Lemmi vagy nem lemmi

Novák Attila, Novák Borbála

Pázmány Péter Katolikus Egyetem Információs Technológiai és Bionikai Kar
MTA-PPKE Magyar Nyelvtechnológiai Kutatócsoport
Budapest, Práter u. 50/a.
{novak.attila, siklosi.borbala}@itk.ppke.hu

Kivonat A szövegfeldolgozással foglalkozó kutatások nagy része ma már a magasabb szintű elemzést megvalósító algoritmusok fejlesztésével foglalkozik, az alacsony szintű, leginkább az előfeldolgozás során használt eszközöket azok hiányosságaival együtt késznek, elfogadhatónak tekintve. Azonban ezeknek az eszközöknek a pontossága sok esetben meg sem közelíti a tökéletes eredményt, az ezek által az eszközök által bevezetett hibák így továbbterjednek a feldolgozási lánc magasabb szintű moduljaiba. Jelen kutatásban célunk egy olyan algoritmus létrehozása volt, ami az egyik ilyen alapeszköz, a lemmatizáló pontosságának javítására használható. A létrehozott rendszer igen magas pontossággal (92,13%) képes megítélni, hogy egy szóalakhoz automatikusan létrejött lemma az eredeti szóalakhoz rendelt szófajcímkével együtt helyes-e vagy sem.
Kulcsszavak: lemmatizálás, morfológia, hibajavítás, neurális hálózat

1. Bevezetés

A legtöbb automatikus szövegfeldolgozó lánc önálló modulok egymás után való alkalmazásából áll. Az egyes modulok kimenete az utána következő bemenete lesz. Egy ilyen architektúrában a feldolgozás elején keletkező hibák a rendszerben tovább terjednek, a magasabb szintű feldolgozást végző modulok a saját hibaarányukon túl a megelőző moduloktól kapott hibás bemenetre is hibás választ fognak adni. Ezért nagyon fontos, hogy a sztenderd alapfeldolgozást végző eszközök a lehető legnagyobb pontossággal működjenek.

Cikkünkben egy ilyen eszköz minőségének javítására teszünk javaslatot: a szófaji egyértelműsítéssel párhuzamosan történő lemmatizáló guesser által létrehozott lemmák helyességét értékelő algoritmust mutatunk be, melynek használatával jelentősen csökkenthető az automatikus elemzés során létrejövő hibás lemmák száma.

Kutatásunk során a PurePos [1] szófaji egyértelműsítő lemmatizáló algoritmusát vizsgáltuk (felmerült más lemmatizálók tesztelése is, ám ezek teljesítménye ránézésre is sokkal rosszabb volt a PurePosénál). A PurePos lehetőséget biztosít morfológiai elemző integrálására, így a lemmatizálás során a morfológiai elemzőt használja azoknak a szavaknak az elemzésére és lemmatizálására, amik benne vannak a morfológiai elemzőben. Azoknak a szavaknak a tövesítéséhez

viszont, amik nem szerepelnek a morfológiai elemző lexikonában, egy egyszerű lemmatizáló algoritmust használ. Ez az algoritmus a TnT tagger eredeti szuffix guesser algoritmusának [2] egy olyan kiegészítése, amely az adott szóalakból a lemmát levezető kódot is tartalmazza az adott szóhoz hozzárendelt morfoszintaktikai címkén kívül[1]. Ennek az algoritmusnak a pontossága viszonylag alacsony (89,01% sztenderd szövegen mérve, tehát minden 10. szóhoz hibás lemmát generál [1]).

Ugyanakkor, nem csak a morfológiai elemző számára ismeretlen szavakból jöhetnek létre hibás lemmák. Számos esetben a morfológiai elemző is olyan elemzéseket produkál, amit valamilyen produktív minta alkalmazásával hoz létre. Ezek azonban nem feltétlenül lesznek értelmes elemzések, a lemma sem lesz helyes.

A hibásan lemmatizált alakok fő forrását a rossz minőségű szövegek jelentik. Ha a szöveg sok hibás, elírt vagy nem sztenderd szóalakot tartalmaz, nem tud ezekre a szavakra a morfológiai elemző elemzést adni, így a rendszer a kevésbé pontos, végződés alapú guessert használja. Ezeknek a szavaknak az esetében azonban tulajdonképpen nincs is esélye az adott szóalakból a lemmát algoritmikusan kiszámolni próbáló eszközöknek arra, hogy helyes eredményt kapjanak: ehhez a szöveget előbb ki kellene javítani.

Kutatásunk során az volt a célunk, hogy egy olyan algoritmust hozzunk létre, amelyik a hibásan lemmatizált szavakat azonosítja, függetlenül attól, hogy a guesser vagy a morfológia hozta-e létre ezeket.

2. Módszer

Kutatásunkban a PurePos által létrehozott lemmáknak három osztályát definiáltuk:

1. Nem lemma (`notlem`): a létrejött szótőjelölt nem lemma, hanem ragozott alak. Pl. *nyelvészt[FN]*
2. Rossz elemzés (`badana`): a létrejött szótőjelölt a hozzárendelt szófajcímkével nem helyes. Pl. *vár[MN]*
3. Helyes lemma (`lem`): a létrejött szótőjelölt a hozzárendelt szófajcímkével együtt helyes. Pl. *vár[IGE]*

A lemmajelöltek e három osztályba való besorolására két módszert próbáltunk ki. Először egy karakteralapú rekurrens neurális hálózatot tanítottunk be, majd egy SVM (Support Vector Machine) osztályozót hoztunk létre. Az SVM bemenetére a lemmajelölteknek egy olyan reprezentációja került, ami az eredeti szóalak és a lemma szóbeágyazási vektorából és a szófajcímkéből áll. A következő alfejezetekben a rendszer létrehozásának és tanításának lépéseit ismertetjük.

2.1. A tanítóanyag létrehozása

A tanítóanyag előállításához egy 1,2 milliárd tokenből álló magyar nyelvű webkorpuszt használtunk [3]. Mivel a webről gyűjtött korpuszban elég sok a zajos,

illetve nem sztenderd nyelvhasználattal írt szövegrész (fórumok, kommentek, blogok, stb.), ezért ez a típusú forrás elég sok hibásan elemzett és lemmatizált szót tartalmazott. A korpuszt a Humor morfológiai elemzővel [4,5] kiegészített PurePos eredeti verziójával elemeztettük, majd kigyűjtöttük belőle azokat a szavakat, amik legalább ötször előfordultak a korpuszban. A létrejött listában szereplő lemmák vagy az integrált morfológiai elemzőből származnak, vagy a guesser algoritmus hozta őket létre. A kapott szólistán ezért lefuttattuk a morfológiai elemzőt, és azokat a szóalakokat, amik a Humor lexikonában lemmaként szerepelnek, tehát a morfológia felismerte, az adott szófajjal helyes lemmának tekintettük (`lem`). Azokat a szavakat, amik a morfológia szerint helyes szóalakok, de nem lemmák, hanem egy szónak valamilyen ragozott alakjai, azokat a „nem lemma" osztályba soroltuk (`notlem`).

A harmadik osztályba pedig olyan szavakra kellett példákat gyűjtenünk, amiknek hibás volt az elemzése, amik biztosan nem helyes lemmák az adott szófajcímkével. Az ilyen jellegű rosszul lemmatizált szóalakok igen jellemzőek egy bizonyos típusú hibát nagy mennyiségben tartalmazó szövegtípusra, mint például ékezet nélküli szövegrészek, rosszul tokenizált szövegrészek, stb. Ezek tehát a tanítóanyag létrehozásához használt korpusznak jellemzően ilyen részleteiből kerültek ki, bár nem szűkítettük le a korpuszt ilyen részkorpuszokra a hibás szavak gyűjtésekor sem. Az ebbe a csoportba tartozó szóalakoknak a kigyűjtését jobb híján úgy oldottuk meg, hogy először kerestünk olyan szavakat, amik egyrészt lemmaként szerepeltek az elemzett korpuszban és volt is olyan szófajú elemzésük, amilyen szófajúként az adott szó az adott alakban ténylegesen szerepelt, másrészt viszont volt egy másik elemzésük más szófajjal, amilyen szófajjal az adott szó az adott alakban nem szerepelt a korpuszban. Például a *vár* lemma főnévi (FN) és igei (IGE) elemzése mellett melléknévként (MN) is megjelenik az elemzés után, mint melléknévi lemma. Ezek közül viszont csak a *vár[FN]* és a *vár[IGE]* alakok szerepelnek önmagukban is, a *vár[MN]* elemzést a PurePos lemmatizáló guessere állította elő valamilyen más szóalak egyébként hibás elemzéseként. Néhány ilyen szóalakból kiindulva feltérképeztük a korpuszból épített szóbeágyazási modellben (lásd a 2.2. részt) az ezek környezetében szereplő szavakat, kihasználva azt a korábbi megfigyelést, hogy a szóbeágyazási térben a hasonló jellegű hibát tartalmazó szóalakok egymáshoz közel helyezkednek el [6]. A 1. táblázatban néhány ilyen hibásan lemmatizált szó és legközelebbi szomszédaik láthatók. Néhány ilyen szóalak szomszédsági listáját kézzel ellenőrizve hoztuk létre a hibás lemmák osztályába (`notlem`) tartozó tanítómintát.

Végül 11066 rosszul elemzett szó, 82582 helyes lemma és 80000 ragozott szóalak alkotta a tanítóanyagot, amiből mindhárom kategóriából 1000-1000 véletlenszerűen kiválasztott elemet különítettünk el a teszteléshez.

2.2. Szóbeágyazási modellek

A neurális szóbeágyazási modellek a tanítóanyagban (korpuszban) szereplő szavakat egy néhány száz dimenziós térben helyezik el, ahol a hasonló jelentésű, és/vagy hasonló szintaktikai, grammatikai, stiláris, stb. tulajdonságokkal rendelkező szavak egymáshoz közel, a kevésbé hasonlóak távolabb helyezkednek el

vár[MN]	alma[IGE]	szép[IGE]
tud[MN]	funda[IGE]	ilo[IGE]
kér[MN]	mangus[IGE]	nari[IGE]
jé[MN]	manda[IGE]	evian[IGE]
fogad[MN]	spiritualité[FN]	insa[IGE]
cso[MN]	Tibi[IGE]	manam[IGE]

1. táblázat. Példa hibásan lemmatizált és elemzett szavakra, illetve a hozzájuk legközelebb eső elemek az elemzett szövegből épített szóbeágyazási modellben

[7,8]. A már korábban létrehozott modelljeink közül [9] kettőt használtunk fel ebben a kutatásban. Az egyiket a nyers korpuszon tanítottuk, így a szavak eredeti alakjukban szerepelnek benne. A másik modell tanításakor pedig a korpuszra szófaji egyértelműsítést, illetve morfológiai elemzést alkalmaztunk, majd a tanítás előtt a szótöveken csak a fő szófajcímkét hagytuk meg, a címkék további morfoszintaktikai információt tartalmazó részét leválasztottuk, és az önálló tokenként szerepelt a tanítóanyagban. A modellek részletes összehasonlítását lásd [9]-ben. Mindkét modell esetében a tanítás során a reprezentációt létrehozó neurális hálózat a CBOW architektúrát valósította meg 5 token sugarú ablakméret alkalmazásával 300 dimenziós vektorokat hozva létre.

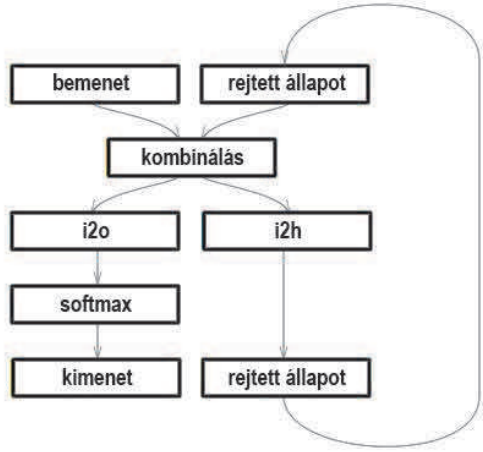
2.3. Karakteralapú rekurrens neurális hálózat használata az osztályozási feladatra

Az utóbbi évek egyik legeredményesebb kutatási iránya a neurális hálózatok, illetve a mélytanulás módszereinek alkalmazása számos, köztük szövegfeldolgozási, területen előforduló problémákra. Ezért első megoldásként egy rekurrens neurális hálózatot tanítottunk be a létrehozott tanítóanyag alapján. Ez a modell egyáltalán nem használja az előző részben bemutatott szóbeágyazási modelleket. A neurális hálózat bemenetén a vizsgálandó szó karakterei egymást követve jelennek meg, végül pedig a szófajcímke. Az egyes szavakat alkotó karaktereket one-hot vektor formában reprezentáltuk. Ennek a vektornak a hossza megegyezik a tanító- és tesztanyagban szereplő különböző karakterek és szófajcímkék számával. Az egyes pozíciókon pedig 0-k szerepelnek, kivéve az aktuális karakter/címke indexének megfelelő pozíciót, ahol 1-es áll.

A létrehozott rekurrens neurális hálózat (RNN) egy kétrétegű hálózat, egy rejtett állapottal. Minden lépésben a következő bemenetet (karakter, illetve címke), illetve az előző lépésben kapott eredményt használja fel (az első lépésben ez nulla). Az egyes lépések kimenete pedig egy, a három osztálynak megfeleltethető háromelemű vektor, ami a bemenet addigi feldolgozása alapján az egyes osztályokba való tartozás valószínűségét tartalmazza. Ezt a részeredményt használja rejtett állapotként is, amit a bemenetre visszacsatolva a következő lépésben felhasznál a következő karakter vagy szófajcímke beolvasása alapján adott predikció során. A kimeneti vektor normalizálására egy LogSoftmax réteget használtunk. A

neurális hálózat architektúrája az 1. ábrán látható, aminek az implementációjához a PyTorch keretrendszert[1] használtuk a következő paraméterbeállításokkal: 256 volt a rejtett réteg dimenziószáma, a rendszert 200000 epochon keresztül tanítottuk, a tanulási faktor 0,005 volt.

A vizsgált `szóalak[CÍMKE]` formájú bemenet feldolgozása során tehát az RNN megtippeli, hogy az adott elem mekkora valószínűséggel tartozik az egyes osztályokba. Bár a tanítóanyagban egy-egy elem akár több osztályban is szerepelt, a kiértékelés során az adott szóhoz legnagyobb valószínűséggel rendelt osztályt vettük csak figyelembe.



1. ábra: A rekurrens neurális hálózat sematikus ábrája

2.4. Support Vector Machine alapú osztályozó

Egy másik kísérletben azt vizsgáltuk, hogy az említett szóbeágyazási modellek felhasználásával egy hagyományos osztályozó algoritmus milyen pontossággal oldja meg az adott osztályozási feladatot. A support vector machine (SVM) algoritmus egy olyan felügyelt tanítási módszert valósít meg, ami minden egyes elemet egy n-dimenziós tér egy pontjának tekint és ebben a térben keresi meg azokat a hipersíkokat, amik a különböző osztályokba tartozó elemeket egymástól elkülönítik. A használat során pedig minden egyes új bemenetet ugyanebbe a térbe helyez el, az adott elem pozíciójának megfelelő osztály lesz a rendszer által hozzárendelt besorolás. Ebben az esetben tehát minden elemhez egyetlen osztályt (és nem az összes osztályba való tartozás valószínűségeinek vektorát) kapjuk eredményként.

Az SVM bemenete tehát egy n-dimenziós vektor, ami esetünkben három részből állt:

[1] http://pytorch.org/

- a vizsgált szónak (szófajcímke nélkül) a szavak felszíni alakjából épített szóbeágyazási modellben szereplő 300 dimenziós vektora. Ha nem szerepelt a szóalak a modellben, akkor egy nullákból álló 300 dimenziós vektort rendeltünk hozzá. A reprezentációnak ez a része azzal kapcsolatos információt tárol, hogy a lemmajelölt szerepel-e a modellben (illetve a korpuszban), és ha igen, akkor a kapott vektor jellemzi az adott szóalak grammatikai tulajdonságait. Lemmaként normális esetben a szó valamelyik ténylegesen létező alakját használjuk, és hogy konkrétan melyiket, az a szó szófajától függ. Ez az összetevő tehát egyrészt arra a feltevésre épül, hogy a toldalékolt szavak lemmája is nagy valószínűséggel előfordul a nyers korpuszban,[2] másrészt hogy mind a valódi lemmák, mind a ragozott szóalakok esetében a szófajcímke és a lemmajelölt vektorreprezentációja között rendszeres összefüggést feltételezhetünk.
- a vizsgált szónak (szófajcímkével együtt) a lemmákból és morfoszintaktikai jegyekből épített szóbeágyazási modellben szereplő 300 dimenziós vektora. Ha nem szerepelt a szóalak a modellben, akkor egy nullákból álló 300 dimenziós vektort rendeltünk hozzá.
- a szófajcímkék one-hot vektora, amit az RNN-ben használt reprezentációhoz hasonlóan úgy hoztunk létre, hogy egy, az összes különböző szófajcímke számával megegyező hosszúságú vektort nullákkal töltöttünk fel, csupán az aktuális szó szófajához tartozó pozícióban szerepelt 1-es. Az általunk használt tanítóanyagban és tesztanyagban összesen 23 különböző szófajcímke szerepelt, tehát ez az összetevő egy 23 dimenziós vektor.

Így minden szóhoz egy 623 dimenziós vektor állt elő a fenti összetevők alapján, ami az SVM bemeneteként szerepel. Azért, hogy ne csupán lineáris szeparáció jöhessen létre a tanítóminták alapján, a tanítás során RBF kernelt használtunk.

3. Eredmények

A kiértékelés során mindhárom osztályból 1000-1000, a tanítóanyagban nem szereplő mintát használtunk tesztanyagként. Mivel a tanítóanyagban egy-egy szó akár több osztályban is szerepelhetett, a véletlenszerűen választott mintában előfordultak ismétlődések, ezeket a mérések pontossága érdekében kitöröltük, így végül 2974 szón értékeltük ki a modelleket. A számszerű eredményeket a 2. táblázat tartalmazza.

A rendszerek minőségét két szempontból vizsgáltuk. Az első esetben a három osztályra vonatkozó pontosságot mértük, csak azt az eredményt tekintve helyesnek, amikor a ténylegesen helyes osztályt eltalálta a rendszer (lásd *Pontosság (3 osztályra)*). Ez az RNN esetében 74,37% volt, míg az SVM esetében 87,86%. Mindkét esetben tehát a vizsgált szavaknak legalább a háromnegyedéhez a helyes osztály került hozzárendelésre, de az SVM esetén ez az arány majdnem elérte a

[2] Ez alól van néhány kivétel, mint pl. az *ága-boga* szó, amelynek esetében a lemmatizáló az *ág-bog* alakot állítja elő lemmaként, illetve más hasonló kifejezések.

90%-ot is. A fő célunk azonban nem a pontos besorolás volt, hanem egy olyan algoritmus létrehozása, ami egy lemmajelöltről el tudja dönteni, hogy a hozzárendelt szófajjal együtt helyes-e vagy sem. Ezért a rendszerek teljesítményét olyan formában is kiértékeltük, hogy azt nem vettük figyelembe, hogy a `badana` és a `notlem` osztályokat meg tudják-e különböztetni. Így az RNN 80,53%, míg az SVM 92,13% pontossággal tudta megítélni, hogy egy lemmajelölt helyes-e vagy sem. Mindkét kiértékelés során látható tehát, hogy beágyazási vektorokat használó SVM felülmúlta a kizárólag a karaktersorozat alapján dolgozó RNN alapú rendszert. Bár a szóvégződés igen meghatározó tényező, az utóbbinak azonban semmilyen azzal kapcsolatos információ nem állt rendelkezésére, hogy az adott szó milyen kontextusokban szokott megjelenni.

	Pontosság (3 osztályra)	Pontosság (2 osztályra)
RNN	74,37%	80,53%
SVM	87,86%	92,13%

2. táblázat. A két rendszer pontossága 3, illetve 2 osztályra vizsgálva (`lem` vs. {`notlem`|`badana`})

Az eredmények részletesebb vizsgálata során a konkrét tévesztéseket is számszerűsítettük, amit a 3. táblázatban összesítettünk. Látható, hogy az RNN sokkal több szót sorolt a `badana` osztályba, ezeknek egy jelentős része (31,64%) azonban valójában vagy szófajilag helyesen osztályozott ragozott szó, vagy helyes lemma. Az SVN azonban – bár alacsonyabb fedéssel találta el ennek az osztálynak az elemeit – csupán 5 olyan szót sorolt ebbe az osztályba, amik valójában helyes lemmák az adott szófajcímkével (az 5 példa: *sziasztok[MSZ], Salaam[FN], szí[IGE], jobbul[IGE], Karmelina[FN]*). Elmondható tehát, hogy az SVM alapú megoldás igen pontos, amelyik lemmajelöltet hibásnak jelzi, az nagy valószínűséggel tényleg hibás. A `notlem` és `lem` osztályok közötti tévesztések számát tekintve is sokkal kisebb a hibák száma az SVM esetén. Azoknak a lemmajelölteknek, amik az SVM alapú rendszer szerint nem lemmák, 97,8%-a valóban nem az. Sajnos az SVM-es rendszer által jó lemmaként azonosított jelöltek pontossága alacsonyabb: ez csak 83,3%. A két rendszert összevetve csupán a `badana` osztály fedését tekintve mutat az RNN valamivel jobb teljesítményt, az ebbe az osztályba tartozó tesztesetekből az SVM többet sorolt a másik két osztály valamelyikébe. A két rendszer kombinálásával feltehetőleg javítani lehetne a hibásként azonosított lemmák fedését a pontosság jelentős romlása nélkül.

Az eredmények vizsgálata során az is látható volt, hogy sok esetben a hibás lemmatizálás forrása helyesírási hiba, elírás volt. Ezeket a szavakat, amiknek így a lemmája is hibásan szerepelt, a `badana` osztályba soroltuk mind a tanításnál, mind a kiértékelésnél akkor is, ha az eredeti szó helyes alakja esetén jó lenne az elemzés a helyes lemmával. Például a *butít* szó rövid *i*-s *butit* alakja a *butit[IGE]* elemzéssel szerepel, ami egyébként helyes lenne a hosszú *í*-s változat esetén. A létrehozott algoritmus ezeket az alakokat gyakran helyes lemmának és

RNN		eredmény		
		badana	notlem	lem
cél	badana	713	93	156
	notlem	90	840	70
	lem	240	113	659

SVM	eredmény		
	badana	notlem	lem
badana	646	127	189
notlem	0	994	6
lem	5	34	973

3. táblázat. A két rendszer tévesztési mátrixa a vizsgált tesztanyagon mérve

elemzésnek tekinti, ami tulajdonképpen zajos szövegek feldolgozása esetén akár elfogadható eredmény is lehet.

4. Összegzés

Cikkünkben bemutattunk két olyan algoritmust, amik annak eldöntésére képesek, hogy egy szóalakhoz automatikusan létrejött lemma az eredeti szóalakhoz rendelt szófajcímkével együtt helyes-e vagy sem. A rekurrens neurális hálózatot, illetve a support vector machine módszert megvalósító rendszerek közül az utóbbi teljesítménye lett kiemelkedően jó, 92% fölötti pontosságot érve el. Az algoritmusban használt modell betanításához egy automatikusan létrehozott és kézzel ellenőrzött tanítóanyagban szereplő szavaknak két különböző jellegű, magyar nyelvű szóbeágyazási modellből származó reprezentációját, illetve a szó eredeti szófaját használtuk fel. Bár a bemutatott algoritmus nem képes arra, hogy a hibásnak ítélt lemmák esetén javaslatot tegyen a helyes alakra, a szófaji egyértelműsítő rendszerbe való integrálásával akár a helyes lemma előállítása is lehetséges lenne. A beépített lemmatizáló ugyanis a morfológiai elemző által nem ismert szavakhoz több lehetséges lemmát is generál, ezek közül választva ki azt, amelyikhez a legnagyobb valószínűséget rendelte. A bemutatott rendszer azonban alkalmas lehetne arra, hogy az ebben a listában esetleg nagyobb valószínűséggel szereplő, de nem helyes lemmákat elvesse, így a kisebb „valószínűségű", de helyes lemmák is megjelenhetnének az eredményekben.

Köszönetnyilvánítás

Jelen kutatás az FK 125217 és a PD 125216 számú projekt keretében a Nemzeti Kutatási Fejlesztési és Innovációs Alapból biztosított támogatással az FK 17 és a PD 17 pályázati program finanszírozásában valósult meg.

Hivatkozások

1. Orosz, Gy., Novák, A.: PurePos 2.0: a hybrid tool for morphological disambiguation. In: Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2013), Hissar, Bulgaria, Incoma Ltd. Shoumen, Bulgaria (2013) 539–545

2. Brants, T.: Tnt: A statistical part-of-speech tagger. In: Proceedings of the Sixth Conference on Applied Natural Language Processing. ANLC '00, Stroudsburg, PA, USA, Association for Computational Linguistics (2000) 224–231
3. Endrédy, I., Prószéky, G.: A pázmány korpusz. Nyelvtudományi Közlemények **112** (2016) 191–206
4. Novák, A.: Milyen a jó Humor? In: I. Magyar Számítógépes Nyelvészeti Konferencia, Szeged, SZTE (2003) 138–144
5. 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. ACL '99, Stroudsburg, PA, USA, Association for Computational Linguistics (1999) 261–268
6. Novák, A.: Improving corpus annotation quality using word embedding models. Polibits **53** (2016) 49–53
7. Mikolov, T., Yih, W., Zweig, G.: Linguistic regularities in continuous space word representations. In: Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics, Proceedings, June 9-14, 2013, Westin Peachtree Plaza Hotel, Atlanta, Georgia, USA. (2013) 746–751
8. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States. (2013) 3111–3119
9. Novák, A., Novák, B.: Magyar szóbeágyazási modellek kézi kiértékelése. In: XIV. Magyar Számítógépes Nyelvészeti Konferencia, Szeged, SZTE (2018)