus, védelemhez, naplózás, tartozik, }
cluster:23 {adatbáziskezelésben, adhatjuk, megvalósítandó, rendszer, leírását, adatbáziskezelőnek, adatmodellekről, megjegyezhetjük, központi, játszanak, szerepet, }
cluster:24 {elemi, lépés, eredményének, ismeretében, modulban, döntéshozatal, szűkebb, hatáskörben, komponensnél, bizonyos, műveleteknél, algoritmusváltozat, kiválasztása, történik, }
cluster:25 {interaktív, eszközöktől, kezdve, programozási, segédeszközökig, tartozó, programok, köre, széles, területet, ölel, hagyományos, }
cluster:26 {gyenge, pontjai, költségtényezők, technológiai, megkötöttségek, kiválasztott, ütemezési, módszer, alapján, optimalizálható, gyártás, felmérhetők, gyártási, esetleges, rendszer, }
cluster:27 {szemantikai, kifejezés, önálló, modellt, jelöl, modellcsaládot, különálló, beletartozik, adatmodell, modell, }
cluster:28 {halmazát, tartalmazza, csoportképzés, csoportot, képez, megadott, típusú, egyedekből, csoportképzéssel, megalkotott, egyedtípus, előfordulása, alaptípus, előfordulásának, }
cluster:29 {típusra, típus, előfordulására, értéket, meta, típust, jelent, előfordulásai, típusok, típushoz, hozzárendelhető, tulajdonságérték, előforduláshoz, érték, egyedtípus, }
cluster:30 {trigger, eseménylekezelő, mechanizmus, két, elemből, esemény, választevékenység, megadása, }]
Az első megoldás epszilon (0.25) átmérőjű klasztereiben lévő legtávolabbi szavak távolsága: 0.33
Az első megoldás epszilon (0.25) átmérőjű klasztereiben lévő szavak átlagos száma: 9.366666666666667

2. futtatás eredményei (40 kérdés esetén)

Klaszterezés futási idő: 10608ms.
A klaszterezendő szavak száma: 241
Az első megoldás klasztereinek száma: 30
Az első megoldás epszilon (0.25) átmérőjű klasztereiben lévő szavak: [
cluster:1 {komponenssel, kommunikál, felhasználó, }
cluster:2 {rétege, oldala, programok, alkalmazói, }
cluster:3 {energia, ekvivalenciáját, szemlélteti, átalakult, energiává, információ, }
cluster:4 {előny, szerzésére, kellő, időben, megszerzett, felhasználható, információ, }
cluster:5 {elterjedt, módszere, titkosítás, hozzáférési, védelem, }
cluster:6 {felkereshető, összes, blokk, címének, ismeretében, sorban, }
cluster:7 {jellegű, rekord, logikai, fájlszerkezet, alapvetően, stream, }
cluster:8 {egyedieknek, lenniük, mezők, egyed, tulajdonságai, melyeknek, }
cluster:9 {ábrázolását, tárolását, Adatbázisokon, voltaképpen, adatoknak, kapcsolataikkal, }
cluster:10 {integritási, szabály, módosításakor, automatikusan, ellenőrzi, sérült-e, adatbázis, }
cluster:11 {szinteket, szokás, nézeteknek, nevezni, }
cluster:12 {történik, szinteken, nézetek, megadása, különböző, adatmodellek, segítségével, }

cluster:13 {szerkezettel, könyvtípussal, írunk, rendszerben, könyvet, }
cluster:14 {metaadatokat, jelentés, űrlap, formátumának, meghatározásánál, fejlesztő, felhasználja, tárolt, rendszer, adatbázisban, }
cluster:15 {természetesen, specifikumait, figyelembe, szofverfejlesztés, metodikája, általános, }
cluster:16 {modell, absztrakciós, megvalósítandó, adatbázist, szinteken, különböző, }
cluster:17 {alkalmas, adatmodell, matematikai, formalizmus, valóság, adatorientált, leírására, }
cluster:18 {fontos, jellemzője, felhasználó, számítógépen, tárolt, adatot, használ, számítástechnika, fejlődésének, }
cluster:19 {értelmezésére, megközelítés, általunk, előbb, megadott, értelmezés, információ, fogalma, alapfogalomnak, tekinthető, }
cluster:20 {listák, hierarchia, szintjén, helyezkedik, B+, fa, előnye, közvetlenül, rekordokra, mutató, listája, levél, }
cluster:21 {relációs, adatbáziskezelés, induló, időszak, hierarchikus, hálós, adatmodelljei, években, indult, hódító, útjára, legelterjedtebb, adatbáziskezelő, típus, }
cluster:22 {belső, karbantartási, funkcióinak, végrehajtása, rendszer, programrendszer, feladata, adatbázishoz, történő, hozzáférések, biztosítása, adatbázis, adatbáziskezelő, }
cluster:23 {elvégzi, megfelelő, módosításokat, olvasási, műveleteket, eredményt, továbbítja, alkalmazói, adatbázis, program, }
cluster:24 {helyezkedhetnek, adathordozón, adatfüggetlenség, következő, adatbázisokban, megvalósuló, szintje, mezőszintű, fizikai, adatbázisban, rekordba, tartozó, fizikailag, szétszórtan, mezők, }
cluster:25 {munkát, végezniük, biztosítsák, hatékony, működését, egységből, felépülő, gyárhoz, hasonlítható, egységeknek, összehangolt, különböző, rendszer, }
cluster:26 {Adatvédelmi, adatsérülések, leállások, okozta, veszteségek, minimalizálása, adatok, védelmének, különböző, rendszer, feladata, biztosítása, megfelelő, }
cluster:27 {eszközökkel, jelentéskészítő, komponensek, tipikus, alkalmazás, felépítését, követik, ezekhez, igazítottak, nehéz, speciális, követelményekhez, igazodó, alkalmazások, elkészítése, űrlap, }
cluster:28 {szoftverfejlesztési, irányelvek, érvényesek, program, adatbázisrendszerek, tervezésének, vizsgálatakor, tényből, kiindulni, futó, software, termék, általános, számítógépen, }
cluster:29 {előfordulásának, halmazát, tartalmazza, csoportképzés, csoportot, képez, típusú, egyedekből, csoportképzéssel, megalkotott, egyedtípus, előfordulása, alaptípus, megadott, }
cluster:30 {létező, tulajdonságértéktől, függ, érték, tulajdonságok, megadására, ad, fogalom, lehetőséget, értéke, műveletsorral, határozódik, }]
Az első megoldás epszilon (0.25) átmérőjű klasztereiben lévő legtávolabbi szavak távolsága: 0.25
Az első megoldás epszilon (0.25) átmérőjű klasztereiben lévő szavak átlagos száma: 8.866666666666667

3. futtatás eredményei (50 kérdés esetén)

Klaszterezés futási idő: 12211ms.
A klaszterezendő szavak száma: 248
Az első megoldás klasztereinek száma: 31
Az első megoldás epszilon (0.25) átmérőjű klasztereiben lévő szavak: [
cluster:1 {információ, }
cluster:2 {újdonság, fontos, eleme, }
cluster:3 {objektív, feldolgozótól, független, adat, }
cluster:4 {Nagyon, információhoz, jutunk, nap, }

cluster:5 {rovására, energianövekedést, démon, gázmolekulákra, vonatkozó, birtokában, érhette, }
cluster:6 {hatalmas, károkat, okozhat, elvesztése, megfelelően, }
cluster:7 {mezők, összessége, rekord, logikailag, összetartozó, adatelemek, }
cluster:8 {tekinthető, leghatékonyabb, módszernek, indexelési, módszer, }
cluster:9 {csoportja, alkalmazásfejlesztők, köre, felhasználóknak, tágabb, }
cluster:10 {szolgálnak, adatmodelleknek, informatikában, modelleket, struktúrájának, leírására, nevezik, adatok, }
cluster:11 {programozó, tömbben, letárolva, jelennek, adatok, }
cluster:12 {szintek, intenzív, kapcsolat, fenn, }
cluster:13 {külső, sémájában, fogalmazza, kiad, utasítást, felhasználó, alkalmazás, }
cluster:14 {sémát, használhat, időben, utasítás, keletkezhet, mindegyike, külső, }
cluster:15 {szoftverfejlesztő, cég, független, cégekhez, tartozik, gyártók, }
cluster:16 {operátorai, típuskonstruktorok, kiemelt, szerepet, játszik, csoportképzés, aggregáció, }
cluster:17 {szükséges, elérhető, haszon, elhanyagolhatóan, kicsi, elméletileg, hasznos, alkalmazható, felhasználásához, hosszú, idő, Nagyon, }
cluster:18 {nevezzük, adat, tények, fogalmak, feldolgozásra, alkalmas, reprezentációja, hordozóját, adatnak, }
cluster:19 {fejlesztő, eszközök, használatát, teszik, szükségessé, programfejlesztés, rugalmasság, gyorsaság, igényei, hatékony, }
cluster:20 {kulcsát, tartalmazza, listában, kulcsértékek, rendezetten, helyezkednek, legegyszerűbb, indexszerkezet, indexlista, összes, rekord, }
cluster:21 {alkalmazói, segédprogramok, együttesét, adatbázisrendszernek, nevezik, rövidítésére, használják, adatbáziskezelő, rendszer, }
cluster:22 {megvalósulnak, mai, hagyományos, fájlkezelésnél, szintű, függetlenség, valósul, állományt, sima, rekordsorozatnak, képzelhetjük, valójában, mezőit, megadni, fizikailag, rekord, }
cluster:23 {bizonyos, mértékben, módosítható, programokat, módosítani, kellene, letárolt, logikai, adatmodell, bővíthető, alkalmazói, }
cluster:24 {adatrendszert, adatbáziskezelésnél, felhasználó, szemszögéből, tekintve, adatmodellben, felhasználónak, törődnie, fizikai, tárolás, részleteivel, magasabb, absztrakciós, szinten, értelmezheti, tárolódnak, adatok, }
cluster:25 {alkalmazás, kézben, tudja, tartani, adatintegritás, megőrzését, adathozzáférés, biztosítását, }
cluster:26 {adatmodellt, készíteni, modellezett, valóság, adatmodelljét, hozni, adatbázist, választani, megfelelő, szükséges, alkalmazói, programokat, }
cluster:27 {kötődnek, Személyileg, szétválnak, komponensek, kezelésével, megbízott, posztok, szükség, többek, rendszeradminisztrátorra, operátorra, kezeléséhez, programozók, felhasználók, alkalmazásokhoz, }
cluster:28 {hardware, fölött, elhelyezkedő, operációsrendszer, szolgáltatásait, adatbázishoz, adathordozón, adatszerkezethez, történő, hozzáféréshez, felhasználhatja, külső, letárolt, fizikai, }
cluster:29 {szolgál, keresést, kulcsszavak, megadásával, könnyíteni, Különböző, szöveges, dokumentumokban, információk, karbantartására, visszakeresésére, letárolt, }
cluster:30 {előfordulása, alaptípus, előfordulásának, halmazát, csoportot, képez, megadott, típusú, egyedekből, csoportképzéssel, megalkotott, egyedtípus, csoportképzés, tartalmazza, }
cluster:31 {blokkokban, tárolódnak, blokk, adatátviteli, egységnek, fogható, írás, olvasás, művelete, minimum, adatmennyiséget, mozgat, adathordozó, központi, egység, fájlokon, adatok, }]
Az első megoldás epszilon (0.25) átmérőjű klasztereiben lévő legtávolabbi szavak távolsága: 0.25
Az első megoldás epszilon (0.25) átmérőjű klasztereiben lévő szavak átlagos száma: 8.741935483870968

A.4. Tesztadatok automatikus generálása

A megvalósított klaszterező eljárások teljesítményének összehasonlításához az algoritmusoknak nagyszámú tesztadaton való lefuttatása szükséges. A teszteket képező definíciós mondatok összeszerkesztése azonban időigényes feladat. Szükség volt tehát olyan szoftvermodul (tesztrendszer) kidolgozására, mely képes automatikusan előállítani az algoritmusok teszteléséhez szükséges nagyszámú és változatos adatokat. A tesztrendszerben négydimenziós pontokkal lettek modellezve a definíciós mondatok szavai. A pontokat valós koordinátákkal egy egységnyi kiterjedésű négydimenziós térben helyez el a szoftver. A pontok egyenletes eloszlás szerinti véletlenszerű felvétele azonban nem reprezentálja helyesen a modellezni kívánt szavak valóságos elhelyezkedését. Ennek oka, hogy a valóságos mondatok esetén a szavak egymáshoz viszonyított távolsága a mondatokban való közös előfordulásuk gyakoriságának a függvénye. Ezáltal a mondatok szavainak egymáshoz viszonyított távolsága inhomogén. Ennek az inhomogenitásnak a megőrzése érdekében a tesztadatok generálása során a szavakat reprezentáló pontok egyenletes eloszlás helyett több elszórtan elhelyezkedő fókuszpont körül a normális eloszlás szabályai szerint kerülnek elhelyezésre.

A tesztadatokat automatikusan generáló szoftvermodul bemenetként kapja a klaszterezendő pontok számát, a fókuszpontok számát valamint a normális eloszlás szórását. A tesztadatok előállításának első lépéseként a fókuszpontok kerülnek elhelyezésre a négydimenziós térben az egyenletes eloszlás szabályai szerint. Ezt követően a további pontok elhelyezéséhez a szoftvermodul sorra veszi a fókuszpontokat és mindegyik esetén meghatározza a köré elhelyezendő további pontok darabszámát. Ennek a számnak a meghatározásához különböző stratégiák lehetségesek. A megvalósításra került stratégia minden fókuszpont köré közel azonos számú pontot helyez el. Ezek pontos száma minden fókuszpont esetén a még elhelyezendő pontok számának és a még nem lezárt fókuszpontok számának hányadosával kerül meghatározásra. A megadott számú pont elhelyezését követően a szoftver a Pitagorasz-tétel alkalmazásával kiszámítja minden pont távolságát az összes többi ponttól. Ezek a távolságértékek képezik a távolságmátrix egyes elemeit, melyek a klaszterezés megkezdése előtt még normalizálásra kerülnek a [0, 1] valós intervallumra.

Mivel a BFS, illetve a mélységkorlátos melységben először kereső algoritmusokkal implementált klaszterezés az (5.6) összefüggés klaszterekre való kiterjesztésével speciálisan mondatok szavainak klaszterezésére lett kifejlesztve ezért nem alkalmazható egymástól független pontok klaszterekbe sorolására. A BFS algoritmussal implementált korongos lefedési módszer azonban pusztán a klaszterezendő elemek távolságának ismeretében is képes megoldani a feladatot. Az automatikusan generált tesztadatok klaszterezésére tehát kizárólag ez utóbbi módszer alkalmazható. A tesztek során a lefedő korongok átmérője 0.25 értékre lett beállítva a [0, 1] valós intervallumban, ahol 0 jelenti a legkisebb, 1 pedig a legnagyobb távolságot. Ez az átmérő reprezentálja a klaszterek magjának átmérőjét. A tesztek lefuttatásra kerültek 100, 500, 1000 valamint 5000 pont; 5, 10, 20 fókuszpont körüli 0.1, illetve 0.2 szórású normális eloszlás alapján számított elhelyezése esetén.

A BFS kereső algoritmussal implementált korongos lefedési módszer futási eredményei automatikusan generált tesztadatok esetén az 1. táblázat foglalja össze.

a klasztermag átmérője [0..1 közötti arányszám]	0,25	0,25	0,25	0,25	0,25	0,25	0,25	0,25	0,25	0,25	0,25	0,25
klaszterezendő pontok száma [darab]	100,00	500,00	1000,00	5000,00	100,00	500,00	1000,00	5000,00	100,00	500,00	1000,00	5000,00
fókuszpontok száma [darab]	5,00	5,00	5,00	5,00	10,00	10,00	10,00	10,00	20,00	20,00	20,00	20,00
normális eloszlás szórása	0,10	0,10	0,10	0,10	0,10	0,10	0,10	0,10	0,10	0,10	0,10	0,10
futási idő [ms]	310,00	2964,00	6537,00	1203744,00	291,00	4447,00	10457,00	608912,00	134,00	7173,00	15799,00	646563,00
klaszterek száma [darab]	19,00	45,00	51,00	411,00	18,00	54,00	57,00	282,00	33,00	70,00	84,00	208,00
a klasztermagokban lévő legtávolabbi elemek távolsága [0..1 közötti arányszám]	0,25	0,25	0,25	0,25	0,25	0,25	0,25	0,25	0,25	0,25	0,25	0,25
a klasztermagokban lévő pontok átlagos száma	13,26	69,98	166,25	472,28	10,11	39,70	98,91	425,08	5,39	22,51	66,25	289,19
a klasztermagok kétszeres átmérőjű körzetében lévő legtávolabbi elemek távolsága [0..1 közötti arányszám]	0,50	0,50	0,50	0,50	0,50	0,50	0,50	0,50	0,50	0,50	0,50	0,50
a klasztermagok kétszeres átmérőjű körzetében lévő pontok átlagos száma	28,16	171,02	406,20	1251,54	33,50	148,96	309,54	1368,00	18,24	99,61	356,85	1853,71
a klasztermag átmérője [0..1 közötti arányszám]	0,25	0,25	0,25	0,25	0,25	0,25	0,25	0,25	0,25	0,25	0,25	0,25
klaszterezendő pontok száma [darab]	100,00	500,00	1000,00	5000,00	100,00	500,00	1000,00	5000,00	100,00	500,00	1000,00	5000,00
fókuszpontok száma [darab]	5,00	5,00	5,00	5,00	10,00	10,00	10,00	10,00	20,00	20,00	20,00	20,00
normális eloszlás szórása	0,20	0,20	0,20	0,20	0,20	0,20	0,20	0,20	0,20	0,20	0,20	0,20
futási idő [ms]	287,00	10245,00	67999,00	1957262,00	972,00	6204,00	50089,00	2116411,00	389,00	8469,00	52491,00	2217102,00
klaszterek száma [darab]	35,00	101,00	198,00	641,00	37,00	100,00	186,00	636,00	41,00	112,00	203,00	677,00
a klasztermagokban lévő legtávolabbi elemek távolsága [0..1 közötti arányszám]	0,25	0,25	0,25	0,25	0,25	0,25	0,25	0,25	0,25	0,25	0,25	0,25
a klasztermagokban lévő pontok átlagos száma	6,26	34,42	42,98	300,02	5,03	31,22	42,53	255,09	4,93	24,82	39,42	176,04
a klasztermagok kétszeres átmérőjű körzetében lévő legtávolabbi elemek távolsága [0..1 közötti arányszám]	0,50	0,50	0,50	0,50	0,50	0,50	0,50	0,50	0,50	0,50	0,50	0,50
a klasztermagok kétszeres átmérőjű körzetében lévő pontok átlagos száma	29,06	193,25	288,05	2313,15	20,57	204,09	332,72	2520,29	29,90	235,85	373,71	2071,99

1. táblázat: A BFS kereső algoritmussal implementált korongos lefedési módszer futási eredményei

A.5. A BIRCH alapú klaszterezés eredményeinek elemzése

A BIRCH alapú klaszterező algoritmus tesztelése során a főbb jellemzők alakulását a következő hat paraméter változtatása esetén vizsgáltam:

- a BIRCH fa levél-csomópontjaiban lévő pontok maximális távolsága (küszöbérték),
- a BIRCH fa nemlevél-csomópontjainak maximális elágazási tényezője,
- a BIRCH algoritmusra épülő k-közép algoritmustól várt klaszterek száma,
- a klaszterezendő pontok száma,
- a klaszterezendő pontok csoportosulásainak száma (fókuszpontok),
- a klaszterezendő pontoknak a fókuszpontok körüli normális eloszlás szerinti szórása.

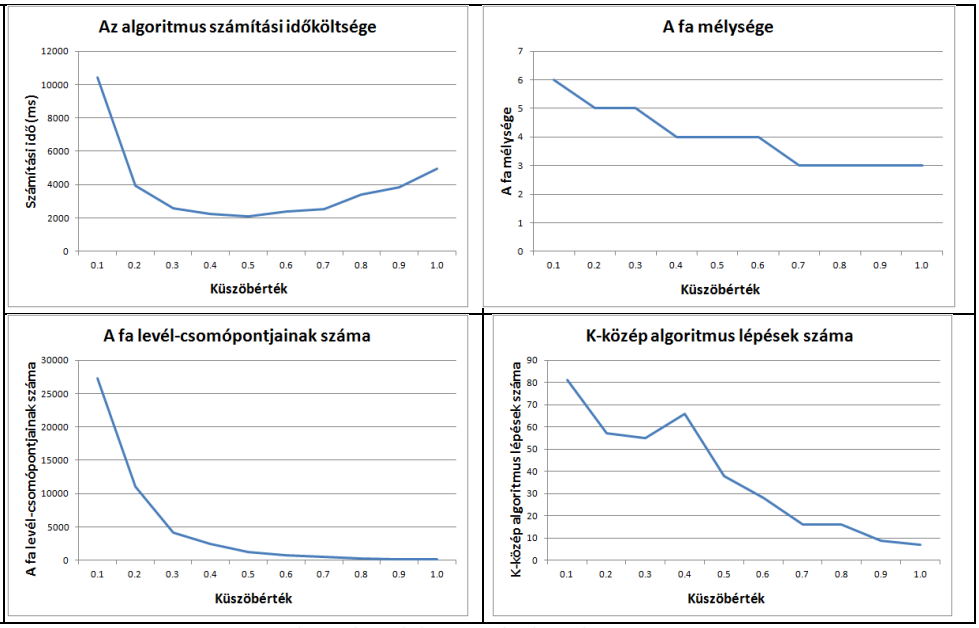
A paraméterek közül az első kettő közvetlenül a BIRCH algoritmus működésére van hatással. A harmadikkal a BIRCH algoritmus elő-klaszterezésének eredményeként kapott halmazok feladatspecifikus klaszterezését végző k-közép algoritmustól várt klaszterek száma állítható be. Az utolsó három paraméterrel a klaszterezendő pontokat automatikusan generáló szoftvermodul működése hangolható. A tesztek során a paraméterek közül mindig csak egy került változtatásra, hogy annak hatása az algoritmus jellemzőire könnyebben kimutatható legyen. A küszöbérték paraméter [0, 1] közötti arányszámként lett beállítva, mivel a klaszterezendő pontokat előállító szoftvermodul is ebben a tartományban állítja elő a pontokat. Ezek ismeretében meghatározásra került:

- a klaszterezés időigénye,
- a BIRCH fa mélysége,
- a BIRCH fa levél-csomópontjainak száma,
- a BIRCH algoritmusra épülő k-közép alapú klaszterezés elvégzéséhez szükséges klaszterátszervezési lépések száma.

Ezek alapján végzett tesztek eredményei a következő diagramokon látható.

A küszöbérték hatásának vizsgálata:

- *küszöbérték:* [0.1 ... 1.0]
- *maximális elágazási tényező:* 10
- *klaszterek száma:* 10
- *klaszterezendő pontok száma:* 50000
- *fókuszpontok száma:* 8
- *normális eloszlás szórása:* 0.2

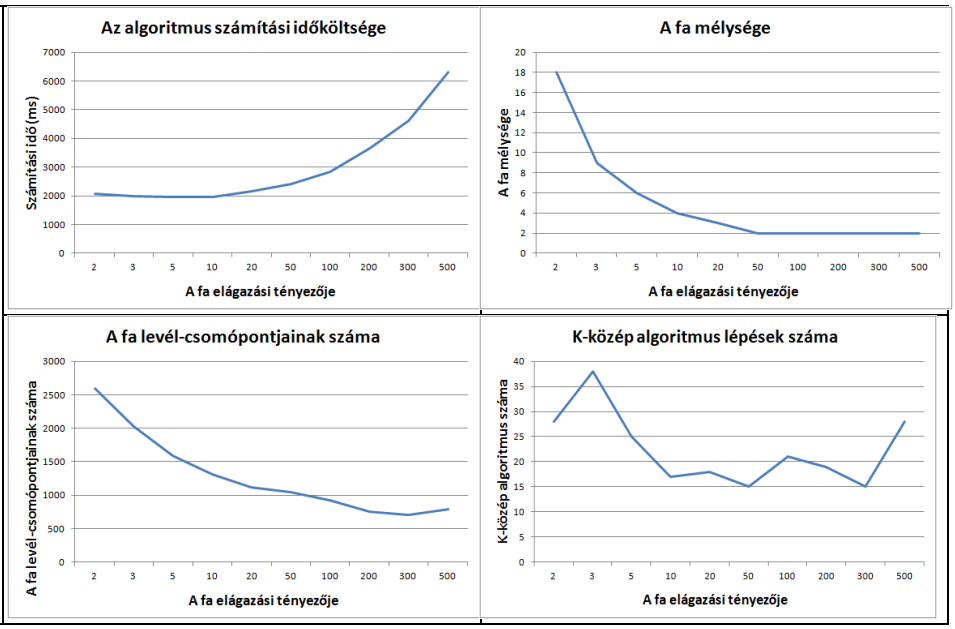


1. ábra: A küszöbérték hatásának vizsgálata

A teszt alapján jól látható, hogy a küszöbérték növelésével exponenciálisan csökken a levél-csomópontok száma. A *k*-közép algoritmusnak jóval kevesebb halmazt kell klasztereznie, ezáltal szintén exponenciálisan csökken a klaszterezés időigénye. A küszöbértéknek egy bizonyos szint fölé emelésével azonban már olyan sok pont gyűlik össze a levél-csomópontokban, hogy a halmazok szétszakadása során már jelentős számú pont elemzése válik szükségessé. Ennek hatása az időigény növekedésében is kimutatható.

A maximális elágazási tényező hatásának vizsgálata:

- *küszöbérték:* 0.5
- *maximális elágazási tényező:* [2 ... 500]
- *klaszterek száma:* 10
- *klaszterezendő pontok száma:* 50000
- *fókuszpontok száma:* 8
- *normális eloszlás szórása:* 0.2

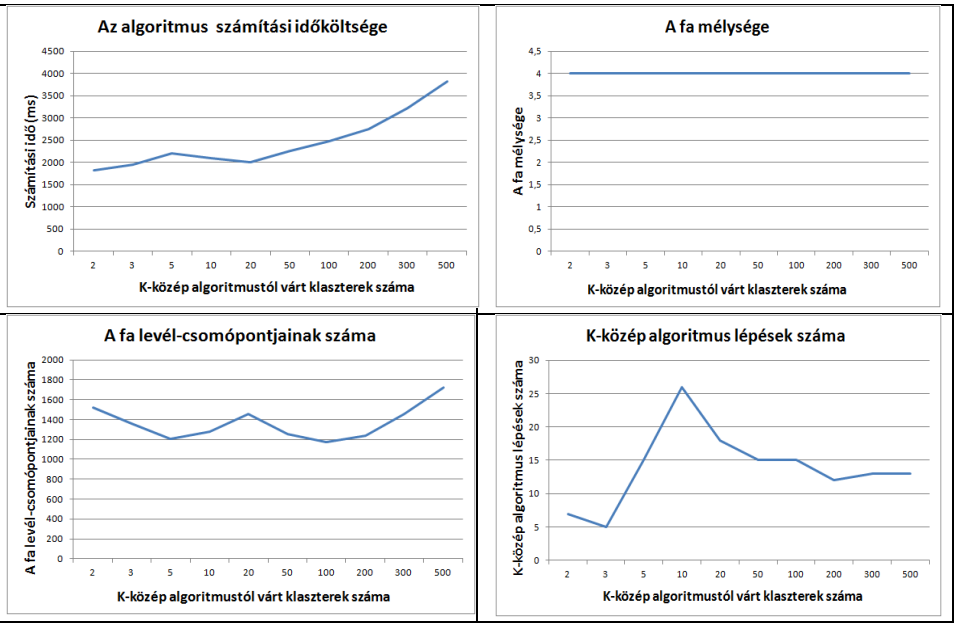


2. ábra: A maximális elágazási tényező hatásának vizsgálata

A fa elágazási tényezőjének növelésével a fa mélysége exponenciálisan csökken. Mivel nagyobb számú elágazás esetén minden pont elhelyezésénél jóval több gyermek csomópont közül kell meghatározni a legjobb menetiránynak számító legközelebbi halmazt. Ez a folyamat jól látható sebességnövekedést eredményez. Az is megfigyelhető, hogy az elágazási tényező változtatása a k-közép algoritmus működésére nincs hatással.

A klaszterszám hatásának vizsgálata:

- *küszöbérték:* 0.5
- *maximális elágazási tényező:* 10
- *klaszterek száma:* [2 ... 500]
- *klaszterezendő pontok száma:* 50000
- *fókuszpontok száma:* 8
- *normális eloszlás szórása:* 0.2

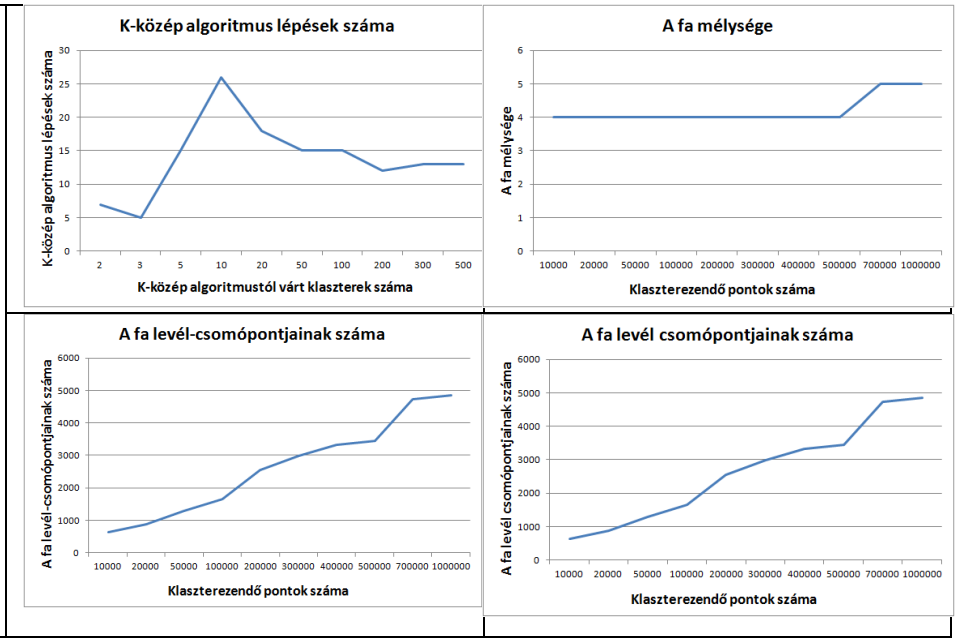


3. ábra: A klaszterszám hatásának vizsgálata

A tesztekből nyilvánvalóvá vált, hogy az előzetes becslésekkel megegyezően a k-közép algoritmustól várt klaszterszám növelése némiképp növeli a futási időt, ám a vizsgált többi tényezőre semmilyen hatással sincs.

A klaszterezendő pontok számának hatása:

- *küszöbérték:* 0.5
- *maximális elágazási tényező:* 10
- *klaszterek száma:* 10
- *klaszterezendő pontok száma:* [10000 ... 1000000]
- *fókuszpontok száma:* 8
- *normális eloszlás szórása:* 0.2

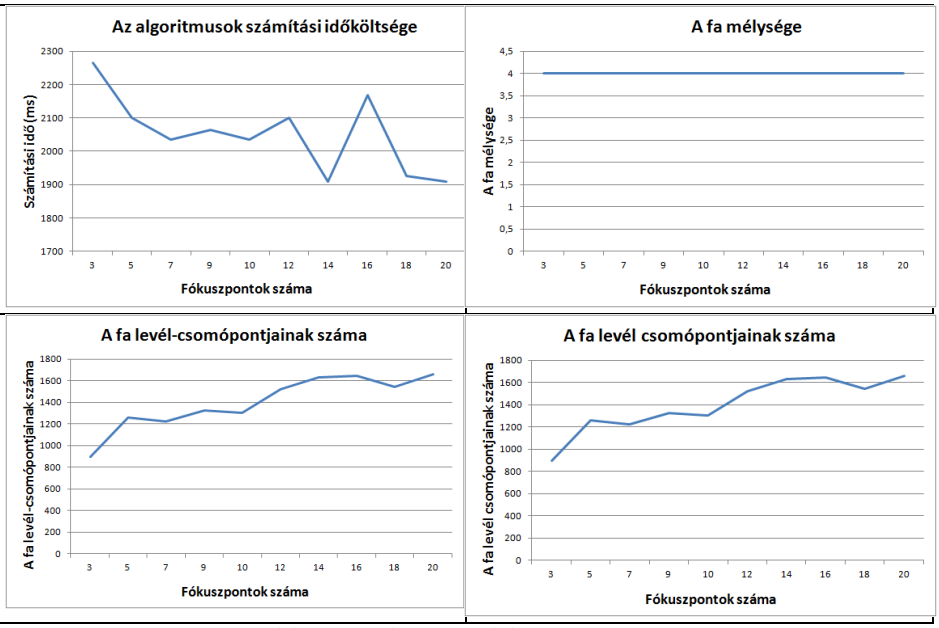


4. ábra: A klaszterezendő pontok számának hatása

A klaszterezendő pontok számának növelése – az elvárásokkal megegyezően – mind a négy vizsgált tényezővel szemben nagyobb igényeket támaszt.

A fókuszpontok számának hatása:

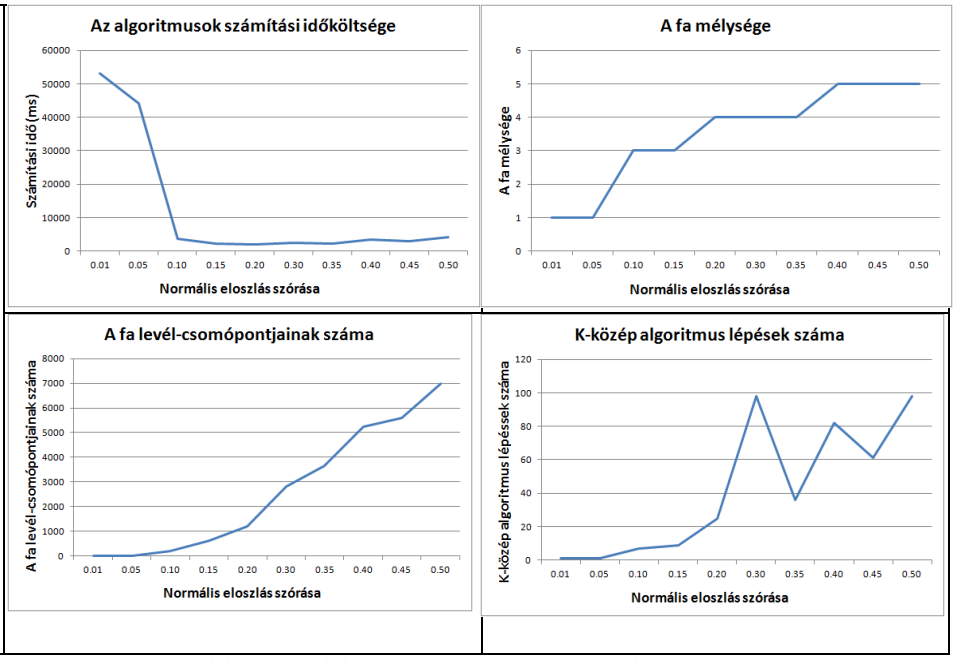
- *küszöbérték:* 0.5
- *maximális elágazási tényező:* 10
- *klaszterek száma:* 10
- *klaszterezendő pontok száma:* 50000
- *fókuszpontok száma:* [3 ... 20]
- *normális eloszlás szórása:* 0.2



5. ábra: A fókuszpontok számának hatása

A fókuszpontok körüli szórás hatásának vizsgálata:

- *küszöbérték:* 0.5
- *maximális elágazási tényező:* 10
- *klaszterek száma:* 10
- *klaszterezendő pontok száma:* 50000
- *fókuszpontok száma:* 8
- *normális eloszlás szórása:* [0.01 ... 0.5]



6. ábra: A fókuszpontok körüli szórás hatásának vizsgálata

Érdemes megvizsgálni még a normális eloszlás szórásának hatását, melyből kiderül, hogy a jobban szétszórt pontok halmazokba gyűjtéséhez több levél-csomópontra és azáltal – adott maximális elágazási szám esetén – nagyobb mélységű fára van szükség. Ha a szórás értéke nagyságrendileg nagyobb a küszöbértéknél, akkor a levél-csomópontokban sok pont halmozódik fel, melyek szétbontásakor az új halmazokba rendezésük időigénye jelentőssé válik.

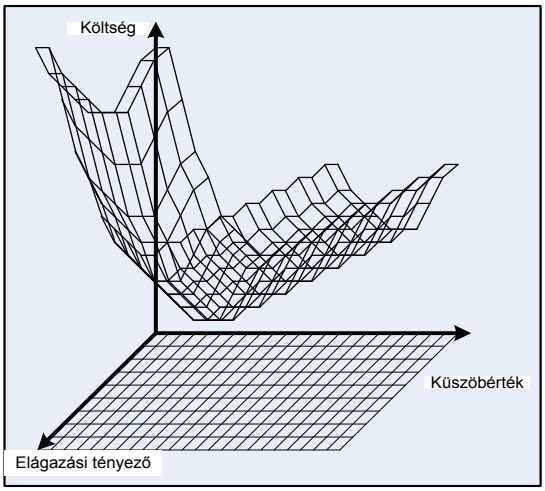
A diagramok látványosan alátámasztják a folyamat várt viselkedését. A kisebb ingadozások oka egyrészt a számítógép tesztenként különböző melléktevékenységeiből adódó leterheltségből ered, másrészt a klaszterezendő pontokat előállító véletlenszám-generátor tesztenként különböző eredményei okozzák.

A.6. A BIRCH algoritmus továbbfejlesztési lehetőségei

A BIRCH algoritmus alapú klaszterezéssel végzett tesztek eredményeinek alapos tanulmányozását követően felfedeztem néhány lehetőséget, melyekkel speciális esetekben az algoritmus hatékonysága javítható. Ezek közül a legígéretesebb irányokat az algoritmus főbb paramétereinek szoftveresen támogatott beállítása, illetve a nagy elágazási tényezőjű fában való keresés időigényének csökkentése jelentik.

Az algoritmus főbb paramétereinek szoftveresen támogatott beállítása

A BIRCH algoritmusra épülő klaszterező szoftverrel végzett tesztek kimutatták, hogy a feladat költségigénye legnagyobb mértékben az algoritmus küszöbérték paraméterének értékétől függ. A költséget kisebb mértékben ugyan, de szintén jelentősen befolyásoló tényező a CF fa maximális elágazási tényezője. Ezeknek a paramétereknek a klaszterezés költségére gyakorolt hatását az 1. ábra szemlélteti.



1. ábra: A küszöbérték és a maximális elágazási tényező paraméterek hatása a klaszterezés költségigényére

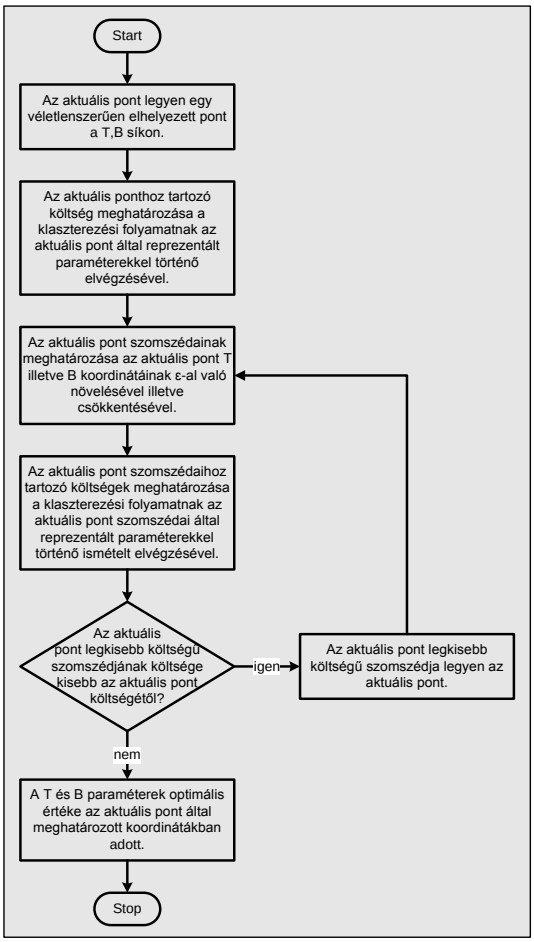
A kísérletek igazolták, hogy a küszöbérték paraméter optimális értékétől negatív irányba való eltérésével exponenciálisan megnő a BIRCH algoritmus által eredményezett halmazok száma. Ennek oka, hogy a kisebb küszöbérték minden halmaz esetén a klaszterezendő pontok nagyobb közelségét követeli meg. Ez a követelmény csak a halmazok számának növelésével teljesíthető. Az exponenciálisan növekedő halmazszám szintén exponenciális költségnövekedést eredményez a BIRCH algoritmus elő-klaszterezésére épülő k-közép algoritmusnál is. A küszöbérték paraméter optimális értékétől pozitív irányban eltérve növekszik a halmazokba kerülő pontok száma, melyek a halmazokat reprezentáló levél-csomópontok szétbontásakor igényelnek folyamatosan növekvő többletköltséget.

A tesztek azt is kimutatták, hogy a küszöbérték paraméter mellett az algoritmus maximális elágazási tényezőjének is nagy hatása van a klaszterezés költségigényére. A maximális elágazási tényezőnek az optimális értékétől negatív irányba való eltérésével exponenciálisan nő a fa mélysége. A levél-csomópontok számának állandó értéke mellett ez a nemlevél-csomópontok számának lényeges növekedését eredményezi. A megnövekedett számú csomópontok adminisztrálása (létrehozás, szétbontás, memóriafoglalás stb.) növekvő költséget jelent, melynek mértéke a paraméter optimális értékétől való távolság függvényében nő. A maximális elágazási tényezőnek az optimális értékétől pozitív irányba való eltérése a fa minden nemlevél-csomópontja esetén növekvő számú gyermek-csomópont megjelenését eredményezi. Nagyobb számú gyermek-

csomópont esetén minden új pont elhelyezéséhez minden nemlevél-csomópontnál arányosan nagyobb számú alternatíva közül kell kiválasztani azt a gyermek-csomópontot, amely CF értéke alapján a legközelebb van az elhelyezendő ponthoz. Ez a szintenként arányosan növekvő számítási igény a teljes fára nézve a mélységek hatványával arányos költséget igényel minden új pont elhelyezésénél.

A tesztek eredményeinek elemzése alapján látható, hogy a klaszterezési folyamat hatékonyságának maximálása érdekében az említett két paraméter optimális értékének ismerete kulcsfontosságú. A paraméterek optimális értékei azonban feladatspecifikusak. Mivel nagymértékben függnek a klaszterezendő pontok számától és elhelyezkedésétől, ezért nem határozható meg olyan konkrét érték, mely minden feladat esetén minimum-közeli költséget eredményez. Ezáltal a paraméterek értékeit minden konkrét feladat esetén egyedileg kell meghatározni a minimális költségű klaszterezés eléréséhez. Az optimális értékeik meghatározása ugyanannak a klaszterezési feladatnak az egymást követő többszöri elvégzését igényli a paraméterek különböző értékei esetén. Ezáltal a paraméterek optimális értékeinek meghatározáshoz szükséges költség általában többszöröse a klaszterezés egyszeri elvégzésével járó költségnek. Az optimális értékek feltárása tehát csak azokban az esetekben jelent valódi költségcsökkenést, amikor darabszám, tartomány, szórás szempontjából hasonló pontokból álló, egymástól független nagyszámú minta klaszterezése a feladat.

A „hegymászó algoritmus” folyamatábrája a 2. ábrán látható.

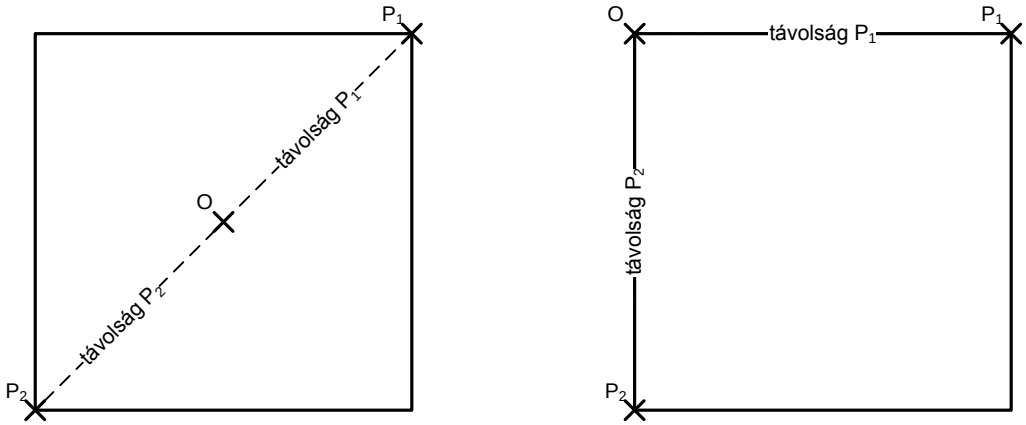


2. ábra: A paraméterek optimális értékének meghatározásához javasolt „hegymászó algoritmus” folyamatábrája (T: küszöbérték; B: elágazási tényező)

A szakirodalomban lokális kereső algoritmusok néven ismert módszerek mindegyike alkalmas az említett paraméterek optimális értékeinek meghatározására. A BIRCH algoritmussal végzett tesztek eredményei alapján azonban megállapítható, hogy az optimálási tartomány nem tartalmaz lokális szélsőértékeket. Mindkét paraméter esetén az érték megváltoztatása egyértelműen mutatja a változtatás irányának megfelelő változást a költség értékében.

Nagy elágazási tényezőjű fában való keresés időigényének csökkentése

A tesztek eredményeit szemléltető diagramokon jól látható, hogy a fa maximális elágazási tényezőjének növelésével egyrészt jelentősen csökken a fa mélysége (mely a nemlevél-csomópontok számának csökkenését is eredményezi), másrészt jelentősen csökken a fa levél-csomópontjainak száma. Mindkét csomóponttípus számbeli csökkenésének a költség egyértelmű csökkenésében is meg kell nyilvánulnia. A tesztek eredményei azonban azt mutatják, hogy a maximális elágazási tényező növelésével az algoritmus futási ideje – a csomópontok számának csökkenése ellenére – nő. A költségnövekedés oka a BIRCH algoritmus működési elvében rejlik. Az algoritmus minden egyes új pont elhelyezéséhez a fa gyökér-csomópontjától kiindulva folyamatosan küldi az elhelyezendő pontot egyre alacsonyabb szintekre a fában, míg végül az a neki megfelelő levél-csomópontba kerül. Az új pont minden szinten ahhoz a gyermek-csomóponthoz kerül továbbításra, amelyik CF értékéhez a legközelebb van. Látható, hogy nagyszámú elágazás (azaz a gyermek-csomópont) esetén minden szinten jelentős költségnövekedést eredményez a helyes út meghatározása. Jelentkezik tehát az igény, hogy a gyermek-csomópontok valamilyen feladatspecifikusan definiált sorrend szerint legyenek tárolva a szülő csomópontjuknál, hogy a legközelebb távolság meghatározásához ne legyen szükség mindegyikük elemzésére. A sorba-rendezés helyes kritériumának meghatározása azonban a feladat többdimenziós jellege miatt nem kivitelezhető. Ennek okát a 3. ábra szemlélteti.



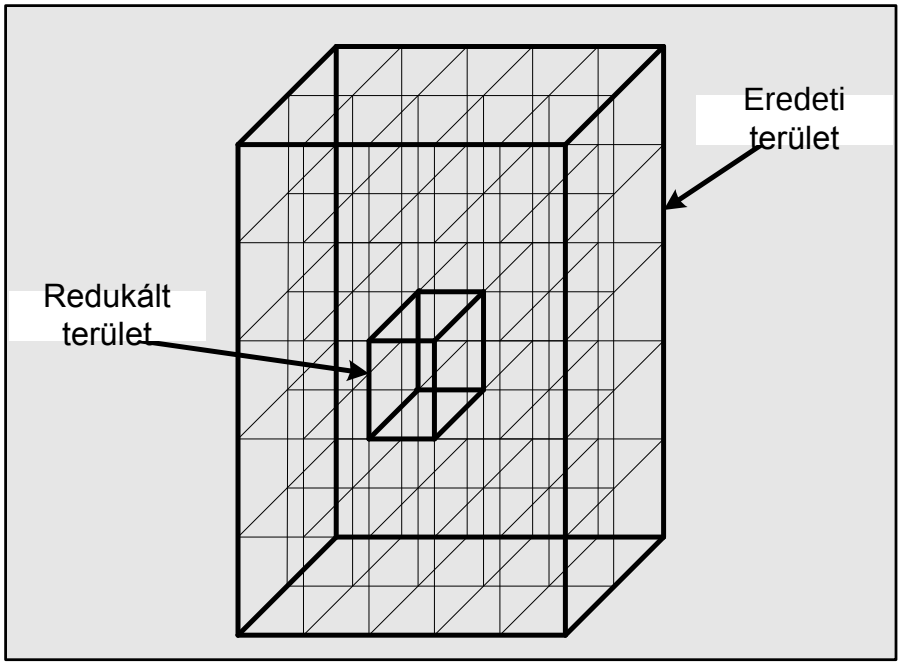
a, a távolságmérés kiinduló pontja a tartomány középpontja

b, a távolságmérés kiinduló pontja a tartomány bal felső pontja

3. ábra: A pontok sorba-rendezésének problémája többdimenziós terek esetén

Függetlenül a távolságmérés kiindulópontjának kijelölésétől egymással azonos távolságra lévő pontok valóságos elhelyezkedése egymástól nagyon távol is lehet. Többdimenziós terekben ez a tény indokolja a térfelosztásra épülő algoritmusok használatát. A konkrét klaszterezési feladat megoldására javasolt térfelosztási algoritmus lényege, hogy a klaszterezendő pontokat tartalmazó tér minden dimenziója mentén egyenlő lépésközzel felosztásra kerül. A lépésköz dimenziónként akár eltérő is lehet. Ezek a lépésköz-értékek képezik a térfelosztásos algoritmus paramétereit. A térnek ez a felosztása egy tömbbel kerül reprezentálásra, melyet a CF fa minden nemlevél-csomópontja tárol. A tömb minden rekesze csomópontok listáját tartalmazza, mely listán lévő

csomópontok CF értéke az adott cella által reprezentált térrészben helyezkedik el. Minden új elhelyezendő pont érkezése esetén egyszerű matematikai kerekítési műveletekkel meghatározhatóak a tömb azon rekeszének koordinátái melyben lévő csomópontok esélyesek az érkező pont továbbküldésére. Ezáltal a BIRCH algoritmusból ismert iránymeghatározási folyamatot már csak az adott cellában lévő csomópontokra kell elvégezni. Az így redukált keresési tartomány mérete az eredeti tartománynak az egyes dimenziók menti felbontások szorzataiból számítható hányada.



4. ábra: A térfelosztással kapott keresési tartomány aránya a teljes térhez képest
Az egyes dimenziók 5, 2, illetve 5 részre kerültek felosztásra tehát a térfelosztással kapott keresési tartomány $\frac{1}{50}$-ed része a térfelosztás előtti tartománynak

A térfelosztással dolgozó BIRCH algoritmus nagy jelentősége, hogy a lépésköz-értékekkel beállítható az algoritmus hatékonyság és költség viszonyának optimális értéke. Abban a szélső esetben, amikor a lépésköz-érték a klaszterezendő pontokat tartalmazó tér minden dimenziója mentén megegyeznek az adott dimenzió kiterjedésével, akkor az algoritmus a teret egyetlen cellaként kezeli. Ez a térfelosztás nélkül implementált BIRCH algoritmus esetével egyezik meg. A lépésköz-értékek csökkentésével a tér egyre kisebb részekre kerül felosztásra, amivel jelentősen csökken az egy térrészbe kerülő gyermek-csomópontok száma. Ezzel arányosan csökken az elhelyezendő új ponthoz legközelebb lévő gyermek-csomópont kiválasztásának időköltsége is. A lépésköz-értékek csökkentésével azonban jelentősen nő a teret felosztó mátrix mérete, amely a memóriaigényben jelent lényeges többletköltséget. Ezek alapján látható, hogy lépésköz-értékek leghatékonyabb beállítása egy optimálási feladat megoldását igényli. Erre szintén alkalmazható a küszöbérték és a maximális elágazási tényező optimális értékének meghatározásánál ismertetett algoritmus.

A.7. Annotált szöveges dokumentum

1. Információ és adat fogalmai
A számítástechnika fejlődésének egyik fontos jellemzője, hogy egyre több felhasználó egyre több, számítógépen tárolt adatot használ fel yzq3. Az egyre növekvő információmennyiség egyre szélesebb körben válik elérhetővé yzq3. Az információ-hozzáférés így jelentkező demokratizálódásának hatása teljes joggal mérhető a könyvnyomtatás jelentőségéhez, ezért szokás ezt a jelenséget elektronikus Gutenberg forradalomnak is nevezni yzq3.
Az elektronikus Gutenberg forradalom legfontosabb jellemzője, hogy
* egyre nagyobb információmennyiség
* egyre szélesebb tömegek számára
* egyre demokratikusabban válik elérhetővé
* a számítógép használatával.
Az elkészített és alkalmazott számítógépi programrendszereknek növekvő adatmennyiséggel kell megbirkózniuk yzq3. A hétköznapjainkban is egyre gyakrabban találkozhatunk a számítógépes információs rendszerek alkalmazásával yzq3. Számítógépes információs rendszer fut az üzemekben, gyárakban a termelés irányítására, a pénzügyi, személyzeti, raktári, anyaggazdálkodási feladatok elvégzésére yzq3.
A felsorolásban gyakran találkozhattunk két, egymással igen rokon értelmű szóval e rendszerek jellemzésében, az információ és az adat fogalmaival. Az információt és adatot gyakran használják azonos értelemben, pedig e két fogalom között létezik jelentésbeli különbség, melyre oda kell figyelni, s célszerű tudatosítani magunkban e fogalmak pontos jelentését yzq3.
1.1. Az információ
Az információt jelsorozathoz kapcsolódó új jelentésnek, hasznos közlésnek tekinthetjük yzq1. Az információ fogalma alapfogalomnak tekinthető, értelmezésére is igen sokféle megközelítés létezik, melyek közül az egyik az általunk megadott értelmezés yzq1. Az információ egyik fontos eleme az *újdonság* yzq1. Információ lehet például egy újsághír a minket érintő törvényváltozásokról yzq3. Nagyon sokféle módon, sokféle információhoz jutunk nap mind nap yzq3.
Az információ, tehát mindig szubjektív fogalom, függ a feldolgozótól yzq3. Az információt igen sok oldalról lehet vizsgálni és elemezni yzq3.
A szemantikai oldal
A szemantikai oldal az információt hordozó jelek előfordulási gyakoriságait vizsgálja yzq3. E terület elméletének kidolgozása Shannon nevéhez kötődik yzq3. Statisztikai értelmében egy jelsorozat információtartalma a jelsorozat előfordulási valószínűségének reciprokával arányos yzq2. Erre egy szemléletes példa, hogy az nem hír, ha egy kutya megharapja a postást, de az már hírnek számít, ha egy postás harapja meg a kutyát.
A szintaktikai oldal a jelsorozatok formális azonosságait vizsgálja, mely alapján a Jani szereti a sört mondat hasonló felépítésű a Jani szereti Marit mondattal. Az adatok viszont a feldolgozás során nyernek értelmet, s ekkor már a két mondat egészen más hatást válthat ki. Az adatok ezen rejtett tartalma a szemantikai oldal, mely a jelsorozat mögött húzódó jelentést, lényeget hangsúlyozza yzq1.

Az annotált mondatok jelölésére felhasznált leíró kód:
ahol: $yzq1$ fogalom, $yzq2$ definíció, $yzq3$ kijelentő mondat.

B. Függelék

B.1. Az osztályozás hatékonyságának mérőszámai és eredményei

Az osztályozás hatékonyságának mérőszámai és eredményei 30 kérdés esetén

Relevancia:
- Helyesen osztályozott igaz állítások száma (TP) : 158
- Hibásan osztályozott igaz állítások száma (FP) : 0
- Helyesen osztályozott hamis állítások száma (TN) : 632
- Hibásan osztályozott hamis állítások száma (FN) : 0
- Az osztályozott állítások száma: 790
- Érzékenység (R) : 1.0
- Pontosság (P) : 1.0
- F-mérték (F) : 1.0
- Szabatosság : 158.8
- Fals pozitív arány (FPR) : 0.0
- Specifikusság (SPC) : 1.0
- Negatív prediktív érték (NPV) : 1.0
- Hibás találatok aránya (FDR) : 0.0

Specialitás:
- Helyesen osztályozott igaz állítások száma (TP) : 147
- Hibásan osztályozott igaz állítások száma (FP) : 9
- Helyesen osztályozott hamis állítások száma (TN) : 623
- Hibásan osztályozott hamis állítások száma (FN) : 11
- Az osztályozott állítások száma: 790
- Érzékenység (R) : 0.930379746835443
- Pontosság (P) : 0.9423076923076923
- F-mérték (F) : 0.9363057324840763
- Szabatosság : 147.7886075949367
- Fals pozitív arány (FPR) : 0.014240506329113924
- Specifikusság (SPC) : 0.9857594936708861
- Negatív prediktív érték (NPV) : 0.9826498422712934
- Hibás találatok aránya (FDR) : 0.057692307692307696

Értelmesség:
- Helyesen osztályozott igaz állítások száma (TP) : 158
- Hibásan osztályozott igaz állítások száma (FP) : 0
- Helyesen osztályozott hamis állítások száma (TN) : 632
- Hibásan osztályozott hamis állítások száma (FN) : 0
- Az osztályozott állítások száma: 790
- Érzékenység (R) : 1.0
- Pontosság (P) : 1.0
- F-mérték (F) : 1.0
- Szabatosság : 158.8
- Fals pozitív arány (FPR) : 0.0

- Specifikusság (SPC) : 1.0
- Negatív prediktív érték (NPV) : 1.0
- Hibás találatok aránya (FDR) : 0.0

Nehézség:
- Helyesen osztályozott igaz állítások száma (TP) : 158
- Hibásan osztályozott igaz állítások száma (FP) : 0
- Helyesen osztályozott hamis állítások száma (TN) : 632
- Hibásan osztályozott hamis állítások száma (FN) : 0
- Az osztályozott állítások száma: 790
- Érzékenység (R) : 1.0
- Pontosság (P) : 1.0
- F-mérték (F) : 1.0
- Szabatosság : 158.8
- Fals pozitív arány (FPR) : 0.0
- Specifikusság (SPC) : 1.0
- Negatív prediktív érték (NPV) : 1.0
- Hibás találatok aránya (FDR) : 0.0

Helyettesítésre kerüljön-e:
- Helyesen osztályozott igaz állítások száma (TP) : 154
- Hibásan osztályozott igaz állítások száma (FP) : 1
- Helyesen osztályozott hamis állítások száma (TN) : 157
- Hibásan osztályozott hamis állítások száma (FN) : 4
- Az osztályozott állítások száma: 316
- Érzékenység (R) : 0.9746835443037974
- Pontosság (P) : 0.9935483870967742
- F-mérték (F) : 0.9840255591054312
- Szabatosság : 154.49683544303798
- Fals pozitív arány (FPR) : 0.006329113924050633
- Specifikusság (SPC) : 0.9936708860759493
- Negatív prediktív érték (NPV) : 0.9751552795031055
- Hibás találatok aránya (FDR) : 0.0064516129032258064

A teljes osztályozásra összesített adatok:
- Helyesen osztályozott igaz állítások száma (TP) : 775
- Hibásan osztályozott igaz állítások száma (FP) : 10
- Helyesen osztályozott hamis állítások száma (TN) : 2676
- Hibásan osztályozott hamis állítások száma (FN) : 15
- Az osztályozott állítások száma: 3476
- Érzékenység (R) : 0.9810126582278481
- Pontosság (P) : 0.9872611464968153
- F-mérték (F) : 0.9841269841269842
- Szabatosság : 775.7698504027618
- Fals pozitív arány (FPR) : 0.003723008190618019 5
- Specifikusság (SPC) : 0.996276991809382
- Negatív prediktív érték (NPV) : 0.9944258639910813
- Hibás találatok aránya (FDR) : 0.012738853503184714

Az osztályozás hatékonyságának mérőszámai és eredményei 40 kérdés esetén

Relevancia:
- Helyesen osztályozott igaz állítások száma (TP) : 207
- Hibásan osztályozott igaz állítások száma (FP) : 21
- Helyesen osztályozott hamis állítások száma (TN) : 875
- Hibásan osztályozott hamis állítások száma (FN) : 17
- Az osztályozott állítások száma: 1120
- Érzékenység (R) : 0.9241071428571429
- Pontosság (P) : 0.9078947368421053
- F-mérték (F) : 0.915929203539823
- Szabatosság : 207.78125
- Fals pozitív arány (FPR) : 0.0234375
- Specifikusság (SPC) : 0.9765625
- Negatív prediktív érték (NPV) : 0.9809417040358744
- Hibás találatok aránya (FDR) : 0.09210526315789473

Specialitás:
- Helyesen osztályozott igaz állítások száma (TP) : 208
- Hibásan osztályozott igaz állítások száma (FP) : 19
- Helyesen osztályozott hamis állítások száma (TN) : 877
- Hibásan osztályozott hamis állítások száma (FN) : 16
- Az osztályozott állítások száma: 1120
- Érzékenység (R) : 0.9285714285714286
- Pontosság (P) : 0.9162995594713657
- F-mérték (F) : 0.9223946784922394
- Szabatosság : 208.78303571428572
- Fals pozitív arány (FPR) : 0.021205357142857144
- Specifikusság (SPC) : 0.9787946428571429
- Negatív prediktív érték (NPV) : 0.9820828667413214
- Hibás találatok aránya (FDR) : 0.08370044052863436

Értelmesség:
- Helyesen osztályozott igaz állítások száma (TP) **:** 211
- Hibásan osztályozott igaz állítások száma (FP) : 16
- Helyesen osztályozott hamis állítások száma (TN) : 880
- Hibásan osztályozott hamis állítások száma (FN) : 13
- Az osztályozott állítások száma: 1120
- Érzékenység (R) : 0.9419642857142857
- Pontosság (P) : 0.9295154185022027
- F-mérték (F) : 0.9356984478935697
- Szabatosság : 211.78571428571428
- Fals pozitív arány (FPR) : 0.017857142857142856
- Specifikusság (SPC) : 0.9821428571428571
- Negatív prediktív érték (NPV) : 0.9854423292273237
- Hibás találatok aránya (FDR) : 0.07048458149779736

Nehézség:
- Helyesen osztályozott igaz állítások száma (TP) : 211
- Hibásan osztályozott igaz állítások száma (FP) : 12
- Helyesen osztályozott hamis állítások száma (TN) : 884
- Hibásan osztályozott hamis állítások száma (FN) : 13
- Az osztályozott állítások száma: 1120
- Érzékenység (R) : 0.9419642857142857
- Pontosság (P) : 0.9461883408071748
- F-mérték (F) : 0.9440715883668902
- Szabatosság : 211.7892857142857
- Fals pozitív arány (FPR) : 0.013392857142857142
- Specifikusság (SPC) : 0.9866071428571429
- Negatív prediktív érték (NPV) : 0.9855072463768116
- Hibás találatok aránya (FDR) : 0.053811659192825115

Helyettesítésre kerüljön-e:
- Helyesen osztályozott igaz állítások száma (TP) : 219
- Hibásan osztályozott igaz állítások száma (FP) : 4
- Helyesen osztályozott hamis állítások száma (TN) : 220
- Hibásan osztályozott hamis állítások száma (FN) : 5
- Az osztályozott állítások száma: 448
- Érzékenység (R) : 0.9776785714285714
- Pontosság (P) : 0.9820627802690582
- F-mérték (F) : 0.9798657718120807
- Szabatosság : 219.49107142857142
- Fals pozitív arány (FPR) : 0.017857142857142856
- Specifikusság (SPC) : 0.9821428571428571
- Negatív prediktív érték (NPV) : 0.9777777777777777
- Hibás találatok aránya (FDR) : 0.017937219730941704

A teljes osztályozásra összesített adatok:
- Helyesen osztályozott igaz állítások száma (TP) : 1056
- Hibásan osztályozott igaz állítások száma (FP) : 72
- Helyesen osztályozott hamis állítások száma (TN) : 3736
- Hibásan osztályozott hamis állítások száma (FN) : 64
- Az osztályozott állítások száma: 4928
- Érzékenység (R) : 0.9428571428571428
- Pontosság (P) : 0.9361702127659575
- F-mérték (F) : 0.9395017793594306
- Szabatosság : 1056.7581168831168
- Fals pozitív arány (FPR) : 0.018907563025210083
- Specifikusság (SPC) : 0.9810924369747899
- Negatív prediktív érték (NPV) : 0.9831578947368421
- Hibás találatok aránya (FDR) : 0.06382978723404255

Az osztályozás hatékonyságának mérőszámai és eredményei 50 kérdés esetén

Relevancia:
- Helyesen osztályozott igaz állítások száma (TP) : 208
- Hibásan osztályozott igaz állítások száma (FP) : 36
- Helyesen osztályozott hamis állítások száma (TN) : 936
- Hibásan osztályozott hamis állítások száma (FN) : 35
- Az osztályozott állítások száma: 1215
- Érzékenység (R) : 0.8559670781893004
- Pontosság (P) : 0.8524590163934426
- F-mérték (F) : 0.8542094455852155
- Szabatosság : 208.77037037037036
- Fals pozitív arány (FPR) : 0.037037037037037035
- Specifikusság (SPC) : 0.962962962962963
- Negatív prediktív érték (NPV) : 0.9639546858908342
- Hibás találatok aránya (FDR) : 0.14754098360655737

Specialitás:
- Helyesen osztályozott igaz állítások száma (TP) **:** 214
- Hibásan osztályozott igaz állítások száma (FP) : 29
- Helyesen osztályozott hamis állítások száma (TN) : 943
- Hibásan osztályozott hamis állítások száma (FN) : 29
- Az osztályozott állítások száma: 1215
- Érzékenység (R) : 0.8806584362139918
- Pontosság (P) : 0.8806584362139918
- F-mérték (F) : 0.8806584362139918
- Szabatosság : 214.7761316872428
- Fals pozitív arány (FPR) : 0.029835390946502057
- Specifikusság (SPC) : 0.970164609053498
- Negatív prediktív érték (NPV) : 0.970164609053498
- Hibás találatok aránya (FDR) : 0.11934156378600823

Értelmesség:
- Helyesen osztályozott igaz állítások száma (TP) : 205
- Hibásan osztályozott igaz állítások száma (FP) : 43
- Helyesen osztályozott hamis állítások száma (TN) : 929
- Hibásan osztályozott hamis állítások száma (FN) : 38
- Az osztályozott állítások száma: 1215
- Érzékenység (R) : 0.8436213991769548
- Pontosság (P) : 0.8266129032258065
- F-mérték (F) : 0.835030549898167
- Szabatosság : 205.76460905349794
- Fals pozitív arány (FPR) : 0.044238683127572016
- Specifikusság (SPC) : 0.9557613168724279
- Negatív prediktív érték (NPV) : 0.9607032057911065
- Hibás találatok aránya (FDR) : 0.17338709677419356

Nehézség:
- Helyesen osztályozott igaz állítások száma (TP) : 202
- Hibásan osztályozott igaz állítások száma (FP) : 43
- Helyesen osztályozott hamis állítások száma (TN) : 929
- Hibásan osztályozott hamis állítások száma (FN) : 41
- Az osztályozott állítások száma: 1215
- Érzékenység (R) : 0.831275720164609
- Pontosság (P) : 0.8244897959183674
- F-mérték (F) : 0.8278688524590164
- Szabatosság : 202.76460905349794
- Fals pozitív arány (FPR) : 0.044238683127572016
- Specifikusság (SPC) : 0.9557613168724279
- Negatív prediktív érték (NPV) : 0.9577319587628866
- Hibás találatok aránya (FDR) : 0.17551020408163265

Helyettesítésre kerüljön-e:
- Helyesen osztályozott igaz állítások száma (TP) : 221
- Hibásan osztályozott igaz állítások száma (FP) : 16
- Helyesen osztályozott hamis állítások száma (TN) : 227
- Hibásan osztályozott hamis állítások száma (FN) : 22
- Az osztályozott állítások száma: 486
- Érzékenység (R) : 0.9094650205761317
- Pontosság (P) : 0.9324894514767933
- F-mérték (F) : 0.9208333333333332
- Szabatosság : 221.46707818930042
- Fals pozitív arány (FPR) : 0.06584362139917696
- Specifikusság (SPC) : 0.934156378600823
- Negatív prediktív érték (NPV) : 0.9116465863453815
- Hibás találatok aránya (FDR) : 0.06751054852320675

A teljes osztályozásra összesített adatok:
- Helyesen osztályozott igaz állítások száma (TP) : 1050
- Hibásan osztályozott igaz állítások száma (FP) : 167
- Helyesen osztályozott hamis állítások száma (TN) : 3964
- Hibásan osztályozott hamis állítások száma (FN) : 165
- Az osztályozott állítások száma: 5346
- Érzékenység (R) : 0.8641975308641975
- Pontosság (P) : 0.8627773212818406
- F-mérték (F) : 0.8634868421052632
- Szabatosság : 1050.7414889637112
- Fals pozitív arány (FPR) : 0.04042604696199467
- Specifikusság (SPC) : 0.9595739530380053
- Negatív prediktív érték (NPV) : 0.9600387503027368
- Hibás találatok aránya (FDR) : 0.1372226787181594

C. Függelék

C.1. Kérdésgeneráló mintarendszer által generált kérdések

Tesztkérdések
az Adatbázisrendszerek I. tantárgyhoz

A kérdésekhez tartozó válaszokból válassza ki azt a szót, amelyiket leginkább alkalmasnak találja a mondatból kivett szó helyére. A kiválasztott szót írja a kipontozott helyre!

1. A számítástechnika <.....> egyik fontos jellemzője, hogy egyre több felhasználó egyre több, számítógépen tárolt adatot használ fel.
   1.) felhasználó 2.) fejlődés 3.) adatbáziskezelő 4.) fájlszervezés 5.) túlcsordul
2. Az információt <.....> kapcsolódó új jelentésnek, hasznos közlésnek tekinthetjük.
   1.) leírás 2.) tekint 3.) adatszerkezet 4.) jelsorozat 5.) érint
3. Az <.....> ezen rejtett tartalma a szemantikai oldal, mely a jelsorozat mögött húzódó jelentést, lényeget hangsúlyozza.
   1.) fontos 2.) alkalmazás 3.) kapcsolódik 4.) adat 5.) tartalom
4. A logikai <.....> alapvetően lehet stream jellegű vagy rekord jellegű.
   1.) adatszerkezet 2.) fájlszerkezet 3.) adatbáziskezelés 4.) helyfoglalás 5.) rekord
5. Minden <.....> időszükséglettel jár, mégpedig minél nagyobb volt a távolság, annál több idő szükséges.
   1.) jelenség 2.) távolság 3.) megkötés 4.) komponens 5.) sávváltás
6. A blokkok a fájlok fizikai szerkezetét <.....>, de emellett a fájl egy belső logikai struktúrával is rendelkezik.
   1.) belső 2.) technika 3.) használ 4.) mutat 5.) dönt
7. A rekordok a fájlban tetszőleges sorrendben, például a felvitel sorrendjében <.....> el, azaz nincs kapcsolat a rekord kulcsértéke és a rekord fájlon belüli pozíciója között.
   1.) kell 2.) pozíció 3.) adatmodell 4.) felhasználható 5.) helyezkedik
8. A fejlesztőknek a DB oldaláról ismernie kell az adatbázis adatmodelljét, a metaadatok megadásának módját, az <.....> programok fejlesztésének lehetőségeit, hogy csak a legfontosabb követelményeket említsük.
   1.) oldal 2.) egzakt 3.) alkalmazó 4.) választevékenység 5.) fogalom
9. A blokkok a fájlok fizikai szerkezetét mutatják, de emellett a fájl egy belső logikai <.....> is rendelkezik.
   1.) információ 2.) közvetlenül 3.) beszúrás 4.) logikai 5.) struktúra
10. A statikus védelem egyik eszköze a mentés, a dinamikus védelemhez pedig a <.....> tartozik.
    1.) tartozik 2.) különböző 3.) miközben 4.) naplóz 5.) megközelítés
11. Az egyes elérési, <.....> módok közötti különbség megértéséhez szükség van egy újabb fogalom a rekordkulcs megismerésére.
    1.) szervezési 2.) szabványos 3.) induló 4.) adat 5.) megértés

12. Az első hálózatos, konkurens hozzáférést <.....> adatbank 1965-ben jelent meg az IBM-nél, s a SABRE nevet kapta.
    1.) látható 2.) ők 3.) megismétlődik 4.) biztosító 5.) elhelyezkedik

13. Az adatbázis összetartozó adatok azon rendszere, mely <.....> több felhasználó között, és az elérést egy központi vezérlő program szabályozza, és a felhasználónak nem kell ismernie az adatok fizikai tárolási mechanizmusát.
    1.) használ 2.) idő 3.) központi 4.) megoszt 5.) jellegű

14. Az adatstruktúra viszont az alkalmazások passzív elemeit jelentik, s kell egy algoritmus, egy program mellyel felhasználhatók ezek az adatok, életre <.....> az információk.
    1.) kelt 2.) elvégez 3.) elem 4.) végrehajtás 5.) stream

15. Az adatbázis módosításakor <.....> ellenőrzi a DBMS, hogy nem sérült-e meg az integritási szabály.
    1.) automatikus 2.) speciális 3.) fejlődés 4.) szabály 5.) saját

16. Az <.....> azonban már most is megjegyezhetjük, hogy központi szerepet játszanak az adatbáziskezelésben, hiszen ezen keresztül adhatjuk meg a megvalósítandó rendszer leírását az adatbáziskezelőnek.
    1.) foglalkozik 2.) módszer 3.) adatmodellek 4.) emellett 5.) rugalmasság

17. A DBMS-ek egyik <.....> jellemzője, hogy mely adatmodellhez kapcsolódnak.
    1.) fő 2.) komponens 3.) külső 4.) adatmodell 5.) megváltozott

18. A rekord mindig egy egyed előforduláshoz <.....>, miközben az adatbázis több előfordulást is tárolhat.
    1.) szabályoz 2.) tárol 3.) beolvas 4.) ellenőriz 5.) tartozik

19. A műveletek összehangolására, vezérlésére minden utasítást le kell fordítani a koncepcionális szint <.....>.
    1.) minden 2.) logikai 3.) vezérlés 4.) hely 5.) séma

20. Az <.....>, mint a külső adathordozón letárolt fizikai adatszerkezethez történő hozzáféréshez a DBMS is felhasználhatja a hardware fölött elhelyezkedő operációsrendszer I/O szolgáltatásait.
    1.) belső 2.) hely 3.) mechanizmus 4.) vezérlő 5.) adatbázis

21. A fizikai szintű műveletek elvégzéséhez ismerni kell az adatok <.....> sémáját is, tehát szükség van a koncepcionális és a fizikai séma közötti leképzésre is.
    1.) mutat 2.) információs 3.) pozíció 4.) fizikai 5.) felhasználható

22. DC <.....> feladata az interface biztosítása a segédprogramok, a felhasználók felé.
    1.) szükség 2.) összehangol 3.) elérés 4.) túlcsordul 5.) komponens

23. A szoftverfejlesztés <.....> fejlődő területe egyre újabb módszereket, technikákat fejleszt ki, és az adatbázis- és alkalmazásfejlesztő rendszerek is követik és alkalmazzák ezen új eszközöket.
    1.) rohamos 2.) adatbázis- 3.) összetett 4.) elvont 5.) koncepcionális

24. Az objektum <.....> modell célja az objektumorientáltság szemléletmódjának alkalmazásával minél valóság hűbb adatmodellt megalkotása.
    1.) idő 2.) oszt 3.) valósághű 4.) kitűnik 5.) orientál

25. A backup rendszer az adatok adathordozó sérüléséből vagy egyéb okokból eredő elvesztésének megelőzése mellett <.....> DBMS rendszer különböző verziószámú változatai közötti adatkonverzióhoz is felhasználható.
    1.) egyazon 2.) sérülés 3.) adatkezelés 4.) hálózat 5.) alkalmazó

26. Az adatbázis azonos rekordtípusokat tartalmazó <.....> épül fel, ahol minden tábla teljesen egyenértékű, s nincs semmilyen az adatdefiníciókor véglegesen lerögzített kapcsolat, váz, mint ami az előző modelleknél előfordult.

1.) struktúra 2.) gyár 3.) elvégzés 4.) előfordul 5.) tábla

27. A szemantikai adatmodellek célja a valóság leírását a számítógépnél megszokott egyszínű, szintaktikai kezelés, leírás helyett <.....> gazdagabbá tenni.
1.) modellezés 2.) hely 3.) szemantika 4.) tipikus 5.) rejt

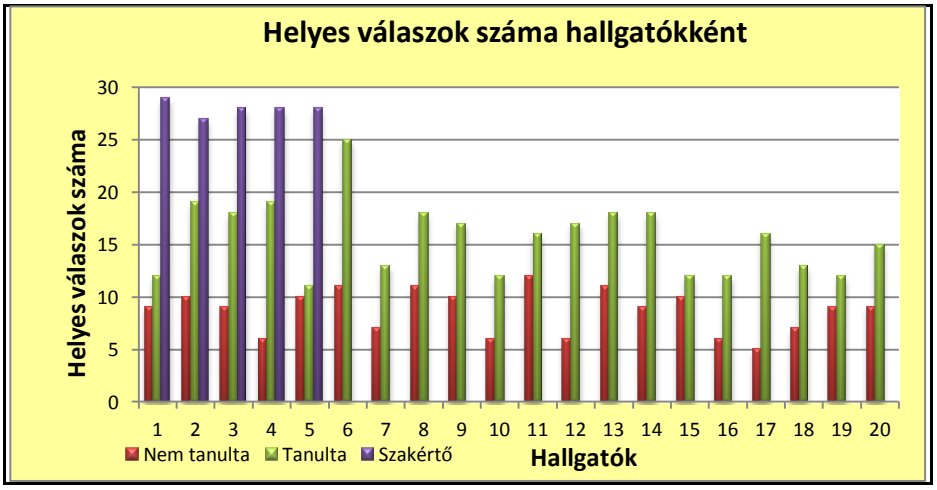
28. Az <.....> feltételek az egyedek között fennálló kapcsolatokat írják le és szabályozzák.
1.) szempont 2.) automatikus 3.) adat 4.) eset 5.) integritás

29. A meta egyedtípusokat az elvontabb fogalmak leírására lehet felhasználni, melyek a létező egyedtípusok alapján <.....>.
1.) ismer 2.) létező 3.) változat 4.) fejleszt 5.) definiál

30. A megkötések tipikus <.....>, amikor a műveleteket leszűkítjük speciális egyedtípusokra.
1.) többek 2.) leszűkít 3.) eset 4.) idő 5.) egyféle

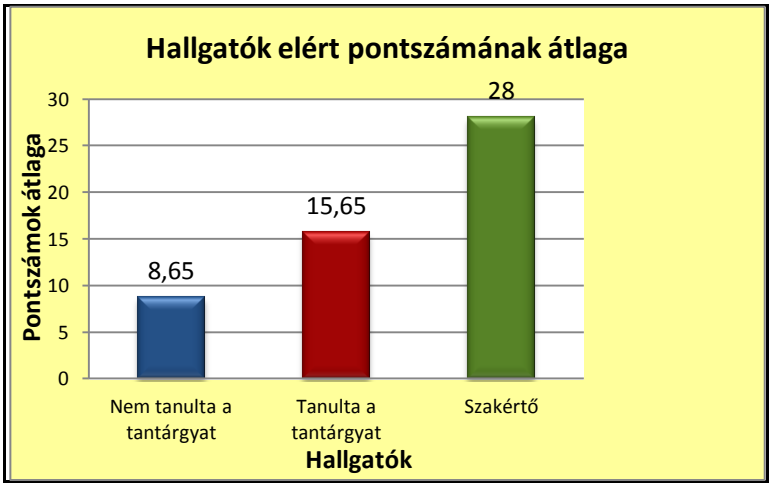
C.2. A hallgatók és eredményeinek értékelése

Helyes válaszok száma hallgatókként



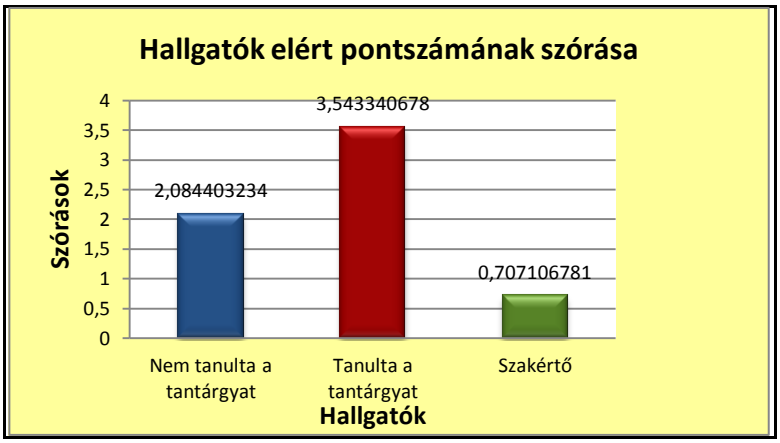
Az eredmények összehasonlíthatósága érdekében a diagram egyben ábrázolja a hallgatók által elért pontszámokat. A tesztben résztvevő 45 főből 5 szakértő, 20 témát ismerő hallgató, 20 pedig a témát nem ismerő hallgató. Az eredmények alapján megállapítható, hogy az automatikusan előállított feladatsor alkalmas a hallgatók tudásának mérésére, mivel a három kategória eredményei jól elkülönülnek egymástól.

Hallgatók elért pontszámának átlaga



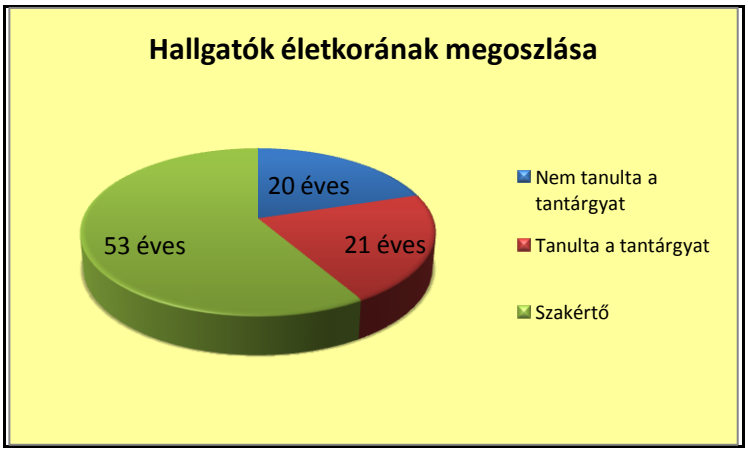
Az ismert három kategóriába tartozó hallgatók átlagosan elért pontszámai jól elkülönülnek egymástól. Ezáltal a tesztek eredményei alkalmasak azoknak a határoknak a megállapítására, melyek alapján az előre nem ismert tudású hallgatók is kategorizálhatóak.

Hallgatók elért pontszámának szórása



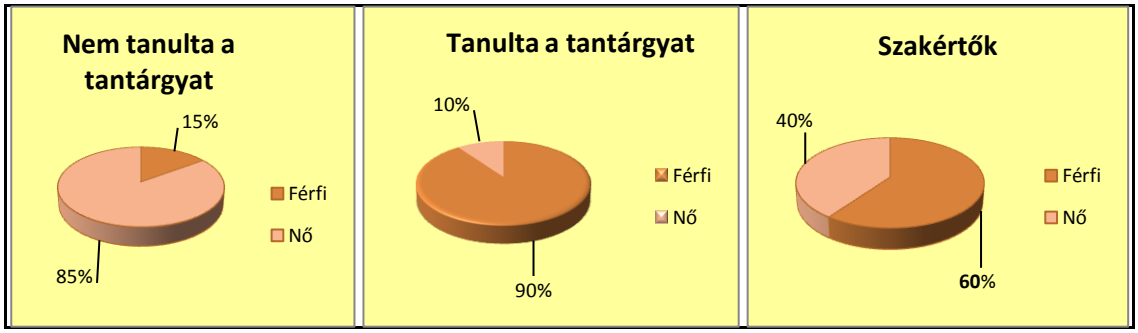
A hallgatók elért pontszámának szórása alapján megállapítható, hogy a választott téma szakértőinek eredményei kis szórással változnak az előzőekben kimutatott maximum közeli értékhez képest. Vagyis, a szakértők szinte mindegyike stabil tudással rendelkezik. A témát nem ismerők viszonylag nagyobb szórással képviselik a megismert alacsony pontszámot. Ennek oka, hogy a véletlen vagy a részleges tudás révén ők is tudtak jó válaszokat adni. Az eredmények legnagyobb szórását a tantárgyat ismerők mutatják. Ennek oka, hogy ebbe a kategóriába tartozó személyek között vannak a témát jól és gyengén ismerők is.

Hallgatók életkorának megoszlása



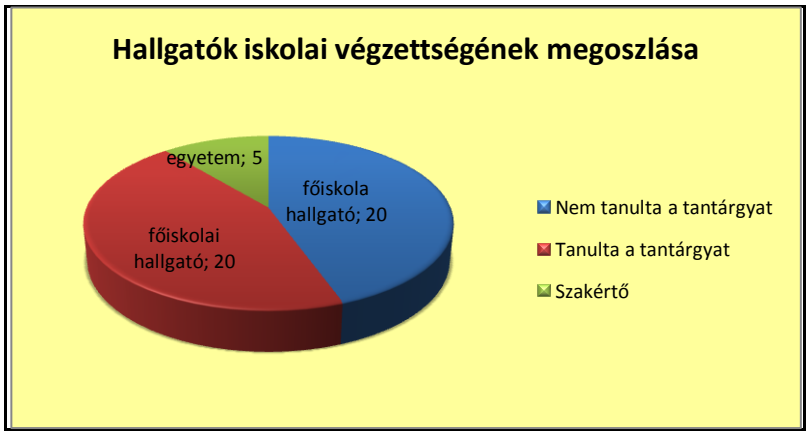
A választott téma szakértőinek átlagos életkora 53 év, míg a témát ismerő illetve nem ismerő hallgatóké 20-21 év. A téma szakértőinek életkora közel háromszorosa a többiekének. Ez az információ is indokolhatja a szakértők magasabb tudásszintjét.

Hallgatók nemének megoszlása



Az ismert kategóriákba tartozó hallgatók nemek alapján való megoszlása a későbbi kutatások folyamán használható lesz a kérdések és válaszok automatikus kiválasztásában.

Hallgatók iskolai végzettségének megoszlása



A választott téma szakértőinek mindegyike egyetemi végzettségű, míg a többi tesztelő a főiskolai hallgatók köréből került kiválasztásra. Ez az információ is indokolhatja a szakértők magasabb tudásszintjét.