Az emMorph morfológiai elemző annotációs formalizmusa

Novák Attila[1,2], Rebrus Péter[3], Ludányi Zsófia[3]

[1] MTA-PPKE Magyar Nyelvtechnológiai Kutatócsoport,
[2] Pázmány Péter Katolikus Egyetem, Információs Technológiai és Bionikai Kar
1083 Budapest, Práter utca 50/a, e-mail:novak.attila@itk.ppke.hu
[3] MTA Nyelvtudományi Intézet
1068 Budapest, Benczúr utca 33
e-mail:{rebrus.peter, ludanyi.zsofia}@nytud.mta.hu

Kivonat A morfológiai elemző – lévén minden nyelvfeldolgozási lánc kezdeti lépése – a nyelvtechnológiai alkalmazásokban kiemelkedő szerepű. A kimenet értelmezése szempontjából rendkívül fontos a morfológiai elemzés kimenetének egységesítése. Cikkünkben az *emMorph* morfológiai elemzőrendszer és az *emLem* lemmatizáló implementációjának ismertetése után bemutatjuk ezen eszközök kimeneti formalizmusát, különös tekintettel a morfológiai címkékre.

1. Bevezetés

A *Nyílt, integrált magyar nyelvtechnológiai kutatási infrastruktúra fejlesztése* projekt (`e-magyar`) az MTA Nyelvtudományi Intézete vezetésével, az MTA SZTAKI, a SZTE, a PPKE és az AITIA International Zrt. közreműködésével valósult meg[4]. Célja egy olyan nyílt forrású, szabadon hozzáférhető nyelvtechnológiai infrastruktúra kiépítése volt, melynek elemei a magyar nyelv gépi elemzésének alapvető eszközeit tartalmazzák integrált, szabványos keretben [11]. A rendszer részét képezi egy új magyar morfológiai elemző, amelynek implementációja a nyílt forráskódú véges állapotú transzducertechnológiát alkalmazó hfst rendszer felhasználásával valósult meg. Jelen cikk célja a megvalósult *emMorph* morfológiai elemző[5] és az arra épülő *emLem* lemmatizáló[6] implementációjának ismertetése és az elemző, illetve a lemmatizáló kimeneti formalizmusának bemutatása, különös tekintettel a morfoszintaktikai címkékre.

2. A morfológiai elemző implementációja

A morfológiai elemző adatbázisa elsősorban az eredetileg a Humor morfológiai elemző motorhoz [8] készült magyar morfológiai adatbázison alapul [5], amelyet

[4] `http://e-magyar.hu`
[5] `https://github.com/dlt-rilmta/emMorph`
[6] `https://github.com/dlt-rilmta/hunlp-GATE/tree/master/Lang_Hungarian/resources/hfst/hfst-wrapper`

kiegészítettünk olyan szavakkal, amelyek az eredeti Humor leírásban nem, a *morphdb.hu* [10] adatbázisban viszont szerepeltek, miután az utóbbi listából kiszűrtük a hibás, illetve elhanyagolhatóan ritka szavakat. A morfológiai leírást kezelő keretrendszer egy procedurális szabályrendszer felhasználásával magas szintű és redundanciamentes morfémaleírásokból állítja elő az egyes morfémák lehetséges allomorfjait, azok tulajdonságait (jegyeit) és azokat a jegyalapú megszorításokat, amelyeknek az egymással szomszédos morfok között teljesülnie kell. Emellett a helyes szószerkezetek leírását egy kiterjesztett véges állapotú szónyelvtan-automata ábrázolja.

Az eredeti Humor elemzőprogram ezeket az allomorflexikonokat, az allomorfok közötti szomszédossági megszorításokat és a véges állapotú szónyelvtan-automatát közvetlenül használja a szóalakok elemzése közben. Az új hfst-alapú implementációban [3] mindezek az adatszerkezetek egyetlen véges állapotú transzducerben jelennek meg.

A véges állapotú transzduceren alapuló morfológiai rendszerek létrehozásánál általában az a szokásos eljárás, hogy a *lexc* lexikondefiníciós nyelv [1] segítségével létrehoznak egy alap-morfémalexikont, amelyben a morfémák valamiféle mögöttes reprezentációban szerepelnek, és a leírás e mellett tartalmaz egy az *xfst* újraírószabály-formalizmusa [1] segítségével megadott vagy a Kimmo-féle kétszintű megszorításokon alapuló szabálykomponenst, amelyet a mögöttes alakokat tartalmazó lexikonnal komponálva előáll a morfémák mögöttes és felszíni alakjai közötti, az adott kontextusban megfelelő leképezés. A hagyományos megközelítésben tehát a *lexc* lexikon és az *xfst* szabályrendszer kompozíciója hozza létre a morfológiai elemző transzducert.

Az általunk készített véges állapotú magyar morfológiai leírás ezzel szemben nem tartalmaz külön sem *xfst* újraíró szabályokat, sem Kimmo-féle kétszintű megszorításokat tartalmazó szabálykomponenst, hanem a morfémák allomorfjait és a hozzájuk tartozó szomszédossági megszorításokat folytatási osztályok formájában tartalmazó adatbázist közvetlenül egy a lexc formalizmus segítségével leírt lexikonná konvertáljuk, amely a mögöttes alakok (lemmák) és a felszíni alakok közötti helyes leképezést már tartalmazza, így további szabályokra nincs szükség. Az eredeti Humor formalizmus szónyelvtan-automatáját a véges állapotú leírásban a *flag diacritcs* konstrukció [1] alkalmazásával ábrázoltuk. Ez a leírás tartalmazza a morfémák közötti nem lokális megszorításokat is (pl. hogy a melléknevek felsőfokát jelölő prefixumot a szón belül valahol vagy egy középfokjelnek vagy valamilyen más felsőfokjelet engedélyező morfémának követnie kell). A Humor formalizmusban leírt adatbázis véges állapotú leírássá konvertálására alkalmazott algoritmusok részletes leírását l. [7] 6. fejezetében, illetve itt: [6].

3. Lemmatizálás

A morfológia az összetett és képzett szavak esetében az összetételi tagokat, illetve a képzőket is azonosítja. Amennyiben az összetett vagy képzett szó a lexikonba egyben is fel van véve, több elemzés is kijöhet, amelyek különböző részletességű elemzését adják az adott szónak. A *fejetlenség* főnév elemzésekor például

az elemző ezt egyben is megtalálja, ugyanakkor visszavezeti a *fejetlen* melléknévre, a *fej* főnévre és a *fej* igére is. Bár ezek az elemzések részben különböző szemantikai tartalmakat tükrözhetnek *(káosz, átgondolatlanság, fejnélküliség, a fejés elmaradása)*, ezek közül a jelentések közül némelyik szinte egyáltalán nem jelenik meg ténylegesen előforduló szövegekben, ráadásul a morfológiai elemzésre épülő és a nyelvi elemzés egyéb szintjeit végző eszközöknek általában nincs is szükségük ilyen részletességű elemzésre. Amire viszont szükségük van, az az adott szó lemmája (szótári töve), valamint (elsősorban a ragok, illetve bizonyos nagyon produktív képzők, pl. az igenévképzők által megtestesített) morfoszintaktikai jegyei. A lemma magában foglalja a szóban levő töveket és képzőket, mindazt, amit nem morfoszintaktikai jegyek formájában szeretnénk a további nyelvi elemzést végző eszközök számára továbbadni.

A hfst rendszer [3] morfológiai elemzést végző eszközei (a *hfst-lookup*, illetve a *hfst-optimized-lookup*) alapesetben nem olyan elemzést állítanak elő, amely közvetlenül alkalmas lenne a lemma előállítására, ugyanis kizárólag az adott elemzést alkotó morfémák mögöttes alakját és a morfoszintaktikai címkéket adják vissza, az ezeknek megfelelő felszíni alakot nem, így a képzőt tartalmazó tövek teljes szótári alakja nem mindig számítható ki. A hfst-lookup fejlesztője kérésünkre kiegészítette az eszközt egy olyan funkcióval, amely az elemzett szót alkotó morfémák felszíni és mögöttes alakját egyszerre adja vissza (illetve ténylegesen működőképessé tette ezt a korábban nem működő funkciót). Ugyan ez a kimenet emberi fogyasztásra nem igazán alkalmas[7], de lehetővé tette, hogy ennek felhasználásával létrehozzuk a morfológiai elemző kimenetére épülő Java nyelven implementált, ezért platformfüggetlen lemmatizáló eszközt (emLem), amely a tőalkotó elemek (tövek, képzők) összevonásával kiszámolja az adott elemzéshez tartozó lemmát (ehhez az utolsó tőalkotó elem kivételével a felszíni alakra van szükség), annak eredő szófaját, és ehhez hozzáadja a nem tőalkotó morfémák által hordozott morfoszintaktikai jegyek címkéit.

Az azonos lemmát, szófajt és egyéb morfoszintaktikaicímke-sorozatot eredményező különböző részletességű elemzések (pl. a *fejetlenség* főnév elemzései) a lemmatizáló kimenetén egyetlen elemzésként jelenhetnek meg, hiszen ezek a magasabb nyelvi szinteket feldolgozó elemzők számára (szófaji egyértelműsítő, szintaktikai elemző stb.) ekvivalensek. Ugyanakkor a lemmatizáló képes a részletes elemzések visszaadására is úgy, hogy az az elemzést alkotó morfok felszíni alakját is tartalmazza olvasható és jól kereshető formában[8]. A lemmatizáló viszonylag bonyolult algoritmust valósít meg, amely nem triviális morfológiai konstrukciók (pl. ikerszavak) és különleges beállítások (pl. ha az igenévképzőket nem tekintjük tőalkotónak) esetén is helyes lemmát ad.[9] Az alkalmazott lemmatizáló algoritmussal kapcsolatos további részletek [7] 4.3 fejeztében olvashatók.

[7] t:t e:e h:h e:é n:n :[/N] e:e c:c s:s k:k é:e :[_Dim:cskA/N] j:j é:e :[Poss.3Sg] t:t :[Acc]

[8] tehén[/N]=tehen+ecske[_Dim:cskA/N]=ecské+je[Poss.3Sg]=jé+t[Acc]=t

[9] Léteznek igenévképzőt tartalmazó alaktani konstrukciók, amelyekre hibás tövet kapunk, ha az igenévképző(vel azonos alakú képző)t nem tekintjük a tő részének: pl. *húsdarál(ó)*.

4. Kiértékelés

A morfológia elemző fedésével kapcsolatban Kornai András és kollégái készítettek független kiértékelést az elemző 2016 augusztusi verziójával. Bár ezen cikk célja elsősorban az elemző által generált annotáció ismertetése, itt röviden bemutatjuk ennek a kiértékelésnek az eredményét. A kiértékeléshez két nagyméretű magyar nyelvű korpuszt, az MNSZ2-t (Magyar Nemzeti Szövegtár V2.0[10]) és a WebKorpusz 2.0-t (WK2[11]) használták. A korpuszokból azokat a szavakat választották ki, amelyek legalább három MNSZ2-részkorpuszban szerepeltek, és a WebKorpuszban is legalább háromszor előfordultak. A kiválasztott 1363692 szóalak az MNSZ2 95,65%-át és a WK2 94,66%-át fedi le. A kiválasztás során a két korpusz tokenjeinek 5,12%-a esett ki. A tesztanyagból az elemző által felismert szóalakok korpusztokenekre visszavetített aránya 92,63%, a nem elemzetteké 2,25%. Kornaiék ezt az fedést „kiemelkedően jó"-nak minősítették.[12]

5. A morfológiai elemző által generált annotáció

5.1. Motiváció

A morfológiai elemzés kimenetének egységesítése rendkívül fontos a kimenet értelmezése szempontjából, legyen az elemzés automatikus vagy nyelvészeti alapú, és a kimenet feldolgozása automatizált vagy emberi erővel történő. Az ilyen kimeneti annotációs rendszerekben a morfológiai elemzők tipikusan kétfajta információt jeleníthetnek meg: morfológiait és morfoszintaktikait. A morfoszintaktikai információ megadja, hogy az adott szóalak milyen szintaktikai környezetben és funkcióban fordulhat elő, előre megadott morfoszintaktikai tulajdonságokhoz rendelt értékek használatával. A morfológiai információ megmutatja, hogy mely morfémaváltozatokból (morfokból) áll össze a szó, és ezekhez a morfokhoz mely morfoszintaktikai jegyek rendelhetők. E két információtípust tipikusan egyszerre szokták az annotációs rendszerek megjeleníteni, de különböző rendszerek különböző arányban. A két szélsőség egyikét a nyelvészeti morfo(fono)lógiai elemzés képviseli, ahol az explicite nem megjelenő morfoszintaktikai információk nem lényegesek (hiányozhatnak), viszont a morfokra való szegmentálás általában központi jelentőségű. Ezekkel szemben állnak azok a formális annotációs rendszerek, amelyekben csak morfoszintaktikai jegyek vannak, és az annotáció nem tartalmaz a morfszegmentálásra vonatkozó információt (ez utóbbira példa az ún. Universal Dependencies [4], az MSD-kódolás vagy a hunmorph rendszerben működő ún. KR-kódolás [9]). Több rendszerben a kétféle információt az annotáció egyszerre tartalmazza (pl. ilyen a Humor [5,8] vagy a Xerox magyar morfológiai elemzője), de ezek megjelenítése sokszor némileg ad hoc módon történik.

[10] `http://mnsz.nytud.hu`

[11] `http://mokk.bme.hu/en/resources/webcorpus`

[12] A jelenlegi verzió az itt ismertetettnél jobb fedést mutat, mert egy jelentős hibaosztály (Kornaiék a kötőjeles szavak egy nagy osztályára nem kaptak elemzést) megszűnt.

Ennek praktikus okai vannak: az írott szóalakok szegmentálása bizonyos esetekben szükségszerűen önkényes: pl. a *hússzal* szóalak morfokra bontásakor a *húsz* tő és a *szal* eszközhatározó-rag közötti határ meghúzása a helyesírás sajátosságai miatt sehogy sem lesz igazán jó. A Humor rendszerben használt *hússz+al* tagolás mellett praktikus (a lexikonmérettel és a jegyrendszer komplexitásával kapcsolatos) szempontok szólnak: a kétjegyű betűre végződő szavakhoz mindenképp elő kell állítani egy-egy plusz allomorfot, ugyanakkor az ezekhez kapcsolódó eszközhatározórag-allomorfból ebben az esetben elég, ha egy van a lexikonban.

Az emMorph elemző kimeneti formalizmusa kialakításakor abból indultunk ki, hogy az egyszerre kell szolgálja a számítógépes nyelvfeldolgozást és a nyelvészeti elemző munkát. Ennek megfelelően igyekeztünk arra törekedni, hogy az annotáció mind a releváns morfológiai szegmentálást, mind a szükséges morfoszintaktikai jegyeket tükrözze, és belőle ezek külön-külön is kinyerhetők legyenek. Ugyanakkor mivel az elemző alapvetően a Humor rendszer számára implementált szabályrendszeren alapszik, a szegmentálás tekintetében megmaradt néhány a Humor leírásból örökölt kompromisszum. Egy másik megszorítás az volt, hogy szerettük volna a korábban használt annotációs sémák és az új rendszer közötti konverziót lehetőleg minél teljesebb mértékben lehetővé tenni. Ezért azokat a komplex toldalékokat, amelyekhez tartozó címke a korábbi rendszerek valamelyikében nem tagolódott világosan elkülöníthető elemekre (pl. az *-i* „birtoktöbbesítő jel”-et tartalmazó birtokos végződések), nem szegmentáltuk szét különálló elemekre az új annotációs sémában sem, hanem azokat a fúziós morfémáknak megfelelő módon ábrázoltuk (l. a 5.5 részt).

Az annotációs rendszer egyben szabványosítási javaslat a magyar nyelvű automatikus morfológiai elemzők kimeneti formátumára, és a magyar alaktan nyelvészeti glosszáinak formátumára. A korábbi magyar morfológiai elemzők egyedi és mind egymástól, mind az esetleges nemzetközi szabványoktól eltérő címkéket használtak. A projekt keretében megvalósult elemző címkekészletét ezzel szemben igyekeztünk nemzetközi szabványhoz igazítani: amennyire lehetséges volt, a nyelvészeti annotációra széles körben egyfajta szabványként használt Leipzig Glossing Rules (LGR) [2] javaslatait követtük. A címkék meghatározásakor emellett az ott leírtakat kiegészítő lényegesen kibővített listára (List of glossing abbreviations = LOGA)[13] támaszkodtunk, amelyet az ezekben a dokumentumokban leírtak szellemében kiegészítettünk a hiányzó (elsősorban képzőkkel kapcsolatos) címkékkel.

5.2. Az annotáció felépítése

Míg a Leipzig Glossing Rulesban javasolt annotációs séma szerint az annotáció külön sorokban tartalmazza a morfokra szegmentált elemzett alakot és a morfokhoz tartozó morfoszintaktikai jegyeket (amely csak a tövek esetén tartalmaz alaki információt: a lemmát), a véges állapotú morfológiai elemző kimenetén ezek az elemek szekvenciálisan jelennek meg: az egyes morfok mögöttes és felszíni alakja, illetve a hozzá tartozó morfoszintaktikai címke együtt jelenik meg

[13] `https://en.wikipedia.org/wiki/List_of_glossing_abbreviations`

a kimeneten. A szegmentálás jelölésére a Leipzig Glossing Rulesban a kötőjel használatát javasolják. Ennek használata – tekintettel arra, hogy a sztenderd helyesírásban ez igen gyakran eleve a szóalak része – nem lett volna praktikus.[14] Ehelyett az elemző kimenetén szögletes zárójelbe tett morfoszintaktikai címkék jelölik implicit módon a szegmentálási határokat. A Leipzig Glossing Rulesban javasolt gyakorlattól még abban a fontos kérdésben tértünk el, hogy az LGR-t követő kiadványokban – némileg meglepő módon – gyakran egyáltalán nem használnak szófajcímkéket: a tövek szófaját semmilyen módon nem jelölik. Hogy ennek a gyakorlatnak mi az oka, azt nem érdemes találgatni, mi mindenesetre nem követtük.

Az emMorphban használt annotációban a címkék egyes alaki tulajdonságai egyértelmű összefüggésben vannak az adott morféma típusával. A tőmorfémák címkéje /-lel kezdődik (`fej[/N]` főnév), a képzőké _-sal, és a képző címkéjét követő / után a képző eredő szófaja áll (`etlen[_Abe/Adj]` névszói fosztóképző „abesszívusz"), az inflexiók címkéje pedig nem tartalmaz speciális karaktert (`t[Acc]` tárgyesetrag). A szófajcímkék elé helyezett / a morphdb.hu-ban használt KR-kódrendszerből származik, a képzők _-sal való megjelölése pedig a Humor-kódkészlet sajátossága volt.

További eltérés az LGR-hez képest, hogy az emMorph kimenete a toldalékmorfok lexikai alakjait is tartalmazza. Ez nem valamiféle absztrakt fonológiai alak, hanem azzal az allomorffal azonos, amelyet az adott toldalékmorféma akkor vesz fel, amikor a szó végén áll. Ennek elsősorban a képzők esetében van jelentősége és a lemmatizáláshoz szükséges. Az emMorphra épülő emLem lemmatizáló az adott elemzéshez tartozó lemma kiszámolásakor azt a tőalkotó morfokból állítja össze. Az utolsó tőalkotó elem a lexikai, a többi a felszíni alakjában szerepel a lemmában (1. táblázat).

surface form	**butá**	**cská**	**bb**	**já**	**tól**	**nadrág**	**ocská**	**tól**
lexical form (lemma)	*buta*	*cska*	*bb*	*ja*	*tól*	*nadrág*	*ocska*	*tól*
`abstract lex. form`	`buta`	`LVcskA`	`LAObb`	`LjA`	`LtŐl`	`nadrág`	`LVcskA`	`LtŐl`
tag	/Adj	_Dim/Adj	_Comp/Adj	Poss.3Sg	Abl	/N	_Dim/N	Abl
lemma 1	**butá**	**cská**	*bb*					
lemma 2	**butá**	*cska*				**nadrág**	*ocska*	
lemma 3	*buta*					*nadrág*		

1. táblázat. Képzett és ragozott szavak lemmatizálása

5.3. Szegmentálás és alternációk

A kötőhangzót általában az utána álló toldalékhoz kapcsoljuk:
`nap[/N]ok[Pl]at[Acc]`. Az epentetikus mássalhangzókat ezzel szemben (pl. *bőv+en, ven+ne*) általában a tőhöz számítjuk.

A morfsorozat az aktuális alakban szereplő tőallomorf részsztringjeit tartalmazza. A lemma neve viszont általában a paradigma alapalakja, mely az izoláltan

[14] Az LGR formalizmusát eleinte elsősorban a helyesírási normával nem rendelkező „bennszülött" nyelvekkel kapcsolatos terepmunkagyűjtések eredményének lejegyzésére használták.

megjelenő alakkal azonos (ha ez létezik). Váltakozó tő esetén a tőallomorf nem mindig egyezik meg a lemma nevével: pl. *fá-* ∼ *fa, bokr-* ∼ *bokor, tav-* ∼ *tó, nyar-* ∼ *nyár, ve-* ∼ *vesz, vol-* ∼ *van.* Az ikes igék esetén az alapalak (és így a lemma neve) az ikes alak, függetlenül attól, milyen tőváltozat jelenik meg a szóban forgó alakban: *laktok:* `lakik[/V]tok[Prs.NDef.2Pl]`.

Ha az alapalak is több alakban jelenhet meg (mint az sz∼d váltakozást mutató igéknél), akkor a gyakoribb alakot vesszük lemmának – az, hogy ez melyik, az egységes lemmaazonosíthatóság miatt előre rögzíteni kell minden egyes ilyen lemmánál: *növekednek:* `növekszik[/V]nek[Prs.NDef.3Pl]`.

5.4. Hiányos és helyettesítő paradigmák

Ha egy morfológiailag hiányos paradigmájú elem alapalakja hiányzik, akkor a lemma neve a morfológiailag legjelöletlenebb alak. Plurale tantum (pl. *üzelmek, bélbolyhok, légutak*) esetén ez a nem birtokos nominativusi többes számú alak. Possessivum tantum (pl. *eleje, alja, hóna, öccse*) esetén a lemma neve az egyes számú E.3 birtokos nominativusi alak. Egyes esetekben a kétféle defektivitás egyszerre érvényesül (pl. *eleik, feleink*), ekkor a lemma a többes számú E.3 birtokos alak: *eleiknek* `elei[/N]ik[Pl.Poss.3Pl]nek[Dat]`.

Az igei defektivitás azon eseteinél, ahol nem áll rendelkezésre a jelen idő kijelentő mód indefinit E.3 alak (pl. *sínyli, kétli*), akkor a definit E.3 kijelentő mód jelen idejű alak lesz a lemma neve: *sínylitek:* `sínyli[/V]itek[Prs.Def.2Pl]`.

5.5. Fúziós morfémák

Ha egy morfhoz több jegyet kell rendelni (fúziós morféma), akkor a szóban forgó jegyek egy []-en belül jelennek meg, és ponttal választjuk el őket. Például egyes birtokosjelölős alakokban a toldalék egyszerre utal a birtoklásra (Poss) és a birtok számára/személyére (pl. 1Sg): *nadrágomat* `nadrág[/N]om[Poss.1Sg]at[Acc]`. Az elemzések Humor-elemzésekre és címkékre való leképezhetősége érdekében így jártunk el néhány olyan toldalék esetében is, amelyek esetében a szegmentálás egyébként nem lenne lehetetlen (bár bizonyos dilemmák felmerülnének): (`jaim[Pl.Poss.1Sg]`, `nátok[Cond.Def.2Pl]`, `nátok[Cond.NDef.2Pl]`, `tatok[Pst.NDef.2Pl]`, `tátok[Pst.Def.2Pl]`). A zérusmorfok jelölése nem különleges, egyszerűen üres a felszíni alakjuk (és általában a lexikai is).

Az igeidőt és a módot egymással komplementáris viszonyban levőnek tekintettük, így külön kijelentő mód jegyet nem vettünk fel, hanem valamely időjegy (`Prs`, `Pst`) meglétéből következik a kijelentő mód.

5.6. Unáris jegyek

Vannak olyan morfoszintaktikai dimenziók, amelyeknek csak egy értéke jelenik meg – ezek az ún. unáris jegyek. Azt az információt, hogy ilyen értékkel az alak nem rendelkezik, az annotáció nem jelöli (pontosabban az adott jegy hiányával jelöli). A modális igei alakokban (pl. *adhatsz* `ad[/V]hat[_Mod/V]sz[Prs.NDef.2Sg]`)

unáris jegy áll, ahogyan az összes képzett alakban is. Ezzel szemben az inflexiós jegyek nagy része nem unáris, például az igeragozás definitsége tekintetében az `Def` jegy szemben ál az `NDef` jeggyel, az alanyesetet is megjelöljük a `Nom` jeggyel. A jelen implementációban sajátos kivételként a névszóragozás paradigmájának leírásában az egyes szám jelöletlenül maradt. Ennek oka az volt, hogy a morfokra szegmentálás szempontjából ennek a jegynek mind a tőhöz, mind a toldalékokhoz rendelése ellentmondáshoz vezetett volna.

5.7. Az alkalmazott címkék

Mint korábban említettük, az elemzőben igyekeztünk következetesen az LGR és a LOGA dokumentumokban felsorolt címkéket használni, illetve az ott megadott alternatív jelölések közül választani. Azon címkék ügyében szavazással döntöttünk, amelyekkel kapcsolatban az előkészítő fázisban nem jutottunk konszenzusra. Így született többek között az igekötők `/Prev` (preverb), a igenevek `Ptcp` a névelők `Det`, a melléknevek, illetve a számnevek `Adj`, illetve `Num` címkéje. Az alkategóriára utaló jegyek a címkén belül |-lal elválasztva jelennek meg, pl. `/Adj|Pro|Int`: melléknévi kérdő névmás (pl. *milyen*). Zárójelben szerepel a vonzatos névutók vonzatát jelölő esetrag kódja: `/Post|(Abl)`. A (szinte) azonos funkciót nem fonológiailag vagy lexikailag kondicionált módon, hanem lényegében szabadon választhatóan különböző formában kifejező toldalékok esetében a funkció mellett a formára is utal a használt címke (a formára utaló címkerész előtt mindig kettőspont áll): `EssFor:ként, EssFor:képp, EssFor:képpen`, illetve `_Adjz_Type:fajta/Adj, _Adjz_Type:forma/Adj, _Adjz_Type:féle/Adj, _Adjz_Type:szerű/Adj` (Adjz: adjectivizer 'melléknévképző'). A képzők esetében a formára sokszor egyébként is utalunk. Sőt, időnként – amikor a funkció viszonylag heterogén, illetve nem volt egyszerű egy rövid címkében egyértelműen megnevezni – csak a formára (és az eredő szófajra) utal a címke: `_Adjz:i/Adj, _Adjz:s/Adj, _Adjz:Ő/Adj, _Adjz:Ű/Adj`.

6. Konklúzió

A cikkben bemutattuk az `e-magyar` projekt keretében megvalósult új, nyílt forráskódú morfológiai elemzőeszközt. Kitértünk a lemmatizáló és a morfológiai elemző implementációjának főbb kérdéseire, majd részletesen ismertettük a nyílt forráskódú emMorph morfológiai elemző és emLem lemmatizáló kimeneti formalizmusát, az általuk generált annotációt. Az emMorph által generált annotáció formalizmusa sztenderdizált, automatikus és kézi feldolgozásra is alkalmas. A jegyek elnevezése (rövidítése) és sorrendje a nemzetközi nyelvészeti konvenciókhoz kötődik, így jól olvasható, és a nyelv ismerete nélkül is értelmezhető.

7. Köszönetnyilvánítás

Az `e-magyar` eszközlánc az MTA 2015. évi Infrastruktúra-fejlesztési Pályázat 2. kategóriájában elnyert támogatás segítségével valósult meg. Köszönetet mon-

dunk Kornai Andrásnak és kollégáinak az elemző fedésének a 4. részben ismertetett kiértékelésért.

Hivatkozások

1. Beesley, K., Karttunen, L.: Finite State Morphology. No. 1 in CSLI studies in computational linguistics: Center for the Study of Language and Information, CSLI Publications (2003)
2. Comrie, B., Haspelmath, M., Bickel, B.: The Leipzig glossing rules: Conventions for interlinear morpheme-by-morpheme glosses (2008), `https://www.eva.mpg.de/lingua/pdf/Glossing-Rules.pdf`
3. Lindén, K., Silfverberg, M., Pirinen, T.: HFST tools for morphology – an efficient open-source package for construction of morphological analyzers. In: Mahlow, C., Piotrowski, M. (eds.) State of the Art in Computational Morphology, Communications in Computer and Information Science, vol. 41, pp. 28–47. Springer Berlin Heidelberg (2009)
4. McDonald, R., Nivre, J., Quirmbach-Brundage, Y., Goldberg, Y., Das, D., Ganchev, K., Hall, K., Petrov, S., Zhang, H., Täckström, O., Bedini, C., Bertomeu Castelló, N., Lee, J.: Universal dependency annotation for multilingual parsing. In: Proceedings of ACL 2013. pp. 92–97. Association for Computational Linguistics, Sofia, Bulgaria (August 2013)
5. Novák, A.: Milyen a jó Humor? In: I. Magyar Számítógépes Nyelvészeti Konferencia. pp. 138–144. SZTE, Szeged (2003)
6. Novák, A.: A Humor új Fo(r)mája. In: X. Magyar Számítógépes Nyelvészeti Konferencia. pp. 303–308. SZTE, Szeged (2014)
7. Novák, A.: A model of computational morphology and its application to Uralic languages. Ph.D. thesis, Roska Tamás Doctoral School of Sciences and Technology Pázmány Péter Catholic University, Faculty of Information Technology and Bionics, Budapest (2015)
8. Prószéky, G., Kis, B.: A unification-based approach to morpho-syntactic parsing of agglutinative and other (highly) inflectional languages. In: Proceedings of ACL '99. pp. 261–268. Association for Computational Linguistics, Stroudsburg, PA, USA (1999)
9. Rebrus, P., Kornai, A., Varga, D.: Egy általános célú morfológiai annotáció. Általános Nyelvészeti Tanulmányok XXIV., 47–80 (2012)
10. Trón, V., Halácsy, P., Rebrus, P., Rung, A., Vajda, P., Simon, E.: Morphdb.hu: Hungarian lexical database and morphological grammar. In: Proceedings of LREC 2006. pp. 1670–1673 (2006)
11. Váradi, T., Simon, E., Novák, A., Indig, B., Farkas, R., Vincze, V., Sass, B., Gerőcs, M., Iván, M.: e-magyar.hu: digitális nyelvfeldolgozó rendszer. In: XIII. Magyar Számítógépes Nyelvészeti Konferencia (MSZNY2017) (2017)

Az e-magyar rendszer GATE környezetbe integrált magyar szövegfeldolgozó eszközlánca

Sass Bálint, Miháltz Márton, Kundráth Péter

MTA Nyelvtudományi Intézet, e-mail: sass.balint@nytud.mta.hu, mmihaltz@gmail.com, peter.kundrath@gmail.com

Kivonat A jelen tanulmányban bemutatott új magyar nyelvfeldolgozó eszközlánc az emberi intelligenciát igénylő szövegértési feladatnak egy jelentős szeletét automatikusan valósítja meg: a szövegben rejlő információkat automatikus módon fedi fel, teszi explicitté. Egy tetszőleges magyar nyelvű szövegrészt feldolgozva megtudjuk az egyes szavak szófaját, szótövét, alaktani elemzését, a mondatok kétféle mondattani elemzését, megkapjuk a főnévi csoportokat és a tulajdonneveket. A rendszer egybegyűjti, egy egységes láncba integrálja és közzéteszi az elemzési lépéseket megvalósító számítógépes magyar szövegfeldolgozó eszközöket. Ezáltal széles körben elérhetővé, közvetlenül felhasználhatóvá válnak ezek az eszközök a különféle igényű felhasználói körök – kiemelten a humán tudományok és a digitális bölcsészet – számára. A tanulmányban áttekintjük az eszközlánc felépítését és használatát.

Kulcsszavak: szövegfeldolgozás, szövegfeldolgozó eszközlánc, szövegelemzés, GATE, integráció

1. Bevezető példa

Mit csinál egy szövegfeldolgozó eszközlánc? Mit csinál konkrétan az e-magyar digitális nyelvfeldolgozó rendszer szövegfeldolgozó eszközlánca? Magyar nyelvű írott szöveget elemez, és lát el különféle kiegészítő információkkal az elemzés eredményeképpen. Egymásra épülő eszközökből áll: az egyes eszközök működésük során felhasználják a korábbiak eredményét.

Tekintsük a következő példaszöveget:

> *Bár külföldre menekülhetett volna, nem tette meg. Támogatta a haladó eszméket, barátságban állt pl. Jókai Mórral is.*

A rendszer a szöveg automatikus feldolgozása során először megállapítja a szavak – ún. tokenek – és mondatok határát. A példában a *Támogatta* új mondatot kezd, a *Jókai* viszont nem, bár itt is pont után nagybetűs szó következik, ami tipikusan mondathatárra utal. Külön tokenként kezeli az írásjeleket, kivéve a rövidítéseknél, ahol a záró pont a rövidítés részét képezi, így a *pl.* egy egység lesz, az *is* és az azt követő pont viszont kettő.

A morfológiai elemzés megadja az egyes szavakról az alaktani információkat: a *menekülhetett* szóalak például múlt idejű ige, mely a *menekül* szótőből,

a *het* képzőből és az *ett* igeragból épül fel. A magyar szóalakok jelentős részének, akár 30%-ának több alaktani elemzése van. A rendszer a szövegkönyezet alapján automatikusan dönt ilyen esetekben, kiválasztja a helyes elemzést, ez az ún. egyértelműsítési lépés. A többértelműség sokszor nem olyan nyilvánvaló, mint a *várnak* vagy a *terem* esetében, hanem rejtetten jelenik meg: fontos, hogy példánkban a *haladó* melléknévként elemződjön, ne pedig összetett főnévként, ami valamiféle vízi élőlényekre vonatkozó járulékot jelentene.

Ezt követően megtörténik – kétféleképpen – az egyes mondatok mondattani elemzése. A függőségi elemzés eredményeként az egyes szavak egymáshoz való kapcsolatai jelennek meg, mint például, hogy a *barátságban* az *állt* igéhez kapcsolódó határozó. Az összetevős elemzés ugyanakkor a mondat egységeit adja ki: a második mondat két nagyobb egységből áll, melyek felsorolás viszonyban vannak egymással. A függőségi elemzés alapján az ige–igekötő kapcsolatok is rendelkezésre állnak, erre építve egy külön segédmodul megjelöli az elváló igekötőket, és a hozzájuk tartozó igéket, példánkban a *tette* és a *meg* kapcsolatát. A főnévi csoportokat – pl. *a haladó eszméket* – is azonosítja egy erre a célra készített modul.

Végül a lánc utolsó tagja megjelöli a tulajdonnevek fontos alosztályait, a személy-, hely- és intézményneveket, példánkban a *Jókai Mórral* nevet.

Látjuk tehát, hogy a feldolgozás során a puszta szöveg számos explicit hozzáadott információval gazdagodik.

2. A konkrét klasszikus nyelvfeldolgozó eszközök

Tekintsük át röviden a konkrét eszközöket. Az eszközökről részletesebb információt az `e-magyar.hu` honlapon, illetve az [1] tanulmányban találunk. Az eszközök elnevezése az `e-magyar`-ra utaló *em* előtagot tartalmaz. Az `e-magyar` integrált magyar szövegfeldolgozó eszközlánc jelenleg a következő eszközökből áll.

A mondatokra bontást és a tokenizálást az emToken [2] eszköz végzi. A bemenetként megadott UTF-8 kódolású magyar nyelvű szövegben megállapítja a mondat- és szóhatárokat. Eltérő módon megjelöli a szavakat és az írásjeleket. Megőrzi a szóközöket és egyéb white space karaktereket is, ezáltal lehetővé teszi a tokenizált szövegből az eredeti szöveg visszaállítását. Széles körűen fel van készítve az egyes Unicode karakterek megfelelő értelmezésére, kezelésére.

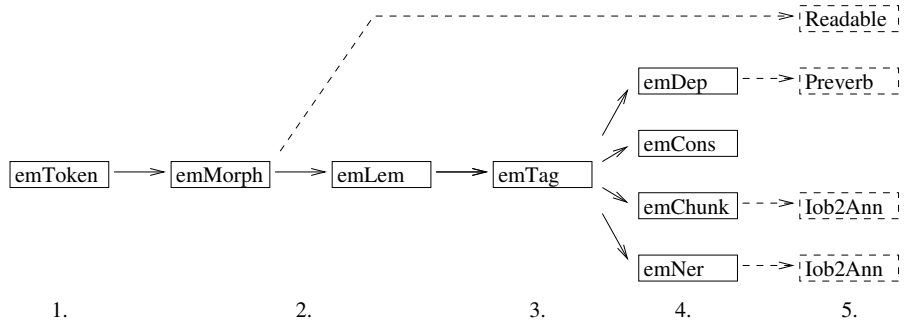
A morfológiai elemzést az emMorph [3] eszköz végzi. A szöveg minden – a tokenizáló által pontosan behatárolt – szóalakjához (tokenjéhez) hozzárendeli az összes lehetséges morfológiai, morfoszintaktikai elemzését, a szóalak aktuális környezetétől függetlenül. Megállapítja a szófaji főkategóriát, megadja a szóalak morfémákra bontásának lehetőségeit (így a szóösszetételi határokat is), és elemzést rendel az egyes morfémákhoz. Az emMorph a korábbi magyar morfológiai elemzők tudását összegzi, a leggyorsabb működést biztosító véges állapotú technológiára alapul, futtatáshoz a HFST [4] véges állapotú eszközkészletet használja.

Az eszközláncba illesztett fenti eszközök újonnan készültek, az alábbiak már korábban is meglévő eszközök legújabb verziói.

Az adott részletes morfológiai elemzéshez tartozó kívánt szótövet az elemzés alapján – a képzőket is tekintetbe véve – a morfológai elemzővel egybeépített emLem [5] szótövesítő eszköz határozza meg. A szóalakokhoz szótövet, a szótőnek megfelelő eredő szófajcímkét és inflexiós jegyeket tartalmazó egyszerűsített elemzést rendel.

A többféle lehetséges morfológiai elemzés közül a megfelelőt az emTag egyértelműsítő eszköz választja ki. Ez a *PurePOS* [6] eszköznek a *Magyarlánc* 3.0 verziójába [7] integrált változata. Ez az eszköz gépi tanulási módszerrel a szöveg minden tokenjéhez meghatározza az aktuális szövegkörnyezetben érvényes szótövét, szófaját és inflexiós jegyeit.

Az egyértelműsítőig az eszközök közvetlenül egymásra épülnek, a további eszközök viszont egymástól függetlenül alkalmazhatók. E további eszközök tokenizált és morfológiailag egyértelműsített bemenetet várnak, azaz használatuk előfeltétele az egyértelműsítőig tartó eszközlánc előzetes lefuttatása (ld. az 1. ábra folytonos vonallal írt részét).



1. ábra. Az `e-magyar` szövegfeldolgozó lánc elemeinek egymásra épülése. A fő nyelvfeldolgozó eszközök folytonos vonallal, a kiegészítő eszközök szaggatott vonallal szerepelnek.

Két különböző felfogású szintaktikai elemző kapott helyet az eszközláncban. Az emDep [7] a mondatok függőségi elemzését valósítja meg. Minden szóról megállapítja, hogy mely másik szóval áll függőségi (dependencia) viszonyban, azaz mely másik szó alárendeltje. Minden tokenhez hozzárendeli tehát a szülőcsomópontot, valamint a függőségi viszonyt leíró megfelelő szintaktikai címkét. Ezek a függőségi viszonyok az elemzett mondat szavait egy elemzési fába rendezik, az elemzési fa csomópontjai a szavak, élei pedig a függőségi viszonyok.

Az emCons [7] eszköz a mondatok összetevős szerkezeti elemzését végzi el. Az összetevős szerkezeti elemzés azt tárja fel, hogy a szavak egymással kombinálódva milyen csoportokat/kifejezéseket alkotnak, és hogy ezek a csoportok/kifejezések hogyan állnak össze mondattá. Az eredményként kapott elemzési fa az elemzési címkékkel ellátott szavakat, a belőlük képzett csoportokat és a csoportok

hierarchiáját ábrázolja. Az elemzés minden tokenhez hozzárendeli a megfelelő elemzésifa-részlet zárójelekkel kódolt formáját.

Az emChunk eszköz azonosítja a szövegben a főnévi csoportokat (NP-ket), egész pontosan a maximális főnévi NP-ket, vagyis azokat, melyek nem részei magasabb szintű NP-nek. Itt az eddigiekkel ellentétben olyan annotációt adunk hozzá a szöveghez, mely több tokenre is kiterjedhet. Ezt a feladatot az eszköz, az alapját képező HunTag3 [8] szekvenciális tagger révén, kizárólag egyes tokenekre vonatkozó annotációk használatával oldja meg: minden tokenhez hozzárendel egy kódot, mely azt mondja meg, hogy az adott token az NP eleje (`B` kód), vége (`E` kód), közbülső eleme (`I` kód), vagy NP-n kívül esik (`O` kód), illetve külön jelet használ az egytokenes NP jelölésére (`1` kód). A többtokenes egységek ilyenfajta tokenenkénti annotációját nevezzük általánosságban IOB-típusú kódolásnak [9].

A szintén a HunTag3 rendszerre alapuló emNer tulajdonnév-felismerő eszköz a fentiekhez hasonló módon működik. Utolsó lépésként ez azonosítja és IOB kódolással megjelöli a szövegben található tulajdonneveket, ezenkívül besorolja őket az előre meghatározott névkategóriák valamelyikébe (személynév, intézménynév, földrajzi név, egyéb).

3. Kiegészítő eszközök

A fenti klasszikus szövegfeldolgozó eszközöket kiegészíti néhány olyan apróbb eszköz (ld. az 1. ábra szaggatott vonallal írt részét), ami az egész lánc hasznosságát, kényelmét növeli, könnyebben értelmezhetővé, felhasználhatóvá teszi az elemzések eredményét, az annotációkban lévő információt.

Az emMorph által szolgáltatott részletes morfológiai elemzés emberi fogyasztásra kevéssé alkalmas. A leírás olvashatóbbá tételét szolgálja a *ReadableMorphoAnalysis* nevű kiegészítő eszköz (2. ábra). Az `e-magyar.hu` honlap szövegelemző felületén is ezzel az eszközzel tesszük olvashatóbbá a részletes morfológiai elemzést.

```
amely[/N|Pro|Rel]=amely+ek[Pl]=ek+ről[Del]=ről
→
amely[/N|Pro|Rel] + ek[Pl] + ről[Del]
```

2. ábra. Az *amelyekről* szó emMorph szerinti morfológiai elemzése, és az elemzés könnyebben olvasható formája.

A magyarban az igekötő elválhat. Nyilván nem elvált (pl.: *elkészít*) és elvált (pl.: *készítette el*) esetben is ugyanarról az igekötős igéről van szó. Hasznos az igekötős igék összes alakját egyben látni, ezért jött létre a *PreverbIdentifier* kiegészítő eszköz, mely a függőségi elemzés alapján az igéhez kapcsolja a hozzá tartozó elváló igekötőt, és az igekötő és az igealak szótövének egybeírásaként megadja az igekötős szótövet – elváló esetben is.

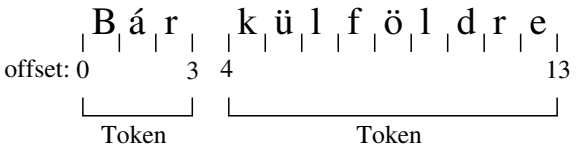
Ahogy említettük, az emChunk és az emNer IOB kódolással látja el a tokeneket. Ezt a kényelmesebb feldolgozhatóság érdekében az *Iob2Annot* kiegészítő eszköz önálló annotációvá: elkülönült egységeken lévő attribútumok sorozata helyett egy egységgé, a szöveg egy részletéhez rendelt 1 db címkévé alakítja.

A hasznos kiegészítő eszközök között említjük meg a GATE rendszerben eleve meglévő ún. *JAPE* transzdúcert is, mely az annotációk fölött megfogalmazott reguláris kifejezésekkel teszi lehetővé a szövegekben való szofisztikált keresést, vagy ha úgy tekintjük, új annotációk létrehozását.

4. A GATE annotációs modellje és a GATE-integráció

Az `e-magyar` rendszer szövegfeldolgozó eszközláncát alkotó, fentiekben áttekintett különféle modulok integrációját a GATE [10] (`https://gate.ac.uk`) nyelvfeldolgozó keretrendszerben valósítottuk meg. A Java nyelven implementált GATE előnye, hogy kényelmes módszert biztosít tetszőleges számú nyelvfeldolgozó eszköz (ún. *Processing Resource*) rendszerbe illesztésére. A másik fontos előnyös tulajdonsága az egyszerű, univerzális annotációs modell, melyre építve biztosítható a gördülékeny kommunikáció az egyes modulok között.

A feldolgozás legelején a szövegben a *karakterközök* kapnak egy sorszámot (ez az ún. *offset*), és onnantól kezdve *minden* annotáció kiterjedését egy offset-pár fejezi ki, mely az annotáció elejét és végét adja meg (3. ábra). Ez, a szöveget és az annotációt szétválasztó (standoff) annotációs modell lehetőséget biztosít arra, hogy az annotációk tetszőleges módon átfedjék egymást. Az annotációknak attribútumai is lehetnek. Az elemzés során hozzáadott információ közvetlenül konkrét annotáció (címke) formájában (pl.: az egyes tokenekhez rendelt *Token* annotáció) vagy az annotációk attribútumaiban (pl.: a *Token* annotáció *lemma* attribútuma, mely az egyértelműsített szótövet tartalmazza) kap helyet.



3. ábra. A GATE annotációs modellje. Minden karakterköz rendelkezik egy offset azonosítóval, az annotációk elejét és végét ezekkel az offsetekkel adjuk meg. A példában szereplő első *Token* annotáció 0-tól 3-ig tart, és a *Bár* szót ragadja meg.

A GATE az annotációkat annotációs halmazokban (*AnnotationSet*-ekben) tárolja. Ez azt jelenti, hogy az igényeknek megfelelően más-más halmazba téve bizonyos annotációkat teljesen elkülöníthetünk egymástól. Tipikus eset: ha HTML-fájl a kiinduló szöveganyagunk, akkor a GATE az eredeti HTML-annotációt

automatikusan feldolgozza, leválasztja, és eltárolja egy *Original markup* nevű annotációhalmazba, így lehetővé teszi, hogy a szöveganyagot az egyszerű szöveg inputtal megegyező módon dolgozzuk fel, a nyelvi elemzés során keletkező hozzáadott annotációkat pedig másik halmazba téve külön kezeljük.

A GATE azt is megengedi, hogy az attribútumok értéke nemcsak puszta szöveget, hanem tetszőleges Java objektumot (például stringek listáját) tartalmazzon. Látjuk, hogy az annotációs modell általánosabb, erősebb egy sima XML-szerű markupnál egyrészt az átfedés lehetősége, másrészt az annotációs halmazok, harmadrészt az attribútumokban megengedett adatszerkezetek miatt.

E modell használatával az annotációk függetlenek egymástól, nem zavarják egymást. Ez hasznos megoldás: így minden modul csak a számára releváns annotációt kell, hogy beolvassa, az eredményét pedig kiírhatja a megfelelő meglévő vagy újonnan létrehozott annotációba, attribútumba. Például: a tokenizáló *Token* és *SpaceToken* elemeket hoz létre a szavaknak és a szóközöknek megfelelően, a morfológiai elemző már csak a *Token*ek listáját fogja lekérni, ezen végzi el a morfológiai elemzést, a *SpaceToken*eket pedig érintetlenül hagyja. A tokenizálót követő elemző lépések tipikusan a tokenek bizonyos attribútumait felhasználva új token-attribútumokat hoznak létre. A modulok paraméterezhetők abban a tekintetben, hogy mely annotációkon dolgozzanak, ezzel a rendszer rugalmassága még tovább növelhető.

Az alapvető integrációs feladat tehát az volt, hogy minden modult alkalmassá tegyünk arra, hogy a bemenetét a GATE annotációs modelljének megfelelő formából tudja venni, és a kimenetét is ennek megfelelő formában prezentálja. Más szóval minden eszközhöz egy GATE-es wrappert kellett gyártani, ami a szükséges adatkonverziókat elvégzi a GATE-s formátum és a modul saját formátuma között. Kiegészítő feladat, hogy ha az egymástól független offset-alapú annotációk között kapcsolatot akarunk megadni, akkor azt explicit meg kell tenni. Jó példa erre az emNer által felismert tulajdonneveket és az őket alkotó tokeneket összekötő kapcsolat: itt az lett a megoldás – és ezt valósítja meg az *Iob2Annot* kiegészítő eszköz –, hogy a tulajdonnév annotáció egy attribútumában soroljuk fel listaként a tulajdonnevet alkotó tokenek azonosítóit.

Egy fontos, technikai kérdés volt, hogy milyen módon integráljuk a nem Java nyelven írt (emToken, emMorph, emChunk, emNer) eszközöket. Úgy döntöttünk, hogy az adott más nyelvű program binárisát vagy a más nyelvű szkriptet közvetlenül hívjuk meg. Ennek ellenére, megfelelő technikák alkalmazása révén a legtöbb esetben (ld. 7. rész) nincs szükség minden hívásnál az eszközök jelentős időigénnyel bíró újbóli inicializálására. Az elkészült elemzőlánc Linux és Windows operációs rendszeren futtatható.

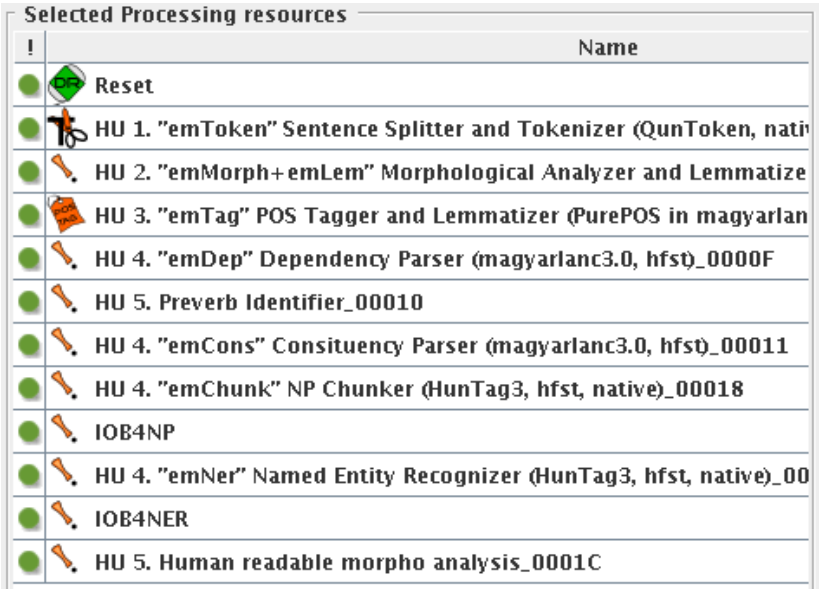
A működtetéshez szükség volt arra is, hogy az összes eszköz az új morfológiai elemző által adott kódkészlettel működjön. Ehhez elő kellett állítani a Szeged Treebank [11] új morfológiai kódokkal annotált változatát, majd ezen be kellett tanítani az emTag egyértelműsítőt, valamint a rá épülő eszközöket. A fenti feladatokat valósítottuk meg az integráció során.

A felhasználók számára előnyös, hogy a GATE annotációs modelljének és az integrációnak köszönhetően tehát nem kell törődni azzal, hogy: (1) hogyan

kell futtatni az egyes eszközöket, (2) milyen inputot várnak, és milyen outputot adnak az egyes eszközök, (3) egy-egy eszköz hogyan kezeli (használja fel, hagyja figyelmen kívül, őrzi meg) a meglévő annotációkat, milyen belső formátumot használ, és milyen annotációba helyezi el a saját elemzési eredményét. Az ilyen kérdéseket a GATE elrejti, elfedi, automatikusan megoldja. Az *Iob2Annot* pont egy egyszerű ilyenfajta GATE-es elrejtő megoldás: a HunTag3-alapú eszközök belső formátumának tekinthető IOB kódolást közvetlenül értelmezhető, szokásos önálló annotációvá alakítja.

5. Az elemzőlánc összeállítása a GATE Developerben

A GATE rendszer egyik fontos eleme a *GATE Developer* nevű grafikus felhasználói felület, ahol igényeink szerint állíthatjuk össze az elemzőláncokat az eszközökből (GATE Processing Resource-okból, PR), és lefuttathatjuk különféle szövegeken és a belőlük összeállított korpuszokon (GATE Language Resource-okon, LR). Ezek kívül számos kiegészítő funkció is rendelkezésre áll.



4. ábra. Az `e-magyar` szövegfeldolgozó eszközlánc a GATE Developer felületén.

A szövegfeldolgozó láncot tartalmazó *Lang_Hungarian* GATE plugin installálása után be kell tölteni az egyes eszközöket, és össze kell állítani belőlük a kívánt feldolgozó láncot. Először (1) jobb kattintás a bal panelen a *Processing Resources*-ra, és válasszuk ki a listából a kívánt eszközöket, majd (2) a bal panel

Applications részében hozzunk létre egy új `e-magyar` elnevezésű *Corpus Pipeline*-t, végül (3) a létrehozott *Corpus Pipeline*-ra kattintva állítsuk össze a fő panelen a láncot úgy, hogy a kívánt eszközöket a kívánt sorrendben a jobb oldali listába rendezzük. A teljes összeállított lánc a 4. ábrán látható.

Az ábrán a korábban leírt sorrendben szerepel az összes szövegfeldolgozó és kiegészítő eszköz. A számok arra utalnak, hogy hogyan épülnek egymásra az eszközök, hogy hányadik „rétegben" szerepel egy adott eszköz: 1. réteg az emToken, 2. réteg az egybeépített emMorph + emLem, 3. réteg az emTag, a 4. rétegben szintaktikai elemzők és a HunTag3-ra alapuló eszközök vannak, az 5. rétegben pedig a kiegészítő eszközök szerepelnek (vö: 1. ábra).

Az elemzés többszöri lefuttatása esetén hasznos, ha a lista elejére elhelyezünk egy *Document Reset PR*-t, ami minden futtatás előtt alaphelyzetbe állítja a dokumentumot, azaz törli az összes hozzáadott annotációt. Ezt a mindig rendelkezésre álló *ANNIE* pluginból tölthetjük be.

A legtöbb eszközt egyszerűen csak be kell töltenünk, és be kell tennünk az ábrán szereplő Corpus Pipeline-ba. Kivétel az *Iob2Annot*, melynek két példányára van szükségünk eltérő paraméterezéssel: egyszer a főnévi csoportok, másszor pedig a tulajdonnevek annotációjának átalakítására. Ahogy az ábrán látható, a két példányt értelemszerűen *IOB4NP* és *IOB4NER* elnevezéssel láttuk el. A paramétereket ezekre az eszközökre kattintva a képernyő alján állíthatjuk be az 1. táblázat szerint.

1. táblázat. Az *Iob2Annot* paraméterezése. A `inputIobAnnotAttrib` paraméter annak az attribútumnak a neve, amelyből a bemenő IOB annotációt veszi az eszköz, a `outputAnnotationName` pedig az új önálló annotáció neve. A megfelelő paraméterértékek a táblázatban láthatók.

	`inputIobAnnotAttrib`	`outputAnnotationName`
emChunk	`NP-BIO`	`NP`
emNer	`NER-BIO1`	`NE`

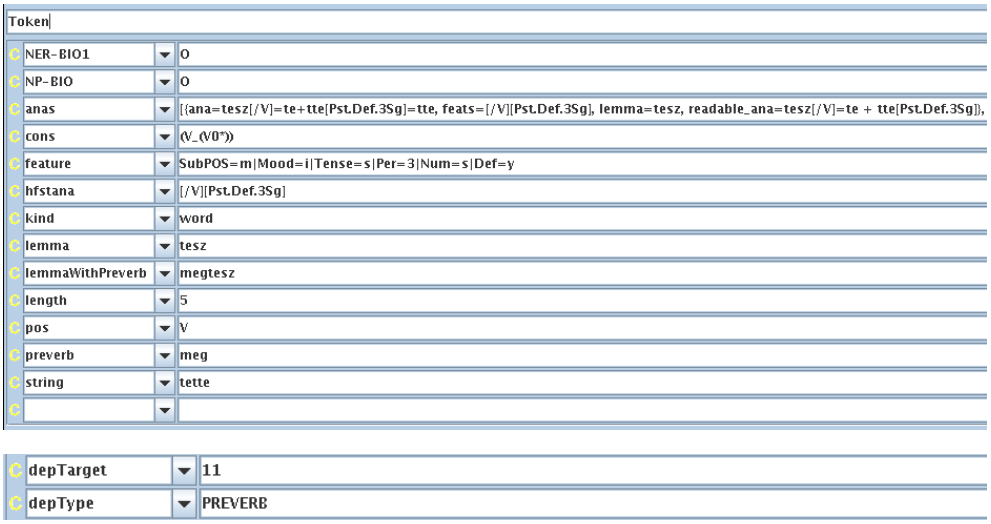
A paraméterező felületen igény szerint átállíthatjuk az eszközök alapbeállításait, ha például egy másik annotációs halmazba szeretnénk irányítani az elemzés eredményét.

6. Az elemzőlánc futtatása és az elemzés eredménye a GATE Developerben

Az összeállított elemzőlánc futtatásához (1) a bal panelen hozzunk létre egy *Language Resource*-ot: egy új *GATE Document*-et, ez fogja tartalmazni a feldolgozandó szöveget. A *GATE Developer* hatékonyan kezel számos formátumot (txt, HTML, XML, doc, stb.), belőlük automatikusan kinyeri a szöveges tartalmat. (2) Készítsünk a dokumentumból korpuszt: jobb kattintás a létrehozott

GATE Document-re, majd válasszuk a *New Corpus with this Document* lehetőséget. Végül (3) kattintsunk az `e-magyar` *Corpus Pipeline*-ra, a képernyő közepén, a *Corpus*-nál adjuk meg az imént létrehozott korpuszt, és kattintsunk lent a *Run this Application* gombra.

Az eredményeket a létrehozott *GATE Document*-re kattintva tekinthetjük meg az *Annotation Sets*, az *Annotation List*, és a kívánt annotált egység (*Token*, *NP*, *NE*) bekapcsolásával. A szövegben az egyes egységek fölé állítva az egeret megjelennek az adott egység attribútumainak értékei. Az elemzés eredményeként hozzáadott információ túlnyomó része a *Token* egységek attribútumaiként látható az 5. ábra és a 2. táblázat szerint.



5. ábra. Példamondatunk *tette* szavának attribútumai az `e-magyar` szövegfeldolgozó lánc lefuttatása után a GATE Developer felületén (fent). Részlet a példamondat *meg* szavának annotációjából a függőségi elemzés bemutatására (lent).

A *tette* szó természetesen nem része NP-nek, a *haladó* esetén a `NP-BIO` attribútumban `I-NP` szerepelne, ami azt jelenti, hogy a szó egy NP közbülső eleme.

A GATE Developerből az elemzett dokumentum GATE XML formátumban menthető el. Ez egy speciális XML formátum, amiben a GATE annotációs modellje ábrázolható. Tartalmazza a szöveget a szükséges offsetekkel együtt, és a szövegtől standoff módon elkülönítve az annotációkat attribútumaikkal. A kimentett XML-fájl pontosan a GATE Developer felületén is látható imént leírt annotációkat tartalmazza.

Itt a GATE Developernek csak a legalapvetőbb használatát mutattuk be, illetve a legszükségesebb információkat közöltük az `e-magyar` szövegfeldolgozó lánc

2. táblázat. A *Token* annotáció attribútumainak értelmezése. Az `anas` egy listában tartalmazza a részletes morfológiai elemzésekhez tartozó információkat, ezen belül: `ana` = részletes morfológiai elemzés, `feats` = emLem egyszerűsített elemzés, `lemma` = emLem szótő, `readable_ana` = `ana` olvashatóbb formája.

attribútum	értelmezés	a példában (5. ábra)
`NER-BIO1`	emNer IOB kód	a szó nem része tulajdonévnek
`NP-BIO`	emChunk IOB kód	a szó nem része NP-nek
`anas`	emMorph + emLem kimenet	
`cons`	emCons annotáció	egytagú igei csoport
`depTarget`	emDep szülő azonosítója	a *meg* szülője a *tette*
`depType`	emDep relációtípus	`PREVERB` (igekötő)
`feature`	az emTag kimenetéből a szintaktikai elemzők számára meghatározott jellemzők	a *tette* jegyei
`hfstana`	az emTag által választott elemzéshez tartozó egyszerűsített elemzés a HunTag3 eszközök számára	a *tette* szófaja és elemzése
`kind`	emToken szótípus	`word` (szó)
`lemma`	emTag szótő	*tesz*
`lemmaWithPreverb`	elváló igekötő esetén az igekötős szótő	*megtesz*
`length`	emToken szóhossz	5 karakter
`pos`	emTag szófaj	`V` (ige)
`preverb`	elváló igekötő esetén az igéhez tartozó igekötő	*meg*
`string`	emToken szóalak	*tette*

által szolgáltatott elemzés, annotáció értelmezéséhez. A GATE használatának további részletei és lehetőségei tekintetében a GATE rendszer dokumentációjára utalunk.

7. Négyféle hozzáférési, használati mód

A különböző felhasználói csoportok igényei szerint négy különböző módon lehet hozzáférni a rendszerhez, ezt tekintjük át az alábbiakban.

A legszélesebb érdeklődői kör számára lehetőség az `e-magyar.hu` honlap használata, mely a teljes elemzési láncot lefuttatja korlátozott szövegmennyiségen, az eredményt megjeleníti, letölthetővé teszi, a szintaktikai elemzések eredményét grafikusan is ábrázolja. Nem kell semmit installálni, az elemzés böngészőből futtatható, csupán annyi a teendő, hogy az elemzendő szöveget be kell másolni a honlap *Szövegelemző* felületére. A honlap a közoktatásban is használható demonstrációs eszköz, részletesebb leírás [1]-ben olvasható róla.

Komolyabb szövegelemzési feladathoz, digitális bölcsészeti kutatáshoz, ha az elemzőlánc bővítésére, új eszközök beépítésére van igény, illetve ha nagyobb mennyiségű az elemzendő szöveg, a GATE rendszer *GATE Developer* nevű grafikus felületének használata ajánlott, ahogy ezt a fentiekben bemutattuk. A GATE Developerben összeállítható a kívánt lánc, lefuttatható és az elemzések eredmény megjeleníthető. Az `e-magyar` elemzőláncon kívül megkapjuk a teljes GATE arzenált, a különféle létező nyelvfeldolgozó eszközeivel (az annotációtörlőtől a gépi tanulásig), és hasznos kiegészítő funkcióival, mint például a többféle bemeneti szövegformátum kezelése, az elemzőeszközök paraméterezhetősége, a kiértékelő modul, a kézi annotáló eszköz vagy a JAPE transzdúcer. Első futtatáskor lassabb működést tapasztalunk, mert ekkor töltődnek be a szükséges erőforrások, modellek, a statikus erőforrásoknak köszönhetően azonban a további futtatásoknál ez a plusz időigény nem jelentkezik. A GATE rendszer telepítése után csupán a GATE Developer saját egyszerű telepítési mechanizmusát kell használni, mely az általunk publikált GATE Plugin repozitóriumból letölti és beilleszti a rendszerbe a *Lang_Hungarian* plugint, mely a teljes láncot tartalmazza. Ennek leírása megtalálható az `e-magyar` szövegfeldolgozó lánc github repozitóriumában: `https://github.com/dlt-rilmta/hunlp-GATE`.

Jelentősebb méretű szöveganyag elemzéséhez a GATE parancssori hozzáférését az ún. *GATE Embedded* technológiát ajánljuk. Ennek révén beépíthetjük az eszközöket nagyobb szoftverrendszerekbe, vagy használhatjuk őket önállóan. A *pipeline*-ként aposztrofált megvalósításunk a GATE alapműködésének megfelelően – a használati módok közül egyetlenként – minden indításkor betölti az erőforrásokat, ezért használata csak nagyobb szövegmennyiség esetén célszerű.

A negyedik használati mód – az ún. *gate-server* – szintén a GATE Embedded technológiára épül, kliens-szerver architektúrában működik. A GATE-et egy HTTP szerverbe csomagoltuk, és kívülről, URL letöltésekkel használjuk. Így lehetővé válik az eszközök hatékony inicializálása: a GATE Developerhez hasonlóan ez a megoldás is csak egyszer tölti be az erőforrásokat, a szerver indításakor. Ez nagyon hasznos tulajdonság összevetve azzal az esettel, amikor az eszközöket a GATE-től függetlenül futtatnánk. A gate-server egyszerre kis méretű szövegdarabot dolgoz fel, nagy korpusz elemzése a korpusz feldarabolásával oldható meg. Egy gate-server üzemel az `e-magyar.hu` honlap mögött is. A harmadik és negyedik módszer leírása szintén az említett github repozitóriumban található, a használatba vételhez szükséges a github repozitórium klónozása.

8. Köszönetnyilvánítás

Az elemzőlánc integrációja az MTA 2015. évi Infrastruktúra-fejlesztési Pályázat 2. kategóriájában elnyert támogatás segítségével készült.

9. Konklúzió

Létrejött egy olyan magyar szövegfeldolgozó eszközlánc, mely egyesíti a legtöbb jelenleg elérhető nyílt forrású magyar szövegelemző eszközt. Tartalmaz egy új

Unicode-képes tokenizálót, valamint az eddigi magyar morfológiai elemzők jó tulajdonságait egyesítő jó minőségű új morfológiai elemzőt. Mindegyik eszköznek a legújabb verziója van beépítve, illetve mindegyik eszköz fel van készítve az új morfológiai kódkészlet használatára. Bízunk benne, hogy a különféle használati módoknak köszönhetően hasznos lesz nem csak a nyelvtechnológusok számára, nagy korpuszok elemzésére, hanem a kényelmes grafikus felhasználói felület révén a humán tudományok kutatói számára, illetve a honlap által a nagyközönség számára is. Reméljük, hogy a jövőben számos további újonnan létrehozott vagy akár korábban már meglévő magyar szövegfeldolgozó eszköz épül majd be ebbe a rugalmasan bővíthető keretrendszerbe, így egyre gazdagabb elemzési lehetőségek válnak elérhetővé. Ehhez lehetőségeinkhez mérten támogatást is nyújtunk.

Hivatkozások

1. Váradi, T., Simon, E., Sass, B., Gerőcs, M., Mittelholcz, I., Novák, A., Indig, B., Prószéky, G., Farkas, R., Vincze, V.: Az `e-magyar` digitális nyelvfeldolgozó rendszer. In: XIII. Magyar Számítógépes Nyelvészeti Konferencia (MSZNY2017), Szeged: JATEPress (2017) (jelen kötetben)
2. Mittelholcz, I.: emToken: UTF-8 képes tokenizáló magyar nyelvre. In: XIII. Magyar Számítógépes Nyelvészeti Konferencia (MSZNY2017), Szeged (2017) (jelen kötetben)
3. Novák, A., Siklósi, B., Oravecz, Cs.: A new integrated open-source morphological analyzer for Hungarian. In: Proceedings of the 10th International Conference on Language Resources and Evaluation (LREC2016), Portorož (2016) 1315–1322
4. Lindén, K., Silfverberg, M., Pirinen, T.: HFST tools for morphology – an efficient open-source package for construction of morphological analyzers. In Mahlow, C., Piotrowski, M., eds.: State of the Art in Computational Morphology. Volume 41 of Communications in Computer and Information Science. Springer Berlin Heidelberg (2009) 28–47
5. Endrédy, I., Novák, A.: Szótövesítők összehasonlítása és alkalmazásaik. Alkalmazott Nyelvtudomány **XV**(1–2) (2015) 7–27
6. Orosz, Gy.: Hybrid algorithms for parsing less-resourced languages. PhD thesis, Roska Tamás Doctoral School of Sciences and Technology, Pázmány Péter Catholic University (2015)
7. Zsibrita, J., Vincze, V., Farkas, R.: magyarlanc: A toolkit for morphological and dependency parsing of Hungarian. In: Proceedings of RANLP. (2013) 763–771
8. Endrédy, I., Indig, B.: HunTag3: a general-purpose, modular sequential tagger – chunking phrases in English and maximal NPs and NER for Hungarian. In: 7th Language & Technology Conference (LTC '15), Poznań, Poland, Poznań: Uniwersytet im. Adama Mickiewicza w Poznaniu (2015) 213–218
9. Ramshaw, L.A., Marcus, M.P.: Text chunking using transformation-based learning. In: Proceedings of the 3rd Annual Workshop on Very Large Corpora. (1995) 82–94
10. Cunningham, H., Maynard, D., Bontcheva, K., Tablan, V., Aswani, N., Roberts, I., Gorrell, G., Funk, A., Roberts, A., Damljanovic, D., Heitz, T., Greenwood, M.A., Saggion, H., Petrak, J., Li, Y., Peters, W.: Text Processing with GATE (Version 6). (2011)
11. Vincze, V., Szauter, D., Almási, A., Móra, Gy., Alexin, Z., Csirik, J.: Hungarian Dependency Treebank. In: Proceedings of LREC 2010, Valletta, Malta, ELRA (2010)