Az e-magyar digitális nyelvfeldolgozó rendszer

Váradi Tamás[1], Simon Eszter[1], Sass Bálint[1], Gerőcs Mátyás[1], Mittelholcz Iván[1], Novák Attila[2], Indig Balázs[2], Prószéky Gábor[2,4], Farkas Richárd[3], Vincze Veronika[3]

[1] MTA Nyelvtudományi Intézet,
1068 Budapest, Benczúr u. 33., e-mail:
VEZETEKNEV.KERESZTNEV@nytud.mta.hu
[2] MTA–PPKE Magyar Nyelvtechnológiai Kutatócsoport,
1083 Budapest, Práter utca 50/a, e-mail:
VEZETEKNEV.KERESZTNEV@itk.ppke.hu
[3] Szegedi Tudományegyetem, Informatikai Intézet,
6720 Szeged, Árpád tér 2., e-mail: {rfarkas,vinczev}@inf.u-szeged.hu
[4] MorphoLogic Kft.
1122 Budapest, Ráth György u. 36., e-mail: {novak,proszeky}@morphologic.hu

Kivonat Cikkünkben átfogó ismertetést adunk az `e-magyar` rendszerről, amely a magyar nyelvtechnológiai közösség összefogásaként jött létre. Az új infrastruktúra fő célja az eddig előállított különböző eszközök továbbfejlesztése, egységesítése és egyetlen koherens technológiai láncba szervezése volt. Az interoperabilitás mellett fontos cél volt a modularitás, a nyílt rendszer és a széles körű elérhetőség. A modulok szabadon használhatók a GATE rendszer keretein belül, emellett pedig igénybe vehető az `e-magyar.hu` oldal webszolgáltatása is. Az írott szöveg feldolgozása mellett az `e-magyar` rendelkezik egy olyan résszel is, amely a beszédfeldolgozást segíti egy beszédadatbázissal és beszédelemző modulokkal.

Kulcsszavak: kutatási infrastruktúra, természetesnyelv-feldolgozás, egységesítés, GATE, integráció, interoperabilitás

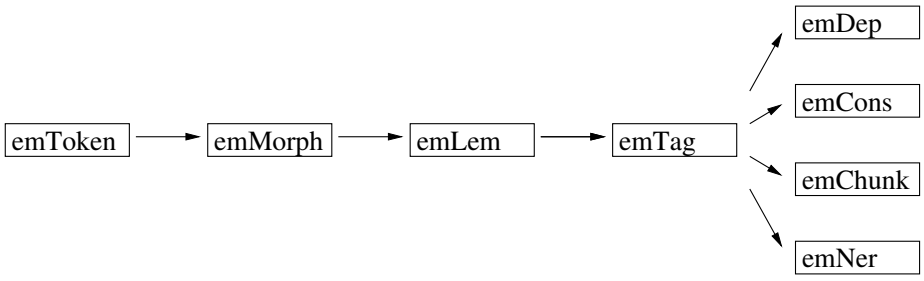
1. Bevezetés

Cikkünkben átfogó ismertetést adunk az `e-magyar` rendszerről, amely a magyar nyelvtechnológiai közösség összefogásaként jött létre 2016-ban. A munkálatok a Magyar Tudományos Akadémia támogatásával, a 2015-ben kiírt infrastruktúra-fejlesztési pályázat keretében folytak a Nyelvtudományi Intézet koordinálásával. A munkában részt vett még a Szegedi Tudományegyetem, az MTA SZTAKI, a Pázmány Péter Katolikus Egyetem, az AITIA International Zrt., valamint a MorphoLogic Kft.

Az új infrastruktúra fő célja a munkában részt vevő műhelyekben eddig előállított különböző eszközök továbbfejlesztése, egységesítése és egyetlen koherens technológiai láncba szervezése volt. Az interoperabilitás mellett fontos cél volt a modularitás, a nyílt rendszer és a széles körű elérhetőség.

A központi gondolat az `e-magyar` kialakításában az integráció volt. A magyar nyelvtechnológiai közösség külön-külön, de az utóbbi évtizedben egyre inkább

együttműködve számos kiváló erőforrást és eszközt hozott létre. Ezek a Nyelv- és Beszédtechnológiai Platform, illetve a META-NET[5] hálózaton keresztül is publikálásra kerültek. Az eszközök egy része nyílt forráskódú (ilyen a `hun*` eszközcsalád[6]), mások csak bináris formában érhetők el kutatás-fejlesztési célokra (ilyen a Humor morfológiai elemző [11]). A mostani infrastruktúra interoperábilissá tette ezeket az eszközöket abban az értelemben, hogy az infrastruktúra egyes eszközei modulárisan egymásra épülnek, vagyis önállóan is működnek, de olyan elemzési láncba is szervezhetők, amelyben zökkenőmentesen halad az adat a különböző eszközökön át. Ez azt jelenti, hogy a nyers szövegből kiindulva az `e-magyar` szövegfeldolgozó eszközlánca elvégzi a szöveg elemeinek a szegmentálását (`emToken`), megállapítja az egyes szavak tövét és teljes morfológiai elemzését (`emMorph`, `emLem` és `emTag`), majd ezek után megadja a mondatok összetevős (`emCons`), valamint függőségi elemzését (`emDep`); de ha csak egy gyors elemzésre van szükségünk, felismeri a mondatban szereplő frázisokat (`emChunk`), továbbá a szövegben előforduló tulajdonneveket (`emNer`). Az eszközök egymásra épülése az 1. ábrán látható.



1. ábra. Az `e-magyar` szövegfeldolgozó lánc elemeinek egymásra épülése.

Fontos megemlíteni, hogy a magyarra már létezik egy szövegfeldolgozó eszközlánc, a `magyarlánc` [25], amely szintén megvalósítja ezt a moduláris architektúrát, de egy zárt rendszeren belül. Az `e-magyar.hu` fejlesztésénél fontos szempont volt, hogy nyílt rendszer legyen, vagyis hogy az infrastruktúra egésze és annak minden eszköze külön-külön is elérhető, letölthető, világos licenccel publikált és kutatás-fejlesztési célra, de adott esetben üzleti felhasználásra is ingyenesen használható legyen.

A láncban részt vevő szoftverek licence GNU GPLv3 vagy GNU LGPLv3, a nem-szoftver elemekre vonatkozó licenc pedig Creative Commons Attribution-ShareAlike 4.0 (CC BY-SA), kivéve az `emMorph` morfológiai elemző alatt működő adatbázist, amelyre az üzleti felhasználást kizáró Creative Commons Attribution-NonCommercial-ShareAlike 4.0 (CC BY-NC-SA) licenc vonatkozik.

[5] `http://www.meta-net.eu/`
[6] `http://hlt.bme.hu/en/resources/hun-toolchain`

Az e-magyar nem csak a nyelvtechnológiai szakma vagy a nyelvtechnológiát használó ipari fejlesztők igényeit kívánja szolgálni. A szakmabeli felhasználók mellett támogatjuk a számítógépes eljárások iránt fogékony, de a nyelvtechnológiában nem jártas diákok és kutatók körét is: a bölcsészet- és társadalomtudomány művelőit, valamint akár az érdeklődő nagyközönséget is. Nekik két szinten kínálunk támogatást. Egyrészt egyedi igények alapján különböző elemzőláncokat állíthatnak össze az e-magyar eszközeinek a felhasználásával a GATE szövegelemző rendszer keretein belül. Másrészt igénybe vehetik az e-magyar.hu oldal webszolgáltatását, amely rövidebb szövegek azonnali elemzését végzi.

Az infrastruktúra két fő részből áll. Az írott szöveg feldolgozása mellett az e-magyar rendelkezik egy olyan résszel, amely a beszédfeldolgozást segíti egy beszédadatbázissal és beszédelemző modulokkal. Ennek megfelelően alakul a cikk felépítése is. A szövegfeldolgozó rész moduljainak ismertetése a 2. fejezetben található, míg a 3. fejezet a beszédarchívumot és a beszédfeldolgozó eszközöket mutatja be. A 4. fejezetben ismertetjük az infrastruktúra integrálását a GATE keretrendszerbe, az 5. fejezet pedig a webes felületet írja le.

2. A szövegfeldolgozó modulok

Az e-magyar.hu a szöveg automatikus feldolgozása során a szöveget először tokenekre bontja, és megállapítja a mondatok határát; bővebben lásd a 2.1. fejezetben. Ezután megkapjuk az egyes szavakról a morfológiai információkat; ezt a lépést ismerteti a 2.2. fejezet. Mivel a magyar szóalakok jelentős részének több lehetséges elemzése van, szükség van egy egyértelműsítési lépésre; erről szól a 2.3. fejezet. Az egyes mondatok mondattani elemzése kétféleképpen is megtörténik; ezt a két modult mutatja be a 2.4. fejezet. Egy következő lépésben a főnévi csoportokat is azonosítja egy erre a célra készített modul, lásd részletesebben a 2.5. fejezetben. Végül a lánc utolsó tagja megjelöli a tulajdonneveket, ami a 2.6. fejezetben kerül bemutatásra.

2.1. Tokenizáló és mondatra bontó

Az emToken tokenizáló felépítésében és szabályaiban közel áll a magyar nyelvű szövegek tokenizálására széles körben használt Huntoken-hez[7]. Ahhoz hasonlóan az elemzendő szöveg több elemzőrétegen megy át, amelyek elvégzik a mondatszegmentálás, majd a tokenizálás feladatát. Az egyes elemzőrétegek mögött a Quex[8] lexergenerátor dolgozik.

A tokenizáló bemenetéül UTF-8 kódolású sima szöveg szolgál. A program kezeli az összes olyan karaktert, amely a latin, a görög és a cirill karaktereket, valamint a szimbólumokat tartalmazó Unicode-kódtáblákban szerepel. A kimenet szintén UTF-8 kódolású szöveg. Kimeneti formátumként az XML és a JSON közül lehet választani (a kimenet önmagában nem valid, csak csonk, gyökér elemet és

[7] http://mokk.bme.hu/resources/huntoken/
[8] http://quex.sourceforge.net/

fejlécet nem tartalmaz). A használt címkekészlet a `Huntoken`-t veszi alapul. Az **s** jelöli a mondatokat, a **w** a szavakat, és a **c** a punktuációk címkéje. Egy új címke is bevezetésre került: a **ws** a szóközjellegű karaktersorozatokat jelöli. Az `emToken` ugyanis kimenetében megőrzi a szóközöket is, ez teszi lehetővé a feldolgozott szöveg detokenizálhatóságát. Az `emToken` részletes ismertetéséhez lásd még: [10].

2.2. Morfológiai elemző és lemmatizáló

A rendszer részét képező új magyar morfológiai elemző (`emMorph`) implementációja a véges állapotú transzducertechnológiát alkalmazó `HFST` rendszer [9] felhasználásával valósult meg. A morfológiai elemző adatbázisa elsősorban az eredetileg a Humor morfológiai elemző motorhoz [16] készült magyar morfológiai adatbázison alapul [11], amelyet kiegészítettünk a `morphdb.hu` adatbázisban [23] szereplő szavakkal. A morfológiai leírást kezelő keretrendszer egy procedurális szabályrendszer felhasználásával magas szintű és redundanciamentes morfémaleírásokból állítja elő az egyes morfémák lehetséges allomorfjait, azok 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 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 mindezek az adatszerkezetek egyetlen véges állapotú transzducerben jelennek meg [12]. Az eredeti Humor-formalizmus szónyelvtan-automatáját a véges állapotú leírásban a *flag diacritcs* konstrukció alkalmazásával ábrázoltuk. Ez a leírás tartalmazza a morfémák közötti nem lokális megszorításokat is.

A nagybetűsített és a csupa nagybetűvel írt szóalakok kezelésére a Humorban külön mechanizmus szolgál: a lexikonban kisbetűsítve tároljuk a morfok felszíni alakjait, és a nagybetűsítést bitvektorok írják le. A nagybetűsíthető morfémákat a szónyelvtan-automatában levő leírás adja meg. A `HFST`-ben ezzel szemben ezeket a transzformációkat is újraíró szabályokkal lehet megadni. Ezeknek a szabályoknak a lexikonnal való kompozíciója a lexikon méretének és ezzel az elemző egyébként is elég jelentős futásidejű memóriaigényének megháromszorozódását eredményezi. Ezért kezdeményeztük a `HFST` elemzőmotorjának fejlesztőjénél, hogy implementáljon egy olyan mechanizmust, amely lehetővé teszi transzducerek futásidőbeli dinamikus kompozícióját úgy, hogy az egyik (itt a kisbetű–nagybetű-konverziót végző) transzducer kimenetét a hozzá kapcsolt következő transzducer (itt a morfológiai elemző) bemenetére vezeti. A módszer elenyésző elemzésisebesség-csökkenés árán harmadára csökkenti az elemző memóriaigényét.

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 morfológiai elemzésre épülő és a nyelvi elemzés egyéb szintjeit végző eszközöknek általában nincs szükségük

ilyen részletes elemzésre, hanem csak az adott szó lemmájára és morfoszintaktikai jegyeire. A lemma magában foglalja a szóban levő töveket és képzőket is.

A lemma előállításához szükség van a tövet alkotó morfok lexikai és felszíni alakjára is. A `HFST` rendszer morfológiai elemzést végző eszközei (a `hfst-lookup`, illetve a `hfst-optimized-lookup`) alapesetben a felszíni alakot (illetve annak szegmentálását) nem adják vissza, így a képzőt tartalmazó tövek 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. Ez lehetővé tette, hogy létrehozzuk a morfológiai elemző kimenetére épülő `Java` nyelven implementált, ezért platformfüggetlen lemmatizáló eszközt (`emLem`), amely a tőalkotó elemek összevonásával kiszámolja az adott elemzéshez tartozó lemmát, annak eredő szófaját, és ehhez hozzáadja a nem tőalkotó morfémák által hordozott morfoszintaktikai jegyek címkéit. Az azonos lemmát, szófajt és egyéb címkesorozatot eredményező különböző részletességű elemzések a lemmatizáló kimenetén egyetlen elemzésként jelenhetnek meg, ezek a magasabb nyelvi szinteket feldolgozó elemzők számára ekvivalensek. Ugyanakkor a lemmatizáló képes a részletes elemzések visszaadására is úgy, hogy az az elemzést alkotó morfok felszíni alakját is tartalmazza olvasható formában. 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.

A korábbi magyar morfológiai elemzők általában az adott elemzőhöz *ad hoc* módon kifejlesztett címkekészletet használtak. Az `emMorph` morfológiai elemző és az `emLem` lemmatizáló kimenetén megjelenő címkekészletet ezzel szemben a nyelvészeti leírásokban elterjedten használt lipcsei notációnak megfelelő készlethez igazítottuk. A címkék meghatározásakor a Leipzig Glossing Rules-ra [3] és az ott leírtakat kiegészítő lényegesen kibővített listára[9] támaszkodtunk, amelyet kiegészítettünk a hiányzó (elsősorban képzőkkel kapcsolatos) címkékkel. Az elemző és a lemmatizáló által generált annotációval kapcsolatban lásd még: [13].

2.3. Morfológiai egyértelműsítő

A morfológiai egyértelműsítés két fő komponensből áll. Az első komponens a morfológiai címke, míg a második a címkéhez tartozó lemma meghatározása. Az előbbit a Thorsten Brants által a `TnT`-ben [2] bevezetett HMM-alapú módszer szolgáltatja, amelyet nyílt forráskódú, tanulmányozható és továbbfejleszthető implementációban `HunPos` néven ismert meg a világ [7]. A `HunPos` hátránya, hogy csak a morfológiai címkét határozza meg, a lemmát nem. Ezért volt szükség a rendszer újraimplementálására `PurePos` néven [14], amely egy lineáris modell segítségével képes kombinálni a lehetséges szótöveket és a megtalált címkéket. Ezt a programot technikai újításokkal és sebességbeli optimalizálással fejlesztettük tovább `emTag` néven, hogy egy kiforrott, robusztus rendszert hozzunk létre.

A legnagyobb problémát a tanítóanyagban lévő szóalakok, illetve azok hiánya okozza. Mivel a tanítóanyag nem tartalmaz minden lehetséges alakot, szükség

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

van egy morfológiai elemzőre, amely a statisztikai címke-guesser segítségére siet, és a képzési szabályok segítségével kiszűri a lehetetlen alakokat.

A bemeneti szavakat az `emTag` három osztályra osztja. A tanítóanyag segítségével az adott szóhoz legenerálja a címkevalószínűségeket, amennyiben a címke előfordult a tanítóanyagban. Ezt követően a kapott elemzéseket elmetszi a morfológia által adott elemzésekkel, hogy a címkézés hamis ágait lenyesse. Amennyiben az eredmény egyelemű halmaz, az egyértelműsítés megtörtént, más esetben a Viterbi-algoritmusra bízzuk az egyértelműsítést. Ha az adott szó nem szerepelt a tanítóanyagban vagy a morfológiában, vagy üres halmaz a morfológiával közös metszete, akkor a statisztikából tanult ragozási szabályok alapján a legvalószínűbb elemzéseket rendeli a szóhoz, akár az átmenetvalószínűség romlása árán is. Ez a működés azt feltételezi, hogy a morfológia és a tanítóanyag címkézési szisztémája szinkronban van. Ha előfordul olyan eset, hogy a morfológia által ismert, de a tanítóanyagban nem szereplő változat lenne a helyes, akkor további bonyolult simítási modelleket kellene alkalmazni, amely nem várt mellékhatásokkal járhatna. Az ilyen esetekben egyszerűbb hozzáadni néhány mondatot a tanítóanyaghoz, hogy a két forrás szinkronban legyen.

2.4. Összetevős és függőségi szintaktikai elemző

A mondatok összetevős szerkezeti elemzése azt tárja fel, hogy a szavak egymással kombinálódva milyen kifejezéseket alkotnak, illetve hogyan állnak össze egy mondattá; a függőségi elemzés pedig a mondatok szerkezeti egységei közötti függőségi viszonyokat (pl. alany, tárgy, jelző) tárja fel. A szintaktikai elemzők a már tokenizált és morfológiailag egyértelműsített mondatokat kapják meg bemenetként, felhasználva a korábbi modulok kimenetét.

A morfológiailag gazdag nyelvek, köztük a magyar, szintaktikai elemzésére hozták létre a Statistical Parsing of Morphologically Rich Languages (SPMRL) workshopsorozatot. Az elemzőláncba beépített rendszer a workshop keretében megrendezett SPMRL 2014 Shared Task első helyezést elért rendszere által bemutatott technikákra épül. A rendszerbe a valószínűségi környezetfüggetlen nyelvtanokat alkalmazó Berkeley Parser [15] egy módosított változatát [20] integráltuk. A magyar nyelv gazdag morfológiájának köszönhetően a szóalakok száma rendkívül nagy, ezért a tanítóhalmazban nem vagy csak ritkán látott szóalakokat lecseréljük a szófaji egyértelműsítés során megkapott fő szófaji kódra. A Berkeley Parser tanítása során kis mértékben szerepe van a véletlennek is, ennek a véletlennek a kiküszöbölésére 8 különböző modellt tanítottunk (eltérő random seed mellett), és predikáláskor a különböző modellek által egy mondatra adott valószínűségek szorzatát vettük, így kiátlagolva a véletlen szerepét.

A `magyarlanc` [25] moduljai között szerepel egy függőségi elemző is, mely a Bohnet parser [1] nevű nyelvfüggetlen függőségi elemzőre épül, a Szeged Dependencia Treebanken [24] betanítva. Ezt az elemzőt integráltuk az `e-magyar` rendszerébe. Ugyanakkor mindkét szintaktikai elemző esetében szükségesnek bizonyultak kisebb átalakítások, hogy azok az `emMorph` modul által kiadott elemzéseket hasznosítani tudják.

A Szeged Treebank eredetileg az MSD kódrendszert alkalmazza, illetve elkészült a Universal Dependencies keretrendszerben alkalmazott morfológiai annotációra való konvertálása is. Az `emMorph` ugyanakkor egy ezektől különböző formátumot használ, és több nyelvészeti elvben is eltér a fenti kódrendszerektől, illetve a Szeged Korpusz annotációs elveitől. Ezért annak érdekében, hogy továbbra is a Szeged Korpuszt tudjuk tanítóadatként használni, szükségesnek bizonyult új konverterek létrehozása. Így összevetettük az `emMorph` nyelvészeti elveit az MSD 2.5 kódrendszer, illetve a UD alapelveivel. Az összehasonlítás eredményeképpen a Szeged Korpusz morfológiai annotációját átalakítottuk az `emMorph` formátumára. Ugyanakkor az elemzőláncban szereplő összetevős és függőségi elemzők UD vagy MSD 2.5 formátumú morfológiai kódokat várnak inputként, így ezen elemzési lépésekhez az `emMorph` elemzését automatikusan visszaalakítjuk UD és MSD 2.5 formátumra.

2.5. Sekély szintaktikai elemző

Az `emChunk` modul sekély szintaktikai elemzést valósít meg, vagyis azonosítja a mondatot alkotó frázisokat, de a köztük levő viszonyokról, illetve az általuk a mondatban betöltött funkciókról nem mond semmit. Ez utóbbit megteszi a függőségi elemző, amelyet a 2.4. fejezet ismertet.

Az `emChunk` modullal a mondatban található maximális főnévi csoportokat (NP-ket) nyerjük ki, vagyis olyan NP-ket, melyek nem részei egy magasabb szintű NP-nek sem. Tervbe van véve egy másik működési mód integrálása is, amelynek során minden típusú frázist azonosítunk a mondatban, így a főnévi csoportok mellett például a határozói (AdvP) és a névutói csoportokat (PP) is.

Az `emChunk` modul a `HunTag3`-ra [6] épül, amely egy maximum entrópiát és HMM-et használó, többféle szekvenciális címkézési feladatra alkalmas rendszer. Elődje a `HunTag`[10], amely többek között magyar nyelvű tulajdonnév-felismerésre [19] és sekély szintaktikai elemzésre [17] volt használva. Felhasználási köre igazából csak a tanítóadaton múlik. A gold standard chunk címkékkel ellátott tanítóadat a Szeged Treebank [4] összetevős elemzéséből lett a megfelelő formátumra konvertálva.

2.6. Tulajdonnév-felismerő

Az `emNer` automatikus tulajdonnév-felismerő rendszer azonosítja a folyó szövegben található tulajdonneveket, és besorolja őket az előre meghatározott névkategóriák valamelyikébe (személynév, intézménynév, földrajzi név, egyéb). Az elemzőlánc többi lépéséhez hasonlóan, az előző szinteken már feldolgozott szövegekkel dolgozik, vagyis a bemeneti szöveg tokenekre és mondatokra van bontva, valamint minden egyes tokenhez hozzá van rendelve a töve és a teljes morfológiai elemzése. Ezek az információk szükségesek a tulajdonnév-felismerő rendszer hatékony működéséhez [19].

[10] `https://github.com/recski/HunTag`

Az `emChunk`-hoz hasonlóan, e mögött a modul mögött is a `HunTag3` szekvenciális címkéző rendszer fut. A gold standard tulajdonnévi címkékkel ellátott tanítóadat a Szeged NER korpusz [21], illetve annak az új morfológiai formalizmusra konvertált változata volt. A nyelvmodell a teljes korpusz felhasználásával készült.

3. Beszédfeldolgozás

Az `e-magyar` beszédtechnológiai része tartalmaz egy beszédarchívumot, valamint három, az automatikus beszédfeldolgozást támogató modult.

Az `e-magyar` Nyílt Beszédarchívum (Open Speech Archive, `emOSA`) létrehozásával három fő célunk volt. Az első és legfontosabb a magyar beszédtechnológiára annak kezdetei óta jellemző zárt kutatási és publikációs modell felváltása egy szabad, nyílt forrású modellel. Második célunk a hagyományos, gondosan felcímkézett, és mind artikulációsan, mind aukusztikailag tiszta adatokon alapuló felügyelt tanulási módszerek felváltása gyengén felügyelt, illetve felügyeletlen módszerekkel. Harmadik célunk pedig egy a digitális bölcsészeti munkát, elsősorban a szociológiát, történelemtudományt és etnográfiát beszédtechnológiai oldalról támogató platform alapjainak megteremtése.

Az `emSad` beszéddetektáló modul beszédszegmentálást végez audió fájlokon. A fájlokat háromféle szegmensre bontja: beszéd, csend és zaj. Ez az első lépés minden további beszédfeldolgozási művelet előtt. A következő modul a beszélődiarizáló (`emDia`), amely egy több beszélő beszédét tartalmazó hangfelvétel esetében arra a kérdésre ad választ, hogy ki mikor beszélt. Képes tehát különbséget tenni a beszédhangok között, és felismerni, amikor az egyik beszélő átveszi a szót a másiktól. A harmadik modul, az `emPros` (korábbi nevén ProsoTool [22]) program az élőnyelvi kommunikációban előforduló verbális megnyilatkozások intonációjának elemzésére és lejegyzésére szolgál. Az archivált hangfelvételekből kinyerhető akusztikai paraméterek beszélőnkénti feldolgozása és stilizálása után az interakcióban részt vevők egyedi hangterjedelméhez viszonyítva címkézi fel a megnyilatkozások hanglejtését. Elsősorban több résztvevős interakciók elemzéséhez készült, de felolvasott szövegek, monologikus közlések feldolgozására is használható. Az `e-magyar` teljes beszédtechnológiai részének részletesebb leírásához lásd még: [8].

4. GATE-integráció

Az `e-magyar` szövegfeldolgozó láncát alkotó, 2. részben ismertetett eszközöknek egy egységes elemzőláncba való integrálását a GATE keretrendszerben [5] valósítottuk meg. Az integráció során az alapvető feladat az volt, hogy a modulokat alkalmassá tegyük arra, hogy a bemenetüket a GATE *offset*-alapú annotációs modelljének megfelelő formából vegyék, és a kimenetüket is ebben a formában állítsák elő. Ehhez minden eszközhöz GATE-es wrappert készítettünk, amely elvégzi a szükséges adatkonverziókat. Szükséges volt ezen túl a nem `Java` nyelven

írt eszközök illesztése a Java nyelvű keretrendszerhez: ezt a más nyelvű progamok binárisának közvetlen hívásával oldottuk meg.

Az eszközök az 1. ábra szerint épülnek egymásra: a tokenizálóból, morfológiai elemzőből, lemmatizálóból és egyértelműsítőből álló alapvető kötött feldolgozólánc eredményére épülnek a további eszközök, melyek már egymástól függetlenül futtathatók.

Az eszközöket további kényelmi eszközök egészítik ki. Az egyik a részletes morfológiai elemzésből állít elő egy ember számára is olvasható formát. A másik a függőségi elemzés alapján külön megjelöli az elváló igekötőt, és megadja az igekötős igei tövet. A harmadik pedig az emChunk és az emNer eszközök által szolgáltatott IOB-típusú (konkrétan BIE-1) kódolást alakítja kényelmesebben kezelhető önálló (NP és NE) annotációkká.

Az eszközlánc négyféleképpen használható. A honlapon keresztül (lásd az 5. fejezetet) egy rövidebb szöveget egyszerűen bemásolva kipróbálhatjuk az eszközláncot. Szövegelemzéshez, digitális bölcsészeti kutatáshoz a GATE rendszer *GATE Developer* nevű grafikus felhasználói felületét ajánljuk, amelybe az e-magyar lánc egyszerűen telepíthető. A telepítés leírása elérhető a https://github.com/dlt-rilmta/hunlp-GATE oldalon a teljes rendszerrel együtt. Itt lehetőség van a rendszer továbbfejlesztésére, vagyis az elemzőlánchoz saját készítésű modulok is hozzáadhatók. Nagyobb korpuszok feldolgozásához a GATE parancssori hozzáférését ajánljuk, ennek használata szintén az említett honlapon található, szükséges hozzá a github repozitórium használata. Negyedik módszerként használatba vehető az ún. *gate-server* is, ez szintén parancssori technológia, és ez az egyébként, amely a honlap mögött is üzemel. Az integrációról és a rendszer használatáról részletesebben lásd még: [18].

5. Webes felület

A projekt célkitűzései között szerepelt, hogy az elemzőlánc hozzáférhető és érdemben használható legyen olyan felhasználók körében is, akik nem feltétlenül járatosak az informatika területén. Ennek az igénynek igyekszik megfelelni az e-magyar.hu webes szövegelemző szolgáltatása[11], amely lehetővé teszi, hogy egy webes interfészen keresztül bárki egyszerűen kipróbálhassa az egyes elemző modulokat vagy akár a teljes elemzőláncot, anélkül, hogy ehhez a böngészőn kívül bármilyen egyéb szoftvert használnia kellene.

A szövegelemző egy olyan webszolgáltatásra épül, amely a GATE-es könyvtárakat használja, bemenetként az elemzett szöveget és a futtatni kívánt elemző modulok listáját várja, kimenetként pedig a GATE által generált, az annotációkat tartalmazó XML-t adja vissza. A weboldal a visszakapott XML-t feldolgozza, és a kinyert adatokat megjeleníti egy könnyen értelmezhető, vizualizált formában.

Az elemző felülete két fő részből áll: egy bemeneti és egy kimeneti panelből. A bemeneti panelen található szövegmező segítségével tudja megadni a felhasználó az elemezni kívánt szöveget (a bevihető szöveg hossza jelenleg 6000 karakterre

[11] http://e-magyar.hu/parser

van korlátozva), valamint itt tudja összeállítani azoknak a moduloknak a listáját, amelyeket le szeretne futtatni a szövegen.

A feldolgozás eredménye a kimeneti panelre kerül. Az elemzés kétféle elrendezésben jeleníthető meg: 'szöveg' és 'lista' nézetben. 'Szöveg' nézetben a tokenek sorfolytonosan követik egymást, az egyes tokenekhez tartozó annotációk kis buborékokban jelennek meg a kurzort egy adott token fölé mozgatva vagy kattintásra. Elváló igekötők esetében az igei tagot tartalmazó token is automatikusan kiemelődik. 'Lista' nézetben az egyes tokenek táblázatos formában, külön sorban egymás alá, az elemző modulok által hozzáadott annotációk pedig az egymást követő oszlopokba kerülnek. Ez utóbbi elrendezés alkalmasabb sok információ együttes megjelenítésére. Ebben a nézetben lehetőség van a tokenek szűrésére különböző szempontok alapján: a felhasználó szűrhet szóalakra, a morfológiai elemzés egy részletére (az `emMorph` kimenete), szófaji címkére (az `emTag` kimenete) és grammatikai funkcióra (az `emDep` kimenete). Mindkét nézetben lehetőség van az egyes modulok által létrehozott egy vagy több tokenből álló szegmensek kiemelésére: a tokenizáló esetében ezek a tokenek és a mondatok, a sekély szintaktikai elemző esetében a főnévi csoportok, a tulajdonnév-felismerő esetében pedig a tulajdonnevek. A szintaktikai elemzések eredménye a 'szöveg' nézetben érhető el az egyes mondatokat követő ikonokra kattintva: a függőségi elemző kimenete egy függőségi fa, az összetevős elemező kimenete pedig egy ágrajz formájában.

Az elemzés eredményét a felhasználó letöltheti magának további felhasználásra. A letöltött csomag három fájlt tartalmaz: a feldolgozásra elküldött nyers szöveget sima szövegfájlként, a GATE által generált `XML`-fájlt, valamint a 'lista' nézet kivonatát `tsv` formátumban.

6. Köszönetnyilvánítás

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

A munkálatokat Váradi Tamás koordinálta, a szövegfeldolgozó részt Oravecz Csaba, a beszédtechnológiai munkát Kornai András irányította. Rajtuk és a szerzőkön kívül számos kutató és fejlesztő vett részt a projektben, akik nélkül az itt ismertetett eszközlánc nem jött volna létre. Ők a következők: Ács Judit, Bobák Barbara, Both Zsolt, Falyuna Nóra, Fegyó Tibor, Kovács Réka, Kundráth Péter, Ludányi Zsófia, Makrai Márton, Miháltz Márton, Nemeskey Dávid, Pajkossy Katalin, Rebrus Péter, Schreiner József, Siklósi Borbála, Szekrényes István, Takács Dávid, Zsibrita János.

Hivatkozások

1. Bohnet, B.: Top accuracy and fast dependency parsing is not a contradiction. In: Proceedings of Coling 2010. pp. 89–97 (2010)
2. Brants, T.: Tnt: A statistical part-of-speech tagger. In: Proceedings of the Sixth Conference on Applied Natural Language Processing. pp. 224–231. Association for Computational Linguistics, Stroudsburg, PA, USA (2000)

3. 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
4. Csendes, D., Csirik, J., Gyimóthy, T., Kocsor, A.: The Szeged Treebank. In: Lecture Notes in Computer Science: Text, Speech and Dialogue. pp. 123–131. Springer (2005)
5. 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), http://tinyurl.com/gatebook
6. 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, Human Language Technologies as a Challenge for Computer Science and Linguistics (LTC '15). pp. 213–218. Poznań: Uniwersytet im. Adama Mickiewicza w Poznaniu, Poznań, Poland (November 2015)
7. Halácsy, P., Kornai, A., Oravecz, C.: Hunpos: An open source trigram tagger. In: Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions. pp. 209–212. Association for Computational Linguistics, Stroudsburg, PA, USA (2007)
8. Kornai, A., Szekrényes, I.: Az e-magyar beszédarchívum. In: XIII. Magyar Számítógépes Nyelvészeti Konferencia (MSZNY2017). p. (jelen kötetben). Szeged (2017)
9. 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)
10. Mittelholcz, I.: emToken: Unicode-képes tokenizáló magyar nyelvre. In: XIII. Magyar Számítógépes Nyelvészeti Konferencia (MSZNY2017). p. (jelen kötetben). Szeged (2017)
11. Novák, A.: Milyen a jó Humor? In: I. Magyar Számítógépes Nyelvészeti Konferencia. pp. 138–144. SZTE, Szeged (2003)
12. Novák, A.: A Humor új Fo(r)mája. In: X. Magyar Számítógépes Nyelvészeti Konferencia. pp. 303–308. SZTE, Szeged (2014)
13. Novák, A., Rebrus, P., Ludányi, Zs.: Az emMorph morfológiai elemző annotációs formalizmusa. In: XIII. Magyar Számítógépes Nyelvészeti Konferencia (MSZNY2017). p. (jelen kötetben). Szeged (2017)
14. Orosz, G.: Hybrid algorithms for parsing less-resourced languages. Ph.D. thesis, Roska Tamás Doctoral School of Sciences and Technology, Pázmány Péter Catholic University (2015)
15. Petrov, S., Barrett, L., Thibaux, R., Klein, D.: Learning accurate, compact, and interpretable tree annotation. In: Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics. pp. 433–440 (2006)
16. Prószéky, G., Kis, B.: A unification-based approach to morpho-syntactic parsing of agglutinative and other (highly) inflectional languages. In: Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics. pp. 261–268. ACL '99, Association for Computational Linguistics, Stroudsburg, PA, USA (1999)
17. Recski, G., Varga, D.: A Hungarian NP Chunker. The Odd Yearbook. ELTE SEAS Undergraduate Papers in Linguistics pp. 87–93 (2009)

18. Sass, B., Miháltz, M., Kundráth, P.: Az `e-magyar` rendszer GATE környezetbe integrált magyar szövegfeldolgozó eszközlánca. In: XIII. Magyar Számítógépes Nyelvészeti Konferencia (MSZNY2017). p. (jelen kötetben). Szeged (2017)
19. Simon, E.: Approaches to Hungarian Named Entity Recognition. Ph.D. thesis, PhD School in Cognitive Sciences, Budapest University of Technology and Economics (2013)
20. Szántó, Zs., Farkas, R.: Special techniques for constituent parsing of morphologically rich languages. In: Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics. pp. 135–144. Association for Computational Linguistics, Gothenburg, Sweden (April 2014)
21. Szarvas, G., Farkas, R., Felföldi, L., Kocsor, A., Csirik, J.: A highly accurate Named Entity corpus for Hungarian. In: Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC'06). pp. 1957–1960. ELRA (2006)
22. Szekrényes, I.: Prosotool, a method for automatic annotation of fundamental frequency. In: 6th IEEE International Conference on Cognitive Infocommunications (CogInfoCom). pp. 291–296. IEEE, New York (2015)
23. 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)
24. Vincze, V., Szauter, D., Almási, A., Móra, Gy., Alexin, Z., Csirik, J.: Hungarian Dependency Treebank. In: Proceedings of LREC 2010. ELRA, Valletta, Malta (May 2010)
25. Zsibrita, J., Vincze, V., Farkas, R.: magyarlanc: A toolkit for morphological and dependency parsing of Hungarian. In: Proceedings of RANLP. pp. 763–771 (2013)