M Ű E G Y E T E M 1 7 8 2
BUDAPESTI MŰSZAKI ÉS GAZDASÁGTUDOMÁNYI EGYETEM
VILLAMOSMÉRNÖKI ÉS INFORMATIKAI KAR
MÉRÉSTECHNIKA ÉS INFORMÁCIÓS RENDSZEREK TANSZÉK

Publikációgyűjtemény tudásbázisának építése természetes nyelven

TDK dolgozat

Készítette:
Hornyák Zsuzsanna Éva
MSc., I. évfolyam

Konzulens:
Mészáros Tamás
mestertanár

2013.10.25.

Tartalomjegyzék

1. Bevezető

Nap mint nap új információkhoz jutunk, melyeket valamilyen formában és rendszerben eltárolunk, legyen az a saját memóriánk vagy esetleg egy elektronikus mentés. A tárolás módja a későbbi hatékony felderítés szempontjából fontos kérdés. Az emberek többnyire az információt annak szemantikája, avagy jelentése, és asszociációi alapján jegyzik meg – de hogyan ültethető ez át a számítógépes világba? Megelégszünk hierarchikus könyvtárstruktúrákkal és a kulcsszavak alapján történő kereséssel? A legnagyobb információs tárház, az internet szemantikus rétegének létrehozása már az ezredfordulón felmerült[1]. Noha erre alapozva sok technológia fejlődés történt az elmúlt évtizedben, a Berners-Lee-ék által megálmodott univerzális tudáshálóhoz szinte alig kerültünk közelebb.

A szemantikus web elképzelés lényege, hogy az internet statikus dokumentumaihoz elkészül egy olyan szemantikus reprezentáció a dokumentum tartalmáról, mely gépek által értelmezhető és feldolgozható. Ehhez azonban szükség van közös szabványokra, a technológiák ismeretére, és a publikáló felhasználók részéről plusz energia-befektetésre. Az online objektumokhoz ma már széles körben elérhetőek egyszerű metaadatok (cím, szerző, címkék, stb.), amelyeknek köszönhetően könnyebb a dokumentumok elérése, ez azonban csak a felszínét érinti a szemantikus reprezentáció problémakörének.

Különösen fontos az információ intelligens reprezentálása a tudományos társadalom számára, hiszen a publikálásra kerülő tudományos eredmények száma rohamosan nő. Két problémát emelnék ki ezen a területen:

1. A *releváns publikációk felderítése*: A tartalomban megjelenő kulcsszavak alapján történő keresés már széles körben támogatott – de nem biztos, azt az eredményt adja, amelyre tényleg szükségünk van. Sokat segítene, ha nem csak a publikáció szöveges tartalmára, hanem az abban megjelenő logikai állításokra lehetne keresni.
2. Az *összegyűjtött publikációk átláthatósága*: A kutatás közben szerzett információk áttekintése a publikációk számának növekedésével egyre nehezebb feladata. A következtetés gépi úton támogatható lehet, ha a publikációkhoz valamilyen tudásábrázolás is elérhető.

A fent említett problémákra megoldást nyújthat a publikációk mögött egy szemantikus réteg létrehozása, mely tartalmazza például a cikkekhez kapcsolódó logikai állításokat. Ez az ötlet már a szemantikus web megjelenésekor felmerült [2]. A formális leírások elkészítése azonban a publikálás amúgy is bonyolult folyamatában egy új lépést jelent a szerzők vagy szerkesztők

számára. Ennek betanulása és elvégzése is plusz erőfeszítéseket igényel, ezért egy általános szabvány elterjesztése nehéz feladat.

Munkám során azt vizsgálom, hogyan lehet egyszerűen elérhetővé tenni a szemantikus technológiákat a mindennapi felhasználók (kutatók) számára. Ehhez egy természetes nyelvű beviteli eszköz fejlesztettem, melynek segítségével természetes nyelven fogalmazhatóak meg állítások egy adott tárgyterületen. A program segítségével lehetséges publikációk olvasása vagy készítése mellett a cikk „szemantikus kivonatát" létrehozni. Ahhoz, hogy ez a kivonat géppel elemezhető és értelmezhető legyen, a szövegalkotás egy kontrollált természetes nyelvre van szűkítve. Ez a nyelv egy dinamikusan bővíthető, de alapvetően korlátozott nyelvtanra épül. A felhasználói felület segítséget nyújt a korlátozott nyelv gördülékeny használatához.

A dolgozatomban bemutatott rendszer célja, hogy tudományos közlemények fontosabb megállapításaiból – egyszerű eszközökkel – olyan tudásbázist építhessünk, amely a mainál jelentősen hatékonyabbá teheti a bennük rejlő információk elérését és felhasználását. A szoftverrel lehetséges a felhasználó által elkészített kivonatok alapján a cikkek fontosabb információinak valamilyen formális reprezentációját létrehozni, és így egy tudásbázist felépíteni az összegyűjtött publikációk mögé. Ezen a tudásbázison már végrehajthatóak szemantikai, logikai keresések, illetve következtetések is.

Az alkalmazást orvosbiológiai publikációk szervezéséhez mutatom be, mivel itt égető szükség van a hatalmas mennyiségben elérhető eredmények hatékony leírására és szervezésére. A bemutatott technológia azonban, ahogy az majd látható, könnyen adaptálható más alkalmazási területekre is.

A dolgozat vázlatos felépítése:

2. fejezet: A publikációgyűjtés és a szemantikus tudásábrázolás problématerületének részletesebb bemutatása, az online tudományos publikációk területére fókuszálva. A szemantikus publikációs kezdeményezések rövid áttekintése. Szintén ebben a fejezetben ismertetem a kontrollált természetes nyelvek kutatási területét és a munkámhoz kapcsolódó alkalmazásokat.
3. fejezet: Ismertetem a publikációk gyűjtéséhez és rendszerezéséhez létrehozott intelligens rendszeremet. Részletesen bemutatom a természetes nyelvű szövegbevitelhez szükséges nyelvtanalkotás, és a nyelvtani elemzés lépéseit, melyekre alapozva lehetséges a rendszer dinamikus bővítése. A rendszer működését példa esetre demonstrálom.

4. fejezet: A dolgozatban bemutatott rendszer értékelése. Felvetítem, hogy a létrehozott technológiának milyen további hasznos alkalmazásai lehetnek.

2. Problématerület

Ebben a fejezetben röviden összefoglalom a publikációgyűjtés, szemantikus publikáció és tudásábrázolás területén milyen fő kérdések illetve megoldások léteznek – megmutatva, hogy itt hol helyezhető el általam létrehozott alkalmazás. Ismertetem továbbá az alkalmazás alapját képező kontrollált természetes nyelvek témakörét, röviden kitérve a hasonló alkalmazások megközelítéseire.

2.1 Publikációgyűjtés ma

Digitális könyvtáraknak nevezzük az elektronikusan tárolt irodalmak, és a hozzájuk tartozó metaadatok (adat az adatról) gyűjteményét. Sok különböző digitális könyvtár érhető el az interneten, amelyek több folyóirat cikkeit gyűjtik össze, ám az egyes könyvtárak szervezése, illetve leírása nem egységes [3]. Digitális könyvtárnak tekinthető a *Google Scholar*[1] is, amely indexeli a weben található publikációkat és olyan keresőszolgáltatást biztosít, amellyel a cikkekre történő hivatkozásokat is könnyen el lehet érni. Szintén lehetséges saját publikációkra történő hivatkozások követése is. A *Google Scholar* segítségével szinte az összes online publikáció elérhető, azonban olyan szolgáltatást (még) nem biztosít, amellyel a számunkra érdekes publikációkat összegyűjthetjük és saját elképzelés szerinti rendszerezésben tárolhatjuk.

A publikációgyűjtemények készítésére és kezelésére több, főként webes megoldás is létezik. Ezek többnyire a dokumentumok *Uniform Resource Identifier* (*URI*) vagy *Digital Object Identifier* (*DOI*) azonosítóit használják a rendezéshez. A gyűjteménykezelők között kétféle típus figyelhető meg:

- *Rendszerezés-központú*: ezek az alkalmazások egyszerűen a menteni kívánt publikációk könnyebb rendezését teszik lehetővé. Ilyen például a *Zotero*[2], a *Mendeley*[3] és a *Papers*[4]. Ezen szoftverek segítségével hivatkozásokat vagy *PDF*-eket is lehet rendezni, a megfelelő metaadatok illetve kategorizálás segítségével, továbbá különböző annotálási szolgáltatásokat is nyújtanak.
- *Megosztás-központú*: más alkalmazások célja, hogy ne csak gyűjteni lehessen a publikációkat, hanem ezeket másokkal megosztani. Ezek közül a legnépszerűbb a *CiteULike*[5], amelynek segítségével hivatkozásokat menthetünk el és oszthatunk meg

[1] http://scholar.google.hu/
[2] http://www.zotero.org/
[3] http://www.mendeley.com
[4] http://www.papersapp.com/mac/
[5] http://www.citeulike.org/

könnyen. Érdekessége az oldalnak, hogy új nézőpontot ad a publikációk értékelésére – nem csak a rájuk történő akadémiai hivatkozás, hanem így az úgymond "olvasói népszerűség" is elérhető.

A publikációgyűjtemény-kezelő megoldások nagy mértékben tudják segíteni a kutatók mindennapi munkáját, hiszen lehetőségük van a publikációk számukra kézenfekvő rendszerezésére, elmentésére. Ez segíti az információ visszakeresését, illetve a későbbiekben az olvasottakra történő hivatkozások előállítását is. A jelenleg létező ilyen eszközök azonban a publikációkhoz csak metaadatokat illetve kulcsszavakat rendelnek, a cikkek tartalmát leíró intelligens reprezentációt nem támogatnak, így az összegyűjtött publikációk lényegi információinak elérését és összekapcsolását ezek a megoldások sem segítik.

A továbbiakban röviden bemutatom a *Zotero* alkalmazást, amelyet jómagam is használok publikációk gyűjtésére.

2.1.1 Zotero

A *Zotero* egy publikáció rendszerező szoftver, mely elérhető, mint a *Firefox* böngésző kiegészítője vagy különálló asztali alkalmazás (további böngészőkbe való részleges integrációval). A *Zotero* képes automatikus felismerni, ha olyan weblapot látogatunk meg, amely egy publikációnak felel meg (cikk vagy könyv), és képes annak automatikusan kinyerni az olyan adatait mint például: szerző, cím, DOI, publikálás éve, stb. Ezeket az adatokat később magunk is kiegészíthetjük, és a publikációkhoz hozzáadhatunk jegyzeteket illetve csatolt fájlokat is.

A *Zotero*-ban lehetséges a publikációk kategóriákba való rendszerezése, illetve címkék (tag) szerinti keresése. Az adatokat különböző információk szerint is rendezhetjük. A publikációkból kinyerhetőek hivatkozások különböző stílusokban, és egy *Word* plugin (kiegészítő) segítségével a *Word* szövegszerkesztőből is elérhetjük a *Zotero*-s gyűjteményünket.

A *Zotero* használatához nem szükséges a regisztráció, de ha ez mégis megtesszük, ingyen biztosítanak 300 MB tárhelyet a szerverükön, és így szinkronizálhatjuk a publikációgyűjteményeinket akár több gép között is. Lehetséges továbbá a gyűjteményeink megosztása más felhasználókkal és csoportok létrehozása, amelyeken belül kollaboráció valósítható meg.

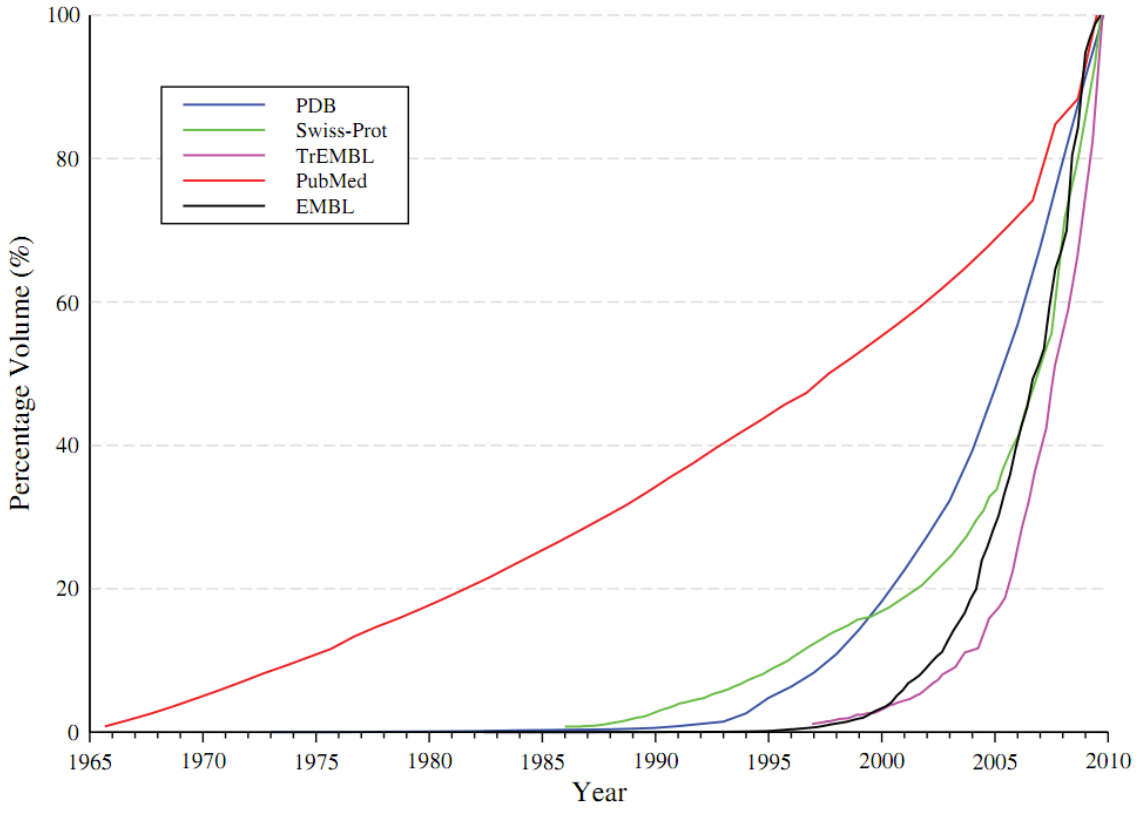
A *Zotero* egy könnyen kezelhető, sok funkcionalitással rendelkező program. A program saját bővítményekkel is kiegészíthető, amelyeket *Firefox* kiegészítő formájában kell létrehozni. A *Firefox* kiegészítők felülete az *XUL* (*XML User Interface Language*) nyelven készíthetőek el,

amely egy *XML* alapú leíró séma. Az *XUL* leírásban lehetőség van hivatkozni a más kiegészítők által felvett elemekre is azonosítóik alapján, így lehetséges egy különálló bővítménnyel módosítani a *Zotero* felhasználói felületét. Az utóbbi forráskódja a *Zotero* letöltésével szabadon rendelkezésre áll.

2.2 Szemantikus publikálás

Az előbbiekben bemutatott publikációgyűjtemény-kezelők mind metaadatok, illetve címkék alapján történő rendszerezést támogatnak. Hatalmas haszonnal járna, ha a weben elérhető publikációkhoz valamilyen szemantikus reprezentáció is társulna, amelyek segítségével a gyűjteménykezelők intelligensebben tudnának dolgozni.

A technológiai fejlődések több tudományos területen is hatalmas adatözönt eredményeztek, például az élettudományokban elérhető kísérleti eredményeknek hála ezek a területek elkezdtek adat-központúvá válni [4]. Az orvosbiológia területén különösen szemléletes a publikált adatok mennyiségének illetve típusának változása az elmúlt 50 évben, ahogy az az *1. ábrán* látható.



1. ábra. Orvosbiológiai kutatási eredmények [5]. Jelölés: piros – publikációk (~19 millió), fekete – nukleinsav szekvenciák (~163 millió), lila – számítógéppel generált fehérje szekvenciák (9 millió), zöld – kézileg előállított fehérje szekvenciák (500 ezer), kék – fehérje struktúrák (60 ezer)

Az orvosbiológiai publikációk hivatkozásait összegyűjtő *MEDLINE* oldalán 2012-ben több mint 760 ezer új bejegyzés született, és az elmúlt évek növekvő tendenciát mutatnak [6]. Az *UniProt* adatbázis protein szekvenciákat és adataikat tárolja, egy részükhöz kézi szerkesztéssel rendelve információkat (*Swiss-Prot*), de nagyobb részt képeznek azok a szekvenciák, melyekhez különböző automatikus módszerekkel kerülnek pl. a hivatkozások illetve relációk hozzárendelésre (*TrEMBL*) [7]. A manuális felülvizsgálás a legtöbb adatbányászati módszer mellett is szükséges, de a szakértői csoportok nem tudnak lépést tartani az adatmennyiség növekedési sebességével – hacsak nem áll rendelkezésre gépek által értelmezhető szemantikai információ, amely segíti a munkát.

A szemantikus publikáció több irányból is megközelíthető. A bevezetőben már felvetítettem, hogy nagyon hasznos lenne tudományos cikkek szemantikus reprezentálása, és azoknak ilyen módon történő összekapcsolása. Hasonlóan fontos kérdéskör azonban a már széleskörűen elérhető adatok (pl. protein szekvenciák) összekapcsolása a publikációkkal, és ehhez itt is szükség van valamilyen szemantikus értelmezésre. Sokan úgy látják, hogy a biológia területén folyóiratok és adatbázisok közötti határnak el kell mosódnia – mivel a biológia cikk többnyire tulajdonképpen leírása egy kutatási eredménynek, amelyet konkrét adatként is kezelhetünk. Rinaldi ezen gondolatmenet alapján a cikkek valódi tudástartalmát a cikkekből kinyert „*nano-publikáció*”-nak hívja [8].

2.2.1 Szemantikus publikáció megközelítései

A publikációk szemantikus értelmezésének egyik iránya az online (illetve elektronikus) cikkek olyan módon történő megjelenítése, amely segíti a megértést illetve a különböző helyen elérhető adatok összekapcsolását. Shotton és társai egy rövidebb tanulmányon keresztül mutatták be, milyen kreatív megoldások lehetségesek a cikkek online megjelenítésekor [9], például *Google Maps* integrációt is alkalmaztak a mintacikkben[6]. A kutatási eredményekre (nyers adatokra) való hivatkozás az ő elképzelésük szerint az adatokra kiterjesztett *DOI* használatával történhet – ez természetes bővítése a publikálásnak, hisz a legtöbb cikk el van látva ilyen azonosítókkal. A tanulmányban szemantikusan dúsított cikk manuálisan készült el, de bizonyos részeire már léteznek automatizált megoldások. Az online ingyenesen elérhető *Reflect*[7] alkalmazással lehetséges tetszőleges weboldal szövegében kiemelni fehérjék illetve molekulák neveit, és azokhoz automatikusan betöltődik a struktúrájukat leíró másik weboldal

[6] http://dx.doi.org/10.1371/journal.pntd.0000228.x001
[7] http://reflect.ws/

[10]. Szemantikus megjelenítésű cikkek elérhetőek a *PenSoft ZooKeys*[8] folyóiratában illetve *Article of the future*[9] weboldalon is.

Látható, hogy noha ezek az elképzelések a cikkek szemantikus értelmezését segítik elő, inkább a felhasználókra fókuszálnak, és nem a gépi feldolgozásra. Az egyes fogalmak kiemelése automatikus, adatbányászati technológiákkal történik, de mellettük még mindig szükség van manuális felülvizsgálásra. A kapott eredmény statisztikát nyújt, illetve lehetővé teszi a fogalmak feloldását, de nem kerültünk közelebb a tartalom formális értelmezéséhez. DeWaard cikkében a tudományos publikációkat a tündérmesékhez hasonlítja: a cikk hosszú szövege csupán „narráció" a mögöttes mondanivalóhoz, amely a lényegi tudás [11]. A mérési eredmény nem helyettesíti a tudományos állításokat, hanem azokat alátámasztja, így a konkrét eredmények publikálásán kívül a következtetések megfogalmazására is kell sémát alkotni (például entitás – reláció – entitás hármasokkal *Resource Description Framework* (RDF)


Structured summary of protein interactions

SIMPK3physically interacts with SIMKK2 by anti tag coimmunoprecipitation (View Interaction: 1, 2, 3, 4)

TFT7physically interacts with TFT7 by anti tag coimmunoprecipitation (View interaction)

2. ábra. Strukturált digitális absztrakt példa (forrás: doi:10.1016/j.febslet.2013.03.033)

modellezéssel).

A probléma egyik első megközelítése a *FEBS Letters* folyóiratban bevezetett *strukturált digitális absztrakt* (*SDA*) ötlete volt [12]. Az elképzelés lényege, hogy a cikkek klasszikus absztrakt része mellé elkészítenek egy digitális kivonatot, amelyben adott struktúra szerint vannak állítások megfogalmazva a cikkből. A kivonat gépek által olvasható formában mellékelve van a cikkhez, de egyszerű természetes nyelven is megtekinthető (*2. ábra*). Ezek az absztraktok jelenleg kizárólag fehérje interakciók leírására szolgálnak. Az ábrán az aláhúzott kifejezések linkek, amelyek ontológiákba (fehérje és molekula ontológiák), illetve a *MINT* fehérjeinterakció-adatbázisba[10] mutatnak. Az *SDA*-t beharangozó cikk szerint ezek állítások úgy készülnek el, hogy egy adott táblázatban adják meg az ott elérhető struktúra szerint a cikkhez kapcsolódó

[8] http://www.pensoft.net/journals/zookeys/
[9] http://www.articleofthefuture.com/
[10] http://mint.bio.uniroma2.it/mint/

interakció adatait a cikkek szerzői, majd ezt azt interakciós adatbázis szerkesztői felülvizsgálják illetve kiegészítik. Már a „kísérlet" indulásakor felmerült, hogy a táblázat kitöltésekor sok plusz munkát adhat a szerkesztőknek, ha a szerzők szubjektív vagy sajátos terminológiát használnak (amelyet így a közösre át kell konvertálni), de bíztak abban a megalkotók, hogy hamarosan olyan adatbányászati algoritmusokat lehet integrálni a folyamatba, amelynek segítségével például a szerzők a cikkekből kikeresett entitásneveket alapján dolgoznak [13]. A 2008-as indulás óta nem készült újabb publikáció arról, hogy milyen fejlesztések történnek az *SDA* terén, de a *FEBS Letters*-ben sok cikkhez ma is elkészítik ezeket a rövid absztraktokat[11].

Egy érdekes és szemléletes példa a cikkek állításainak formális leírására az *Alzheimer Kutatási Fórumon*[12] létrehozott rendszer. Itt az úgynevezett *SWAN* ontológiát használják, melynek segítségével van lehetőség hipotézisek, tények és felvetések megadására továbbá más ontológiákra történő hivatkozásra [14]. Az oldalon egy szűk felhasználói rétegnek van lehetősége állítások felvételére. Ezek a bejegyzések összekapcsolhatóak különböző relációkkal – milyen állítás milyen evidenciákra alapoz, mely állítások állnak szemben egymással, stb. A webes felületen lehetőség van hipotézisekre vagy génekre is keresni, és többféle vizuális megjelenítési forma segíti a megértést (pl. kapcsolati gráf az állítások között).

Az előbbi két példa jól mutatja, hogy igény van publikációk állításainak formális leírására, mivel az segítheti mind a különböző helyen elérhető adatok összekapcsolását a szemantika alapján, mind pedig a gépek által is segített következtetést és tudás szervezést. Mindegyik alkalmazásnál felmerült azonban problémaként, hogy az adatok formális felvétele vagy nagyon korlátozott formájú (*SDA*), vagy pedig csak annak a szűk rétegnek elérhető, akik ismerik a szükséges technológiát (*AlzSWAN*). Ezekre megoldást nyújthat, ha a formális adatok felvétele természetes, a hétköznapi felhasználók által használt nyelven történik. Munkám során egy olyan alkalmazás fejlesztésével foglalkoztam, melynek segítségével az *SDA*-hoz hasonló absztraktok hozhatóak létre, közvetlenül természetes nyelvű állításokkal, melyek később alakulnak át a gépi reprezentációra.

2.2.2 Ontológiák szerepe

A szemantikus publikációhoz szorosan kapcsolódik az ontológiák fogalma. Az ontológiák segítségével van lehetőség egy terület fogalmait, azok kapcsolatait és tulajdonságait modellezni [15]. A *W3C* (*World Wide Web Consortium*) szemantikus web szabványai között szerepel

[11] Az SDA-val rendelkező cikkek megtekinthetőek egy gyűjtőoldalon: http://www.febsletters.org/content/sda
[12] http://www.alzforum.org/

többféle támogatott technológia, melyek a szabványos adatcserét segítik az alkalmazások között [16] – például az *RDF* erőforrás leíró modell és annak ontológiákra specializált változata, a *Web Ontology Language* (*OWL*). Már a *strukturált digitális absztraktnál* is felmerült, hogy fontos szerepe van annak, hogy a kutatók ugyanolyan néven hivatkozzanak a különböző entitásokra (pl. génekre). A szemantikus tudásbázisok létrehozásakor fontos szempont, hogy a tárgyterület ontológiáit támogassa (ha vannak) és azokat megfelelően bekapcsolja a logikai leírásba.

A biológia kutatások során keletkező rengeteg új adat miatt itt is nagyon fontos az eredmények szabványos leírása. Az *Open Biomedical Ontologies* (*OBO*) alapítvány célja az orvosbiológiai ontológiák szabványos, kompatibilis és szabályozott elkészítése [17]. A kedvezményezés részeként kerül fejlesztésre többek között a gén ontológia (gének különböző organizmusokban) és a fehérje ontológia (protein típusok és elváltozások). Ezek az ontológiák nem csak a terület fogalomtárát írják le, hanem azokról elérhető információkat, relációs kapcsolatokat is tartalmaznak.

2.3 Természetes nyelvű alkalmazások

A szemantikus publikálás során az egyik legnagyobb probléma, hogy a szemantikus reprezentáció létrehozásához szükséges eszközök a felhasználók számára nagyon idegenek. A természetes nyelvű felhasználói felületek előnye, hogy azok a felhasználók számára a legkönnyebben érthetőek illetve használhatóak. A természetes nyelv gépi feldolgozása azonban a mai napig nehéz feladat. A természetes nyelvi elemzés legnagyobb sikerét eddig az *IBM Watson* szuperszámítógép[13] érte el, amely 2011 februárjában megnyert egy amerikai kvízműsort – de ennek komoly ára volt. A *Watson* képes rendkívül gyorsan a szövegben összefüggések felfedésére, azonban a megfelelői logikai következtetéshez és válaszadáshoz 4 TB adat tárolására volt szükség és 2800 processzoron futott párhuzamosan a feldolgozás és az elemzés [18]. Egy kisebb alkalmazás fejlesztésekor viszont többnyire nem érhetőek el ilyen erőforrások – de nem is szükséges, hiszen valószínűleg nem igényli a teljes természetes nyelv támogatását, csupán a problématerülethez kapcsolatos részhalmazának a feldolgozását.

2.3.1 Kontrollált természetes nyelv

A természetes nyelv feldolgozási nehézségeinek egy részétől megszabadulhatunk, ha a vizsgálatot leszűkítjük a *környezetfüggetlen nyelvekre* [19]. Ezek jelentéstartalma, elnevezésükből adódóan, független az adott kommunikációs helyzettől, és mint karakterfüzérek sorozatai határozhatóak meg. Az ilyen nyelveket egyértelműen

[13] http://www-03.ibm.com/innovation/us/watson/index.shtml

modellezhetjük a megfelelő nyelvtan felépítésével, amely tartalmazza az elfogadott szintaktikai struktúrákat, a szemantikától függetlenül. Adott értelmezési tartomány mellett a környezetfüggetlen nyelvtanokkal a természetes nyelv tetszőleges részhalmaza előállítható [20]. Egy konkrét részhalmazt meghatározó szabályok véges csoportját hívjuk *kontrollált nyelvtannak*, amely tehát megszabja, hogy mely természetes nyelvi struktúrák vannak értelmezve az adott rendszerben.

2.3.1.1 Kontrollált nyelvtan

A formális nyelvtan adja meg a nyelvben előfordulható karakterfüzérek szerkezetét és halmazát. A füzéreken belül két kategória különböztethető meg: a záró és a nem záró szimbólumok. A *záró szimbólumok* (terminal symbols) azon fix karakterláncok, amelyek tovább nem bonthatóak, és amelyekre a nyelvtan többi része épül – más néven ezek a nyelv alapszavai. A *nem záró szimbólumok* (nonterminal symbols) további, tetszőleges kategóriájú szimbólumokból épülnek fel: a megfelelő szerkezetek az *átíró szabályok* (rewrite rules) segítségével definiálhatóak. A környezetfüggetlen nyelvtanok esetében az átíró szabályok csak egyedül álló nem záró szimbólumhoz rendelhetnek struktúrát, így azok a környezettől függetlenül mindig felbonthatóak az adott szerkezetre [21].

A nyelvtani szabályokat leírásához használhatunk többféle formalizmust, például a *Backus-Naur-formát* (*BNF*) [22] vagy a *definit klóz nyelvtani* (*DCG*) alakot, mely a *Prolog* programozási nyelv készítése során jött létre [23]. A *DCG* neve onnan ered, hogy a vele megadott nyelvtani szabályok értelmezhetőek *Horn*-logikában szereplő *definit klózként*. A *DCG* természetes kiterjesztése a környezetfüggetlen nyelvtanoknak, és segítségével a nyelvtani elemzés logikai következtetésként kezelhető [24].

Példa egy nyelvtan definíciójára *BNF* illetve *DCG* jelölésrendszerben:

BNF	DCG

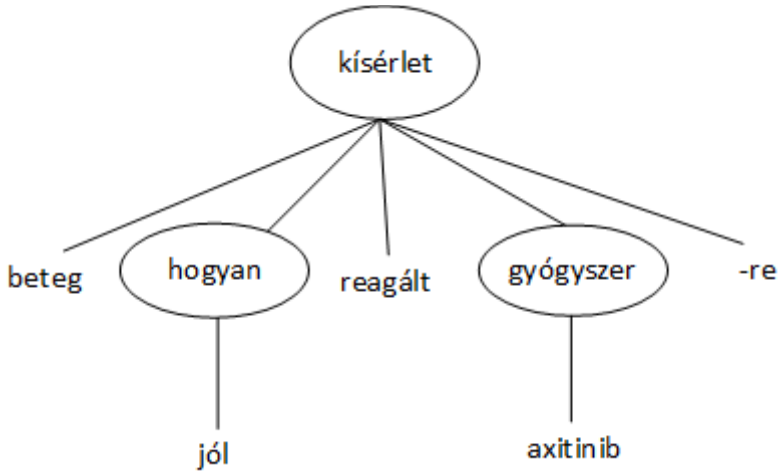
`<kísérlet> ::= beteg <hogyan> reagált a <gyógyszer>-re`
`<hogyan> ::= jól | rosszul`
`<gyógyszer> ::= axitinib | gefitinib`
	`kísérlet --> [beteg], hogyan, [reagált], [a], gyógyszer, [-re].`
	`hogyan --> [jól].`
	`hogyan --> [rosszul]`
	`gyógyszer --> [axitinib]`
	`gyógyszer --> [gefitinib]`

A példanyelvtan által definiált egyszerű nyelvben olyan mondatok szerepelhetnek mint például: "*Beteg jól reagált az axitinib-re*". A nyelvtan gyökere (kiindulópontja) a `kísérlet` nemterminális szimbólum.

2.3.1.2 Nyelvtani elemzés

A nyelvtani elemzés célja annak megállapítása, hogy a kapott szöveg eleme-e a nyelvtan által meghatározott nyelvnek, avagy megfelel-e a nyelvtani szabályoknak. Az elemzés során előáll egy *elemzési fa*, amelyben látható, a vizsgált karakterfüzér hogyan illeszkedik a nyelvtani struktúrákra (amennyiben érvényes). Az előző pont mintanyelvtanára épülve a "Beteg jól reagált az axitinib-re" elemzése során előálló elemzési (avagy levezetési) fa a *3. ábrán* látható.

A belső csomópontok nem terminális szimbólumoknak, míg a levelek a terminális szimbólumoknak feleltethetőek meg. Az elemzési fának megfelelő mondatot a levelek balról jobbra történő olvasásával kapjuk vissza. A levezetési fa által meghatározott struktúra alapján tudjuk szemantikusan értelmezni az adott szöveget, hiszen a szemantika a nyelvtan



3. ábra. Elemzési fa példa.

definíciójában implicit szerepel. A minta nyelvtan mondataihoz az elemzés után például olyan entitás-hármasokat rendelhetünk, mint a `reagál(jól, axitinib)`. A hármasban a két "paramétert" az elemzési fában történő behelyettesítések (`hogyan` illetve `gyógyszer` ágak) alapján kapjuk meg.

A nyelvtani elemzés többféleképpen is történhet. Az elemzési fa viszonylatában megvalósítható *fentről lefelé* (top-down), vagy *lentről felfelé* (bottom-up). A feldolgozás történhet egyszerűen balról jobbra, mindig a legbaloldaliabb szimbólum feloldásával *(LL parser*). A *diagramelemzők* (chart parser) részfüzéreket elemeznek, a gyökér szimbólumból kiindulva, és a felismert rész struktúrákat egy gráf szerű modellben tárolják. A hatékony és gyors nyelvtani elemző megvalósítása fontos az olyan alkalmazásoknál, ahol a természetes

nyelvű "kommunikáció" a felhasználóval valós időben történik, így például egy intelligens szövegszerkesztőnél is.

2.3.2 Prediktív nyelvtani elemzés és alkalmazások

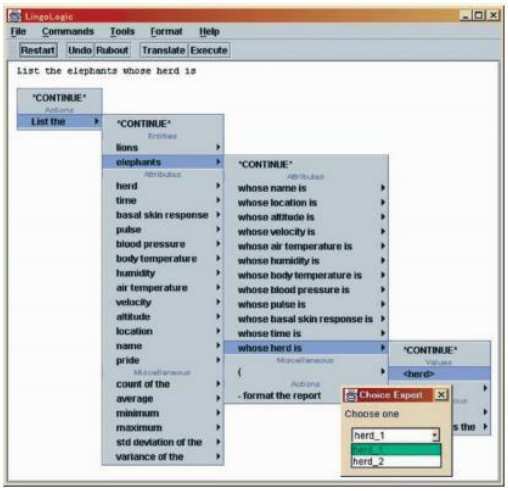
A kontrollált nyelvtanra épülő kommunikáció esetén nem minden bemenet fogadható el, csak az, ami az adott korlátozott nyelvtanra illeszkedik. A gépekkel történő természetes nyelvű kommunikációban a „*habitability*" avagy *otthonossági* problémának hívjuk azt, hogy a felhasználó elvárásai a természetes nyelvű felület iránt eltérnek annak képességeitől [25]. Ezen lehet segíteni, ha a program a bevitel közben támpontokat ad a felhasználónak arról, milyen folytatás engedhető meg. A továbbiakban az ilyen kontrollált természetes nyelvű, prediktív szerkesztőket vizsgálom röviden.

Prediktív elemzőknek inkrementálisan kell elemezniük, tehát például minden új szó után elemezni kell az addig elkészített részmondatot a kontrollált nyelvtani szabályok szerint, hogy megkapjuk a lehetséges folytatásokat. Mivel a diagramelemzők az elemzés során a köztes állapotot is tárolják, jól alkalmazhatóak prediktív elemzésre (tehát a szöveg egy adott pontján megmondani, milyen szimbólum következhet).

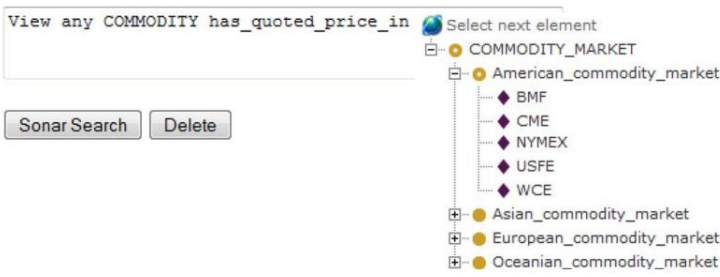
Kuhn és Schwitter egy 2008-as cikkben számoltak be arról, hogy milyen eredményeket értek el egy klasszikus balról-jobbra, illetve egy diagramelemzővel [26]. A balról jobbra elemzőnél az elemzés ideje a *tokenek* (szavak) számával majdnem exponenciálisan nőtt, míg a diagram módszernél egy új elem felvétele konstans, sokkal rövidebb elemzési időt indukált. A valós idejű alkalmazásokban mindkét algoritmus helyt állt, *Prolog* nyelvű implementáció esetén még jobb eredményekkel is. A prediktív elemzésnél tehát, ahol szavanként újrafuttatjuk az elemzőt, érdemesebb a diagramelemző használata.

A prediktív nyelvtani elemzés elkészítése után a következő kérdés, hogy a természetes nyelvű szöveg bevitele milyen módon történjen, illetve hogyan jelenítsük meg a felhasználónak a szerkesztést segítő támpontokat. A felhasználói felületek elkészítése az alkalmazási területtől is függ. A továbbiakban röviden bemutatok 4 különböző megközelítést a prediktív szerkesztők körében.

A *LingoLogic* célja természetes nyelvű adatbázis lekérdezések létrehozása [25]. A mondatokat egy „menü alapú" kezelői felületen lehet elkészíteni. A leugró választási lista dinamikusan épül fel a szerint, addig miből lett összeállítva a mondat (*4/a. ábra*). A nagyon kötött felület biztosítja, hogy csak érvényes kérdés kerüljön bevitelre. Érdekes funkcionalitása még a

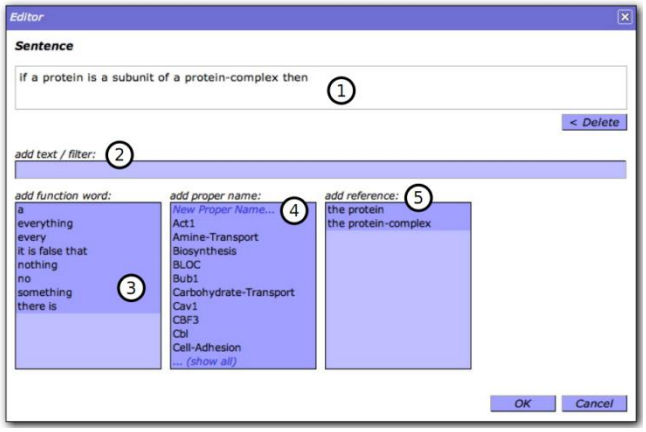


a. LingoLogic



b. OWLPath

c. GINO



d. ACEWiki Editor

4. ábra. Prediktív kontrollált természetes nyelvű szerkesztőfelületek

programnak, hogy betöltött adatbázis séma szerint dinamikusan építhető a nyelvtan (bizonyos strukturális tulajdonságok, pl. „kié” mentén).

Két, ontológia lekérdezést segítő alkalmazás a *Guided Input Natural Language Ontology Editor* (*GINO*) [27] és az *OWLPath* [28]. Mindkét alkalmazásban ontológia formájában van megadva a lekérdezés nyelvtana is, és a lekérdezés építése egy ontológia erdő böngésző segítségével lehetséges (illetve *GINO* esetében egy leugró lista is megjeleníti a lehetséges folytatást) (*3/b. és 3/c. ábrák*). Az ontológiai alapú nyelvtannal egyszerűen adhatóak meg nyelvtani elemekre korlátok (pl. adott helyen új számérték intervallumára).

Az általam is tervezett rendszerhez legszorosabban kapcsolódó alkalmazás az *ACEWiki* [29]. Az *ACEWiki* egy szemantikus wiki, avagy tudástár, amelynek célja formális állítások megadása természetes nyelven, így bővítve az ontológiákat. Az *Attempto Controlled English* egy részhalmazára épül, ami egy az angol nyelv bonyolult struktúráit is lefedő kontrollált nyelv. Az állítások bevitele történhet szabad szöveges mezőben gépeléssel, vagy a megjelenített lehetséges folytatások (kategória szerint csoportosítva) listájában kattintással (*3/d. ábra*). Az

ACEWiki-re alapozva elérhető egy érdekes alkalmazás (kezdeményezés) is, melynek célja természetes nyelvű leírásokból képzett formális logikai ábrázolással az orvosi praktizálás (döntéshozás) segítése [30].

Többféle megközelítése létezik tehát a jóslással segített szövegszerkesztésnek, és noha ezek egyenként mind egyszerűsítették a formális lekérdezések vagy állítások megadásának folyamatát, ahhoz, hogy felhasználó inspirálva legyen a használatára (ami a szemantikus publikálás elterjesztéséhez szükséges), fontos, hogy lehetőleg minél intuitívabb és barátságos legyen a szerkesztőfelület.

3. Publikációgyűjtemény tudásbázisának építése természetes nyelven

Ebben a fejezetben részletesen bemutatom, milyen megoldást hoztam létre a publikációk szemantikus értelmezéséhez és feldolgozásához.

3.1 Célkitűzések

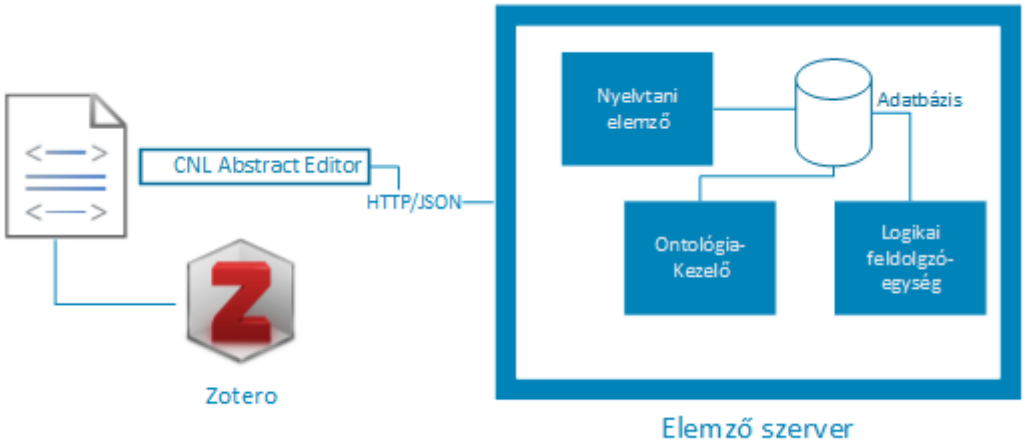
A munkám során egy olyan alkalmazás fejlesztése volt a cél, amelynek segítségével elkészíthető publikációk szemantikus kivonata természetes nyelven, és ezen kivonatokból egy tudásbázis építhető fel a publikációgyűjteményekhez.

A rendszer tartalmaz egy szövegszerkesztő felületet, ahol lehetőség van természetes nyelvű állítások felvételére. Az állításoknak a kontrollált nyelvtan által meghatározott struktúráknak kell megfelelniük. Az ehhez való alkalmazkodást a szerkesztőfelület segíti. A szerkesztőben megadott mondat véglegesítésekor elemzésre kerül, és hozzárendelődnek az elérhető szemantikus információk illetve elkészül a szemantikus reprezentációja.

A gyűjteménybe rendezett publikációkhoz az állítások alapján létrehozható egy formális logikai tudásbázis, melyre alapozva lehetséges logikai keresés és következtetés a gyűjteményben található cikkekben.

A fejlesztés során fontos szempont volt, hogy a program dinamikusan kezelje mind a nyelvtant, mind pedig a betöltendő ontológiákat, amelyekre az állítások hivatkoznak. A továbbiakban a rendszert egy olyan kontrollált nyelvtanon mutatom be, amely egy orvos biológiai alkalmazási területre készült. Az alkalmazás rákbetegségek genetikai kutatásaival kapcsolatos egyszerű angol nyelvű állítások megfogalmazását támogatja.

3.2 Architektúra áttekintése



5. ábra. Rendszer vázlatos felépítése

A kialakított rendszer vázlatos felépítése az *5. ábrán* látható. Egy kliens-szerver architektúrát terveztem meg. A felhasználónak a **2.1.1** alfejezetben bemutatott *Zotero* nevű alkalmazással van lehetősége az ott megszokott módon a publikációk gyűjtésére és elmentésére. Az itt felépített gyűjtemények elemeihez egyenként elkészíthetőek a szemantikus kivonatok a *CNL Abstract Editor* segítségével, amely az általam fejlesztett *Zotero* kiegészítő. Ez a szövegszerkesztő modul folyamatosan kommunikál egy *Elemző szerver*rel, amely a prediktív szövegszerkesztéshez szükséges információt (az éppen szerkesztés alatt álló mondat lehetséges folytatásait az adott ponton) állítja elő.

Az *Elemző szerver*en felhasználónként elkészül egy nyelvtani konfiguráció, a szerveren tárolt nyelvtani leírások alapján. Az *Ontológia-kezelő* tölti be az adott nyelvtanhoz szükséges ontológiákat a nyelvtani leírásba. A szövegszerkesztőtől kapott mondatok elemzését a felhasználóhoz tartozó *Nyelvtani elemző* végzi el, és ez állítja elő, hogy milyen folytatások lehetségesek az adott ponton. Az absztraktok elkészítése után azokból a *Logikai feldolgozóegység* állítja elő a logikai reprezentációt és elmenti a szerver adatbázisában. A szerver komponensei mind *Java* platformon futnak.

3.3 Kontrollált természetes nyelvű kivonat szerkesztője

A cikkekhez tartozó szemantikus kivonatok létrehozásához elkészítettem a *CNL Abstract Editor* nevű *Zotero* plugint. Ez egy *Firefox* kiegészítő is egyben, tehát a korábban említett *XUL* leíró nyelven kellett létrehozni a kiegészítő felhasználói felületét. Ennek két fő része volt:

- A *Zotero* publikáció listájában a publikációkra jobb egérrel való kattintással megjelenő publikáció kezelő menü kiegészítése egy új elemmel (`menuitem`), melyre való kattintással megjelenik a szemantikus szövegszerkesztő. Ennek az `overlay.xul` fájlban kell definiálva lennie, mivel ez épül rá a többi felületre.
- A szövegszerkesztő felugró ablak felépítésének leírása. Itt az *XUL* építőelemei mellett *HTML* elemeket is használtam, amelyek a megfelelő névtér importálásával elérhetőek. A dolgozat készítésekor ennek a felületnek még rendkívül egyszerű a felépítése, és csak a lehetséges folytatások listájából, egy szövegdobozból és irányító gombokból áll. Lehetséges azonban olyan továbbfejlesztés, hogy a szövegszerkesztő a *Zotero* megjegyzéseihez elérhető *WYSIWYG* (*What You See Is What You Get*) szerkesztőhez hasonlóan nézzen ki.

A felületre érkező felhasználói eseményeket (gombkattintás, új karakter gépelése) a felület leírásához csatolt *Javascript* szkriptek végzik. Az XUL leírásban, a szokott módon, lehetséges a

Javascript függvények közvetlen meghívása. Szintén a szkriptek segítségével érhetőek el a *Zotero* adatbázisából a cikk adatai, illetve a szkriptből az adatbázis módosítható is, így lehetséges a szemantikus kivonatokat közvetlenül a *Zotero* adatbázisába elmenteni, mint megjegyzések.

A szövegszerkesztés során minden karakterleütésre (`onkeypress`[14] esemény) küld egy *JSON* (*JavaScript Object Notation*) tartalmú *HTTP* üzenet a szervernek, majd a kapott válasz alapján frissíti a szerkesztői ablakban a lehetséges folytatások listáját, a felületi elem azonosítója alapján (vagy egyéb visszajelzést ad).

3.3.1 Kommunikáció a szerverrel

A *CNL Abstract Editor* és az *Elemző szerver* között aszinkron kommunikáció valósul meg, *AJAX* (*Asynchronous JavaScript and XML*) technológiákra alapozva. Annyi a különbség az *AJAX*-hoz képest, hogy *XML* helyett *JSON* típusú adatok kerülnek cserére a *HTTP* üzenetekben. Azért a *JSON*-t választottam, mivel annak készítése és feldolgozása *JavaScript*-ben és *Java*-ban egyaránt rendkívül könnyű (utóbbihoz az *Apache Jena* keretrendszerben elérhető segédosztályokat használtam – l. **3.4.2** alfejezet).

Egy publikációhoz tartozó szemantikus absztrakt megnyitásakor vagy készítésekor először el kell küldeni egy nyitó üzenetet, mely alapján a szerver inicializál egy elemzőt a felhasználónak az adott publikációhoz. Ezután szerkesztés közben minden új karakter után küldésre kerül egy új üzenet a megfelelő azonosítókkal és az új karakterrel. A kommunikáció aszinkron, így nincs blokkolás a szerver válaszára várva a kliens oldalon, hanem a *HTTP* kérésekhez, megadható egy függvény, ami akkor fut le, mikor a válasz megérkezik (eseménykezelő). A szerver válasza tartalmazza az aktuálisan lehetséges folytatások listáját, illetve hogy érvényes-e az addig elkészített szöveg.

Minta egy kliens HTTP kérés tartalmára szövegszerkesztés közben:

```
{
     "user_id":"z.hornyak",
     "msg_type":"edit",
     "new_char":"a"
}
```

Az erre adott szerver válasz (ha pl. eddig "pa" került begépelésre):

```
{
```

[14] Ez karakter "lenyomás" nem "gépelése" esemény, mely a speciális karakterek (pl. backspace) kezeléséhez kell.

```
2 |      "valid": true,
3 |      "prompts": [
4 |            "patient",
5 |            "patients"
6 |      ]
7 |}
```

3.4 Elemző szerver

A szerver feladata elsősorban az imént bemutatott kliensoldali alkalmazás kéréseinek kiszolgálása. A szerveren egy `ElemzőManager` modul a bejövő kéréseket a felhasználóknak tartozó nyelvtani elemzőknek továbbítja, melyek elvégzik a kapott szöveg elemzését – frissítik belső állapotukat és visszajelzést adnak, hogy az éppen gépelt állítás elfogadható-e (lehetséges helyes befejezése) és amennyiben igen, milyen folytatások következhetnek. Szintén a szerveren fut egy logikai feldolgozó egység, mely képes a nyelvtani elemzés eredménye alapján a formális logikai leírásának elkészítésére és mentésére.

A nyelvtani elemzőtől elvárt funkcionalitások a következőek: szavak (tokenek) egyesével történő fogadása; adott mondatbeli pozícióban lehetséges folytatások listájának megadása; kész mondat elemzésének elvégzése; sikeres elemzés esetén a mondathoz (annak elemeihez) szemantikus értelmezéséhez szükséges információ előállítása. A kontrollált nyelvtan formája összekapcsolódik az elemzőjével, ezzel kapcsolatban elvárás az ontológiákra való hivatkozás bekapcsolása és a szimbólumokhoz szemantikai információ rendelhetősége.

A szerver alkalmazásomba az *ACEWiki* részeként nyílt forráskóddal elérhető[15] diagramelemzőt építettem be, amely az összes szükséges elvárást teljesíti, és integrálása is viszonylag könnyen megoldható feladatnak bizonyult. A program szerkezete szerint a nyelvtani elemző a többi elemtől függetlenül cserélhető, de egyelőre az *ACEWiki* `ChartParser` megfelelőnek bizonyult a megoldáshoz. A továbbiakban az implementációjával kapcsolatos részleteket ismertetem.

3.4.1 Nyelvtan létrehozása

Teljes rendszer szempontjából központi kérdés és fontos tudásmérnöki feladat azon kontrollált nyelvtanok létrehozása, amelyekre alapozva szeretnénk a cikkek szemantikus kivonatait elkészíteni. A **2.3.1.2** alfejezetben már említettem, hogy a nyelvtan felépítése már önmagában hordoz szemantikai információt, tehát a nyelvtani szabályok leírásakor nem csak azt kell szem előtt tartani, hogy milyen természetes nyelvi struktúrák kerüljenek elfogadásra, hanem hogy mondatszerkezetből mely elemek hordoznak olyan információt, amelyet később ki akarunk nyerni. Ezért vettem fel például a mondatokban előfordulható gén neveket, mint egy nem

[15] http://attempto.ifi.uzh.ch/acewiki/

terminális szimbólum egyértelmű kifejtései[16], így a feldolgozás során mikor a mondatban releváns gént keresem, akkor az elemzési fában az annak megfelelő nem terminális szimbólum ágának levelét tudom kiolvasni. Az információkinyerésre példát a **3.5** alfejezetben mutatok.

A nyelvtanok könnyen megtervezhetőek például elemzési fák kézzel történő felépítésével, majd ezeket a megfelelő jelölésrendszerben le kell írni.

Az *ACEWiki* a *Codeco* jelölésrendszert használja a korlátozott nyelvtanok definiálására. A *Codeco* elnevezés a „konkrét és deklaratív nyelvtani jelölés kontrollált természetes nyelvekhez" rövidítéseként jött létre [31].Egy *BNF*-hez hasonló jelölésrendszer, kiterjesztve bonyolultabb nyelvtani konstrukciók támogatására, mint az anaforák illetve a hatáskörrel rendelkező szavak (pl. minden). A jelölésrendszer teljes kifejező erejének én csak egy kisebb részét használom egyelőre a munkám során.

A *Codeco* a *BNF*-hez hasonlóan terminális és nem-terminális szimbólumokból áll, ezeket kiegészítve a speciális *pre-terminális* szimbólummal. A *terminális* szimbólum egy kötött szöveget jelent – az alkalmazás szavankénti feldolgozása miatt az alkalmazás során minden terminális szimbólum alatt egy konkrét szót értek. A *pre-terminális* szimbólumok olyan szimbólumok, amelyek kizárólag terminális szimbólumokra vezethetőek vissza. *A nem-terminális* szimbólumok bármilyen típusú elemek szekvenciájaként felépíthetőek.

A jelölésrendszert egy példán szemléltetve:

```
1 | sentence → [Patient] [had] genechange [of] gene
2 | change  → [overexpression]
3 | change → [mutation]
4 | genechange → change
5 | genechange → change [and] change
```

A példában minden szögletes zárójellel határolt szó terminális, a simán szereplő szavak alkotják a nem-terminális szavakat, az aláhúzottak pedig a pre-terminálisokat. Egy-egy sor egy-egy szabálynak felel meg – az 1., 4. és 5. sor nyelvtani szabálynak, míg a 2. és 3. sor lexikai szabálynak minősül. Lexikai szabályokat a pre-terminálisok kifejtésekor lehet (kell) használni. A gene pre-terminális szimbólum definícióját szándékosan nem szerepeltettem a példában.

A *Codeco* nyelvtan implementációja támogatja, hogy a nyelvtani leírás később módosuljon, így például nem kell inicializáláskor megadni egy pre-terminális összes kifejtését. Mindezek a nyelvtanhoz rendelt dinamikus lexikonnal lehetségesek. Az *ACEWiki* implementációjában a

[16] Avagy pre-terminális, lsd. később.

`DynamicLexicon` interfészből származtathatjuk ezeket a lexikonokat. A lexikontól elvárt funkcionalitás, hogy megadja egy adott szimbólumra vagy szóra a hozzá kapcsolódó lexikális szabályokat. A jelenlegi megvalósításom során ezt a funkcionalitást (egyelőre) két helyen használtam fel:

- A nyelvtanban a gének nevét a `gene` pre-terminális szimbólum jelzi. A tényleges értékeit a szimbólum a kapcsolódó (szűrt) ontológia betöltése után kapja meg. A dinamikus lexikon tárolja a betöltött ontológia információt, és az alapján generálja le a lexikális szabályokat a szimbólumhoz.
- A kísérleti eredményekre hivatkozó mondatszerkezetnél („*For detailed results see computation 234*”) a kísérlet azonosítóját a mondatban egy `number` nevű pre-terminális szimbólum jelöli. Ennek lehetséges értéke nem, csak típusa fix. A dinamikus lexikon lekérdezéskor random szám példákat generál a szimbólumhoz, és a szimbólum helyén olyan bemeneteket fogad el, amelyek tényleges számokként értelmezhetőek (ez reguláris kifejezéssel van ellenőrizve).

A dinamikus lexikális szabályok lehetőséget biztosítanak arra, hogy "új" felhasználói bejegyzés is keletkezhessen egy nyelvtanra illeszkedően. Például a nyelvtanok hivatkozhatnak génvariánsokra úgy, hogy nem kell előre ismerni az összes létező típust (hiszen ezek száma hatalmas és folyamatosan növekszik), hanem csupán létrehozni egy dinamikus szabályt amely például a *HVGS* nomenklatúrára való illeszkedést vizsgálja egy reguláris kifejezéssel [32].

A *Codeco* leírással megadott nyelvtannak létre kell hozni a *Java* nyelvű reprezentációját a működéshez (egy `ch.uzh.ifi.attempto.chartparser.Grammar` típusú objektum). A korábban bemutatott leírás alapján a `ch.uzh.ifi.attempto.codeco` csomagban elérhető `generate_java` *Prolog* függvény segítségével legenerálható a megfelelő Java osztály. A *Prolog* függvény használatához a nyelvtan leírását *DCG*-hez hasonló formában kell megadni, amely az előbbi példára a következőképpen néz ki:

```
1 | sentence => [patient], [had], genechange, [of], $gene.
2 | $change => [overexpression].
3 | $change => [mutation].
4 | genechange => $change.
5 | genechange => $change, [and], $change.
```

A *Prolog* leírásban tehát a `$` jelöli a pre-terminális szimbólumokat. A konverzió egyszerűen megoldható, csupán arra kell vigyázni, hogy a terminálisok mind kisbetűs szavak legyenek, mert egyébként a *Prolog* mint változó próbálná őket értelmezni (ami hibához vezetne). A *Java*

implementációban a nagybetűt tartalmazó terminál is értelmezett, így amennyiben ez szükséges, kézzel módosíthatjuk (vagy létrehozhatunk saját nyelvtan konverziós programot).

Az **F1** függelékben mellékeltem egy részletet a példa alapján generált *Java* nyelvtan implementációból. A struktúra könnyen érthető és elkészíthető például a dinamikus lexikon használatakor is. A lexikális illetve nyelvtani szabályoknak a `LexicalRule` és `GrammarRule` osztályok felelnek meg – ezeknek egy szimbólumlistát (a szabályok balról jobbra olvasott sorrendezése szerint) és egy `Annotation` objektumot kell átadni létrehozáskor. Ez utóbbi objektumot használom fel arra, hogy a szabályokhoz különböző szemantikai információt rendeljek (kulcsszavas `Map` objektum). Mivel egy pre-terminálisnak megfelelő minden érték egy új szabály felvételével adható meg, a szabályon keresztül érték szerinti információ-hozzárendelés lehetséges (például a gének neveihez azok adatbázis azonosítóit).

Amikor a nyelvtan átadásra kerül az elemzőnek, manuálisan kell megadni a nyelvtan elemzési fájának gyökér elemét, ahonnan az algoritmus elindulhat.

3.4.2 Dinamikus ontológia betöltése

Az előbb ismertettek alapján készíthető el tehát nyelvtani elemző működéséhez szükséges nyelvtan. Ezt azonban ki kell egészíteni a megfelelő ontológiai információkkal, például a `gene` pre-terminális feloldásához. Az *Ontológia-kezelő* felelős az ontológiák betöltéséért, feldolgozásáért és a nyelvtanhoz szükséges szóhalmaz előállításáért.

Az alkalmazás fejlesztése során szempont volt, hogy ontológiák használatát dinamikusan lehessen támogatni. A jelenlegi verzióban egyetlen ontológiát használtam, a protein struktúrákat leíró *PRO* ontológiát [33]. Ez az ontológiát a korábban bemutatott *OBO* szervezet szerkeszti. A *PRO* a szabványos *OBO* formátumon kívül *OWL* formátumban is elérhető, a programomban az utóbbit tudtam integrálni.

A *PRO* ontológia fehérjék illetve összetett fehérjék reprezentációját tartalmazza, azokról több különböző adatot tárolva: szöveges leírást, hierarchia kapcsolatot, kapcsolódó adatbázis azonosítót (pl. *UniProt* szekvenciára), kategóriát, állapotokat, szinonimákat, stb.. A nyelvtan építéséhez nekem a sok adatból mindössze az egyszerű szöveges megnevezésre volt szükségem, amely a kivonatokban megjelenhet, illetve az adott fehérje *PRO* azonosítójára, mely alapján a részletes információk az ontológiából elérhetőek a *Protein Information Resource* (*PIR*) weboldalon[17].

[17] http://pir.georgetown.edu/pro/pro.shtml

A *PRO* ontológia legutolsó verziójában kb. 36 ezer különböző fogalom szerepelt, amelyek több mint 90 ezer protein szekvenciához vannak hozzárendelve. Ekkora mennyiségű adattal nehéz, és felesleges is dolgozni (hiszen nincs mindenre szükség), ezért a programom támogatja az ontológia részének egy adott szűrő szerinti betöltését (a szűrő dinamikusan előállítható, de jelenleg nincs hozzá felhasználói felület, hanem a kódban kötött módon történik).

Az ontológiák kezeléséhez az *Apache Jena*[18] keretrendszert használtam. A *Jena* keretrendszer célzottan szemantikus web alkalmazások készítéséhez lett létrehozva, és segíti az ontológiák betöltését, feldolgozását, lekérdezését illetve a bennük történő következtetést is.

A nagyméretű ontológia fájl kezelését (`pro.owl` esetén 80 MB-os méret) a *TDB* interfész segítségével oldottam meg. Ehhez először az elérhető konverziós programokkal átalakítottam az ontológiát a *TDB* formátumnak megfelelő reprezentációra, amely noha nagyobb helyet foglal el utána a merevlemezen (közel kétszeres), az elérési illetve kezelési ideje jelentősen felgyorsult (mivel modell reprezentáció létrehozásakor nem kerül betöltésre az egész ontológia).

Az ontológiában történő szűrést *SPARQL* segítségével oldottam meg. A *SPARQL* egy lekérdező nyelv *RDF* formátumú dokumentumokhoz, az *SQL*-hez hasonló egyszerű felépítéssel. Az objektumértékekre való szűrésnél figyelembe kellett venni, hogy a *TDB*-ben létrehozott ontológia modell tipizált (tehát entitás értékre nem lehet közvetlenül szűrni, csak `FILTER`-rel). A humán gének kiszűrésére generált lekérdezés a *PRO* ontológiához:

```
1 PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
2 PREFIX oboInOwl: <http://www.geneontology.org/formats/oboInOwl#>
3 SELECT ?label ?id
4 WHERE { ?x oboInOwl:id ?id .
5 ?x oboInOwl:hasExactSynonym ?label .
6 ?x rdfs:comment ?c .
7 FILTER (regex(?c,'.*Category=organism-gene.*'))}
```

A szűrés után így kapott objektumok száma (~1000) már teljesen jó sebességgel kezelhető volt a program által. A kapott információ alapján felépítésre került a dinamikus lexikon `gene` része, a `label` információt mint terminális, az `id`-t pedig mint megjegyzés tartalmazva.

3.4.3 Nyelvtani elemző működése

Az előző két pontban ismertettem, hogyan hozhatóak létre kontrollált nyelvtanok egy adott területre, ontológia hivatkozásokkal együtt. A kliens szövegszerkesztő (*CNL Abstract Editor*) indulásakor küld egy inicializáló üzenetet, mely meghatározza, milyen nyelvtani

[18] http://jena.apache.org/

konfigurációval kell létrehozni a felhasználó számára az elemzőt. A konfiguráció egy nyelvtan és a kapcsolódó ontológiák megadását jelenti, továbbá az elemzőnek a nyelvtan kezdőszimbólumának átadását.

A szövegszerkesztőtől kapott információ alapján szavanként frissítem[19] a megfelelő nyelvtani elemzőt, amely egy adott mondatot elemez inkrementálisan. Az elemzés közben bármely ponton lehetőség van eldönteni, a készített mondat még megfelel-e a nyelvtani szabályoknak, és amennyiben igen, milyen folytatások lehetségesek.

Az *ACEWiki*-ben implementált nyelvtani elemző Earley diagramelemzési módszere alapján működik [34], némi kiegészítéssel a nyelvtanban megjelenő különleges konstrukciók kezelésére (pl. hatáskör). Noha nem személyesen implementáltam az algoritmust, most mégis röviden ismertetném, mert megértése fontos lehet a jövőbeli továbbfejlesztések szempontjából. Az implementáció részletesebb leírása Kuhn tanulmányában ([31]) elérhető.

Tekintsünk az alábbi nyelvtani szabályt, ahol A egy nem-terminális szimbólum, α és β pedig szimbólumsorozatok:

$$A \rightarrow \alpha\ \beta$$

Az Earley algoritmus működése során a szabályok, mint élek kerülnek felvételre a bemenetként kapott token (szó) sorozat pozíciói között, az alábbi információkkal:

$$(p_1, p_2)\ A \rightarrow \alpha \bullet \beta$$

Itt p_1 illetve p_2 jelölik a két pozíciót, amely között az A szabály értelmezve van úgy, hogy α sorozat már le van fedve, de a β még nincs ($\bullet$ jelzi a feldolgozottság szintjét).

Az algoritmus futása során tokenek (x_i) feldolgozásával iteratívan kerülnek felvételre az élek (lépésenként halmazokba gyűjtve, a régebbiek nem törölve). A futás ott áll meg, ha már nem lehet több élet felvenni vagy sikerült a gyökér szabálynak megfelelő élet felvenni. Az algoritmusnak minden körben három alaplépése van [35]:

- Keresés: Minden meglévő élre, amelyhez tartozó szabálynak még van lefedetlen része, ellenőrzi, hogy az új szónak köszönhetően felvehető-e új (hosszabb) él a szabállyal. Tehát minden meglévő

  $$(p_j, p_{i-1})\ A \rightarrow \alpha \bullet y\ \beta$$

[19] A CNL Abstract Editor karakterenként küld üzenetet, de a nyelvtani elemzőt csak egy szó vagy mondat befejezése után kell frissíteni.

élre, ha $y = x_i$, akkor egy új él felvételre kerül:

(p_j,p_i) A → α y • β

- Jóslás: a folytatás jóslásához az aktuális végpontokba „hurok éleket" vesz fel. Tehát a keresés után ha van egy

  (p_j,p_i) A → α • B β

  szabály, akkor a p_i pozícióba felveszi az alábbi élet:

  (p_i,p_i) B → • μ

- Teljesítés: A teljesen feldolgozott szabályokkal frissíti a rájuk hivatkozó szabályok éleit. Például A hivatkozik B-re, és B már készen van az adott pozícióban:

  (p_j,p_k) A → α • B β

  (p_k,p_i) B→ μ •

  Akkor felvehető az alábbi él:

  (p_j,p_i) A → α B • β

Az *ACEWiki* implementáció ezt kiegészíti egy negyedik, rezolúciós lépéssel, amely feloldja a *Codeco* nyelvtanon belüli hivatkozásokat is (anafora és társai).

A folytatás jóslásához tehát az elemző algoritmus leállása után az utolsó pozícióhoz tartozó hurok éleket lehet lekérdezni. Ezt kétféleképpen is támogatja az elemző osztály (`ch.uzh.ifi.attempto.chartparser.ChartParser`): `getAbstractOptions` metódussal a következő szimbólumot (és annak megkötéseit), míg a `getConcreteOptions` metódussal a következő terminális szimbólumok kérdezhetőek le. Az elemző az elemzési lépés minden állapotához tartozó éleket menti, így visszalépés esetén nem kell a teljes bemenetet újra elemezni. Ha valamely ponton semmilyen folytatási lehetőséget nem ad a nyelvtani elemző, abból már tudhatjuk, hogy a készülőben lévő mondat nem megfelelő struktúrájú.

A teljes mondat elemzése a `getParseTree` metódus meghívásával történik. Amennyiben a vizsgált mondat (az eddig megadott szavak sorozata) megfelel a nyelvtannak, úgy az elemző visszaadja a mondatnak megfelelő elemzési fát, ellenkező esetben azonban egy `null` értéket.

3.4.4 Logikai tudásbázis

A publikációkhoz létrehozott szemantikus kivonatok létrehozásának befejeztével lehetséges a kivonatok állításait elmenteni formális logikai reprezentációban. Ennek elkészítéséért a logikai feldolgozóegység felel.

3.4.4.1 Logikai reprezentáció elkészítése

A kivonatba a szerkesztői felületen csak olyan mondatok készülhettek el, amelyek a kontrollált nyelvtannak megfelelnek. A kész kivonat bármikor feldolgozható úgy, hogy a benne lévő mondatokat egyesével elküldjük a nyelvtani elemzőnek (a kivonatnak alapját képző kontrollált nyelvtan szerint inicializált). A sikeres elemzés után a nyelvtani elemző visszaadja az adott mondathoz tartozó elemzési fát. Ebben szerepel az illeszkedő szimbólum struktúra, a szabályokból kapott `Annotation` elemekkel együtt, így kinyerhetem belőle a szükséges szemantikai információkat.

Az elemzési fából többféle módon és alakban is felépíthetünk szemantikus reprezentációt. Én az egyes állítások gyökér szimbólumaihoz (megjegyzés: ez nem a teljes nyelvtan gyökere csak egy adott mondaté) egy úgymond "paraméterezhető" szöveget rendelek. Ennek ismereteséhez most egy *Prolog* formális leírás készítését mutatom be, de az elképzelés alkalmazható például *RDF* leírás előállítására is.

A **3.4.1** fejezetben bemutatott nyelvtanrészlet alapján elfogad az elemző olyan természetes nyelvű állítást például, hogy "*Patient had mutation of BRCA1*" ("A páciensnek BRCA1 mutációja volt"). Ezt egy *Prolog* állításnak szeretném megfeleltetni a következő formában:

```
1 |patient(id,"mutation","BRCA1")
```

Itt az `id` annak a publikációnak az azonosítója, amelyhez az adott állítás készült. Ennek létrehozásához az állításnak megfelelő gyökérszimbólumhoz, a `sentence`-hez a következő annotációt rendelem:

```
1 |patient(#pubid#,#change#,#gene#)
```

Az annotáció kulcsának persze jeleznie kell, hogy paraméterezett szövegről van szó. A paraméterek neve két `#` között szerepel.

A szemantikus reprezentációt előállító logikai egység a paraméterezett szövegben a `pubid` kivételével úgy helyettesíti be a paraméter értékeket, hogy megkeresi az elemzési fában nekik megfelelő ágakat, és onnan kiolvassa megfelelő záró – levél – értéket[20].

A logikai feldolgozó egység tehát a cikkek kivonatai alapján felépít úgy egy tudásbázist, hogy mondataikat egyesével elemezteti a nyelvtani elemzővel, majd a kapott elemzési fa és a nyelvtan szimbólumaihoz definiált paraméterezett szövegek alapján elkészíti a mondatoknak

[20] A példában a gene preterminális feloldása egyértelmű, hiszen egyszer szerepel a nyelvtanban, de a change-nek már több előfordulását kell kezelni. Ilyen esetben duplikálható a formális állítás úgy, hogy a többesélyes helyen más behelyettesítés is lesz.

megfelelő *Prolog* logikai állításokat. Ezeket összegyűjtve épül fel a tudásbázis, melyet a szerveren tárolhatunk az adott felhasználó számára.

3.4.4.2 A tudásbázis felhasználása

A *tuProlog*[21] Java keretrendszer segítségével az elkészített *Prolog* tudásbázison többféle következtetést végrehajthatunk, esetleg konzisztenciát lehet vizsgálni (hasonlóan *RDF* formális ábrázolás esetén használhatnánk az *Apache Jena* keretrendszert). A *Zotero* felhasználó felülete könnyen kiegészíthető további elemekkel, így például a tudásbázison történő kereséssel – akár logikai leírással, akár természetes nyelven, hiszen ahogy azt a **2.3.2**-ben láthattuk, a kontrollált természetes nyelveknek egy gyakori és jól működő területe a lekérdezésekhez létrehozott alkalmazások. Egy természetes nyelvű kérdés alapján a megfelelő *Prolog* kérdés felépítése szinkron módon történhet az ebben a fő fejezetben bemutatottakkal.

Elképzelhető tehát, hogy a felhasználó kíváncsi az olyan elmentett publikációira, amelyek konkrétan BRCA2 mutációval foglalkoznak (és nem csak említik a gént, vagy más oldalról foglalkoznak vele). Ekkor elkészíthető az alábbi lekérdezés a tudásbázison:

```
1 | ?- patient(ID,"mutation","BRCA2").
```

Erre a kérdésre válaszként a *Prolog* következtető vissza fogja adni azokat az `ID` értékeket, amelyekkel szerepelt a megfelelő állítás – tehát azoknak a cikkeknek az azonosítóit, amelyekben ténylegesen BRCA2 mutációjú páciensekről volt szó.

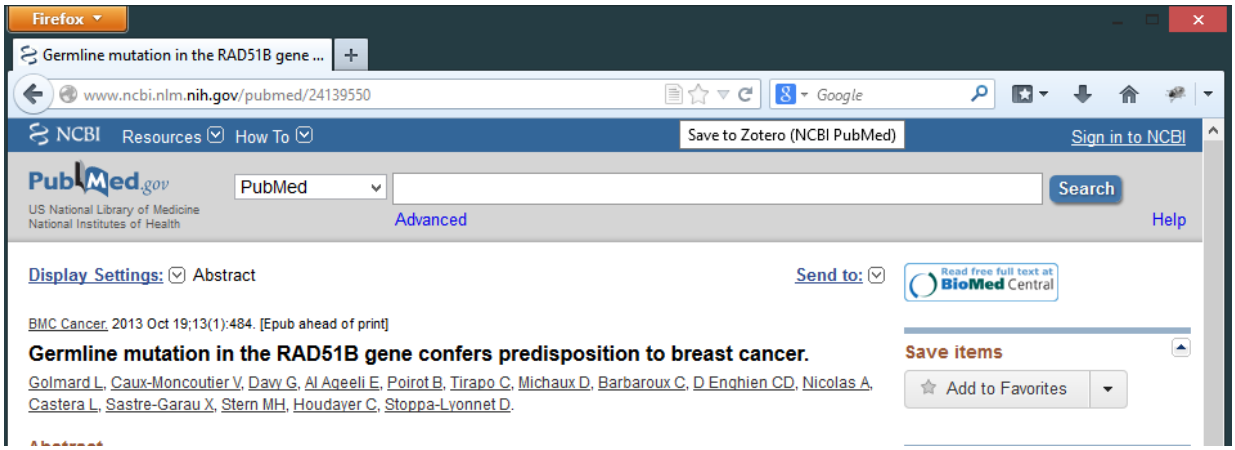
Prolog kérdések persze nem csupán a cikkek kereséséhez használhatóak az ezen elképzelés szerint felépített tudásbázison. A szemantikus kivonatban szerepelhetnek olyan állítások, melyek egy eredményt közölnek, például, hogy az adott gyógyszer használt vagy nem használt egy bizonyos betegség kezelésénél. Ekkor a tudásbázist lekérdezhetjük úgy is, hogy szabad változóként szerepel a kísérlet eredményessége. A kapott válasz halmazon láthatjuk, hogy levonhatóak-e általános következtetések (a cikk gyűjtemény alapján) egy adott gyógyszer és betegség kapcsolatára.

3.5 Alkalmazási példa

Az elkészített rendszer működését egy orvosi kutatási példán szemléltetném. Tegyük fel, hogy egy kutató olyan cikkeket gyűjt, amelyek leírják, az egyes génmutációval rendelkező betegek milyen kezelésre hogyan reagáltak. Ez ma egy nagyon aktív kutatási téma, és rengeteg publikáció elérhető a téren. Az internetes publikációkeresés közben ha talál egy kutató egy cikket, azt elmentheti a saját *Zotero* gyűjteményébe. A *6.ábrán* látható, hogy a talált cikknek

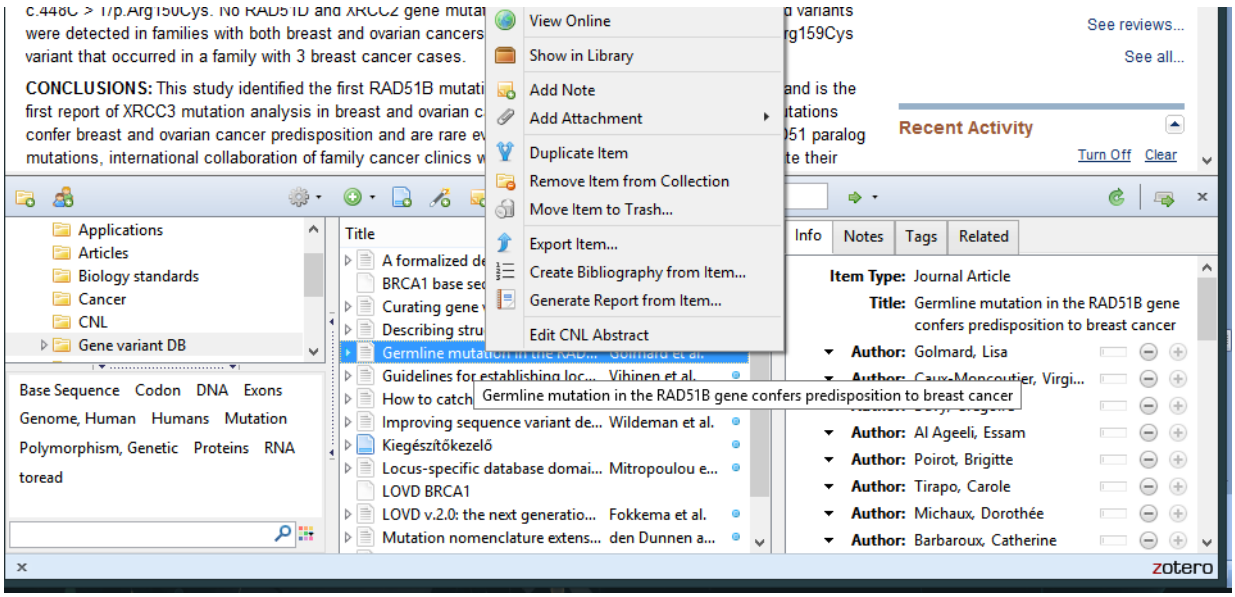
[21] http://apice.unibo.it/xwiki/bin/view/Tuprolog/

megfelelő oldalon a címsávban megjelenik egy kis ikon (csillagtól balra), amelyre kattintva a publikáció elmenthető az összes metaadatával együtt a gyűjteménybe.



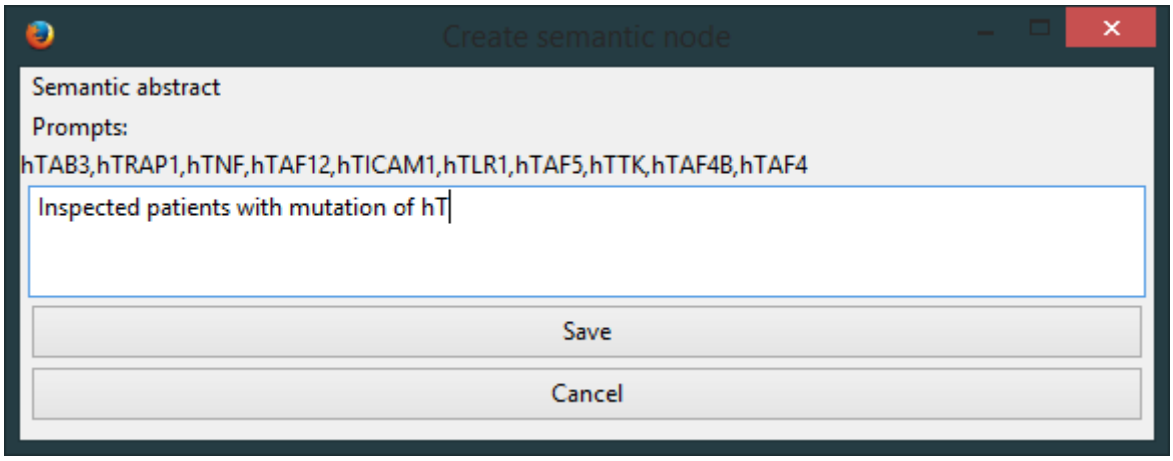
6. ábra. Publikáció mentés Zotero-ba

Ezután megjelenik a publikáció a *Zotero* listájában (a megfelelő gyűjteményen belül). Ha a kutató elkezdi olvasni a cikket, lehetősége van közben belőle a fontosabb állításokat a *CNL Abstract* szerkesztőben felvenni, mely az adott cikkre kattintva érhető el.



7. ábra. Zotero tárolás és szerkesztő megnyitása

Az új ablakban megjelenő szerkesztőn a indulástól fogva megjelenített tippek segítségével történhet a mondatok begépelése, amely alkalmazkodik ahhoz is, hogy az adott részlegesen begépelt szó milyen lehetőségekre illeszkedik – l. *8. ábra*. Az ábrán látható mondat jelentése, hogy valamilyen (hT kezdetű) gén mutációval rendelkező betegeket vizsgáltak.



8. ábra. Absztrakt szerkesztése

Amennyiben a cikkhez az összes releváns (megengedhető) állítást felvette a kutató, akkor a kész absztraktot elmentheti a *Zotero*n belül. Mikor már több cikkhez is ily módon elkészítette az absztraktot, akkor az adott gyűjtemény publikációihoz elkészíttetheti a szemantikus reprezentációt.

A tudásbázist később többféle keresésre is felhasználhatja. Amennyiben egy betegén genetikai vizsgálatot végez el, és valamely génben mutációt fedez fel, a tudásbázisban hatékonyan megkeresheti, mely cikkek foglalkoztak konkrétan azzal a mutációval, ott milyen gyógyszerkezeléseknek milyen eredményei lettek, stb.

4. Összefoglalás

A legtöbb tudományos publikáció ma már digitálisan, online is elérhető – ám az információ feldolgozását segítő technológiák alig kerülnek kihasználásra. A szemantikus publikálás problémaköre foglalkozik azzal, hogyan lehetséges a publikációkat tartalmi alapon, szemantikájuk alapján ábrázolni, feldolgozni illetve összefoglalni. Az elképzelés alapját képező technológiák a szemantikus web koncepció fejlődése során már megjelentek, ám elterjedésüket gátolja, hogy viszonylag nagy időbefektetés igényel megismerésük illetve adaptálásuk. Szintén segíthet az új technológiák elterjesztésében, ha meg tudjuk mutatni a felhasználóknak, milyen haszonnal jár számukra az alkalmazásuk.

4.1 Az alkalmazás értékelése

A dolgozatomban bemutatott rendszer célja a publikációkhoz szemantikus reprezentáció létrehozása oly módon, hogy az a felhasználóknak könnyű és hasznos is legyen. Az ismertetett szerkesztői felület segítségével a szemantikus kivonatok létrehozása olyan kézenfekvő módon történhet, mint egy rendes természetes nyelvű összefoglaló elkészítése. A háttérben futó korlátozott nyelvtani elemzőhöz egyszerű leírásban készíthetünk nyelvtanokat, és azokhoz dinamikusan bekapcsolhatóak ontológiák is. A nyelvtanok és a hozzájuk tartozó szemantikus értelmezés elkészítéséhez természetesen szükség van a technológia és szakterület együttes ismeretére is, azonban amennyiben ezt egy tudásmérnök elvégzi, a kapott konfiguráció már tetszőleges számú, laikus felhasználó által használható.

A bemutatott *Zotero* kutatási segédeszközbe való integráció rengeteg lehetőséget biztosít, melyeket itt még csak felvetíteni tudok. A cikkekhez készített szemantikus kivonatok és a *Zotero* publikáció rendszere alapján akár gyűjteményenként felépíthetünk tudásbázisokat (esetleg különböző nyelvtani konfigurációk alapján). A tudásbázis ezután használható magában a gyűjteményekben való szemantikus keresésre, vagy akár következtetések futtatására is – ezek megvalósítása a felépített architektúrába könnyen beépíthető.

Noha a *Zotero* lehetőséget biztosít gyűjtemények csoportos kezelésére, egy gazdag, érdemi tudásbázis felépítéséhez még így is sok munka lehet, mivel sok cikket kell elolvasni hozzá, és azokat mind szemantikus kivonatokkal ellátni. A dolgozatomban csak említésre került, de összetett probléma lehet a felhasználók általi új tartalmak támogatása illetve kezelése. Érdekes kérdés lehet továbbá, hogy a felhasználók által elkészített tudásbázisok hogyan oszthatóak meg és kapcsolhatóak össze más tudásbázisokkal (akár más nyelvtanra alapozva).

4.2 Merre tovább

A munkám egy olyan alkalmazás készítésére fókuszál, mellyel személyes publikációgyűjtemények rendszerezése, és szemantikus tudásbázisuk létrehozása lehetséges. A részletesen ismertetett technológiák azonban a szemantikus publikálás online, nagyobb közönségű elterjesztéséhez is hasznosak lehet.

A tudományos publikálás folyamata a jövőben nagy valószínűséggel meg fog változni, hogy lépést tartson a technológiai fejlődésekkel [36]. Hogy valaha létrejön-e szemantikus web hajnalán megálmodott nyitott, univerzális tudásháló, annak gazdasági és etikai kérdései is vannak, hiszen átalakítaná a tudományos publikációkról alkotott jelenlegi elképzeléseinket [37]. Napjainkban aktuális vitatéma, hogy a publikációk mindenki számára ingyenes elérhetőek legyenek-e [38]. Az nem kérdéses azonban, hogy egyre nagyobb igény van az online elérhető információk összekötésére. Erről tanúskodik a *Tudományos, technikai és egészségügyi kiadók szövetsége* (STM) által tett brüsszeli nyilatkozat, amelyben a kiadók megfogadják, hogy segítik az információ szabad terjedését és a hatékonyságot növelő technológiák adaptálását [39].

Amennyiben a nagy online publikációgyűjtemények vagy folyóiratokba történő publikálás során a szerzőknek kötelező lenne elkészíteni egy szemantikus kivonatot is, úgy ezen nagy gyűjtemények mögött is létrehozható lenne egy tudásbázis. A dolgozatomban bemutatott természetes nyelvű alkalmazás segítségével ez a szerzőkre sem róna túl nagy terhet, és az egyes publikációgyűjtemények szervezői tudnák a nyelvtanok segítségével kontrollálni, milyen kivonatokra van szükség és milyen felépítésű tudásbázist készítenek el. Egy ilyen szemantikus rendszerben már intelligens keresések hajthatóak végre, amelynek segítségével könnyebben megtalálhatják a kutatók a számukra érdekes kérdésekhez legrelevánsabb kutatási eredményeket. A szerzőket ösztönözheti, hogy amennyiben elkészítik a szemantikus kivonatot, úgy könnyebben találják meg publikációjukat a hasonló területen dolgozók, és így valószínűleg több hivatkozást is kapnak munkájukra.

Köszönetnyilvánítás

Szeretném megköszönni konzulensemnek, Mészáros Tamásnak a problématerület ismertetését és a megoldás hatékony fejlesztését segítő támpontjait és ötleteit. Köszönömöm továbbá Antal Péternek a szemantikus publikálás, és azon belül is az orvosbiológiai alkalmazási terület megismeréséhez nyújtott segítségét.

Függelék

F1. Példa nyelvtan Java implementációja (részlet)

```
// sentence=>[patient],[had],genechange,[of],$gene
l.clear();
featureHash.clear();
ann = new Annotation();
nonterm = new Nonterminal("sentence");
fm = new FeatureMap();
nonterm.setFeatureMap(fm);
l.add(nonterm);
term = new Terminal("patient");
  l.add(term);
  term = new Terminal("had");
  l.add(term);
  nonterm = new Nonterminal("genechange");
  fm = new FeatureMap();
  nonterm.setFeatureMap(fm);
  l.add(nonterm);
  term = new Terminal("of");
  l.add(term);
  preterm = new Preterminal("gene");
  fm = new FeatureMap();
  preterm.setFeatureMap(fm);
  l.add(preterm);
  addGrammarRule(new GrammarRule(ann, l, false));

  // $change=>[overexpression]
  l.clear();
  featureHash.clear();
  ann = new Annotation();
  preterm = new Preterminal("change");
  fm = new FeatureMap();
  preterm.setFeatureMap(fm);
  l.add(preterm);
  term = new Terminal("overexpression");
  l.add(term);
  addLexicalRule(new LexicalRule(ann, l));
```

Irodalomjegyzék

[1] "The Semantic Web: Scientific American." [Online]. Available: http://www.scientificamerican.com/article.cfm?id=the-semantic-web. [Accessed: 17-May-2013].

[2] T. Berners-Lee and J. Hendler, "Publishing on the semantic web," *Nature*, vol. 410, no. 6832, pp. 1023–1024, Apr. 2001.

[3] D. Hull, S. R. Pettifer, and D. B. Kell, "Defrosting the Digital Library: Bibliographic Tools for the Next Generation Web," *PLoS Comput Biol*, vol. 4, no. 10, p. e1000204, Oct. 2008.

[4] "The Fourth Paradigm." [Online]. Available: http://research.microsoft.com/en-us/collaboration/fourthparadigm/4th_paradigm_book_complete_lr.pdf. [Accessed: 10-Oct-2013].

[5] T. K. Attwood, D. B. Kell, P. McDermott, J. Marsh, S. R. Pettifer, and D. Thorne, "Calling International Rescue: knowledge lost in literature and data landslide!," *Biochem. J.*, vol. 424, no. 3, pp. 317–333, Dec. 2009.

[6] "Detailed Indexing Statistics: 1965-2012." [Online]. Available: http://www.nlm.nih.gov/bsd/index_stats_comp.html. [Accessed: 21-Oct-2013].

[7] "The Universal Protein Resource (UniProt) in 2010," *Nucleic Acids Res.*, vol. 38, no. Database issue, pp. D142–D148, Jan. 2010.

[8] A. Rinaldi, "For I dipped into the future," *EMBO Rep.*, vol. 11, no. 5, pp. 345–349, May 2010.

[9] D. Shotton, K. Portwin, G. Klyne, and A. Miles, "Adventures in Semantic Publishing: Exemplar Semantic Enhancements of a Research Article," *PLoS Comput Biol*, vol. 5, no. 4, p. e1000361, Apr. 2009.

[10] E. Pafilis, S. I. O'Donoghue, L. J. Jensen, H. Horn, M. Kuhn, N. P. Brown, and R. Schneider, "Reflect: augmented browsing for the life scientist," *Nat. Biotechnol.*, vol. 27, no. 6, pp. 508–510, 2009.

[11] A. de Waard, "From Proteins to Fairytales: Directions in Semantic Publishing," *IEEE Intell. Syst.*, vol. 25, no. 2, pp. 83–88, Mar. 2010.

[12] A. Ceol, A. Chatr-Aryamontri, L. Licata, and G. Cesareni, "Linking entries in protein interaction database to structured text: The FEBS Letters experiment," *FEBS Lett.*, vol. 582, no. 8, pp. 1171–1177, Apr. 2008.

[13] M. Seringhaus and M. Gerstein, "Manually structured digital abstracts: A scaffold for automatic text mining," *FEBS Lett.*, vol. 582, no. 8, p. 1170, prilis 2008.

[14] P. Ciccarese, E. Wu, G. Wong, M. Ocana, J. Kinoshita, A. Ruttenberg, and T. Clark, "The SWAN biomedical discourse ontology," *J. Biomed. Inform.*, vol. 41, no. 5, pp. 739–751, 2008.

[15] "Ontology (Computer Science) - definition in Encyclopedia of Database Systems." [Online]. Available: http://tomgruber.org/writing/ontology-definition-2007.htm. [Accessed: 19-Oct-2013].

[16] "Ontologies - W3C." [Online]. Available: http://www.w3.org/standards/semanticweb/ontology. [Accessed: 19-Oct-2013].

[17] B. Smith, M. Ashburner, C. Rosse, J. Bard, W. Bug, W. Ceusters, L. J. Goldberg, K. Eilbeck, A. Ireland, C. J. Mungall, N. Leontis, P. Rocca-Serra, A. Ruttenberg, S.-A. Sansone, R. H. Scheuermann, N. Shah, P. L. Whetzel, and S. Lewis, "The OBO Foundry: coordinated evolution of ontologies to support biomedical data integration," *Nat. Biotechnol.*, vol. 25, no. 11, pp. 1251–1255, Nov. 2007.

[18] "IBM Watson Wins Jeopardy, Humans Rally Back," *PCWorld*, 17-Feb-2011. [Online]. Available: http://www.pcworld.com/article/219900/IBM_Watson_Wins_Jeopardy_Humans_Rally_Back.html. [Accessed: 19-Oct-2013].

[19] N. Chomsky, "Three models for the description of language," *IRE Trans. Inf. Theory*, vol. 2, no. 3, pp. 113–124, 1956.

[20] G. K. Pullum and G. Gazdar, "Natural languages and context-free languages," *Linguist. Philos.*, vol. 4, no. 4, pp. 471–504, Dec. 1982.
[21] S. J. Russell, P. Norvig, and E. Davis, *Artificial intelligence: a modern approach*. Upper Saddle River, NJ: Prentice Hall, 2010.
[22] J. W. Backus, "The syntax and semantics of the proposed international algebraic language of the Zurich ACM-GAMM Conference.," pp. 125–131, 1959.
[23] A. Colmerauer, "The Birth of Prolog," in *III, CACM Vol.33, No7*, 1993, pp. 37–52.
[24] F. C. N. Pereira and D. H. D. Warren, "Definite Clause Grammars for Language Analysis - A Survey of the Formalism and a Comparison with Augmented Transition Networks.," *Artif Intell*, vol. 13, no. 3, pp. 231–278, 1980.
[25] C. W. Thompson, P. Pazandak, and H. R. Tennant, "Talk to your semantic Web," *Internet Comput. IEEE*, vol. 9, no. 6, pp. 75–78, 2005.
[26] T. Kuhn and R. Schwitter, "Writing support for controlled natural languages," in *Proceedings of ALTA*, 2008, pp. 46–54.
[27] A. Bernstein and E. Kaufmann, "GINO – A Guided Input Natural Language Ontology Editor," in *The Semantic Web - ISWC 2006*, vol. 4273, I. Cruz, S. Decker, D. Allemang, C. Preist, D. Schwabe, P. Mika, M. Uschold, and L. Aroyo, Eds. Springer Berlin / Heidelberg, 2006, pp. 144–157.
[28] R. Valencia-García, F. García-Sánchez, D. Castellanos-Nieves, and J. T. Fernández-Breis, "OWLPath: An OWL ontology-guided query editor," *Syst. Man Cybern. Part Syst. Humans IEEE Trans.*, vol. 41, no. 1, pp. 121–136, 2011.
[29] T. Kuhn, "AceWiki: A Natural and Expressive Semantic Wiki," *arXiv:0807.4618*, Jul. 2008.
[30] R. Shiffman, G. Michel, M. Krauthammer, N. Fuchs, K. Kaljurand, and T. Kuhn, "Writing clinical practice guidelines in controlled natural language," *Control. Nat. Lang.*, pp. 265–280, 2010.
[31] T. Kuhn, "A Principled Approach to Grammars for Controlled Natural Languages and Predictive Editors."
[32] P. E. M. Taschner and J. T. den Dunnen, "Describing structural changes by extending HGVS sequence variation nomenclature," *Hum. Mutat.*, vol. 32, no. 5, pp. 507–511, May 2011.
[33] D. A. Natale, C. N. Arighi, W. C. Barker, J. A. Blake, C. J. Bult, M. Caudy, H. J. Drabkin, P. D'Eustachio, A. V. Evsikov, H. Huang, J. Nchoutmboube, N. V. Roberts, B. Smith, J. Zhang, and C. H. Wu, "The Protein Ontology: a structured representation of protein forms and complexes," *Nucleic Acids Res.*, vol. 39, no. Database issue, pp. D539–D545, Jan. 2011.
[34] J. Earley, "An efficient context-free parsing algorithm," *Commun. ACM*, vol. 13, no. 2, pp. 94–102, 1970.
[35] J. Aycock and R. N. Horspool, "Practical earley parsing," *Comput. J.*, vol. 45, no. 6, pp. 620–630, 2002.
[36] M. Seringhaus and M. Gerstein, "Publishing perishing? Towards tomorrow's information architecture," *BMC Bioinformatics*, vol. 8, no. 1, p. 17, Jan. 2007.
[37] D. Shotton, "Semantic publishing: the coming revolution in scientific journal publishing," *Learn. Publ.*, vol. 22, no. 2, pp. 85–94, 2009.
[38] C. Shaw, "The future of open access research and publishing – live chat," *the Guardian*, 22-Oct-2013. [Online]. Available: http://www.theguardian.com/higher-education-network/2013/oct/22/future-open-access-research-publishing. [Accessed: 25-Oct-2013].
[39] "Brussels Declaration | STM." [Online]. Available: http://www.stm-assoc.org/brussels-declaration/. [Accessed: 21-Oct-2013].