Kísérletek tudásbázis- és mondatkörnyezet-alapú beágyazásokkal magyar nyelvre

Kardos Péter[1], Berend Gábor[1,2], Farkas Richárd[1]

[1]Szegedi Tudományegyetem, Informatikai Intézet
Szeged, Árpád tér 2.
[2]MTA-SZTE Mesterséges Intelligencia Kutatócsoport
Szeged, Tisza Lajos körút 103.
Kardos.Peter@stud.u-szeged.hu, {berendg,rfarkas}@inf.u-szeged.hu

Kivonat Napjainkban a szavak jelentésének folytonos vektortérbeli leírására a mondatkörnyezet-alapú neurális megoldások a legelterjedtebbek. Ebben a cikkben gráfalapú beágyazások használatával kísérletezünk, amelyek a gráf csúcsait (például WordNet synsetek) ágyazzák be vektortérbe, szomszédaik alapján. Javaslunk egy újszerű módszert is, mellyel kombinálhatók ezen beágyazások, majd a beágyazásokat a magyar WordNetből általunk készített szóasszociációs feladaton teszteljük.

1. Bevezetés

Hogyan is tudjuk megtanítani a gépeknek, hogy a cipő az egy ruhadarab és a lábunkon hordjuk, nem pedig a fejünkön? Több tudásbázis létezik, melyek megpróbálják a szavak jelentését leírni, szinonimák, hipernímák és más szemantikai kapcsolatok használatával. Ezeknek nagy hátránya, hogy kézzel építettek, nem teljesek, hiányoznak újabb szavak. Másik megoldás lexikai szemantikai ábrázolásra a statisztikai szemantika (distributional semantics), aminek napjainkban legelterjedtebb tagja a szóbeágyazások (*word embedding*) használata. A szóbeágyazó módszerek a szavakat vektortérbe ágyazzák be a szavak mondatkörnyezetének felhasználásával, innen is ered megnevezésük. Ezt úgy teszik, hogy a hasonló fogalmak közel kerüljenek egymáshoz. Ezek rengeteg fontos feladathoz segítséget nyújtanak, mint például gépi fordítás, kérdésmegválaszolás, szófaji kódsorozat meghatározás, ami miatt széles körű felhasználásnak örvendenek. A mondatkörnyezeten alapuló reprezentációk is rendelkeznek természetesen hiányosságokkal, amelyek között megemlíthetjük a reprezentációk egyes dimenzióinak interpretációjának korlátjait [1] vagy a folytonos szóreprezentációk azon tulajdonságát, hogy az ellentétes jelentésű szavakhoz gyakran hasonló reprezentáció társul. A szóreprezentációk tudásbázisokkal való kombinálása alkalmas lehet mindkét előbb említett probléma mérséklésére.

A szóbeágyazások létrehozásának eddig legelterjedtebb formája a szövegekből (*mondatkörnyezet alapú*) való generálás volt, de manapság sorra jelennek meg a gráfokat beágyazó algoritmusok. Ezen gráfalapú algoritmusok a gráf csúcsait ágyazzák be vektortérbe, szomszédaik alapján. A tudásbázisok felfoghatók egy

gráfként, melynek csúcsai a szavak, címkézett élei pedig a szavak közötti kapcsolatok. Munkánkban a magyar WordNetből készítettünk ilyen beágyazásokat a diff2Vec [2] algoritmussal. Majd a kapott beágyazásokat összehasonlítottuk a mondatkörnyezet alapú beágyazásokkal.

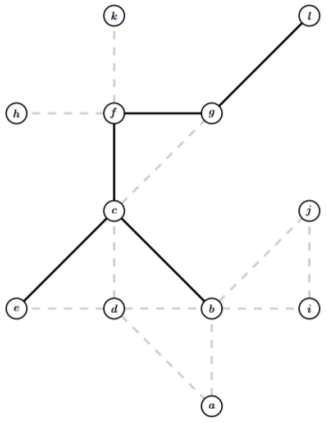
Javaslunk egy újszerű módszert is, mellyel kombinálhatók ezen beágyazások, majd a kombinációkat is összevettük a már előtanított beágyazásokkal. A beágyazásokat a magyar WordNetből általunk készített feladaton teszteltük.

2. Kapcsolódó munkák

2.1. Gráfbeágyazások

A gráfbeágyazó algoritmusok megjelenését a szóbeágyazó algoritmusokkal elért sikerek ihlették meg. Ezen algoritmusok egy gráf csúcsait (V) képzik le d dimenziós lineáris térbe, úgy, hogy az reprezentálja a gráfban lévő távolságokat. Vagyis ezek egy $V \mapsto \mathbb{R}^{|V| \times d}$ leképezést hajtanak végre, ahol $d \ll |V|$. A legtöbb ilyen algoritmus a skip-gram/CBOW beágyazó megoldásra épít, de előtte át kell alakítanunk a gráfot ezzel kompatibilis formába.

A legismertebb gráfbeágyazó algoritmusok a node2Vec [3] és diff2Vec [2], amelyek a csúcsok környezetét adott hosszúságú és számú csúcsból induló bejárással írják le. Ezek a környezet már a mondathoz hasonló szekvenciaként is felfoghatóak. Ezen utakból adott ablak méret mellett, együttes előfordulás mátrixot készítenek a módszerek. Ezek lesznek majd a neurális háló kimenetei, a bemenet pedig a csúcs one-hot vektora. Így már tudjuk tanítani a hálót a skip-gram vagy CBOW módszerrel. A node2vec és diff2vec csak a környezet kinyerésében tér el egymástól. Mivel előzetes kísérleteink azt mutatták, hogy a diff2vec rendre jobb eredményeket ér el és robusztusabb, ezért ebben a munkában csak azt használjuk.



1. ábra. Diffúziós gráf példa. Forrás: [2]

A diff2Vec [2], mint Diffusion to Vector úgynevezett szétterjedés gráfokat hoz létre a következő módon. Először inicializáljuk a $\widetilde{G}$ diffúziós gráfot, melyben az egyetlen csúcs az aktuálisan vizsgált csúcs. Majd minden iterációban a még nem bevett szomszédos élekből véletlenszerűen választunk egyet és a csúccsal együtt bevesszük a $\widetilde{G}$ gráfba. Ezt addig ismételjük amíg a paraméterként kapott p darab csúcs nem lesz a gráfban, vagy már nem tudunk újat bevenni. Most egy útvonalat kell generálnunk ebből a részgráfból. Ezt úgy tesszük, hogy minden élet megduplázunk, így biztos lesz benne Euler séta, amit könnyen meg is tudunk találni. Az Euler sétának megvan az a jó tulajdonsága, hogy megismer minden szomszédsági kapcsolatot a részgráfban. Ezután már tudunk ezekből a csúcs sorozatokból skip-gram/CBOW módszer használatával beágyazásokat tanítani.

A diff2vec a következő paraméterekkel rendelkezik:

- n: Generált szétterjedés gráfok száma csúcsonként.
- p: Szétterjedés gráfokbeli csúcsok száma.
- d: Beágyazás dimenziója.
- $\hat{w}$: Ablak mérete.

A neurális hálók tanításánál a legfontosabb epoch és learning rate paramétereket állítottuk. Az algoritmus implementációja elérhető a `https://github.com/benedekrozemberczki/diff2vec` oldalon.

2.2. Magyar nyelvű beágyazási eredmények

Magyar nyelvű szóbeágyazok minőségével kapcsolatban több korábbi munka is napvilágot látott. A [4] cikk magyar nyelvű szóbeágyazások minőségét a [5] által javasolt szóanalógiás feladat magyarra adaptált verzióján, illetve nyelvek közötti fordítási feladatokon értékelte ki. A [6] szerzői csupán elvétve találtak olyan analógiapéldákat, melyekre helyes eredményt kaptak volna, így a szóbeágyazások minőségének ellenőrzésére a szemantikai csoportok hierarchikus klaszterezéssel történő automatikus kinyerését választották. A magyar nyelvű szó– és karakter szintű információt integráló szóreprezentációk hatékonyságát [7] téma– és véleményosztályozási feladatokba építve mutatta meg. A [8] munka különböző szóbeágyazási modellek minőségét hasonlította össze annotátorok segítségével. Az annotátorok feladata az volt, hogy a különböző szóbeágyazási modelleket aszerint rangsorolják, hogy azok mennyire homogén legközelebbi szóhalmazt adnak vissza bizonyos hívószavak tekintetében.

3. Mondatkörnyezet- és gráfalapú beágyazások kombinálása

Ebben a munkánkban fő célunk, hogy a szavak mondatkörnyezet- és gráfalapú beágyazásait összehasonlítsuk és kombinálással kiaknázzuk azok előnyeit.

A gráfbeágyazó algoritmusokhoz bemenetként a magyar Wordnetet használtuk fel. Az általunk használt algoritmusok nem veszik figyelembe az élek címkéit,

vagyis a szavak közötti kapcsolatok típusát, viszont ezt nem hagyhatjuk figyelmen kívül, hiszen ezek is információval láthatnak el bennünket. Ezért az adatbázist a kapcsolatok szerint több kisebb részre szedtük, majd ezekre külön-külön futtattuk az algoritmusokat. Az így kapott beágyazásokat a magyar WordNetből generált kiértékelő adatbázison teszteltük.

Ha több különböző beágyazást egyszerre szeretnénk használni, akkor találnunk kell egy módszert azok kombinálására. Alább javaslunk, és tesztelünk egy módszert különböző beágyazások kombinálására.

3.1. Beágyazások kiértékelése

A kiértékelő adatbázishoz a magyar WordNetből [9] két gráfot készítettünk, egyet a szinonima, egyet a hiperníma kapcsolatokból. Mondatkörnyezet beágyazásként a Szeged_fasttext-et [7] használjuk. Vettük azon szavak listáját melyek mindkét gráfban és az Szeged_fasttext-ben is szerepelnek. Ezekből a közös szavakból gyűjtöttünk szópárokat melyek 1,2 vagy 3 távolságra (legrövidebb út) helyezkednek el egymástól és mindegyik távolságból véletlenszerűen választottunk 100-100 darab szópárt. A két gráfból külön-külön generáltuk ezeket, így összesen 600 kiértékelő szópárt kaptunk, ahol a távolságok inverzét vettük az asszociáció erősségének. A következőkben ezen a 600 páron hasonlítjuk össze a tudásbázis- és mondatkörnyezet-alapú beágyazásokat úgy, hogy az egyes szópárokra kiszámoljuk a két vektor koszinusztávolságát, majd az egész beágyazás jóságának mértéke a 600 koszinusztávolság és a WordNet szópár távolság inverzének Spearman korrelációja lesz.

Out-of-vocabulary (OOV) A modellből hiányzó, de a teszt adatokban előforduló szavak kezelésére három eljárást is megvalósítottunk.

1. **Zero** - Egyszerűen azt mondjuk, hogy ezen példáról semmit nem tudunk mondani, így 0 a hasonlóságértékük.
2. **Skip** - Egyszerűen átugorjuk az adott példát. Itt problémát jelent, hogy a különböző beágyazásokban más-más példák hiányoznak, eltérő mennyiségben, emiatt az ezzel kapott korrelációk nem hasonlíthatók össze. Ellenben ez egy jó módszer annak tesztelésére, hogy a modell által ismert szavak közötti kapcsolatot mennyire jól tudja.
3. **Fallback** - Előfordulhat, hogy csak a szó végén álló toldalék miatt nem ismeri fel a szót. Innen jött az ötlet, hogy akkor levágjuk mindig az utolsó betűt, amíg nem kapunk egy olyan szót, ami szerepel a modellben. Itt kérdéses, hogy mit is kéne tenni, ha elfogynak a betűink. Ebben az esetben átugorjuk a példát.

Miután a 600 szópárt kigyűjtöttük a kiértékelő adatbázisba, a gráfból töröltük az egyes szópárokat összekötő összes legrövidebb út összes élét, hiszen célunk a tudásbázisban nem szereplő asszociációk predikálhatóságának vizsgálata.

3.2. Beágyazások kombinációja

A legkézenfekvőbb eljárás beágyazások kombinálásra az, ha egyszerűen konkatenáljuk a szavak vektorait. A beágyazásokban szereplő szavak száma viszont eltérő, így előfordulhat, hogy az egyik beágyazásban szereplő szót nem találjuk a másikban. Ezért a beágyazások vektorait úgy konkatenáljuk, hogy amennyiben valamelyik beágyazásban nem szerepel egy szó, akkor annak helyét nullákkal töltjük fel.

A következő lépésben információtömörítést alkalmazunk, ez lehet SVD vagy PCA. Ennek a motivációja az, hogy redundanciát csökkenthetjük, valamint az erősebb dimenziók jobban fogják meghatározni a beágyazást, mellyel általánosságban javulást érünk el. Ez a művelet nagy mátrix esetén elég költséges mind időigény mind pedig memóriahasználat szempontjából. Az erőforrásigények csökkentése érdekében csak az adatsorok 10%-át vettük a legfontosabb dimenziók meghatározására.

Végül a kapott beágyazás oszlopait L1 normalizáljuk, melyet a [10] cikkben mutatott sikerek miatt teszünk. A módszerek paramétereinek finomhangolását a 3.5 fejezetben mutatjuk be. A következőkben a kombinálást a konkatenálás, majd tömörítést és normalizálást mutatjuk be részletesen.

3.3. Diff2Vec beágyazások

Ebben a fejezetben a WordNet szinonima és hiperníma éleiből készítünk beágyazást. Ezt úgy tesszük, hogy előfeldolgozásként a gráfból kivesszük a teszt adatbázisban megtalálható szópárok közötti útvonalat, mely alapján azok létre lettek hozva. Emiatt létrejöttek olyan csúcsok, melyeknek egyetlen élük sincsen és ezeket a beágyazó algoritmus nem tudja beágyazni, vagyis ezek OOV szavak lesznek. Ezeket a fenti lehetőségek közül mindegyik megoldással leteszteltük. A Diff2Vec algoritmushoz a következő paramétereket használtuk.

- dimenziók - 100
- Szétterjedés gráfok száma csúcsonként - 10
- Szétterjedés gráfbeli csúcsok száma - 40
- ablak méret - 10
- learning rate - 0.025
- epoch - 4

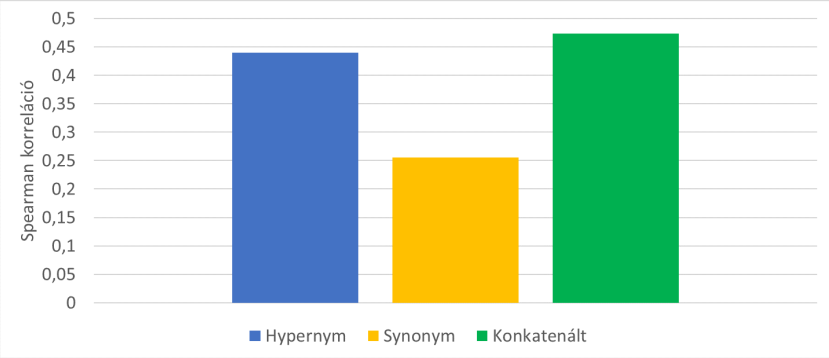
	Zero		**Skip**		**Fallback**	
	Hypernym	Synonym	Hypernym	Synonym	Hypernym	Synonym
Spearman	0.439526	0.254951	0.586998	0.300968	0.526991	0.294253
OOV	109	59	109	59	35	25

1. táblázat. Hypernym és synonym élek felhasználásával épített gráfbeágyazók eredményei különféle OOV-kezelési stratégiák alkalmazása mellett.

3.4. Konkatenálás

Már tudjuk a beágyazások hasznosságát, de kérdés mennyire szerepelnek jól együtt. A konkatenálással kapott beágyazások már többféle kapcsolat típus információját is magában foglalják, így remélhetőleg ez jó irányba befolyásolja a teljesítményt. Ebben a fejezetben ezt teszteljük.

A külön-külön számított gráfbeágyazási vektorok konkatenálásának eredményét a 2. ábra mutatja. A WordNet Hypernym és Synonym éleiből kapott beágyazásokat konkatenálva ez már egy 200 dimenziós beágyazás lesz. A kapott beágyazás 15 darab OOV példát tartalmaz. A kiértékelésnél Zero megoldást használtunk a kezelésükre. A konkatenálás után kapott beágyazás jobban meg tudta oldani a feladatot, mint ezek külön. Ebből következik, hogy a beágyazások konkatenálása hasznos az egyes beágyazások által megtanult információ egyetlen beágyazásban való reprezentálására. A továbbiakban a paraméterek beállítására,illetve tesztelésére ezt a beágyazást fogjuk használni.



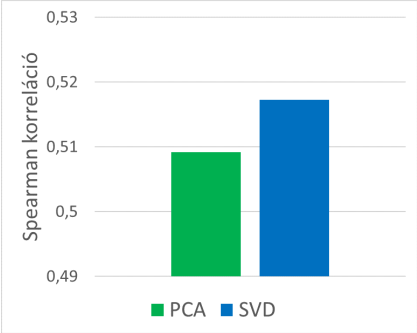
2. ábra. Diff2Vec beágyazások konkatenálásának Spearman korrelációs értékei

3.5. Beágyazások utó- és finomhangolása

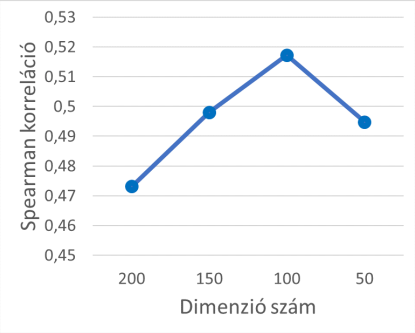
Ahhoz, hogy a kombináló módszerből a lehető legjobbat hozzuk ki minden paramétert tesztelünk, hogy is éri meg beállítani azokat. Ehhez az előző fejezetben kapott konkatenált beágyazást használjuk.

Információtömörítés Az információtömörítés segíthet a beágyazásainkon azzal, hogy elhagyjuk az információt nem hordozó dimenziókat. Két lehetőségünk is van információtömörítésre, ezek a PCA és SVD. A teszteléshez 100 dimenziós beágyazást gyártottunk. Ebből a kísérletből az derül ki, hogy a kombináló módszer csak kis mértékben függ attól, hogy PCA-t vagy SVD-t használunk, melyek közül az SVD minimálisan jobb (3. ábra).

Jó kérdés lehet, hogy mennyi dimenzióra érdemes lecsökkenteni a beágyazásokat. A konkatenált beágyazásból kiindulva 50 lépésközzel legyártottunk tömörített beágyazásokat. Ezt a 4. ábra foglalja össze, melyből az derül ki, hogy 100 dimenzióig monoton nőnek az eredmények. Ebből arra következtettünk, hogy a kiindulási dimenzió feléig biztonságos a redukálás a javulás szempontjából.



3. ábra. SVD és PCA összehasonlítása.



4. ábra. Dimenziócsökkentés eredményei 50-es lépésközzel.

Normalizálás A különböző beágyazásoknál könnyen meglehet, hogy azok nem ugyanazon az értéktartományon mozognak, emiatt van szükség erre a lépésre. Ennél a kísérletnél az teszteltük javít-e a beágyazás oszlopainak normalizálása. Ehhez L1 normalizációt használtunk, majd összevetettük ezen lépés kihagyásával kapott beágyazásokat (5. ábra). A normalizálás szembetűnően sokat javított, így a továbbiakban ezt mindig elvégezzük.

A dimenziócsökkentés előtti normalizálással kapott eredményeket is megnéztük, mellyel kapcsolatos eredményeinket a 2. táblázatban foglaljuk össze. A normalizálás SVD előtti elvégzése csekély javulást eredményezett a beágyazásokon, így a továbbiakban ezt kihagytuk.

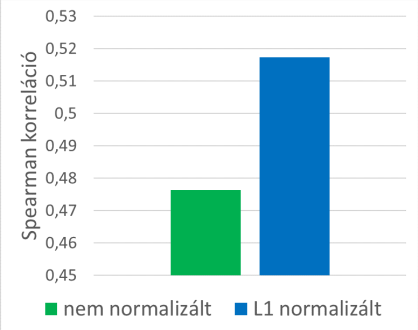
Beágyazás	**Dim**	**WordNet**
Synonym + Hypernym + Szte_fasttext	300	0.52778
Normalizálás nélkül	200	0.52846
SVD előtti normalizálás	200	0.53007
SVD utáni normalizálás	200	**0.59193**

2. táblázat. Normalizálás összehasonlítása.

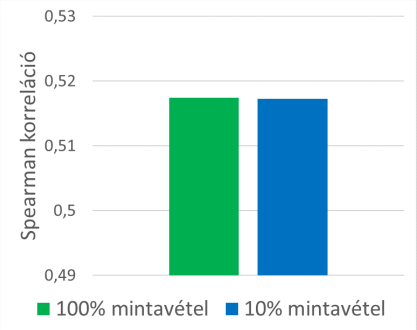
Mintavételezés A dimenziócsökkentő algoritmus nagy dimenzió- és szószám esetén erősen számításigényes. Ahhoz, hogy ezen javítsunk megnéztük mi törté-

nik, ha az adott szóhalmaz csak egy paraméterben kapott százalékát használja fel a leghasznosabb dimenziók meghatározására.

Ezt az összes szó felhasználásával kapott beágyazás és a szavak 10%-ának használatával kapott beágyazás összevetésével teszteltük le. A 6. ábrán látható, hogy ez egyáltalán nem befolyásolja a beágyazások minőségét, viszont ezzel csökkenthetjük az erőforrás igényt.



5. ábra. Beágyazás normalizálásának hatása.



6. ábra. Mintavételezés hatása.

Végső paraméterek A fent elvégzett tesztek alapján a paraméterek a továbbiakban ennek megfelelően fogjuk beállítani:

- Dimenzió csökkentés: SVD
- Mintavétel: 10%
- Dimenzió szám: kiindulási dimenziók feléhez közelítve
- Normalizáció: L1

3.6. Mondatkörnyezet- és gráfalapú beágyazások kapcsolata

A kombináló módszer paramétereinek finomhangolása után, ebben a fejezetben összehasonlítjuk a tudásbázis- és mondatkörnyezet alapú beágyazásokat. A továbbiakban a paraméterek tesztelésére használt él kombinációt fogjuk összevetni más beágyazásokkal.

A Szeged_fasttext [7] egy magyar előtanított szóbeágyazási modell 100 dimenziós vektorokkal. Ezt a beágyazást összehasonlítva az előzőekben tárgyalttal láthatjuk, hogy a gráfalapú modell nem sokkal teljesít rosszabbul. Ha a kettőt konkatenáljuk egy minimális romlás látható, de a kombinálás minden lépésének elvégzése után már sokkal jobban teljesít ezen a feladaton.

Felmerülhet a kérdés mennyire befolyásolja a gráfbeágyazásokat az, hogy voltak OOV szavak. Ezt úgy orvosoltuk, hogy kiszedtük a teszt halmazból azokat a sorokat melyek tartalmaztak OOV-t valamelyik beágyazásban. Az eredmények a 4. táblázatban látottak szerint alakultak:

Beágyazás	Dim	WordNet
Synonym + Hypernym	200	0.47308
Szeged_fasttext	100	0.53936
Synonym + Hypernym + Szeged_fasttext	300	0.52778
Előző tömörítve	200	**0.59193**

3. táblázat. Szeged_fasttext és a gráfbeágyazások kombinálása.

Beágyazás	Dim	WordNet
Synonym + Hypernym	200	0.51540
Szte_fasttext	100	0.56918
Synonym + Hypernym + Szte_fasttext	300	0.55714
Synonym + Hypernym + Szte_fasttext	200	**0.61213**

4. táblázat. Szeged_fasttext és a gráfbeágyazások kombinálása OOV nélkül.

4. Konklúzió

Munkánkban összevetettük a manapság már egyre elterjedtebb gráfokat beágyazó algoritmusokat, a szavakat leíró tudásbázisokból vett kapcsolatokból kiindulva. Általunk vizsgált algoritmus a diff2Vec volt, melyhez a magyar WordNet által leírt szókapcsolatokat vettük kiindulási gráfnak. A beágyazások minőségének kiértékelését a WordNetből generált feladaton végeztük. A WordNet különböző éleit beágyaztuk a diff2Vec algoritmus segítségével, így kiderült mely élek milyen jól tudják ezen feladatokat megoldani. Ezek kombinálására készítettünk egy módszert, mely lehetővé teszi az ezek által megtanult információ egyetlen beágyazásba kombinálását. A módszer a különböző beágyazások konkatenálása után információtömörítést és normalizálást használ.

Mondatkörnyezet alapú előtanított beágyazásokkal (Szeged_fasttext) is összevetettük a csupán tudásbázisokból kapott beágyazások teljesítményét. A gráfalapú beágyazások kombinálása nem sokkal maradt el az előtanított beágyazással szemben. A mondatkörnyezet- és gráfalapú beágyazások kombinálva javítottak egymás eredményein, így kijelenthetjük, hogy a kombináló módszer hasznos különböző beágyazások összeolvasztásában.

A jövőben az olyan úgynevezett heterogén gráfalapú beágyazó algoritmusokkal tervezzük folytatni a kísérletezést, melyek már az élek címkéit is képesek figyelembe venni.

Köszönetnyilvánítás

Kardos Péter munkáját az "Integrált kutatói utánpótlás-képzési program az informatika és számítástudomány diszciplináris területein" című, EFOP-3.6.3-VEKOP-16-2017-0002 számú projekt támogatta. A projekt az Európai Unió támogatásával, az Európai Szociális Alap társfinanszírozásával valósul meg. Berend Gábor munkáját a Nemzeti Kutatási, Fejlesztési és Innovációs Hivatal Mester-

séges Intelligencia Nemzeti Kiválósági Programja támogatta a 2018-1.2.1-NKP-2018-00008 azonosítójú projekt keretében.

Hivatkozások

1. Faruqui, M., Dyer, C.: Non-distributional word vector representations. In: Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers), Association for Computational Linguistics (2015) 464–469
2. Rozemberczki, B., Sarkar, R.: Fast sequence based embedding with diffusion graphs. In: International Conference on Complex Networks. (2018)
3. Grover, A., Leskovec, J.: node2vec: Scalable feature learning for networks. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. (2016)
4. Makrai, M.: Comparison of distributed language models on medium-resourced languages. In Tanács, A., Varga, V., Vincze, V., eds.: XI. Magyar Számítógépes Nyelvészeti Konferencia (MSZNY 2015). (2015)
5. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. CoRR **abs/1301.3781** (2013)
6. Siklósi, B., Novák, A.: Beágyázási modellek alkalmazása lexikai kategorizációs feladatokra. In Tanács, A., Varga, V., Vincze, V., eds.: XII. Magyar Számítógépes Nyelvészeti Konferencia (MSZNY 2016), Szeged, Szegedi Tudományegyetem, Szegedi Tudományegyetem (2016) 3–14
7. Szántó, Z., Vincze, V., Farkas, R.: Magyar nyelvű szó-és karakterszintű szóbeágyazások. In Vincze, V., ed.: XII. Magyar Számítógépes Nyelvészeti Konferencia (MSZNY 2018), Szeged, Szegedi Tudományegyetem, Szegedi Tudományegyetem (2017) 323–328
8. Novák, A., Novák, B.: Magyar szóbeágyázási modellek kézi kiértékelése. In Vincze, V., ed.: XII. Magyar Számítógépes Nyelvészeti Konferencia (MSZNY 2018), Szeged, Szegedi Tudományegyetem, Szegedi Tudományegyetem (2018) 67–77
9. Miháltz, M., Hatvani, C., Kuti, J., Szarvas, G., Csirik, J., Prószéky, G., Váradi, T.: Methods and results of the hungarian wordnet project. In: Proceedings of The Fourth Global WordNet Conference. (2008) 311–321
10. Speer, R., Lowry-Duda, J.: Conceptnet at semeval-2017 task 2: Extending word embeddings with multilingual relational knowledge. CoRR **abs/1704.03560** (2017)