

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 közlést tesz 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 rövidítésekénél, 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 szöveggörnyezet 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* igehez 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.

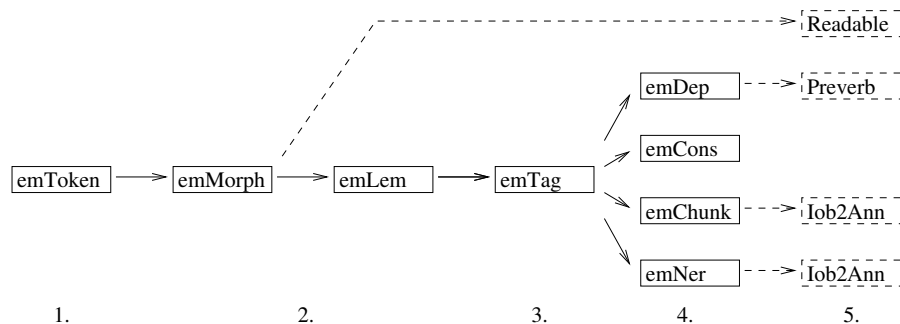
A morfológiai elemzést az *emMorph* [3] eszköz végzi. A szöveg minden – a tokenizáló által pontosan behatárolt – szóalakjához (tokenjéhez) hozzárendeli az összes lehetséges morfológiai, morfoszintaktikai elemzését, a szóalak aktuális környezetétől függetlenül. Megállapítja a szófaji főkategóriát, megadja 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 morfológiai elemzővel egybeépített emLem [5] szótövesítő eszköz határozza meg. A szóalakokhoz szótövet, a szótőnek megfelelő eredő szófajcímket 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 *Magyarlanc* 3.0 verziójába [7] integrált változata. Ez az eszköz gépi tanulási módszerrel a szöveg minden tokenjéhez meghatározza az aktuális szöveggörnyezetben érvényes szótövet, 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 kevésbé 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[P1]=ek+ről[Del]=ről
→
amely[/N|Pro|Rel] + ek[P1] + 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 igeről van szó. Hasznos az igekötős ige összes alakját egyben látni, ezért jött létre a *PreverbIdentifier* kiegészítő eszköz, mely a függőségi elemzés alapján az igehez kapcsolja a hozzá tartozó elváló igekötőt, és az igekötő és az igealak szótövének egybeírásaként megadja az igekötős szótövet – elváló esetben is.

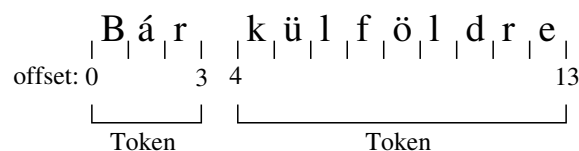
Ahogy említettük, az emChunk és az emNer IOB kódolással látja el a tokeneket. Ezt a kényelmesebb feldolgozhatóság érdekében az *Iob2Annot* kiegészítő eszköz önálló annotációvá: elkülönült egységeken lévő attribútumok sorozata helyett egy egységgé, a szöveg egy részletéhez rendelt 1 db címkévé alakítja.

A hasznos kiegészítő eszközök között említjük meg a GATE rendszerben eleve meglévő ún. *JAPE* transzdúcort 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 (címké) 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 szövegyanyagunk, 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 pusztán szöveget, hanem tetszőleges Java objektumot (például stringek listáját) tartalmazzon. Látjuk, hogy az annotációs modell általánosabb, erősebb egy sima XML-szerű markurnál 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 *Tokenek* listáját fogja lekérni, ezen végzi el a morfológiai elemzést, a *SpaceTokenek* 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 elrejtí, 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.

Selected Processing resources	
!	Name
	Reset
	HU 1. "emToken" Sentence Splitter and Tokenizer (QunToken, native)
	HU 2. "emMorph+emLem" Morphological Analyzer and Lemmatize
	HU 3. "emTag" POS Tagger and Lemmatizer (PurePOS in magyarlan
	HU 4. "emDep" Dependency Parser (magyarlanc3.0, hfst)_0000F
	HU 5. Preverb Identifier_00010
	HU 4. "emCons" Constituency Parser (magyarlanc3.0, hfst)_00011
	HU 4. "emChunk" NP Chunker (HunTag3, hfst, native)_00018
	IOB4NP
	HU 4. "emNer" Named Entity Recognizer (HunTag3, hfst, native)_00
	IOB4NER
	HU 5. Human readable morpho analysis_0001C

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 elhelyezzünk 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.

	<code>inputIobAnnotAttrib</code>	<code>outputAnnotationName</code>
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.

Token	
NER-BIO1	O
NP-BIO	O
anas	[(ana=tesz[/V]=te+tte[Pst.Def.3Sg]=tte, feats=[/V][Pst.Def.3Sg], lemma=tesz, readable_ana=tesz[/V]=te + tte[Pst.Def.3Sg]],
cons	(V_(V0*))
feature	SubPOS=m Mood=i Tense=s Per=3 Num=s Def=y
hfstana	[/V][Pst.Def.3Sg]
kind	word
lemma	tesz
lemmaWithPreverb	megtesz
length	5
pos	V
preverb	meg
string	tette
depTarget	11
depType	PREVERB

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 tulajdonnévnek
NP-BIO	emChunk IOB kód	a szó nem része NP-nek
anas	emMorph + emLem kimenet	
cons	emCons annotáció	egytágú igei csoport
depTarget	emDep szülő azonosítója	a <i>meg</i> szülője a <i>tette</i>
depType	emDep relációtípus	PREVERB (igekötő)
feature	az emTag kimenetéből a szintaktikai elemzők számára meghatározott jellemzők	a <i>tette</i> jegyei
hfstana	az emTag által választott elemzéshez tartozó egyszerűsített elemzés a HunTag3 eszköz számára	a <i>tette</i> szófaja és elemzése
kind	emToken szótípus	word (szó)
lemma	emTag szótő	<i>tesz</i>
lemmaWithPreverb	elváló igekötő esetén az igekötős szótő	<i>megtesz</i>
length	emToken szóhossz	5 karakter
pos	emTag szófaj	V (ige)
preverb	elváló igekötő esetén az igehez tartozó igekötő	<i>meg</i>
string	emToken szóalak	<i>tette</i>

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 grafikusán 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 bemutatunk. 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 repozitóriumban: <https://github.com/dlt-rilmta/hunlp-GATE>.

Jelentősebb méretű szöveganyag elemzéséhez a GATE parancssori hozzáférést 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 nyelvttechnológusok 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 mértén 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)

emLam – a Hungarian Language Modeling baseline

Dávid Márk Nemeskey

Institute for Computer Science and Control
Hungarian Academy of Sciences
nemeskeyd@gmail.com

Abstract. This paper aims to make up for the lack of documented baselines for Hungarian language modeling. Various approaches are evaluated on three publicly available Hungarian corpora. Perplexity values comparable to models of similar-sized English corpora are reported. A new, freely downloadable Hungarian benchmark corpus is introduced.

1 Introduction

Language modeling (LM) is an integral part of several NLP applications, such as speech recognition, optical character recognition and machine translation. It has been shown that the quality of the LM has a significant effect on the performance of these systems [5, 7]. Accordingly, evaluating language modeling techniques is a crucial part of research. For English, a thorough benchmark of n-gram models was carried out by Goodman [10], while more recent papers report results for advanced models [20, 8]. Lately, the One Billion Word Benchmark corpus (1B) [8] was published for the sole reason of measuring progress in statistical language modeling.

The last decade saw dramatic advances in the field of language modeling. Training corpora grew from a few million words (e.g. the Brown corpus) to gigaword, such as 1B, while vocabulary size increased from a few 10k to several hundred thousands. Neural networks [3, 21, 19] overtook n-grams as the language model of choice. State-of-the-art LSTMp networks achieve up to 55% reductions in perplexity compared to 5-gram models [14].

Surprisingly, these developments left few traces in the Hungarian NLP literature. Aside from an interesting line of work on morphological modeling for speech recognition [23, 18], no study is known to the author that addresses issues of Hungarian language modeling. While quality works have been published in related fields, language model performance is often not reported, or is not competitive: e.g. in their otherwise state-of-the-art system, Tarján et al. [28] use a 3-gram model that achieves a perplexity of 400¹ on the test set — a far cry from the numbers reported in [8] and here.

In this paper, we mean to fill this gap in two ways. First, we report baselines for various language modeling methods on three publicly available Hungarian corpora. Hungarian poses a challenge to word-based LM because of its agglutinative nature. The proliferation of word forms inflates the vocabulary and decreases the number of contexts a word form is seen during training, making the data sparsity problem much more

¹ Personal communication with the author.