Eötvös Loránd Tudományegyetem

Társadalomtudományi Kar

Szóbeágyazási vektorterek illesztési problémájának megoldása Prokrusztész transzformációval

Különböző politikai szemléletű online médiumokból származó korpuszokon

2023

Konzulens:

Rakovics Márton

Készítette:

Csaba Enikő

OQE0YC

Survey statisztika és adatanalitika

TARTALOMJEGYZÉK

TÁBLÁZATJEGYZÉK

ÁBRAJEGYZÉK

ABSZTRAKT

Dolgozatomban kísérletet tettem két, eltérő társadalomszemléletű online hírportál cikkeiből létrehozott korpusz összehasonlítására szóbeágyazási vektorterek összeillesztésével, annak érdekében, hogy definiáljam a különböző kontextusból eredő eltéréseket. Emellett dolgozatom további célja annak meghatározása, mennyire alkalmas eszköz a Prokrusztész transzformáció a vektorreprezentációk közös térbe való illesztésére. Különböző szóbeágyazások tanításával először kiválasztottam a feladatra legalkalmasabb modellt, majd végrehajtottam és kiértékeltem a Prokrusztész-transzformációkat. A legalacsonyabb közelítési hibával rendelkező transzformáció kijelölése után pedig sor került az összeillesztett vektortér elemzésére. Az eredmények egyrészről alátámasztják, hogy a Prokrusztész transzformáció alkalmas a beágyazások eltéréséből fakadó illesztési probléma kezelésére, másrészről azonosít téma-specifikus szavakat, melyek eltérő kontextusban jelennek meg a két médiumban.

Kulcsszavak: természetes-nyelvfeldolgozás, szóbeágyazás, Word2Vec, FastText, Skip-Gram, CBOW-modell, nyelvi-modell, neurálisháló, Prokrusztész-transzformáció, vektortér-modellek

1. BEVEZETÉS

1.1. A téma jelentősége, a témaválasztás indoklása

A média elévülhetetlen módon alakítja azt, ahogy a politikai környezetet érzékeljük, hiszen maga is véleményt formál, ezzel egyfajta értékhierarchiát adva a politikai és társadalmi eseményeknek. Ezáltal a médiumokból származó szövegek, és az ebben megjelenő szavak kontextusának elemzése hozzájárulhat ahhoz, hogy feltárjuk a nyelvhasználat és a média politikai szerepének összetettségét, illetve azt, hogy ezek a tényezők miképpen alakítják a közös valóságképünket.

A közös valóságkép kialakításához azonban több, különböző politikai szemléletű médium is hozzájárul, így az ebből fakadó különbségek feltárása segíthet annak megértésében, hogy az azonos szavak eltérő használata miként formálja a köznyelvet.

Ennélfogva dolgozatom fő célja két, eltérő társadalomszemléletű online hírportál cikkeiből létrehozott korpusz összehasonlítása szóbeágyazási vektorterek összeillesztésével. A célból fakadó elsődleges kutatási kérdésem az, hogy az eltérő korpuszokon tanított szóbeágyazásokban jelen lévő vektorreprezentációk mennyiben feleltethetők meg egymásnak, illetve milyen eltérések azonosíthatók a különböző kontextusból eredően.

Mindemellett másodlagos kérdésként megfogalmazható, hogy a szóbeágyazásokkal kialakított vektorreprezentációk közös térbe való illesztésére mennyire alkalmas eszköz a Prokrusztész transzformáció.

Dolgozatom elméleti hátterének kialakításához, illetve a modellek Pythonban történő felépítéséhez három szakirodalom járult hozzá a leginkább; Mikolov és társai 2013-ban megjelent cikke, mely a Word2Vec modelljeit mutatja be, Bojanowski és csapata 2016-es FastText-et ábrázoló tanulmánya, illetve Rong (2016) szóbeágyazási modellek matematikai hátterét bemutató értekezése.

1.2. A dolgozat felépítésének ismertetése

A dolgozat felépítését az alábbiakban szemléltetem; az elméleti háttér fejezetben kifejtem a legfontosabb, általam használt definíciókat, illetve levezetem, hogy a természetes nyelvfeldolgozáshoz elengedhetetlen szóreprezentáció miként vezetett a szóbeágyazási modellek kifejlődéséhez. A különböző beágyazások ismertetése után kitérek arra, hogy az

illesztési probléma miképpen vezet Prokrusztész transzformáció által nyújtott megoldáshoz, és részletezem ennek, az általam felhasznált típusait.

A harmadik fejezet a saját kutatásomat szemlélteti; először az adatelőkészítés és – tisztítás folyamatát mutatom be, majd részletesen bemutatásra kerülnek a szóbeágyazási modellek építéseinek fázisai. Ezek után a vektorterek illesztésének megvalósítását szemléltetem, végül a közös vektortér elemzésére kerül sor.

2. ELMÉLETI HÁTTÉR

2.1. Természetes nyelvfeldolgozás

Az információs technológia és a digitális tartalmak robbanásszerű fejlődése alapvető változásokat hozott a társadalomtudományok kutatási paradigmájában, hiszen az online platformokon megjelenő szöveges adatok mennyiségének növekedése mellett a számítógépek feldolgozási kapacitása is exponenciálisan emelkedett. Ez utóbbi nem csak a tárhely kibővülését, de a sebesség fejlődését is jelentette. Illetve a különböző programnyelvek egyre szélesebb körű felhasználása is hozzájárult ahhoz, hogy számos módszer, mely addig nehezen megvalósíthatónak tűnt, ismertebbé váljon (Kmetty 2022).

Ugyanakkor az új lehetőségek új kihívásokat is jelentettek, mivel a fent említett szöveges tartalmak többnyire strukturálatlan adatok formájában vannak jelen az online térben. Míg a strukturált adatok hagyományos sorokban és oszlopokban rögzítettek, addig a strukturálatlanok nem követnek kötelezően egyetlen formai, sorrendi előírást vagy szabályt sem (Németh, Katona, és Kmetty 2020).

Ezek a strukturálatlan, szöveges tartalmak gazdag forrásai az információnak, hiszen „elképzelhetetlen mennyiségben tartalmaznak adatokat az emberek véleményéről, preferenciáiról, attitűdjeiről, sőt cselekvéseiről is” (Németh, Katona, és Kmetty 2020, 45. oldal).

Tehát a szövegekbe ágyazva mértéktelen információ található arról, hogy a társadalomban miképpen változnak az érdeklődési tendenciák, szokások és az ezekre adott reakciók. Emellett az is fellelhető, hogy az adott reakciók nyelvi mintázata és szóhasználata hogyan alakul az idők során. Azonkívül számos – akár laikusok, akár professzionális szereplők által előállított tartalomról van szó – szöveg alkalmas arra, hogy detektálja az adott időszakban jelenlévő

legégetőbb társadalmi problémákat is (Evans és Aceves 2016). Illetve érdemes még megemlíteni a digitális szöveges tartalmak egy alfaját, az online médiából érkező tartalmakat, melyek elemzése segíthet feltárni a média szerepét és befolyását a társadalom információs rfolyamataiban.

A digitális szöveges tartalmakkal való munka azonban rendkívül komplex, mivel a szöveges jellegükből adódóan eltérő feldolgozási és elemzési módszereket igényelnek. Ez a kvantitatív társadalomtudományok területén különösen izgalmas, új akadályokat jelentett, hiszen a korábbi évtizedek kutatásai rendszerint strukturált adatokra támaszkodtak. Erre a problémára adott választ a természetesnyelv-feldolgozás (Natural Language Processing – NLP) módszere is, amely több tudomány határterületén található, hiszen magába foglalja az informatikát, a mesterségesintelligencia-kutatást és a nyelvészetet is. Az NLP fő célja olyan eljárások fejlesztése, amelyek lehetővé teszik a nagy mennyiségű, emberi nyelven íródott szövegek elemzését, a kívánt információ célzott kinyerését, valamint nyelvi tartalmak létrehozását (Németh, Katona, és Kmetty 2020).

Számos iparág használja a természetes nyelvfeldolgozást az adatok értelmezésére és különböző kihívások kezelésére. Sőt, az NLP-alkalmazások a mindennapi életünkben is megjelennek, ilyen példa a Google Fordító, a Google Asszisztens, de akár az Amazon Alexája vagy a Microsoft Cortana-ja is (Omar és mtsai. 2022).

Az NLP területén tehát jelentős mértékű fejlődés zajlott az elmúlt évtizedekben és zajlik ma is. A természetesnyelv-feldolgozás egy ilyen innovációja volt a szavak reprezentációjának paradigmaváltása is, melynek során egy elmozdulás tapasztalható a hagyományos, gyakorisági alapú szemlélettől új, kontextus-alapú megközelítésig.

2.2. Szavak reprezentációja

A szavak reprezentációja kulcsfontosságú a természetes nyelvfeldolgozás során, mivel ez teszi lehetővé, hogy a gépek megértsék az emberi beszédet és kölcsönhatásba lépjenek vele. Ezek a reprezentációk olyan matematikai modelleket alkotnak, melyek gyakran vektorokban vagy mátrixokban fejezik ki az egyes szavak jelentését. A modellek egyrészről segítenek a gépeknek a természetes nyelvek nyelvstruktúrájának és szemantikájának jobb megértésében, másrészről hatékonyabbá teszik az olyan nyelvi feladatok végrehajtását, mint például a gépi fordítás.

Ennélfogva a következőkben arra keresem a választ, hogy milyen módszerek léteznek arra, hogy a szövegeinket úgy alakítsuk át, hogy azok gépi modellek számára is értelmezhetőek legyenek.

2.2.1. One-hot kódolás

Az egyik legegyszerűbb megoldás, ha létrehozunk egy szótárat, mely a szöveg egyedi szavait tartalmazza. Minden szót egy vektor reprezentál, mely vektor hossza a szótár mérete (N), tehát a vektorok $1 \times N$-es méretűek. A k-adik egyedi szó k-adik eleme 1-es értéket kap, míg a többi érték helyén 0-s szerepel (Turian, Ratinov, és Bengio 2010). Tehát ha a szótárat az egyedi szavak vektoraiként reprezentáljuk $N \times N$-es mátrixot kapunk, amely azonban sok egyedi szóval rendelkező szöveg esetén igen jelentős méreteket ölt. Ennek következtében növekszik a memóriaigény és a számítási komplexitás. Ugyanakkor a one-hot reprezentációk ritka mátrixokat eredményeznek, hiszen az elemek túlnyomó része 0, így lényegesen kevesebb tárhelyet igényelnek, mint az azonos méretű, sűrűbb mátrixok.

A jelentős probléma inkább abban mutatkozik meg, hogy az egyes vektorok önmagukban nem sok információt szolgáltatnak a szótár méretén és a szótáron belüli pozíciójukon kívül. Tehát a reprezentációk nem tartalmaznak szemantikus információt a szavak közötti hasonlóságról vagy kapcsolatról. Ez hátrányos lehet olyan NLP feladatoknál, ahol a szavak szemantikájának megértése kulcsfontosságú, például szinonimák, hasonló értelmű kifejezések vagy szövegazonosítás területén.

Azaz a modell egyszerűsége, könnyen használhatósága nem jelent kárpótlást a hiányosságaival szemben. Így érdemesebb a komplexebb modelleket is megvizsgálni.

2.2.2. Gyakoriság alapú megközelítések

Mivel jelen dolgozat problémája kontextus-alapú, így a modellek megfigyelése során az egyik legmérvadóbb szempont, hogy a szavak reprezentációja tartalmazzon szemantikai információt. A másik alapvetés, amelyből kiindulok, hogy két dokumentum közötti hasonlóság mértéke közvetlenül összefügg azzal, hogy mennyire közös a szókészletük (Tikk 2007). Ezen logika mentén a hasonló jelentésű szavak hasonló kontextusban fedezhetőek fel, mely feltevés a nyelvészetben a disztribúciós hipotézisként jelenik meg (Harris 1954).

Fontos kiemelni, hogy jelen dolgozat nem a dokumentumok reprezentációjára fókuszál, hanem több dokumentumból összeálló szövegben fellelhető szavak kontextusát kívánja

vizsgálni. Ennélfogva az, hogy mely szó mely dokumentumban szerepel elhanyagolható információ.

Ugyanakkor a gyakoriság alapú megközelítések bizonyos típusához elengedhetetlen, hogy a szövegünket dokumentumgyűjteményként értelmezzük. A dokumentumgyűjtemény szavai által alkotott vektortérben minden tengely egy-egy szót jelöl, és a dokumentumok a szavakból álló vektorokként jelennek meg, ezért bizonyos gyakoriság alapú módszerek vektortérmodellekként váltak ismertté.

A legegyszerűbb ilyen vektortérmodellezési módszer a szó-dokumentum (term-document TD) mátrix. Ahol a $D = \{d_1, d_2, \ldots\ldots, d_N\}$ dokumentumgyűjteményt a TD mátrixszal reprezentáljuk, ahol M darab egyedi szó van – mely a vektortér dimenziója -, így tehát a D mátrix $M \times N$-es. A dokumentumokat a mátrix oszlopai reprezentálják, míg az egyedi szavak a mátrix soraiban vannak ábrázolva. Azokban a sorokban találhatók nem nulla értékek, amelyek azon pozíciókat jelölik, ahol a dokumentumokban a szó relevanciája nem nulla (Tikk 2007). Ebből kifolyólag a D mátrix rendkívül ritka (hasonlóan a one-hot kódolás esetén kapott mátrixhoz), hiszen egy dokumentumban az összes egyedi szó közül csak igen kevés fordul elő.

A TD mátrix esetén elmondható, hogy két dokumentum hasonló, ha a hozzájuk tartozó oszlopvektorok hasonlóak. Eszerint két szó akkor hasonló, amennyiben közel azonos dokumentumokban fordul elő. Ez nyilván attól is függ, hogy miket jelölnek a dokumentumok, melyek egészen hosszú (pl.: cikkek, blogbejegyzések), vagy egészen rövid (pl.: egy mondat) szövegek is lehetnek, de összességében elmondható, hogy nem a legalkalmasabb módszer a TD mátrix a szavak hasonlóságának vizsgálatára.

A szó-dokumentum mátrix másik jelentős problémája, hogy teljesen figyelmen kívül hagyja a szavak szövegen belüli pozícióját. Az ilyen modelleket szózsák modelleknek nevezi a szakirodalom (Tikk 2007).

Mindent összevetve tehát a TD módszer egyszerű és könnyen érthető reprezentációját biztosítja mind a szavaknak, mind a dokumentumoknak. Alkalmas továbbá a dokumentumok közötti hasonlóságok meghatározására, ezáltal azok csoportosítására. Ugyanakkor nagyon ritka vektorokat eredményez hosszú dokumentumok esetén, illetve nem célszerű szemantikai információ azonosítására, hiszen csak a szavak jelenlétét vagy hiányát rögzíti.

A TD mátrix fentebb említett problémájára jelent megoldást a szó-kontextus (Word-Context) mátrix (Turney és Pantel 2010). Általánosságban szó-kontextus mátrix esetén a kontextust szavak, kifejezések, mondatok, bekezdések, fejezetek, dokumentumok vagy egzotikusabb lehetőségek, például karaktersorozatok vagy minták adják. A TD mátrixszal ellentétben az oszlopvektorok ebben az esetben nem dokumentumok, hanem választott szövegkörnyezetek, melyek gyakorlatilag a kontextust kívánják meghatározni. Ez egy általunk választott c hiperparaméter, amely azt adja meg, hogy az adott szó mekkora környezetét vizsgáljuk ablakszerűen. Tehát van egy F mátrixunk, ami $N \times N$-es (tehát ismét N darab egyedi szavunk van a korpuszban), f_{ij} érték megmutatja, hogy az adott i szó hányszor fordul elő j meghatározott c nagyságú kontextusban. Amennyiben a c értékét a dokumentum nagyságára állítjuk, visszakapjuk a term-document mátrixot.

Tehát a szó-kontextus mátrix már hordozza különböző szavak szemantikai hasonlóságát, ugyanakkor egy magas dimenziójú, a TD mátrixnál jóval sűrűbb mátrixot kapunk, melynek nagy a memóriaigénye. Továbbá ebben az esetben már igen nagy változásokat hozhat a korpusz előfeldolgozásának minősége, hiszen például a stopszavak szűrése változtat a kontextuson. A másik probléma, hogy ebben az esetben is szimpla szó-előfordulásról beszélünk, holott a szövegek tartalmi jellemzését illetően a szavak jelentősége eltérő, így nem minden szó egyenlően fontos.

A következő gyakoriság alapú modell már arra a problémára fog megoldást kínálni, hogy miképpen tudunk nagyobb jelentőséget adni a „specifikusabb" szavaknak, amelyek alkalmasabbak a korpuszunk leírására.

A Term Frequency – Inverse Document Frequency (TF-IDF) súlyozási eljárás esetében a TF tag egy adott szó előfordulásának gyakoriságát méri egy dokumentumban. Minél többször szerepel egy szó egy dokumentumban, annál magasabb lesz a TF-értéke, tehát ez a tag gyakoriság alapján értékeli a szó fontosságát a dokumentumban. Ezzel szemben az IDF rész egy adott szó előfordulásának gyakoriságát méri. Minél ritkábban fordul elő egy szó az összes dokumentumban, annál magasabb lesz az IDF-értéke.

Tikk (2007) alapján amennyiben az n_{ki} a t_k szó előfordulásainak a száma a d_i dokumentumban logaritmikus súlyfüggvényt használva a term-frequency súly értéke:

$$w_{ki} = \begin{cases} 1 + \log n_{ki}\,, ha\ n_{ki} > 0 \\ 0\ egyébként \end{cases}$$

$$tf(t_{ki}) = \frac{w_{ki}}{\sum_{k=1}^{M} w_{ki}}$$

Míg az Inverse Document Frequency az alábbiképpen számítható ki, amennyiben n_k-val jelöljük azon dokumentumok számát, amelyben a t_k szó előfordul:

$$df = \frac{n_k}{N}$$

$$idf(t_k) = \log \frac{N}{n_k}$$

Így a súlyozás megkapható:

$$tf - idf\left(t_{ki)}\right) = tf(t_{ki}) \cdot idf(t_k)$$

Ennélfogva egy adott kifejezéshez (szűkebb esetben szóhoz) rendelt súlynak az értéke abban az esetben lesz nagy, ha egy adott dokumentumban gyakran, viszont a teljes szövegben ritkábban található meg. Viszont, ha egy bizonyos dokumentumban ritkán, vagy a korpusz többi dokumentumában többször szerepel, akkor az érték alacsonyabb lesz. És akár a nullás súlyt is felveheti, ha szinte minden dokumentumban gyakran előfordul.

Tehát a TF-IDF súlyozás egyrészről kiemeli a fontos szavakat a korpuszban, és képes arra, hogy megkülönböztesse a szavak tartalmi jelentőségét egymástól, viszont szemantikai értelemben ez a modell is gyengén teljesít, hiszen a szavak kontextusát figyelmen kívül hagyja.

Ahogy az a korábbi módszertanoknál világossá vált, a vektortérmodellt alkalmazó modellek egyik nagy problémája a vektortér magas dimenziószáma. Erre jelent egyfajta megoldást a különböző dimenzióredukciós módszerek használata.

Tikk (2007) mentén a szövegelemzésben használt dimenzióredukciós modellek közül a jellemzőkinyerő módszerek egyik változatát fogom bemutatni. Ezen módszerek a redukciós halmaz létrehozására szintetikus jellemzőket használnak, vagyis olyanokat, melyek az eredeti halmazban nem találhatóak meg. A jellemzőkinyerő módszereken belül találhatóak meg a jellemzők csoportosításán alapuló modellek, melyek a szavakat szemantikai kapcsolat alapján csoportokba rendezik, és a csoportokból egy-egy elemet kiválasztva jön létre az új dimenzió.

Ilyen módszer a látens szemantikai indexelés, amely szinguláris értékfelbontáson alapszik. Tételezzük fel, hogy rendelkezünk egy $D \in \mathbb{R}^{M \times N}$ mátrixszal, ahol M az egyedi szavak száma,

N pedig a dokumentumok száma. A mátrix szinguláris értékfelbontását megkapjuk az alábbi módon:

$$D = U\Sigma V^T$$

Ahol $U \in \mathbb{R}^{M\times R}$ és $V \in \mathbb{R}^{N\times R}$, mely mátrixok egymásra ortogonálisok, és a $\Sigma \in \mathbb{R}^{R\times R}$ diagonális mátrix tartalmazza a sajátértékeket. Ha mind a három mátrixot átrendezzük úgy, hogy csökkenő sorrendben tartalmazzák a sajátértékeket, ekkor a Σ mátrix 0, vagy 0-hoz közeli sajátértékei a mátrix jobb alsó részmátrixába kerülnek és amennyiben elhagyjuk őket, elérhető a dimenzióredukció. A redukált mátrix így tehát képes lesz kiszűrni a zajt, és azokra a szavakra koncentrálni, melyek tartalmilag fontosabbak. Illetve arra is alkalmas az LSI, hogy két szót összehasonlítson (ez a korábban felsorolt módszerek közül csak a szó-kontextus mátrixban volt lehetséges), hiszen csak meg kell vizsgálnunk a dimenziócsökkentett $U\Sigma$ megfelelő sorait.

Az LSI tehát egyrészről már kevésbé tárhelyigényes, hiszen csökkentettük az eredeti gyakorisági szó-dokumentum mátrixunkat, illetve segít a hasonló jelentésű szavak azonosításában, feltárható általa a rejtett szemantikai szerkezet a szövegekben (Evangelopoulos 2013). Másrészről megjelenik az előfeldolgozási lépések minőségének a problémája, ahogy azt korábban a szó-kontextus mátrix esetén is láttuk. Továbbá az LSI nem alkalmas olyan feladatokra, ahol a dokumentumok gyakran változnak, mivel újra kell tanítani a modelleket, és ez meglehetősen számításigényes.

Mindent összegezve a gyakoriság alapú megközelítések általában egyszerűek és könnyen megvalósíthatók, nem igénylik a bonyolult matematikai vagy gépi tanulási algoritmusok alkalmazását. Emellett könnyen értelmezhető a kimenetük, mely különösen igaz a fent említett TD, TF-IDF vagy szó-kontextus mátrixokra. Az ilyen modellek általában alacsony számítási erőforrásokat igényelnek, így gyorsan feldolgozhatók akár nagy korpuszokon is.

Ugyanakkor a gyakoriságon alapuló módszereknek megvannak a maguk szemantikai korlátai, ez alól többé-kevésbé kivételt jelent a szó-kontextus mátrix és az LSI. Illetve azt is érdemes kiemelni, hogy főleg a TD és TF-IDF esetén nincs mód a szavak összehasonlítására, vagy legalábbis nincs túlzott jelentősége ezeknek. Ezek a módszerek inkább dokumentumok összehasonlítására alkalmasak. Tehát olyan feladatok esetén, amikor ugyan egy dokumentumgyűjteményből dolgozunk még sincs különösebb jelentősége annak, hogy adott

szavak milyen dokumentumokban vannak jelen, inkább a globális kontextusra vagyunk kíváncsiak, ezek a modellek többnyire eredménytelenek.

De a gyakorisági modellek legnagyobb hiányossága, hogy figyelmen kívül hagyják a szavak sorrendjét. Ezen problémára fognak megoldást kínálni a következő fejezetben bemutatott predikcióra épülő modellek.

2.3. Predikciós eljárások – szóbeágyazási modellek

Tehát a klasszikus, gyakoriságon alapuló eljárások nem tudták kellő eredménnyel megragadni a szavak szintaktikai és szemantikai értelmét, illetve gyakran szenvedtek a nagy dimenzionalitás átkától. Ahogy azt láttuk az LSI esetében, a szemantikai probléma kiküszöbölésére az első jelentősebb megoldást az együttes szó-előfordulási mátrixok jelentették, ahol szinguláris érték-felbontás adta a dimenziócsökkentéshez az alapot. Ezen, az úgynevezett disztribúciós szemantikus, modellek esetén azonban problémát jelentett, hogy a már dimenziócsökkentett kimeneti vektortér igen ferde volt. A probléma tehát az, hogy a dimenziócsökkentés során az LSI vagy más szemantikus modell esetében a gyakori szavak reprezentációi közel kerülnek egymáshoz, míg a ritkább szavaké távolabb maradnak. Ez azért akadály, mert a gyakori szavak közötti kapcsolatok túlsúlyban vannak, míg a ritkább szavak közötti kapcsolatok elvesznek, ami nehezítheti a modell szemantikus pontosságát és alkalmazhatóságát, hiszen a ritkább szavak szerepe és jelentősége kevésbé lesz hangsúlyos a reprezentációkban.

Bengio és munkatársai (2003) a gyakoriságon alapuló mátrix dimenziócsökkentése helyett egy másfajta megközelítést javasoltak, neurális hálókat használtak a szavak kontextusának becslésére. Ezek a neurálisháló alapú nyelvi modellek az előző n-1-edik szavakat/tokeneket használják arra, hogy megbecsüljék az n-edik szót/tokent (Naseem és mtsai. 2021). Tehát e módszerek képesek szavakat prediktálni a kontextus alapján anélkül, hogy a szöveg szemantikai vagy szintaktikai információkat veszítene, vagyis mind a szavak szövegben elfoglalt helyét, mind a szavak jelentését képesek megfogni, így természetesen könnyebben összehasonlíthatóvá válnak.

Ilyen neurálisháló alapú nyelvi modell a szóbeágyazási modell, amely Wang és munkatársai (2019) szerint a szavak valós értékű vektoros reprezentációja. Ez pedig nagy, címkézetlen korpuszokból nyert szemantikai és szintaktikai jelentések beágyazásával jön létre. Míg

Naseem és társai (2021) szerint a szóbeágyazás egy olyan módszer, ahol egy adott szótárból (itt szótár alatt a korpuszban fellelhető egyedi szavakat érti) vett szót N dimenziós vektorra képezünk le. Kmetty (2022) pedig inkább a módszer dimenziócsökkentő jellegére koncentrál, amely azt a célt szolgálja, hogy meghatározzuk az adott szó kontextusát, a redukcióra pedig azért van szükség, mert a szavak többfajta környezetben fordulnak elő, így a szógyakoriság önmagában nem ad képet a kontextusról.

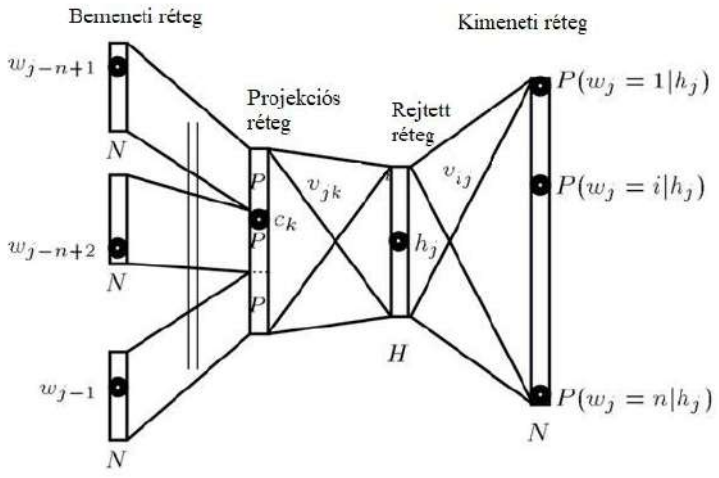
A gyakorlatban a szóbeágyazások minden egyes szóhoz egy többdimenziós, folytonos vektort rendelnek. Ezek a vektorok egy szemantikus teret alkotnak, ahol hasonló jelentésű szavak vektorai közel találhatók egymáshoz (Nemeskey 2020).

2.3.1. Az első neurálisháló alapú nyelvi modellek

A Bengio és munkatársai (2003) által javasolt első NNLM (neural network language model) olyan modellarchitektúra volt, ahol egy lineáris projekciós réteggel és egy nemlineáris rejtett réteggel rendelkező előrecsatolt neurális háló egyszerre tanulja a szóvektor-reprezentációt és a nyelvi modellt.

Ez az architektúra úgy épül fel, hogy az input rétegbe (N) érkező n-t megelőző tokenek már kódolt formában szerepelnek, méghozzá akképpen, hogy minden token V méretű vektor – ahol a V a szótár méretét jelöli -, amelynek minden eleme nulla, kivéve a token szótárbeli sorrendjének megfelelő eleme, mely 1 értéket kap. Ezek után az input réteget a projekciós rétegre (P) vetítjük, amely $N \times D$-s dimenzionalitással rendelkezik, ahol a D a közös vetítési mátrix.

1. Ábra: NNLM modell felépítése



(Popov 2016 mentén saját szerkesztés)

Az NNLM a projekciós és a rejtett réteg között válik igazán számításigényes feladattá, mivel a projekciós réteg már igen sűrű. Ha például N=10-es értékkel indulunk – ami egy viszonylag gyakori választás – a projekciós réteg mérete 500 és 2000 között lehet, míg a rejtett réteg (H) 500 és 1000 közötti méretet vesz fel. Illetve a rejtett réteget a szótár összes szava feletti valószínűségi eloszlás kiszámítására használjuk, ami olyan kimeneti réteget eredményez, melynek dimenziója V. Tehát a paraméterek száma minden tanítási etap esetén:

$$Q = N \times D + N \times D \times H + H \times V,$$

Ahol a domináló rész a $H \times V$. Ennek elkerülésére azonban több gyakorlati megoldást is javasoltak; például a softmax hierarchikus változatait használják, így a szótár bináris fa reprezentációjával a kiértékelendő kimeneti egységek száma $\log_2 V$ körüli értékre csökkenhet. Ezek után a bonyolultság nagy részét az $N \times D \times H$ kifejezés okozza.

Az előrecsatolt NNLM problémáinak kiküszöbölésére javasolták a rekurrens neurális háló alapú nyelvi modellt, amely nem rendelkezik projekciós réteggel. Az ilyen típusú modell különlegessége a rekurrens mátrix, amely a rejtett réteget önmagával köti össze, így rendelkezik egyfajta rövidtávú memóriával. Az RNN (recurrent neural network) modell számításigénye ezáltal az alábbi:

$$Q = H \times H + H \times V$$

Itt is az az eset áll fent, hogy a $H \times V$ kifejezés hatékonyan csökkenthető hierarchikus softmax alkalmazásával (Mikolov és mtsai. 2013).

Habár kifejezetten innovatívak voltak ezek a modellek, viszont számításigényük megakadályozta őket abban, hogy igazán elterjedjenek.

2.3.2. Word2Vec

A szóbeágyazási modellek elterjedése Mikolov és társai (2013) nevéhez fűződik, akik két egyszerűbb modellt javasoltak, melyek egyszerűségük ellenére mégis jobban teljesítettek. Az architektúrák fő célja az volt, hogy minimalizálják a számításigényt. Ahogy láttuk a korábbi NNLM-ek esetén, mind az előrecsatolt, mind pedig a rekurrens esetben a számítási komplexitás jelentős részéért a nemlineáris rejtett réteg felelt, így ezt a réteget mind a két architektúra esetén eltávolították.

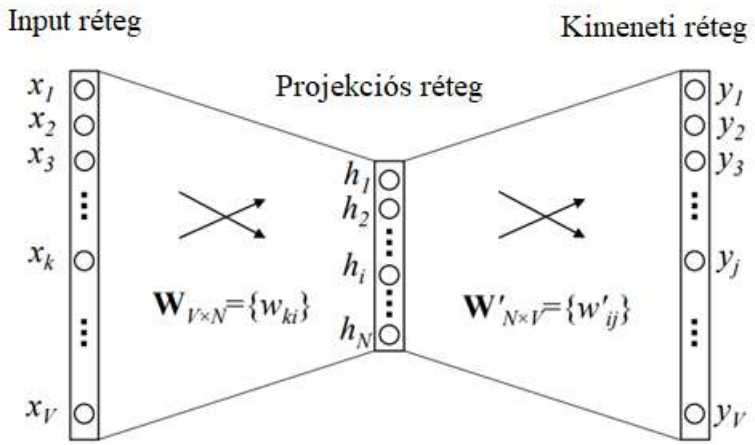
Először az egyszerűbb architektúrát mutatom be, a folytonos szózsák modellt (Continuous Bag of Words, CBOW). Ebben a modellben a cél az, hogy a környező szavak alapján becslést

készítsünk egy adott szóra. A nevét onnan kapta, hogy a projekciós réteg minden szóra közös, az adott szó környezetében lévő szavak ugyanarra a pozícióra vetülnek, mégpedig úgy, hogy a szavak vektorait átlagoljuk, ennélfogva nem számít a szavak sorrendje. Szintén új megközelítést jelentett az is, hogy nem csak a prediktálni kívánt szó előtti szavakat vizsgáljuk, hanem az utániakat is.

2.3.2.1. Folytonos szózsák modell egyetlen kontextusszóval

Rong (2016) mentén az egyszerűbb magyarázat érdekében a modell felépítését azzal a kikötéssel mutatom be, hogy a kontextusban csupán egyetlen szó szerepel. Így a CBOW egy kontextus szó alapján fogja prediktálni a célszót.

2. Ábra: CBOW modell egyetlen kontextusszó esetén



(Rong 2016 alapján saját szerkesztés)

Tételezzük fel, hogy a szótár mérete V, a projekciós réteg pedig N. A bemenet egy one-hot kódolt vektor, esetünkben az adott kontextusszó esetén V vektor $\{x_1, x_2, \dots, x_V\}$, melynek egyetlen értéke 1, a többi 0. A bemeneti és kimeneti réteg közötti súlyokat jelölje a W mátrix, melynek dimenziója $V \times N$. A W minden sora az N dimenziós vektorreprezentációja a bemeneti réteg szavának (v_w). Jelöljük az i-edik sorát a W-nek v_w^T. Adott kontextus esetén, feltételezve, hogy x_k=1 és $x_{k'} = 0$, amennyiben $k' \neq k$:

$$h = W^T x = W^T_{(k,\cdot)} \coloneqq v^T_{wI},$$

amivel gyakorlatilag a W k-adik sorát másoljuk h-ra. v_{wI} tehát a kapott vektorreprezentáció. Ahogy a fenti egyenletből kivehető az aktivációs függvény ebben az esetben lineáris.

A projekciós rétegből a kimeneti rétegbe való eljutáshoz már egy másik súlymátrixot használunk, $W'_{N\times V} = \{w'_{ij}\}$. Ezeket a súlyokat használva megkaphatjuk az $u_j$értéket minden szótárbeli szó esetén:

$$u_j = v'^T_{wj} h,$$

Ahol v'_{wj} a j-edik oszlopa a W'-nek. Ezek után softmax-ot alkalmazunk, amely egy log-lineáris osztályozási modell:

$$p(w_j|w_I) = y_j = \frac{\exp(u_j)}{\sum_{j'=1}^{V} \exp(u_{j'})},$$

ahol y_j a j-edik eleme a kimeneti rétegnek. Visszahelyettesítve a korábbi egyenleteket a fenti egyenletbe:

$$p(w_j|w_I) = \frac{\exp(v'^T_{wj} v_{wI})}{\sum_{j'=1}^{V} \exp(v'^T_{wj'} v_{wI})}$$

v_w és v'_w gyakorlatilag két különböző vektorreprezentációja w szónak. Míg v_w (input vektor) a W soraiból származik, amely az input → projekciós réteg súlymátrixa, addig v'_w (output vektor) a W' oszlopaiból kapható, mely a projekciós → kimeneti réteg mátrixa.

Tehát ahogy láttuk, megkapható a modell paramétereinek ismeretében adott bemeneti kontextus esetén a szóbeágyazás által becsült kimeneti érték. A célunk az, hogy minél közelebb jussunk a kívánt kimeneti értékhez a becsléssel. Werbos (1974) azzal forradalmasította a neurális hálókat, hogy talált egy hatékony algoritmust, a hibavisszaterjesztést, amely segítségével a paraméterek becslése elvégezhető. Ennek értelmében a háló függvények kompozíciójaként felírható, így a paraméterei is megbecsülhetők gradiens módszerrel, ha előállítjuk a paraméterenkénti parciális deriváltjait a veszteségfüggvények (amely az output és a valódi érték eltérését adja meg).

Ez a hibavisszaterjesztés esetünkben (egy kontextus szóval) az alábbi módon néz ki a CBOW modell esetén: a célunk a feltételes valószínűség maximalizálás a súlyok figyelembevételével, ahol a valószínűség a kimeneti szó w_O előfordulása a bemeneti kontextus szó w_I alapján.

$$\max p(w_O|w_I) = \max y_{j*} = \max \log y_{j*} = u_{j*} - \log \sum_{j'=1}^{V} \exp(u_{j'}) := -E,$$

Ahol E a kereszt-entrópia veszteségfüggvény egy speciális esete, amit minimalizálni szeretnénk, és a $j*$ az aktuális kimeneti szó indexe az output rétegben. Így a rejtett és kimeneti rétegek közötti súlyok egyenlete – vagyis az E u_j szerinti deriváltja:

$$\frac{\partial E}{\partial u_j} = y_j - t_j := e_j,$$

Ahol t_j csak akkor lesz 1, ha a j-edik egység a tényleges kimeneti szó, egyébként t_j = 0. Ezek után a projekciós → output réteg súlyának gradiensét szeretnénk megkapni, tehát E w'_{ij} szerinti deriváltját:

$$\frac{\partial E}{\partial w'_{ij}} = \frac{\partial E}{\partial u_j} \cdot \frac{\partial u_j}{\partial w'_{ij}} = e_j \cdot h_i$$

Ennélfogva a sztochasztikus gradiens süllyedési módszer segítségével a projekciós → output réteg súlyainak egyenlete:

$$w'^{(\text{új})}_{ij} = w'^{(régi)}_{ij} - \eta \cdot e_j \cdot h_{i.},$$

Másképpen:

$$v'^{(\text{új})}_{wj} = v'^{(régi)}_{wj} - \eta \cdot e_j \cdot h, \quad \text{ahol } j = 1,2,\dots,V.$$

$\eta > 0$ a tanulóráta, e_j a kimeneti réteg predikciós hibája, h_i az i-edik eleme a projekciós rétegnek, v'_{wj} a kimeneti vektora a w_j-nek. Tehát végigmegyünk a szótár minden szaván, és ellenőrizzük, hogy a kimeneti rétegben szereplő érték megegyezik-e a valódi értékkel. Ha $y_j > t_j$, akkor a projekciós vektor (h) egy részét kivonjuk v'_{wj}-ből, tehát v'_{wj} távolabb (skaláris szorzattal számítva) lesz v_{wI}-től. Ha alulbecslünk, akkor hozzáadjuk h egy részét v'_{wj}-hez. Így tehát egyre közelebb kerül t_j y_j-hez ezzel az iteratív folyamattal.

Miután megkaptuk a W' frissített egyenletét, haladhatunk tovább az input → projekciós réteg súlyokra. Vesszük az E deriváltját a rejtett réteg kimenetén, így:

$$\frac{\partial E}{\partial h_i} = \sum_{j=1}^{V} \frac{\partial E}{\partial u_j} \cdot \frac{\partial u_j}{\partial h_i} = \sum_{j=1}^{V} e_j \cdot w'_{ij} := EH_i,$$

Ahol EH egy N-dimenziós vektor, amely szótárban lévő szavak kimeneti vektorainak összegéből áll elő, a predikciós hibával súlyozva. Ezek után megnézem E deriváltját W-ben, a projekciós réteg az input réteg és a súlyok lineáris kombinációjából áll elő ($h_i = \sum_{k=1}^{V} x_k \cdot w_{ki}$), így:

$$\frac{\partial E}{\partial w_{ki}} = \frac{\partial E}{\partial h_i} \cdot \frac{\partial h_i}{\partial w_{ki}} = EH_i \cdot x_k$$

Ez pedig egyenértékű lesz x és EH tenzorszorzatával:

$$\frac{\partial E}{\partial W} = x \otimes EH = xEH^T$$

Azonban tudjuk, hogy x-nek csupán egyetlen értéke nem zérus, így $\frac{\partial E}{\partial W}$ esetén is csak egyetlen sor kap 0-tól különböző értéket, így:

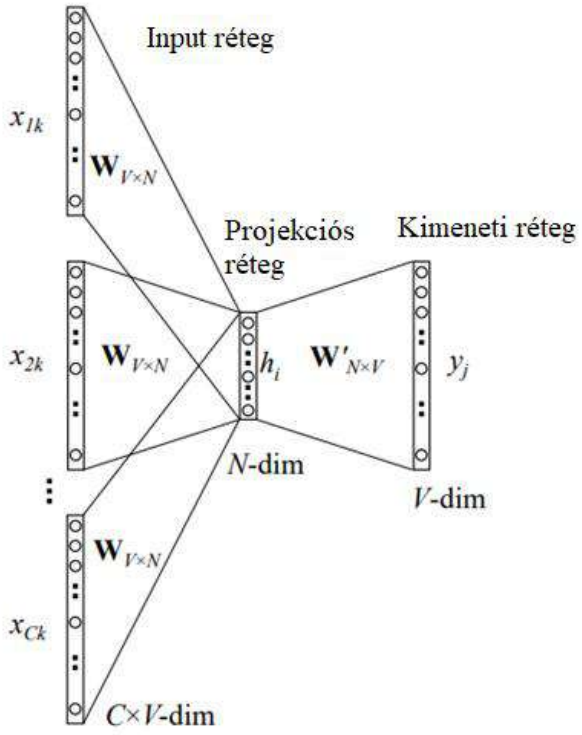
$$v_{wI}^{(új)} = v_{wI}^{(régi)} - \eta EH^T$$

ahol v_{wI} a W egy sora, az egyetlen kontextusszó bemeneti vektora, és a W egyetlen olyan sora, amelynek deriváltja nem nulla.

Az egész iteratív folyamat elképzelhető akképpen, mintha az output vektort ide-oda rángatnánk, annak meghatározásaképpen, hogy milyen súlyokkal lesz a lehető legközelebb az prediktált szó (a kontextus-szó alapján) az eredeti szóhoz képest.

2.3.2.2. Folytonos szózsák modell több kontextusszó esetén

3. Ábra: CBOW modell több, mint egy kontextusszó esetén



(Rong 2016 alapján saját szerkesztés)

Amennyiben a kontextus szélesebben értelmezett, és nem kizárólag egyetlen szó áll a rendelkezésünkre (ez szinte minden esetben hatásosabb) a kívánt szó prediktálására, a folyamat az alábbi módon változik. A projekciós réteg kiszámításakor a CBOW modell ahelyett, hogy közvetlenül a bemeneti kontextus szó vektorát másolná, az input szavak vektorainak átlagát veszi, és a bemeneti → projekciós súlymátrix és az átlagos vektor szorzatát használja kimenetként. Vagyis:

$$h = \frac{1}{C} W^T (x_1 + x_2 + \cdots + x_C) = \frac{1}{C} (v_{w1} + v_{w2} + \cdots + v_{wC})^T,$$

ahol a C a kontextus-szavak száma, $w_1, w_2, \dots, w_C$ a konkrét szavak, és $v_{w1}, v_{w2}, \dots, v_{wC}$ a szavak inputvektorai. A veszteségfüggvény pedig a következőképpen változik:

$$E = -\log p(w_o | w_{I,1}, \dots, W_{I,C})$$
$$= -u_{j*} + \log \sum_{j'=1}^{V} \exp(u_{j'}) = -v'^T_{w_O} \cdot h + \log \sum_{j'=1}^{V} \exp(v'^T_{wj} \cdot h)$$

Ezen kívül az input → projekciós súlyok gradiens számítása változik:

$$v^{(\text{új})}_{wI,C} = v^{(régi)}_{wI,C} - \frac{1}{C} \cdot \eta \cdot EH^T, ahol\ c = 1,2, \dots, C,$$

ahol$v_{wI,C}$ a c-edik kontextus-szó inputvektora.

2.3.2.3. Skip-Gram modell

A Skip-Gram architektúra esetén a célszó a bemeneti, míg a kontextusszavak a kimeneti rétegben találhatóak, tehát ez a felépítés a CBOW ellentétje.

A bemeneti réteg egy szavának vektorát jelölje ebben az esetben is w_I, a projekciós réteg elemeit ugyanúgy kapjuk meg, mint az egy kontextusszavas CBOW modell esetén (az input és projekciós réteg „közötti” súlyokat is ugyanazon elv alapján kapjuk meg). A projekciós → kimeneti réteg közötti súlyt is hasonlóan kapjuk meg, mivel mindegyik kimeneti kontextusszóra egyazon értéket használunk:

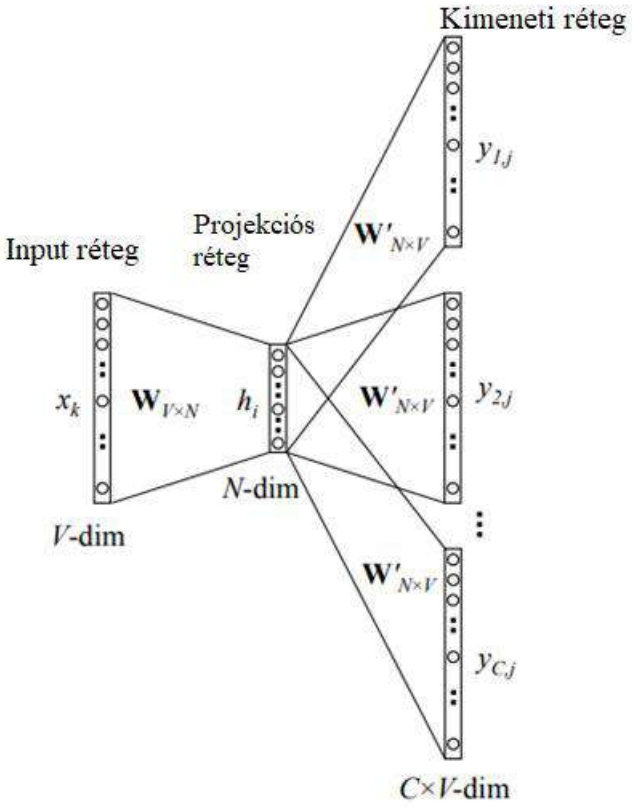
$$u_{c,j} = u_j = v'^T_{wj} h, \text{ amennyiben } c = 1, 2, \dots, C$$

Ebből számítva a kimeneti réteg elemei megkaphatóak úgy, hogy minden értékre ugyanazt a mátrixot használjuk:

$$p(w_{c,j} = w_{O,c} | w_I) = y_{c,j} = \frac{\exp(u_{c,j})}{\sum_{j'=1}^{V} \exp(u_{j'})},$$

ahol $w_{c,j}$ a j-edik szó a c-edik panelben a kimeneti rétegen, $w_{O,c}$ pedig a c-edik szó a kimeneti réteg kontextusszavai közül, w_I az egyetlen input szavunk, és az $y_{c,j}$ a kimenet.

4. Ábra: Skip-Gram modell



(Rong 2016 alapján saját szerkesztés)

A Skip-Gram esetében a hibavisszaterjesztéshez használt veszteségfüggvény az alábbiképpen változik:

$$E = -\log p(w_{O,1}, w_{O,2}, \dots, w_{O,C} | w_I) = -\log \prod_{c=1}^{C} \frac{\exp(u_{c,j*_c})}{\sum_{j'=1}^{V} \exp(u_{j'})}$$

$$= -\sum_{c=1}^{C} u_{j*_c} + C \cdot \log \sum_{j'=1}^{V} \exp(u_{j'}),$$

ahol $j*_c$ az indexe a valódi c-edik output kontextusszónak a szótárban. A rejtett és kimeneti réteg közötti súly-egyenlet – vagyis az E u_j szerinti deriváltja az alábbi

$$\frac{\partial E}{\partial u_{c,j}} = y_{c,j} - t_{c,j} := e_{c,j}.$$

A könnyebb átláthatóság érdekében a kontextusszavakra vonatkozó predikciós hibák összegét úgy definiáljuk, mint egy V dimenziós vektort az alábbi módon:

$$EI_j = \sum_{c=1}^{C} e_{c,j}.$$

Ezek után a projekciós → output réteg súlyának gradiensét vizsgáljuk, vagyis $E\ w'_{ij}$ szerinti deriváltját:

$$\frac{\partial E}{\partial w'_{ij}} = \sum_{c=1}^{C} \frac{\partial E}{\partial u_{c,j}} \cdot \frac{\partial u_{c,j}}{\partial w'_{ij}} = EI_j \cdot h_i$$

Ebből következik, hogy a projekciós → output réteg súlyainak egyenlete:

$$w'^{(\text{új})}_{ij} = w'^{(\text{régi})}_{ij} - \eta \cdot EI_j \cdot h_i,$$

Másképpen:

$$v'^{(\text{új})}_{wj} = v'^{(\text{régi})}_{wj} - \eta \cdot EI_j \cdot h, \text{ ahol } j = 1,2,\dots,V.$$

Az input → projekciós réteg közötti egyenlet majdnem teljesen megegyezik az egy kontextusszavas CBOW architektúra esetén tapasztalt módszerrel, kivéve, hogy a predikciós hiba nem e_j, hanem EI_j. Ennek értelmében:

$$v^{(\text{új})}_{wI} = v^{(\text{régi})}_{wI} - \eta EH^T \text{ , ahol } EH_i = \sum_{j=1}^{V} EI_j \cdot w'_{ij}.$$

Tehát a Skip-Gram modell nagyon hasonló felépítésű, mint a CBOW, viszont ez az architektúra már számításba veszi a szavak sorrendjét, hiszen az aktuális szó alapján jósolja meg az előzetes és utólagos szavakat (előre megadva, hogy mekkora tartományt szeretnénk kimenetként kapni).

2.3.2.4. A számításigény redukálása

A folytonos szózsák modell számításigénye $Q = N \times D + D \times log_2 V$, a Skip-Gram-é pedig $Q = C \times (D + D \times \log_2 V)$, ahol a C az általunk kiválasztott ablaknagyság, vagy távolság. Mind a két esetben jóval kisebb ez az érték, mint akár a rekurrens neurálisháló alapú nyelvi modell esetén. Ez a már korábban említett két okra vezethető vissza: a log-lineáris függvény használatára (tehát a softmaxra) és a rejtett réteg elhagyására (Mikolov és mtsai. 2013).

Ugyanakkor még ezen modellek esetén is fennáll az alábbi probléma: bár az input vektorok által meghatározott szó-reprezentációk megszerzése könnyű feladat, amely hamar véghezvihető, a kimeneti szavak tanításához jóval több idő kell, hiszen végig kell iterálni a

szótár minden szaván (és kiszámolni az ehhez tartozó súlyokat, valószínűségi eloszlásokat stb). Ez főleg akkor jelent gondot, ha jelentős méretű korpuszokkal dolgozunk, így érdemes valamiképpen redukálni az outputként kapott vektorok számát. Erre jelent megoldást egyrészről a hierarchikus softmax függvény használata (a softmax helyett), illetve a negatív mintavételezés.

A hierarchikus softmax egyfajta közelítése a teljes softmaxnak, amely számítási szempontból sokkal hatékonyabb. A fő előnye, hogy ahelyett, hogy a szótár összes szavának kimeneti csomópontjához tartozó valószínűségi eloszlást kellene meghatároznunk, csupán körülbelül $log_2 V$ csomópontot kell kiértékelni. A modell egy bináris fát használ az összes szó reprezentálására a szótárban, ahol a szavak a fa levelei. Minden egyes levélhez létezik egy egyedi út a gyökértől az egységig; ez az út szolgál a levél által képviselt szó pszeudó-valószínűségének becslésére (Rong 2016).

Amennyiben ezt a fát egy olyan döntési folyamatnak, vagy véletlen sétának tekintjük, amely a fa gyökerénél kezdődik, és minden egyes lépésben a levélcsomópontok felé halad, az lehetővé teszi, hogy minden egyes szót bináris döntési sorozatok eredményének tekintsünk (Mikolov és mtsai. 2013). Ez azért jelent segítséget, mivel így az egyes szavak becslése csak a hozzájuk vezető úton található csomópontoktól függ (nem pedig az összes szótárbeli szótól).

Az iterációnkénti frissítési feladat számításigényét kiküszöbölendő a negatív mintavételezés softmax helyett logisztikus regressziót használ célfüggvényként (ezt a korábbiakban E-vel jelöltem), illetve egy-egy iteráció alatt csak $k + 1$ darab szóhoz tartozó paramétert frissít – azaz a szavak egy mintáját. Az új célfüggvény meghatározásával a negatív mintavétel célja, hogy maximalizálja a szavak hasonlóságát ugyanabban a kontextusban, és minimalizálja azt, amikor különböző kontextusban fordulnak elő. Ahelyett azonban, hogy a minimalizálást a szótárban lévő összes szóra elvégezné, nyilván kivéve a kontextusszavakat, véletlenszerűen kiválaszt egy maroknyi szót ($2 < k < 20$) a képzési mérettől függően, és ezeket használja az optimalizálásra.

A negatív mintavételezés során olyan kontextus-centrális szópárosokat állítunk fel, melyek a kontextus szavaikban megegyeznek, de a centrálisakban eltérnek. Egy pozitív példát biztosítunk minden kontextus szó mellé, illetve k darab véletlenszerűen kiválasztott – mintavételezett - negatív példát. Tehát gyakorlatilag egy osztályozó modellt építünk (Mikolov és mtsai. 2013).

2.3.3. FastText

Azonban a word2vec modellek figyelmen kívül hagyják a morfológiai információkat, hiszen minden szóhoz külön vektort rendelnek. Példának okáért a katasztrofális és a katasztrófa szó teljesen különböző módon lesz reprezentálva. Ez különösen a jelentős szókinccsel és sok ritka szóval rendelkező nyelvek esetében jelent korlátozást – amely jelzők a magyarra abszolút igazak.

Bojanowski és társai (2016) egy új megközelítést javasoltak, amely egy olyan modellen alapul, ahol a szavakat n-grammok halmazaként reprezentálták – pontosabban az n-grammok által meghatározott vektorok összegeként. Ez azért előremutató, mivel a morfológiailag gazdag nyelvek esetén a szavak belső szerkezetének vizsgálatával a szavak vektorreprezentációja jelentős mértékben javítható, hiszen a szavak képzése gyakran szabályokat követ, ami n-grammok segítségével megtanítható a modellnek.

A FastText tehát nem egy szózsák modell, hanem egy úgynevezett karakter n-gram zsák architektúra. Annak érdekében, hogy a modellünk képes legyen az előtagok és utótagok más karaktersorozatoktól való megkülönböztetésére, a szavakat < > jelekkel határolták. A szavak reprezentációjában nem csak annak n-gramjai fedezhetőek fel, hanem maga a szó is. Például

a kevés szó a <kev, evé, vés, kevé, evés> és <kevés> n-gramok vektorreprezentációjának összegeként ábrázolható, amennyiben 3 és 5 közötti n-gramokat használunk.

Bojanowskiék Skip-Gram architektúrában (negatív mintavételezéssel) szemléltetik a FastText modell felépítését az alábbi módon. Tételezzük fel, hogy rendelkezünk egy n-gram szótárral (G). Egy w szó esetén a $G_w \in \{1, \dots, G\}$ az n-gramok halmaza. Minden g n-gramhoz hozzárendelünk egy vektorreprezentációt z_g. És minden szót ezen vektorreprezentációk összegeként ábrázolunk. Így kialakítunk egyfajta pontozási rendszert az összegzés során:

$$s(w, c) = \sum_{g \in G_w} z_g^T v_c,$$

ahol a v_c a kontextusszó vektorreprezentációja. Ez gyakorlatban úgy néz ki, hogy először kiszámoljuk a centrális szó beágyazását a karakter n-gramok és az adott szó vektorainak összegéből. Ezután a kontextusszavak reprezentációját keressük meg, amit a beágyazásból veszünk az n-gramok kihagyásával. Majd negatív mintavételezést alkalmazunk (softmax helyett), végül vesszük a centrális szó és a kontextusszavaknak (pontosabban azok

vektorreprezentációinak) a skaláris szorzatát (ezt mutatja meg a fenti egyenlet). Ezek után hibavisszaterjesztéssel frissítjük az egyenleteket – és ezzel a paramétereket. Vagyis ez a módszer egyfajta továbbfejlesztése a Word2Vec algoritmusnak.

A FastText képes arra, hogy akár a ritka szavak pozícióját is helyesen megtalálja, ha ezen szavak karakter n-gramjai más szavakban is fellelhetőek, sőt akár olyan szavakét is, amelyek a korpuszunkban nem is szerepelnek (Kmetty 2022). A későbbiekben bemutatom, hogy ez a tulajdonság a szintaktikai feladatok elvégzése során fog nagy előnyt jelenteni. Ugyanakkor a szavak szemantikai elemzésében elmarad a Word2Vec modelljeitől. Ez amiatt lehetséges, hogy a FastText nem tesz különbséget a szótő és a toldalék között. Nyilván azok az n-gramok, amelyek közelebb vannak a szótőhöz, sokkal jobban kifejezik a szavak jelentését, mint a toldalékok vagy előtagok. Choi és Lee (2020) ennek kiküszöbölésére javasolt egy metódust, amely IDF-súlyozással adna nagyobb szerepet a vektorösszegzés során a szótőhöz közelebb álló n-gramoknak.

Mindent összevetve míg a FastText inkább a szintaktikai feladatokban teljesít sikeresebben, mivel figyelembe veszi a szavak belső struktúráját, addig a Word2Vec a szemantikai feladatok megoldásában tehetségesebb. A későbbi fejezetekben egyéb praktikusabb megfontolások alapján is összehasonlításra kerül a kétféle szóbeágyazási modell (többfajta architektúrával).

2.4. Az illesztési probléma

Mivel dolgozatom fő célja két eltérő forrásból felépített korpusz összehasonlítása a kontextus szempontjából, így evidens módon felmerült a kérdés, hogy miképpen lehet szóbeágyazási vektortérmodelleket egyazon térbe illeszteni. Ezt a problémát Bojanowski, Mikolov és munkatársaik (2019) után egyfajta szóvektor-illesztési problémaként definiáltam. Ebben a tanulmányukban a szóvektor-alapú modellek új szöveges adatokhoz való adaptálásának kérdéskörével foglalkoznak; adott egy nagy referenciaadatokon előre betanított modell, hogyan igazíthatjuk azt egy kisebb, kissé eltérő nyelvi eloszlású adathoz. Ez a probléma azért analóg a jelen dolgozatban bemutatott illesztési problémával, mivel mind a két esetben kombinálni szeretnénk eltérő korpuszból származó adatokat, méghozzá úgy, hogy minimalizáljuk az információvesztést.

Az egynyelvű illesztési probléma pedig a kétnyelvű lexikon indukcióból építkezik, amely esetén két különböző nyelvű korpuszon tanult szóbeágyazási vektortérmodelleket adaptálnak

egymáshoz, így létrehozva egy többnyelvű szóbeágyazási modellt, amely aztán képes a kétnyelvű szófordítási párok létrehozására például legközelebbi szomszéd algoritmus felhasználásával (Artetxe, Labaka, és Agirre 2019).

Bojanowski és társai a szóvektor illesztési problémát az alábbi módon fogalmazzák meg: rendelkezünk két $m \times n$-s mátrixszal, melyek esetében feltételezzük, hogy az azonos szót jelölő vektorok azonos indexszel szerepelnek a mátrixokban. A megoldáshoz egy lineáris legkisebb négyzetek probléma definícióval jutunk el, ami egy kvadratikus optimalizálási problémát eredményez. A leképezési mátrix $Q \in R^{n \times n}$ megkapható a következő módon:

$$\min_{Q} \frac{1}{n} \|XQ - Y\|_F^2,$$

amely zárt formájú eredményt ad. Tehát a cél az, hogy megtaláljuk azt az egyetlen transzformációt, amely minimalizálja az összes adatpár közötti négyzetes távolságok összegét, ezáltal minimalizálva az eloszlások közötti teljes távolságot.

Amennyiben a Q-t korlátozzuk ortogonális mátrixokra a kapott probléma ortogonális Prokrusztész problémaként definiálható.

2.4.1. Prokrusztész probléma

„Prokrusztész – attikai útonálló, Szinisz apja, valódi nevén Polüpémón. ~ nevét ('kinyújtó') arról a szokásáról kapta, hogy az arra járó utasokat házába invitálta, majd ágyába fektette, s ha annál rövidebbek voltak, kegyetlenül kinyújtotta őket, a túl hosszú vendégnek pedig levágott egy darabot a lábából. Az ifjú Thészeusz Athénba vezető útján ~ kényszerítette bele a saját ágyába, és mivel túl hosszú volt hozzá, egy fejjel megkurtította.” (Arcanum Adatbázis Kiadó, é. n.)

A fent idézett történetnek három eleme van: az utazó, jelölje X, a Prokrusztész ágy -Y, illetve maga a ‘kezelés’: T. Ennélfogva a Prokrusztész probléma legegyszerűbb algebrai megoldása azt a T mátrixot keresi, ami minimalizálja az:

$$\| XT - Y \|$$

egyenletet T felett, adott X $(m \times n_1)$ és Y $(m \times n_2)$ esetén. Ezen probléma különböző változatai függnek az X, Y alakjától, illetve attól, hogy milyen megkötéseket alkalmazunk a T-re (Gower és Dijksterhuis 2004). Valójában ez egy általánosabb megfogalmazása a fenti illesztési problémának, ahol a T mátrix nem kapott ortogonális megkötést. Ebben az általános

megfogalmazásban nem feltételezzük, hogy X és Y egyaránt n darab oszlopból áll. Itt érdemes kiemelni, hogy a Prokrusztész transzformáció ezt nem is várja el, mivel nem feltétlenül ugyanazokra a változókra vonatkoznak a két mátrixban az oszlopok. Éppen ebben van a Prokrusztész egyik nagy előnye, hogy amennyiben az X és Y olyan adatmátrixok, amelyek oszlopai különböző változókra utalnak, és így akár különböző számú oszlopokkal rendelkeznek, akkor a probléma szinte egyetlen megoldása egy Prokrusztész módszer (Gower 2010).

A Prokrusztész problémát már 1962-ben megfogalmazta Hurley és Catell: megtalálni azt az ortogonális transzformációt, ami egy X mátrixot maximálisan hasonlóvá tesz Y célmátrixhoz legkisebb négyzetes értelemben. Pár évvel később Schönemann (1966) talált egy szinguláris érték felbontás alapú megoldást a problémára, amely felírható a következőképpen, ha $X \in \mathbb{R}^{m\times n}$ és $Y \in \mathbb{R}^{m\times n}$:

$$\min_{S,T} \|SXT - Y\|_F^2$$ [1]

Ahol $\|\cdot\|_F$ a Frobenius normát jelöli, azaz:

$$\|X\|_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}\left|x_{ij}\right|^2} = \sqrt{Tr(X^T X)},$$

ahol a $Tr(X)$ az X mátrix nyomát jelöli. Ahogy korábban már szó volt róla, a különböző Prokrusztész problémák különböző választási lehetőségeket használnak az S és T transzformációs mátrixok megválasztására, például ortogonális, rotációs, szimmetrikus vagy permutációs mátrixokat. Amennyiben S identitás mátrix, akkor az egyenletet egyoldalú Prokrusztész problémának nevezzük, ha pedig megegyezik T-vel kétoldalúnak.

2.4.1.1. Általános Prokrusztész

Meng és munkatársai (2022) mentén amennyiben nem alkalmazunk a transzformációs mátrixra semmilyen megkötést, általános Prokrusztész transzformációról beszélünk. Ezáltal, az eddigi jelölésekkel a mátrixkeresési probléma az alábbi, ha $X \in \mathbb{R}^{m\times n}$ és $Y \in \mathbb{R}^{m\times n}$ (tehát ők feltételezik az oszlopegyenlőséget a két mátrix között, ugyanakkor a gyakorlati

[1] Vegyük észre, hogy ez a felírás csak abban különbözik a Bojanowskiék által megfogalmazottal, hogy SVD-t (singular value decomposition) használ.

megvalósításban bemutatásra kerül, hogy ez elérhető, ha a kisebb mátrixot nullás sorokkal és/vagy oszlopokkal töltjük fel):

$$T_{opt} = \arg\min \|XT - B\|_F^2 = (X^T X)^{-1} X^T Y$$

Ha az $m < n$, akkor a transzformációs mátrix T_{opt} nem egyedi, mivel az egyenletrendszer aluldeterminált (azaz kevesebb egyenlet van, mint ismeretlen).

2.4.1.2. Ortogonális Prokrusztész

Amennyiben a T mátrixot korlátozzuk az ortogonális mátrixok halmazára, a probléma annak az ortogonális ($Q \in \mathbb{R}^{n \times n}$) mátrixnak a megkeresése, amely elforgatja és tükrözi az X-et, hogy a legjobban illeszkedjen Y-hoz. Ez elérhető a két mátrix szorzatának szinguláris érték felbontásával, $X^T Y$. Az első lépés ehhez az, hogy a mátrixok közötti távolság minimalizálását átírjuk a mátrixok átfedésének maximalizálására:

$$\arg \min_{\{Q | Q^{-1} = Q^T\}} \|XQ - Y\|_F^2 = \arg \min_{\{Q | Q^{-1} = Q^T\}} Tr[(XQ - Y)^T (XQ - Y)] =$$

$$= \arg \min_{\{Q | Q^{-1} = Q^T\}} Tr[(XQ)^T XQ - Q^T X^T Y - Y^T XQ + Y^T Y] =$$

$$= \arg \min_{\{Q | Q^{-1} = Q^T\}} Tr[-Q^T X^T Y - Y^T XQ] = \arg \max_{\{Q | Q^{-1} = Q^T\}} Tr[Q^T X^T Y]$$

Az átalakítás után a megoldás megkapható a mátrixszorzás szinguláris értékfelbontásával:

$$X^T Y = U \Sigma V^T,$$

ahol $U \in \mathbb{R}^{m \times m}$ és $V \in \mathbb{R}^{n \times n}$ unitárius mátrixok, míg $\Sigma \in \mathbb{R}^{m \times n}$ egy diagonális mátrix, melynek $r = min\{m, n\}$ nemnegatív diagonális elemei $\sigma_1 \geq \cdots \geq \sigma_r \geq 0$ az $X^T Y$ szinguláris értékei, máshol pedig nullák. Így a probléma átírható az alábbiképpen:

$$\max_{Q^{-1} = Q^T} Tr[Q^T X^T Y] = \max_{Q^{-1} = Q^T} Tr[Q^T U \Sigma V^T] = \max_{Q^{-1} = Q^T} Tr[\Sigma V^T Q^T U] =$$

$$= \max_{Q^{-1} = Q^T} Tr[\Sigma Z] = \max_{Q^{-1} = Q^T} \sum_i \Sigma_{i,i} Z_{i,i},$$

ahol $Z = V^T Q^T U$. Mivel Q ortogonális, ezért a függvény értéke akkor maximális, ha $Z = I$, vagyis az identitásmátrix. Ezért $Q_{opt} = UV^T$.

2.4.1.3. Rotációs-ortogonális Prokrusztész

Amennyiben a T-vel szembeni elvárásunk nem csak az, hogy ortogonális legyen, hanem a transzformációt rotációra korlátozzuk, abban az esetben a probléma felírható – felhasználva a fentiekben bemutatott ortogonális transzformációs levezetést -:

$$R_{opt} = \arg \min_{\left\{R \middle| \begin{matrix} R^{-1} = R^T \\ |R| = 1 \end{matrix}\right\}} \|XR - Y\|_F^2 = \arg \max_{\left\{R \middle| \begin{matrix} R^{-1} = R^T \\ |R| = 1 \end{matrix}\right\}} Tr[R^T X^T Y]$$

alakban. Így a megoldás:

$$R_{opt} = USV^T,$$

Ahol az $S_{n \times m}$-es mátrix közel egy identitásmátrix, kivéve az utolsó elemet a diagonálisban, ami:

$$sgn(|UV^T|) = \begin{cases} +1, ha\ |UV^T| \geq 0 \\ -1, ha\ |UV^T| < 0 \end{cases}$$

Ez biztosítja, hogy az R_{opt} determinánsa 1, tehát mindenképp ortogonális.

2.4.1.4. Permutációs Prokrusztész

Míg abban az esetben, ha permutációra korlátozzuk a transzformációt, az optimális permutációs mátrix megkapható az alábbi módon:

$$P_{opt} = \arg \min_{\left\{P \middle| \begin{matrix} p_{ij} \in \{0,1\} \\ \sum_{i=1}^{n} p_{ij} = \sum_{j=1}^{n} p_{ij} = 1 \end{matrix}\right\}} \|XP - Y\|_F^2 =$$

$$= \arg \max_{\left\{P \middle| \begin{matrix} p_{ij} \in \{0,1\} \\ \sum_{i=1}^{n} p_{ij} = \sum_{j=1}^{n} p_{ij} = 1 \end{matrix}\right\}} Tr[P^T X^T Y]$$

Ennek a megoldásához a problémát lineáris programozási problémává alakítjuk át, amelynek a megoldása mindig felírható permutációs mátrixként, mely illesztési probléma megoldható a magyar módszerrel (ennek kifejtése jelen dolgozat tárgyát nem képezi) (Meng és mtsai. 2022).

A transzformációs mátrixot illető megkötésekre azért van szükség, mert gyakran ezek kisebb hibával képesek közelíteni egyik mátrixot a másikhoz; ennek gyakorlati megvalósításban tapasztalt eredményei a 3.4-es fejezetben kerülnek bemutatásra.

3. GYAKORLATI PÉLDA - SAJÁT KUTATÁS

Az online médiumokból származó szövegek elemzése hozzájárulhat a nyelvészeti és politikai kutatások fejlődéséhez, hiszen az egyes médiumok különböző kontextusban alkalmazzák a nyelvet, így ennek eredményeként bizonyos szavak eltérő szóhasználatban jelennek meg, ezzel bővítve és árnyalva az adott szó jelentését. Ezen kontextusbeli különbségek feltárása pedig segíthet minket megérteni, hogy a szavak használata hogyan alakítja a köznyelvet.

Ezenkívül a kontextuselemzés politikai szempontból is kiemelkedő jelentőséggel bír, Kubin és Von Sikorski 2021-es cikke szerint a média egyre inkább alakítja azt, ahogyan a politikai környezetet érzékeljük, hiszen jelentéssel látja el a különböző társadalmi eseményeket azzal, hogy maga is véleményt formál, és értékhierarchiát fogalmaz meg (Béni 2020).

Tehát a kontextuselemzés és az eltérő médiumok összehasonlítása segíthet bennünket abban, hogy feltárjuk a nyelvhasználat és a média politikai és társadalmi szerepének összetettségét, és azt, hogy miképpen alakítják ezek a tényezők a közös valóságképünket.

Ennélfogva e dolgozat célja két különböző online hírportálból létrehozott korpusz összehasonlítása szóbeágyazási vektorterek összeforgatásával. A médiumok kiválasztásakor arra törekedtem, hogy két eltérő társadalomszemléletű portált válasszak, ezért döntöttem a közismerten baloldali-liberális 444.hu és a jobboldali 888.hu mellett. Fontos megjegyezni, hogy a 888.hu 2023 februárjában megszűnt, és azóta az origo.hu alportáljaként működik.

888.hu 2015 szeptemberében indult, a lap főszerkesztője G. Fodor Gábor ismert kormánypárti politológus (Mandiner 2015), a portál legutóbbi jelmondata „Mi vagyunk a Soros ellenzéke”. Mind a lap neve, mind pedig a jelmondata értelmezhető egyfajta reakcióként a 2013 áprilisában Uj Péter által létrehozott 444.hu hírportálra, amely egy olyan amerikai alapítványtól kapott kölcsön működőtőkét, melynek egyik alapítója az Open Society Foundation (mely részben Soros György nevéhez is köthető) (Bátorfy 2017).

A cikkekből összeállított korpuszok létrehozása után először adattisztítást, és -előkészítést végeztem, majd különböző szóbeágyazási modelleket tanítottam és hasonlítottam össze analógiai kérdések mentén. Ezek után a legjobban teljesítő beágyazásokat többfajta Prokrusztész transzformációval is megpróbáltam összeilleszteni, annak érdekében, hogy a lehető legalacsonyabb hibával rendelkezőt vizsgálhassam meg és elemezhessem ki. Ezen lépések gyakorlati megvalósítása és részletes leírása a következőkben olvasható.

3.1. Az adatok forrásának ismertetése

Az adatokat, korpusz formájában, az RC2S2 kutatóközpont bocsátotta rendelkezésemre. Az elemzett korpusz az Nemzeti Kutatási, Fejlesztési és Innovációs Hivatal által támogatott ELTE RC2S2 kutatás része, melynek címe: A politikai nyilvánosság rétegei Magyarországon (2001-2020). A kutatás célja „a hivatásos politikusi, a professzionális sajtóban megjelenő és a laikus online közbeszéd szociológiai elemzése automatizált szövegelemzés és kritikai diskurzuselemzés segítségével" (Research Center for Computational Social Science 2020). A kutatás az NKFIH által támogatott. A támogatási időszak: 2020. december – 2023. december. A kutatásvezető: Németh Renáta. A kutatás azonosítója: K-134428.

Mind a 444.hu, mind pedig a 888.hu esetében a cikkek az 2013.03.10 - 2021.03.08 időszakot ölelik fel. Ez utóbbi 56 664 darab, míg az előbbi 101 932 darab cikket tartalmaz.

Az adatfeldolgozási és az adatelemzési fázishoz is Python programnyelvet használtam, azon belül is a Python 3.10.11-es verzióját. A Python egy nagy teljesítményű, nyílt forráskódú, objektumorientált programozási nyelv, amelyet Guido van Rossum hozott létre. Tervezési filozófiája a kód olvashatóságát hangsúlyozza, és szintaxisa lehetővé teszi a programozók számára, hogy kevesebb sornyi kódban fejezzenek ki koncepciókat, ellentétben az olyan nyelvekkel, mint a C. A Python legfontosabb jellemzője, hogy több programozási paradigmát támogat, beleértve az objektumorientált, imperatív és funkcionális programozási vagy procedurális stílust is (Srinath 2017).

Emellett a Python nagy előnye, hogy az alapmodulokon kívül további nyílt forráskódú csomagokkal bővíthető. Számos ilyen könyvtárat használtam az adattisztítás és -előkészítés során; mindenekelőtt a táblázatok és mátrixok helyes kezelésére a pandas (McKinney 2010) és NumPy (Harris és mtsai. 2020) csomagokat. Emellett főleg az adatelőkészítéshez az NLTK (Bird, Klein, és Loper 2009) és HuSpacy (Orosz és mtsai. 2023) könyvtárakat. A szóbeágyazások tanításához a Gensim (R Rehr Uv Rek és P Sojka 2010) csomagot, a beágyazások kiértékeléséhez pedig Makrai (2015) által bemutatott magyar nyelvű analógiai kérdésbankot használtam. A dimenziócsökkentéshez a Scikit-learn (Pedregosa és mtsai. 2012) könyvtárat, a vizualizációkhoz pedig a Matplotlib-et (Hunter 2007) és a Seaborn-t (Waskom és mtsai. 2017).

Még mielőtt elkezdhettem volna az adatelőkészítést, tisztítást végeztem. Először is, hogy a két korpusz azonos időszakot öleljen fel, szűrést végeztem a dátumot illetően, ehhez elengedhetetlen volt, hogy az adatbázisom metaadatokat is tartalmazzon, ne csupán a cikkek

szövegét. Ezek után kitöröltem az adatbázisból a duplikált elemeket, mivel mind a két korpusz tartalmazott többet is. A fent említett cikk számok a már megtisztított darabszámot jelölik.

3.2. Az adatok előkészítése

Ezek után következhetett az adatok előkészítése. Először kisbetűsítettem a szöveget, majd eltávolítottam az URL-eket a már említett URLExtract csomag segítségével. Ezután megtörténhetett a korpuszok előkészítésének egyik legfontosabb eleme, a tokenizálás. Ennek során a cikkeket egységekre bontjuk fel, ezek az egységek lehetnek szavak, illetve kifejezések. Emellett a tokenizálás magában foglalja az írásjelek eltávolítását is (Mészáros és Sebők 2018).

Ebben a lépésben távolítottam el a nem alfabetikus karaktereket tartalmazó szavakat is a szövegből, illetve végrehajtottam a stopszavazást. Stopszavaknak olyan szavakat nevezzük, amelyek gyakorlatilag minden dokumentumban előfordulnak, és nincs jelentőségük a tartalmi megkülönböztetés szempontjából. Ilyen szavak közé tartoznak például az "*egy*," "*a*," "*az*," "*hogy*," "*is*", és hasonlók. (Reményi, Sárdi, és Tóth 2016).

A korpuszok leíró statisztikái az 1. és 2. táblában láthatóak. Elmondható, hogy ugyan a cikkek darabszámából adódóan több tokenünk van a 444.hu-s korpuszban, viszont az egyes cikkekben átlagosan kevesebb token van, mint a 888.hu esetén.

Illetve a 444.hu cikkeinek tokenizált változatában nagyobb az eltérés a tokenszámok között, mely látszik a standard hiba értékében. Ezen korpusz esetén a cikkek 25%-a 60 vagy annál kevesebb tokenből épül fel, míg ez az érték a 888.hu cikkeiben jóval magasabb, 105 tokent jelent.

1. Táblázat: Leíró statisztika a 444.hu korpuszának tokeneiről

Összes	Átlag	Medián	Standard hiba	1. kvartilis	3. kvartilis
16281000	159,7	100	226,7	60	170

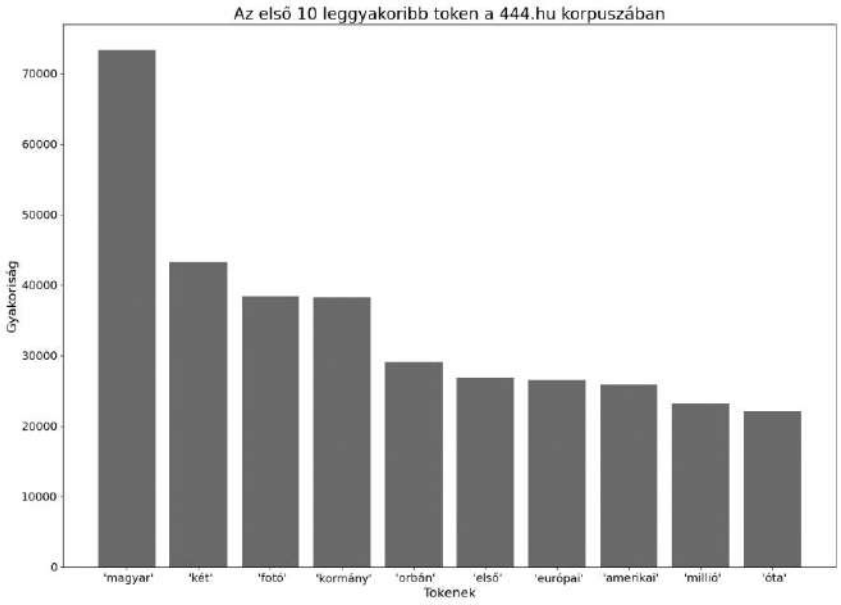
2. Táblázat: Leíró statisztika a 888.hu korpuszának tokeneiről

Összes	Átlag	Medián	Standard hiba	1. kvartilis	3. kvartilis
10531286	185,9	150	150,7	105	219

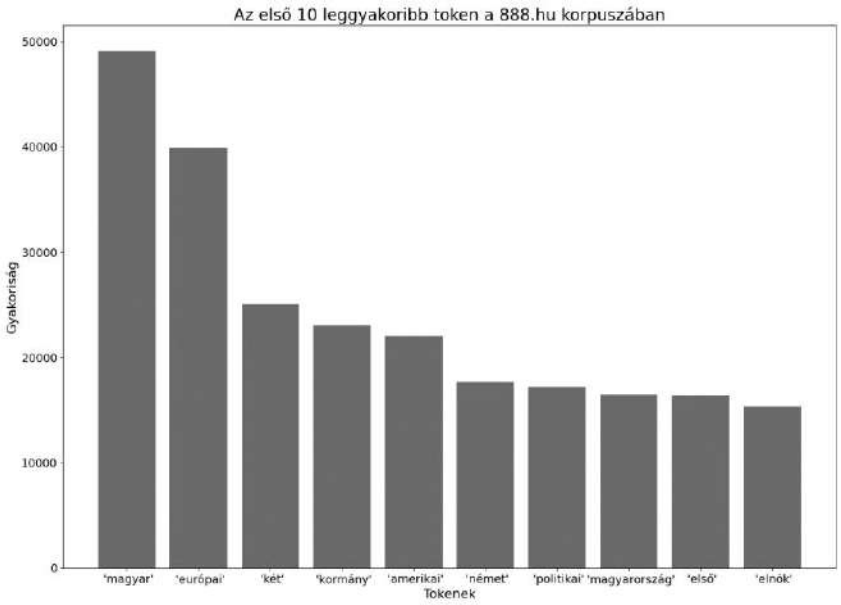
A tokenizálás után megnéztem melyek azok a szavak, amelyek a leggyakrabban szerepelnek a két korpuszban. Ezek a 4. és 5. ábrán szerepelnek. Természetesen mind a két korpusz esetében a *magyar* szó toronymagasan vezet, de a *kormány, amerikai, európai* szavak is közös gyakori

elemek. Míg a 444.hu esetében az *orbán* token is bekerült a top 10 közé, addig a 888.hu-n inkább az *elnök* és *magyarország* szavak fordulnak elő többször.

5. Ábra: Leggyakoribb tokenek a 444.hu korpuszában



6. Ábra: Leggyakoribb tokenek a 888.hu korpuszában



Az is látszik a két ábrából, hogy a stopszavazás eredményesen lezajlott, hiszen a leggyakoribb szavak között nem szerepelnek ezek. Szóbeágyazási modellek tanítása esetén egyébként gyakori, hogy még elsőkörös stopszavazást sem eszközölnek az előkészítés során (Rahimi és Homayounpour 2023), ugyanakkor az én modelljeim rosszabbul teljesítettek enélkül. Viszont további, korpusz-specifikus stopszavazást már nem hajtottam végre, hiszen az ronthatta volna

a beágyazásaim eredményességét. Ugyanezen okból nem alkalmaztam sem szótövezést, sem lemmatizálást (szótári alakra hozást).

Ezek után azonosítottam a szignifikáns n-gramokat a szövegekben; ilyen volt például az *orbán_viktor, iszlám_állam, ursula_von_der_leyen* token. Ezeket tehát kicseréltem az imént felsorolt alakokra mind a két szöveg esetén.

Összességében ezen lépéseket alkalmaztam az előfeldolgozás során. Ugyanakkor érdemesnek tartom megjegyezni, hogy valahol ezek a lépések jelentik a dolgozatom korlátait is, hiszen egyéb lépések beiktatásával (akár egy újabb körös, igen megfontolt stopszavazás, akár egy gyakoriság alapú szűrés) erősen változhat a szóbeágyazási modelljeim eredményessége is. Így a végső token-darabszám a 888.hu esetén 10,392,077, a 444. hu esetén pedig 16,137,033.

3.3. Szóbeágyazás végrehajtása és a beágyazás minőségének vizsgálata

3.3.1. A szóbeágyazási modellek értékelése

Valódi reprezentációra alkalmas vektortér-modellek tanítása rendkívül fontos feladat, ugyanakkor választ adni arra a kérdésre, hogy milyen a jó szóbeágyazási modell kifejezetten fogós probléma.

A modellek tulajdonságainak tesztelésére megkülönböztetünk külső (extrinsic) és belső (intrinsic) értékelőket. A külső értékelési feladat arra keresi a választ, hogy a szóbeágyazási modellek milyen mértékben járulnak hozzá egy konkrét feladat megoldásához, példának okáért a gépi fordításhoz vagy beszédfelismeréshez. Ezzel szemben a belső értékelés magát a reprezentációt teszteli; közvetlenül méri a szavak szemantikai és szintaktikai kapcsolatait.

Wang és munkatársai (2019) nyomán öt olyan tulajdonságát szeretném kiemelni a szóbeágyazási modelleknek, amelyek alkalmassá teszik őket a további elemzésre. Ugyanakkor érdemes hangsúlyozni, hogy annak vizsgálata, hogy egy szó reprezentációja megfelelő-e nehezen objektivizálható. Hiszen gyakran már annak hozzáadott értéke van, hogy az adott korpuszból megkapott kontextus mit mond el a kiválasztott szóról (vagy magáról a korpuszról). Ez különösen igaz olyan specifikus szövegekre, ahol egy meghatározott téma köre csoportosulnak a dokumentumok – ahogy az esetemben a cikkek a politika köré.

Az egyik ilyen tulajdonság a konfláció elkerülése. Ez az a gyakorlat, amikor két különálló fogalmat úgy kezelnek, mintha egy lenne. Tehát a különböző helyi kontextusoknak a szó sajátos tulajdonságait kell eredményezniük – például a többes vagy egyes számú alakot – nem

pedig tévesen eltérő fogalmakat, amelyek hasonló kontextusban szerepelnek. Emellett nagyon fontos tényező a robusztusság a lexikai kétértelműséggel szemben; ez éppen az azonos alakú szavak többjelentésének feltárását igyekszik hangsúlyozni. Vagyis egy szó minden jelentését reprezentálni kell.

A modellnek alkalmasnak kell lennie a sokoldalúság bemutatására; tehát a szó morfológiai, fonetikai, szintaktikai stb. tulajdonságainak mind hozzá kell járulniuk a végső reprezentációhoz. Ez azért is fontos, hogy megtaláljuk az egyes szavak közötti kapcsolatokat. Illetve a szóbeágyazási modell eredményeinek megbízhatónak kell lenniük. Hiszen még ha egy modell a véletlenszerű inicializálás miatt különböző reprezentációkat is hoz létre ugyanabból az adathalmazból, a különböző reprezentációk teljesítményének következetesen kell teljesítenie.

Az utolsó lényeges jellemző a vektortér geometriájára vonatkozik. Általánosságban elmondható, hogy a gyakoribb, egymáshoz nem kapcsolódó szavak kisebb halmazának egyenletesen kell eloszlania a térben, míg a ritka szavak nagyobb halmazának a gyakori szavak köré kell csoportosulniuk. Tehát a modelleknek le kell küzdeniük a szóhasználat következetlen gyakoriságából adódó nehézségeket.

Ezen jellemzők biztosítására Mikolov és munkatársai (2013) által javasolt analógiai mutatókat alkalmaztam. Amennyiben van egy szópárunk a és a^*, illetve rendelkezünk egy b szóval is, akkor az a és a^* közötti analógiai kapcsolat felhasználható a b szóhóz tartozó b^* megtalálására. Matematikailag ez a következőképpen fejezhető ki:

$$a{:}\, a^* :: b{:}\, \_$$

ahol az üres hely a b^*-t keresi. Ilyen példa lehet:

$$olvasás{:}\, olvasni :: írás{:}\, írni$$

Makrai (2015) Mikolovék munkája mentén létrehozott egy analógiai kérdésbankot magyar nyelvre, ezt használtam a modelljeim értékelése során. A kérdésbank két főbb részből áll: morfológiai és szemantikai szópárokból, melyek a 3. táblázatban látható módon további bontásban kerültek megírásra.

3. Táblázat: Az analógiai kérdésbank elemei és példák

		Példák	Elemszám
Morfológiai (nyelvtani)	Melléknév-határozószó	természetesen:természetes:lassan lassú	40
	Ellentétes	ismeretlen:ismert::számtalan:számos	30
	Középfok	nagyobb:nagy::kisebb:kis	40
	Felsőfok	legnagyobb:nagy::legjobb:jó	40
	Jelen idejű melléknévi igenév	egyeztetés:egyeztet::jutás:jut	40
	Nemzeti melléknév	Albánia:albán::India:indiai	41
	Múlt idő	volt:van::kellett:kell	40
	Többesszám – főnév	akik:aki::ezek:ez	40
	Többesszám – ige	lesznek:lesz::tudnak:tud	40
Szemantikai	Főváros – gyakori ország	Pozsony:Szlovákia::Bagdad:Irak	20
	Főváros – világ	Amszterdam:Hollandia:: Zágráb Horvátország	166
	Megye – székhely	Baranya:Pécs::Békés:Békésscsaba	171
	Valuta	Ján:jen::Ausztria:euró	30
	Család	fiú:lány::apa:anya	20

Makrai (2015) mentén saját szerkesztés

3.3.2. Szóbeágyazási modellek illesztése – 1. fázis

Az adatok behívása, megtisztítása, előkészítése majd a megfelelő értékelő-módszer kiválasztása után elkezdtem a modellek tanítását. A szóbeágyazásokat a korábban említett gensim könyvtárat alkalmazva hajtom végre. 24 darab különböző modell kerül betanításra mind a két korpusz esetén, melyek közös jellemzői az alábbiak.

A kezdeti tanulóráta 0,05, ez az érték meghatározza azt, hogy mennyire hajlandó a modell az egyes tanítási példákból származó hibák alapján módosítani a súlyait. A tanuló ráta eleinte magasabb, hiszen ekkor messzebb vagyunk a kívánt optimumtól (általában 10^{-2} körüli kezdőértékkel indítunk), majd egyre csökken (10^{-4}) (Di Gennaro, Buonanno, és Palmieri 2021). Tehát az általam megadott kezdőérték hozzájárul, hogy a modell hatékonyan konvergáljon az optimumhoz.

Annak érdekében, hogy a tanítási időt csökkentsem, negatív mintavételezést alkalmaztam, mely a korábbi fejezetben került kifejtésre. Ezen paraméter értékét 5-re állítottam, tehát minden pozitív példához 5 negatív példát használunk fel a tanítás során.

Emellett a min_count paraméter értékét is meghatároztam, szintén 5-ös értéket adva neki. Így tehát csak azok a szavak vesznek részt a tanításban, amelyek legalább ötször szerepelnek a korpuszban. Ennek a paraméternek a meghatározása azért fontos, hogy kiszűrje a ritka szavakat, amelyek nem hordoznak sok információt a korpuszról. Ugyanakkor érdemes nem túl nagy értéket adni neki, hogy elkerüljük a kevésszer előforduló, viszont jelentős szavak kiiktatását a modellből.

Az epoch paraméter az iterációk számát határozza meg a modellek tanítása során. A Gensim csomag alapértelmezett értéke 5, melyet én is használtam. Érdemes megjegyezni, hogy az iterációk számának növelése egyrészről pozitív eredményt érhet el a szóbeágyazás minőségét illetően, hiszen többször történik a „finomhangolás" a hibavisszaterjesztés során, ugyanakkor a túl sok iteráció hajlamos a túlillesztésre. Illetve mivel igen sok modellt építettem mind a két korpusz esetén, ezért arra törekedtem, hogy a számítási időt lecsökkentsem.

Azonkívül a FastText modellek tanítása során 3 és 6 közötti n-gramokat vizsgáltam. Ezek a közepes méretű n-gramok lehetővé teszik a modell számára, hogy észlelje a rövidebb kifejezéseket, ugyanakkor elég hosszúak ahhoz, hogy az összetettebb kifejezéseket és jelentéseket is kezelni tudja. Ezek az értékek azért is megfelelőek, mivel a tartomány nem túl kicsi, hogy csak a részleteket lássa, de nem is túl nagy, hogy elveszítse az összefüggéseket.

A közös paraméterezés mellett a következőkben felsorolt eltérések mentén különböztettem meg a modelljeimet. Két különböző modelltípust használtam; a FastText és a Word2Vec algoritmust, ezen belül pedig mind a kettő esetén a Skip-Gram és CBOW architektúrákat is megnéztem. Illetve a dimenziószám és az ablakméret paraméterek (melyeket a korábbi fejezetekben kifejtettem) esetén használtam több differens értéket. A különböző modellek és az analógiai kérdésbank által meghatározott teljesítményük a 4. és 5. táblázatban láthatóak.

4. Táblázat: A 444.hu korpuszán tanított különböző szóbeágyazási modellek és eredményeik

444.hu				
Modell	Architektúra	Dimenzió	Ablakméret	Pontosság
Word2vec	CBOW	100	2	22,63%
		100	5	27,12%
		100	10	27,98%
		300	2	19,23%
		300	5	25,17%
		300	10	27,43%

Word2vec	Skip-Gram	100	2	26,56%
		100	5	**30,51%**
		100	10	**33,96%**
		300	2	21,25%
		300	5	25,48%
		300	10	30,06%
FastText	CBOW	100	2	28,21%
		100	5	28,08%
		100	10	27,42%
		300	2	23,65%
		300	5	24,75%
		300	10	23,32%
FastText	Skip-Gram	100	2	**31,96%**
		100	5	**33,57%**
		100	10	27,42%
		300	2	18,49%
		300	5	20,70%
		300	10	22,82%

5. Táblázat: A 888.hu korpuszán tanított különböző szóbeágyazási modellek és eredményeik

888.hu				
Modell	Architektúra	Dimenzió	Ablakméret	Pontosság
Word2vec	CBOW	100	2	24,77%
		100	5	31,29%
		100	10	32,19%
		300	2	20,88%
		300	5	28,16%
		300	10	30,57%
Word2vec	Skip-Gram	100	2	28,71%
		100	5	**34,45%**
		100	10	**35,33%**
		300	2	21,30%
		300	5	25,83%
		300	10	28,46%
FastText	CBOW	100	2	25,97%
		100	5	26,37%
		100	10	25,58%
		300	2	22,40%
		300	5	22,68%
		300	10	21,65%
FastText	Skip-Gram	100	2	**33,34%**
		100	5	**33,63%**
		100	10	29,63%

		300	2	16,93%
		300	5	18,96%
		300	10	19,90%

Az elért pontosságot úgy határoztam meg, hogy a modellben fellelhető analógiai szópárok közül hányat talált el helyesen – tehát relatív értékek mentén került kiszámításra az eredmény. Még mielőtt kitérnék a modellek értékelésére, elöljáróban annyit szeretnék megjegyezni, hogy természetesen nem lehet elvárni egy konkrét téma köré csoportosuló dokumentumokból összeállított korpusztól, hogy általános kérdéseken tökéletesen teljesítsen. Hiszen pont a specifikusságából adódóan rengeteg olyan szót és ehhez tartozó kontextust tartalmaz, melyet a kérdésbank nem szerepeltet. Ennélfogva az értékelést és a tanítást is két, egymást követő fázisban valósítottam meg. Az első, és az előbbiekben kifejtett lépés a 24 modell közül négy olyan választása, amely általánosságban jól teljesít. A következő fázisban pedig ezeket a modelleket fogom a különböző analógiai témakörök mentén tüzetesebben megvizsgálni, hogy a feladatnak és a korpuszoknak legmegfelelőbb beágyazásokat tudjam kiválasztani.

Az első fázisban tapasztalt eredmények alapján általánosságban elmondható, hogy a Skip-Gram modellek sokkal jobban teljesítenek a CBOW architektúránál. Ez nem is meglepő, tekintve, hogy a Skip-gram a környező szavakat próbálja megtanulni egy adott szó alapján, így valamiféle előretekintő mechanizmus jelenik meg benne. Ezáltal képes a ritkább és specifikusabb szavakat is hatékonyabban kezelni, mivel azok kontextusát is figyelembe veszi. Ezzel szemben a CBOW egy adott szót próbál a környező szavak alapján előállítani, amely sokszor kevésbé hatékony a nem túl gyakori, de nagy jelentőségű szavak esetén.

Érdekes, hogy a Word2vec CBOW modellje rosszabbul teljesít a 444.hu korpuszánál, ami következhet akár abból is, hogy specifikusabb a szókészlete a 444.hu cikkeinek, lehetséges, hogy többet használnak ritkább, különlegesebb szavakat, mint a 888.hu esetében. Ugyanakkor egy másik felmerülő indok, hogy a 888.hu átlagosan hosszabb cikkeket ír, tehát terjedelmesebbek a szövegeik, így a lokális kontextusok is nagyobbak.

Meglepő eredmény az is, hogy a dimenziószám növelése nem eredményez jobb modellt egyik korpusz esetén sem. Ez egyrészről fakadhat a korpusz homogén jellegéből, illetve abból is, hogy az adathalmazunk mérete nem túl jelentős. Észrevehető, hogy a nagyobb dimenziószám a 444.hu-nál nem ér el annyival rosszabb pontosságot, mint a 888.hu esetén.

Van, ahol a nagyobb ablakméret is csökkent eredményességhez vezetett, ez különösen igaz a FastText modelljeinél. Ennek a kiváltó oka lehet az, hogy nagyobb ablakméret esetén a modell hajlamos kevésbé specifikus szókapcsolatokat megtanulni, mivel a kontextus túl messze esik a cél szótól.

Az már kevésbé megállapítható, hogy a FastText vagy a Word2vec modell alkalmasabb-e a homogén, politikai témájú cikkeken tanított beágyazások létrehozására. Talán annyi tanulság vonható le, hogy a 444.hu esetén a FastText-es CBOW modellek jobban teljesítenek a Word2Vec azonos architektúrájú modelljeinél. Itt valószínűleg az n-gramokból fakadó információtöbblet segít a helyesebb szókapcsolatok kialakításában. Ugyanakkor, ha a Skip-Gram modelleket vizsgáljuk mind a két modell és mind a két korpusz esetén valóban nehéz lenne egyértelmű döntést hozni azt illetően, hogy melyik beágyazás az eredményesebb.

Hasonlóan a modelltípus teljesítményének homályosságához, az ablakméretek esetén is nehéz általánosítható tanulságot levonni. Míg a Word2Vec rosszabbul teljesít 2-es ablakmérettel, a FastText nem feltétlen. Ez adódhat abból, hogy az n-gramokból kapott információtöbblet kiegyensúlyozza a kisebb kontextus okozta információ-veszteséget.

A fent bemutatott elemzés nyomán látható, hogy míg az architektúra és a dimenziószám mind a két korpusz esetén egyértelmű eredményt adott, addig az ablakméret és a modell típusa további kérdéseket vetett fel. Az előbbi, általánosítható eredményekre hagyatkozva a négy kiválasztott modell - mind a két korpusz esetén - az alábbi:

- Word2Vec Skip-Gram 100-as dimenziószámú 5-ös ablakméretű modellje
- Word2Vec Skip-Gram 100-as dimenziószámú 10-es ablakméretű modellje
- FastText Skip-Gram 100-as dimenziószámú 2-ess ablakméretű modellje
- FastText Skip-Gram 100-as dimenziószámú 5-ös ablakméretű modellje

Érdemes megjegyezni, hogy abszolút nem volt kikötésem ugyanazon modelleket választani a két korpusz esetén, hiszen a Prokrusztész transzformáció arra tökéletesen alkalmas eszköz, hogy különböző-jelentésű változóval (vagy dimenzióval) rendelkező mátrixokat is egymásba forgasson. Viszont a politikai fókuszból adódó homogenitás igen hasonló eredményeket szült a korpuszok esetén, még a tokenek elemszámából adódó különbség ellenére is.

A fent említett bizonytalansági tényezők miatt iktattam tehát be a második fázist a döntéshozatalba, melyet a következő fejezet szemléltet.

3.3.3. Szóbeágyazási modellek illesztése – 2. fázis

Tehát az alábbiakban szeretném szemléltetni a négy kiválasztott modell analógiai kérdésbank elemein mért részletes teljesítményét. Ezen eredményeket a 6. és 7. táblázat illusztrálja.

6. Táblázat: A 4 legjobb modell analógiai kérdésbankon mért pontossága a 444.hu esetén

		Word2Vec		FastText	
		5-ös ablak	10-es ablak	2-es ablak	5-ös ablak
Morfológiai	Melléknév-határozószó	11,67%	11,92%	41,41%	26,03%
	Ellentétes	3,94%	3,20%	9,85%	6,04%
	Középfok	40,26%	29,45%	49,64%	41,39%
	Felsőfok	18,17%	11,56%	29,28%	15,77%
	Jelen idejű melléknévi igenév	0,00%	0,31%	52,62%	40,92%
	Nemzeti melléknév	52,56%	58,46%	48,72%	49,36%
	Múlt idő	80,95%	80,48%	72,22%	70,63%
	Többesszám – főnév	33,16%	30,12%	35,47%	30,84%
	Többesszám – ige	89,19%	87,24%	45,35%	55,41%
Szemantikai	Főváros – gyakori ország	27,45%	45,75%	24,84%	43,14%
	Főváros – világ	21,32%	29,41%	16,93%	28,76%
	Megye – székhely	4,55%	1,52%	3,03%	6,06%
	Valuta	10,84%	12,32%	9,61%	9,11%
	Család	10,83%	13,33%	29,17%	23,33%

7. Táblázat: A 4 legjobb modell analógiai kérdésbankon mért pontossága a 888.hu esetén

		Word2Vec		FastText	
		5-ös ablak	10-es ablak	2-es ablak	5-ös ablak
Morfológiai	Melléknév-határozószó	10,90%	9,49%	49,36%	30,38%
	Ellentétes	7,38%	6,77%	6,15%	4,62%
	Középfok	16,22%	15,22%	39,26%	28,17%
	Felsőfok	4,05%	5,11%	13,06%	4,35%
	Jelen idejű melléknévi igenév	0,00%	0,00%	51,05%	40,00%
	Nemzeti melléknév	53,71%	54,52%	53,04%	56,28%
	Múlt idő	71,43%	63,33%	59,84%	65,24%
	Többesszám – főnév	14,80%	15,33%	32,09%	23,17%
	Többesszám – ige	87,39%	81,83%	25,38%	50,00%
Szemantikai	Főváros – gyakori ország	56,21%	58,82%	49,67%	50,33%
	Főváros – világ	37,34%	43,60%	26,36%	35,90%
	Megye – székhely	1,52%	1,52%	7,58%	3,03%
	Valuta	7,94%	6,35%	1,06%	4,76%
	Család	13,19%	7,69%	16,48%	14,29%

Itt már szeretnék arra is kitérni, amiről a korábbiakban nem volt szó; a FastText modellek tanítása valamivel több időt vett igénybe az n-gramokból adódóan, mint a Word2Vec modelleké. A 888.hu esetén ez azt jelentette, hogy míg a Word2vec 2-es és 5-ös ablakméretű

modelljének tanítása rendre közel 6 és 10 percet vett igénybe, addig a FastText 2-es és 5-ös ablaknagyságú modelljei 9 és 12 perces tanítási időt követtek. Tehát még a kisebb ablakú modell is huzamosabb időt vett igénybe a FastText esetén. Abszolút hasonló tendenciákat figyeltem meg a 444.hu korpuszából létrehozott beágyazásoknál is, itt a tanítási idő természetesen megnövekedett, mivel jóval nagyobb tokenszámból építkezett a modell.

Az elméleti háttér fejezetben bemutatott okokból adódóan a FastText morfológiai analógiákon mért eredményei jobbak. Ez különösen szembetűnő a melléknév-határozószó analógia párok, a középfok és a jelen idejű melléknévi igenevek esetén. De az is általánosan elmondható, hogy a Word2Vec eredményei szélsőségesebbek: amikben jól teljesítenek, abban kitűnőek, amikben viszont rosszul, ott 0-10% közötti pontosságokat találunk. A FastText ezzel szemben a legtöbb alkategóriában középszerű eredményt produkál.

Az is szembetűnő, hogy bizonyos kategóriák analógiai feladatainak megoldására alkalmatlanok a modelljeim, ez valószínűleg a két korpusz jellegéből adódik. Ilyen például az országok valutával való kapcsolatának felismerése, de érdekes módon a megyék székhelyeivel való azonosítása is gondot okoz. Ez utóbbi fakadhat a korpusz kis méretéből is.

Az ablakméret vizsgálata során a Word2Vec esetében mind a két korpusznál a jelentősebb ablaknagyság eredményesebb az analógiai feladatok elvégzésében. Ugyanakkor a FastText modelljeinél ez korántsem ilyen egyértelmű.

Érdekes módon a Word2Vec modelljei különösen jól teljesítenek a többesszámú és a múlt idejű igék analógiai feladataiban; tehát ezek a modellek helyesen megtanultak igét ragozni. Ugyanakkor az én kutatásom szempontjából ez, ha nem is teljesen irreleváns, semmiképp sem a legfontosabb. A Word2Vec modellek szélsőséges eredményessége és a sikereres alkategóriák irrelevanciája miatt a FastText modelljeit választom további vizsgálatra.

A különböző ablakméretek összevetése során azt figyelhetjük meg, hogy a morfológiai feladatoknál a 2-es ablakméret gyakran néhány százalékkal jobban teljesít, ugyanakkor a szemantikai feladatokban sokszor kifejezetten rosszul. Wang és munkatársai (2019) mentén arra törekedtem, hogy a modellem ne kizárólag a szemantikai információkat tartalmazza; egy szó reprezentálása során a morfológiai tulajdonságokat is vegye figyelembe, ugyanakkor a kontextus vizsgálat szemantikai hangsúlyt von maga után.

Ennélfogva a végső választásom mind a két korpusz esetén a FastText 5-ös ablakméretű modelljére esett, hiszen ezek a modellek a szavak strukturális elemeit többé-kevésbé felismerték (még ha kicsit gyengébben is, mint a 2-es ablaknagyságú modellek), illetve egészen jól teljesítettek a szemantikai analógiák felismerése során is. Emellett két alkategóriát különösen fontosnak ítéltem meg a korpusz tematikájából adódóan: ezek azok a kategóriák, amelyek a fővárosokat párosítják országokkal. Ennek az az oka, hogy mind a két médium cikkei gyakran térnek ki külpolitikai témákra és ezáltal ehhez kapcsolódó tulajdonnevekre, melyek felismerése és helyes kontextusba ágyazása kiemelten fontos feladata a modellnek.

3.3.4. Kiválasztott modellek elemzése

A továbbiakban tehát mind a 444.hu, mind a 888.hu esetén 100-as dimenziónagyságú 5-ös ablakméretű FastText Skip-Gram modellen tanult szóbeágyazásokat fogom vizsgálni.

Annak érdekében, hogy a szavak – és az egymáshoz való viszonyuk – vizuálisan ábrázolható legyen dimenziócsökkentést hajtottam végre a szóbeágyazási modell által meghatározott szótáron. A cél az volt, hogy a 100-as dimenziónagyságot kettes vagy hármas dimenzióra szűkítsem, hogy illusztrálásra alkalmas legyen.

Kmetty (2022) mentén a dimenzióredukciót t-eloszlású sztochasztikus szomszéd beágyazással (t-distributed stochastic neighbour embedding, T-SNE) végeztem (van der Maaten és Hinton 2008). A T-SNE úgy kezeli az eredeti pontokat, mintha azokat egy normál eloszlásból vett mintaként modellezné, míg a beágyazott pontokat, mintha Student-eloszlásból származnának (Muhi és Johanyák 2020). A T-SNE nemlineáris redukciós módszer, mely a főkomponens elemzéssel szemben a lokális struktúrákat is jól ábrázolja magas dimenziószámú adatmátrix esetén. A T-SNE tehát felfogható úgy, mint egy magas dimenziós adatmátrix lokális struktúrájának alacsony dimenziós reprezentációja. Ez azt is jelenti, hogy olyan szópárok közötti kapcsolatokat helyez előtérbe, melyek közel vannak egymáshoz az eredeti magas dimenziós térben, viszont a távolabbi szócsoportok közötti viszonyokat nem tudja mindig pontosan megőrizni. Ennélfogva a T-SNE által létrehozott ábrák inkább a lokális hasonlóságokat és kapcsolatokat mutatják be, a távolabbi összefüggéseket gyakran kevésbé helyesen ábrázolják.

A T-SNE végrehajtásához a korábban már említett Scikit-learn könyvtárat használtam. A dimenziócsökkentéshez az alábbi paramétereket vettem igénybe; a perplexitást 40-re

állítottam, amely igen magas értéknek számít. Ez egy olyan hiperparaméter, amely meghatározza, hogy az algoritmus milyen mértékben veszi figyelembe a közeli és távoli szomszédok közötti hasonlóságot. Tehát nagyobb perplexitás esetén az algoritmus jobban szem előtt tartja a távoli szomszédokat is, amely lehetővé teszi a globális szerkezet megőrzését – ez esetemben kulcsfontosságú szempont volt. Emellett azt is meghatároztam, hogy a modell kezdetben egy lineáris PCA-val végezzen dimenziócsökkentést, és ezeket az adatokat felhasználva adja meg a kiindulási pontokat a T-SNE-nek, ami gyorsabb futást eredményez. Többek között azért volt szükségszerű az inicializálásban segítséget adó PCA, mert az iterációk számát 2500-ban határoztam meg; ez magasnak számít, és igen meg tudja növelni a számításigényt.

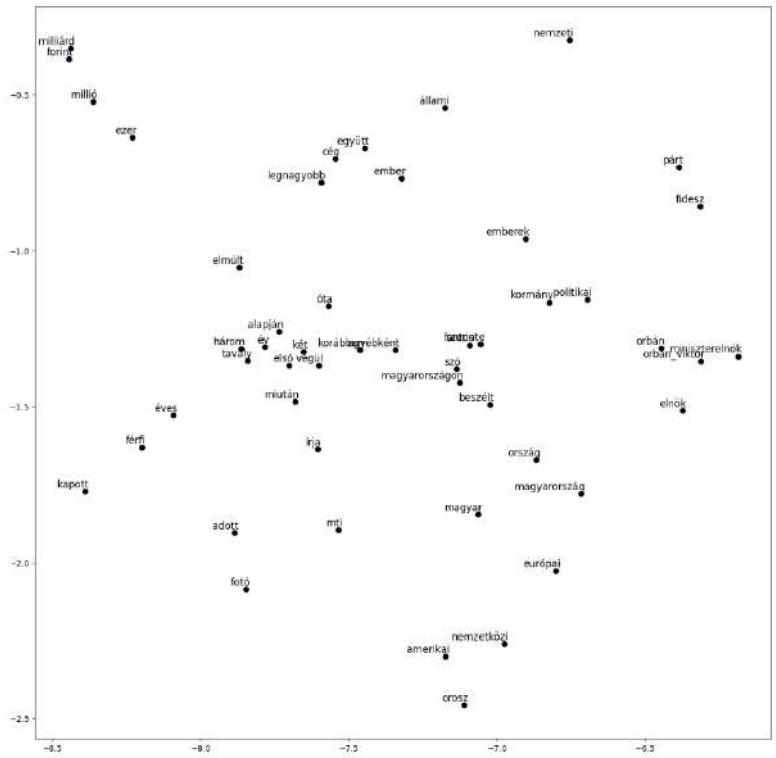
A leggyakoribb szavakat reprezentáló vektorokat a T-SNE segítségével kapott első két komponens szerint ábrázoltam, melyek a 6. és 7. ábrán láthatóak.

A kapott dimenziók értelmezése T-SNE modellben egyáltalán nem triviális. Mivel nem általam válogatott, egy adott téma köré csoportosítható szavakat ábrázolok, így most ebben az esetben eltekintek ezek meghatározásától, és a szavak közötti távolságra koncentrálok.

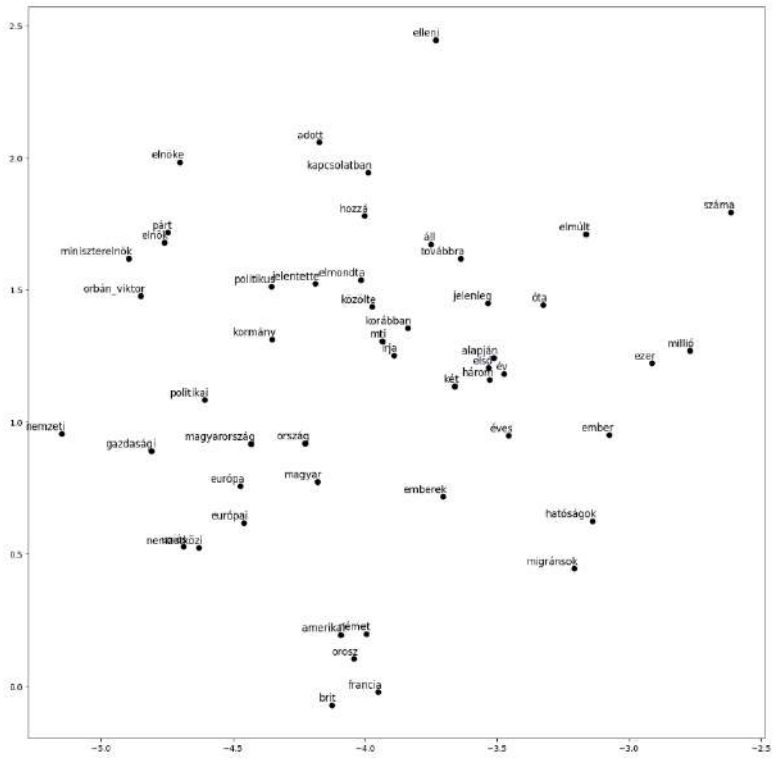
A 444.hu korpuszán tanított beágyazás felismeri az *orbán, orbán_viktor, elnök, miniszterelnök* közötti kontextusbeli hasonlóságot. Illetve szintén helyesen, közel helyezkednek el egymáshoz a *két, három, első* szavak, illetve az *ország, magyar* és *magyarország* tokenek. Ugyanakkor például a *nemzeti* és a *nemzetközi* szavak igen távol helyezkednek el a kétdimenziós térben, illetve a *magyarország* és *magyarországon* sincsenek elég közel.

Az leggyakoribb 50 szó alapján a 888.hu korpuszán tanított beágyazás kétdimenziós reprezentációja igen jól teljesít; *amerikai, német, orosz, francia, brit* szavak kontextusbeli hasonlóságát szépen felismeri, de a *jelentette, elmondta, közölte* elemeket is egymáshoz közel ábrázolja a térben. Itt is megjelenik az *elnök, orbán_viktor, miniszterelnök* hasonlósága, és az ehhez közel álló *párt, kormány, politikus* szavak is megjelennek.

7. Ábra: Az 50 leggyakoribb szó távolsága T-SNE módszer alapján a 444.hu esetén



8. Ábra: Az 50 leggyakoribb szó távolsága T-SNE módszer alapján a 888.hu esetén



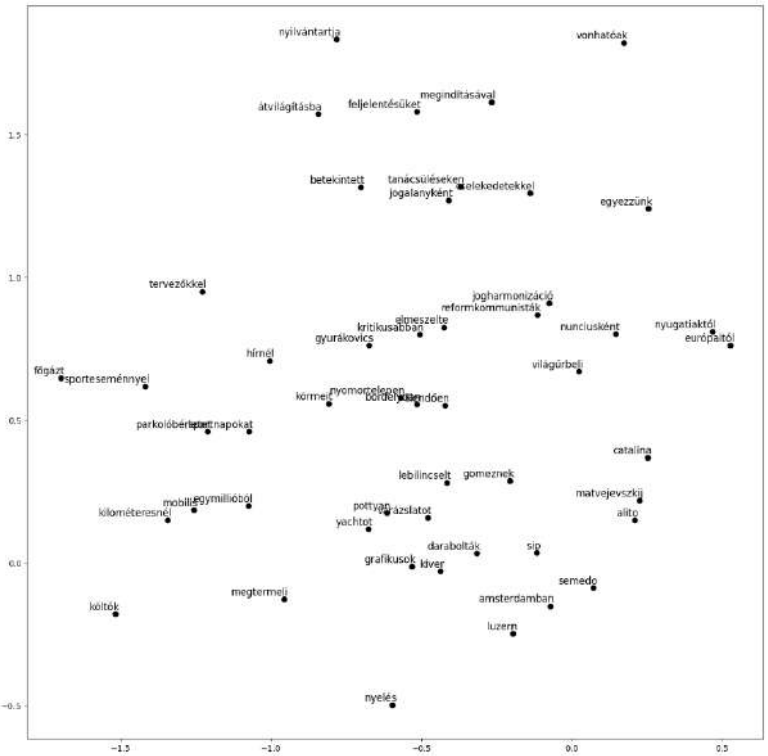
Izgalmas eredménye a modelleknek, hogy egyáltalán melyek azok a szavak, amelyek a leggyakoribbak a korpuszokban. Míg a 444.hu esetén a *fidesz* szó bekerült az első 50-be, addig

a 888.hu valószínűleg inkább a *kormány* szót használja ehelyett. Illetve a 888.hu leggyakoribb szavai közé a *migránsok* is bekerült a 444.hu-val ellentétben. Ugyanakkor ez egy várható különbség volt a kormányközeli és ellenzéki média összehasonlítása során.

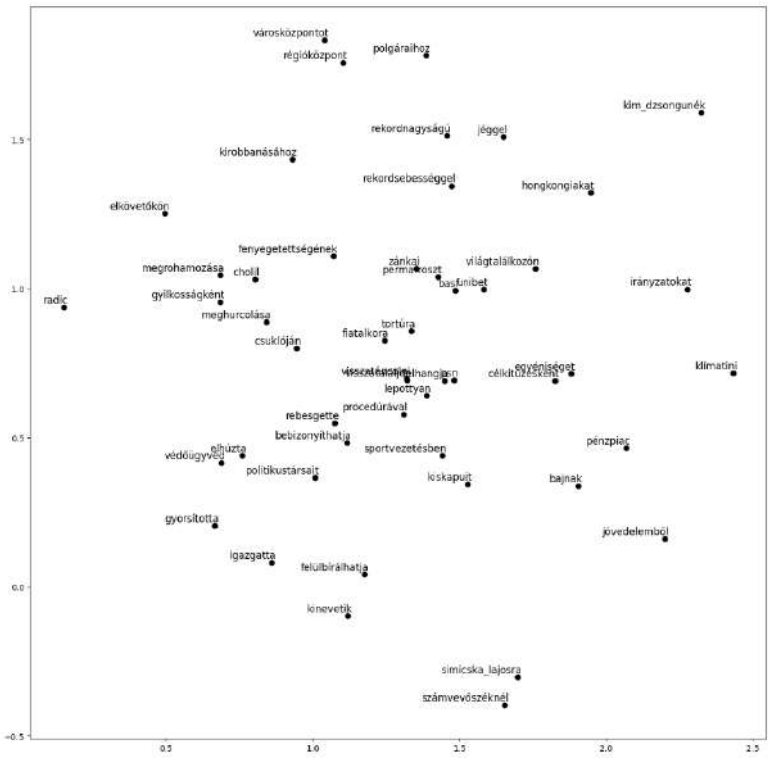
Nyilván a gyakori szavak kontextusát sokkal könnyebben tanulja meg a beágyazás, mint a ritka szavakét. Ennek okán megvizsgáltam hogyan teljesítenek a modellek a legritkább 50 token elemzésében. A 8. ábra reprezentálja a 444.hu korpuszán tanított modellt, melyen látszik, hogy bizony kifogott a beágyazáson a legtöbb ritka szó, kevés valós szemantikai kapcsolatot ábrázoló távolságot lehet azonosítani. A *nyugatiaktól* és *európaiaktól* szavak közelsége helyesnek tűnő kontextusreprezentációt jelöl, szintúgy az *amsterdamban* és a *luzern*. Míg a 888.hu esetén (9. ábra) a *városközpont* és *régióközpont* kapcsolatát tudta ábrázolni a T-SNE. Illetve a *fenyegetettségének, meghurcolása, gyilkosságként* és *megrohamozása* szavak is a kétdimenziós tér azonos részén helyezkednek el.

Tehát azért általános elmondható, hogy míg a gyakori szavak közötti hasonlóságokat többnyire helyesnek vélt módon határolja be a dimenziócsökkentett beágyazás, addig a ritka szavak reprezentációja már sokkal kevésbé sikeres.

9. Ábra: Az 50 legritkább szó távolsága T-SNE módszer alapján a 444.hu esetén



10. Ábra: A 888.hu korpuszából tanított szóbeágyazás legritkább 50 szava



Érdemes kiemelni, hogy ezek az ábrák remek reprezentációi annak, milyen nehéz feladat két korpuszt összehasonlítani ilyen módon. Hiszen az elengedhetetlen dimenziócsökkentés során a kapott két dimenzió a korpuszok esetében teljesen eltérő. Például a gyakori szavakat reprezentáló ábrákon mindkét esetben helyesen jelenik meg az *orbán_viktor, miniszterelnök, elnök* szavak kapcsolata, viszont meglehetősen eltérő helyeken az ábrázolt térben.

Az általánosabb vizsgálat után bizonyos, a kormányközeli és ellenzéki sajtót gyakran megosztó szavakat tanulmányoztam. A *feminista, migráns* és *liberális* 20 legközelebbi szomszédját vizsgáltam meg koszinusz távolság alapján.

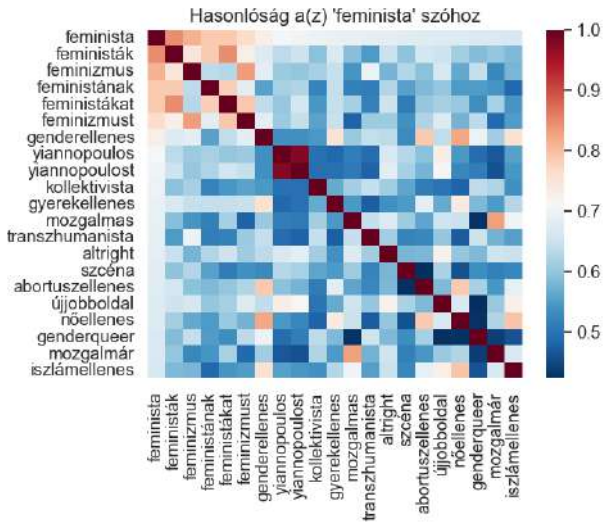
A 10. és 11. ábrán megfigyelhető, hogy természetesen mind a három szó esetében az első néhány legközelebbi szó az adott elem valamilyen ragozott alakja. Az ábrák nem csak a legközelebbi szavakat mutatják, hanem ezen szavak egymáshoz viszonyított közelségét is egy hőtérképen. Természetesen a szavak ragozott és ragozatlan alakjának kapcsolatát a beágyazások az egyhez igen közeli koszinusz-távolsággal azonosítják. Míg például a 444.hu korpuszánál a feminista szó esetén Milo Yiannopoulos (brit-amerikai feminizmus-kritikus újságíró) nevéhez fűződő szavak alacsonyabb, 0,5 körüli értéket kapnak a *gyerekellenes* vagy *transzhumanista* szavak esetében. Ugyanígy viszonylag alacsony értéket látunk a

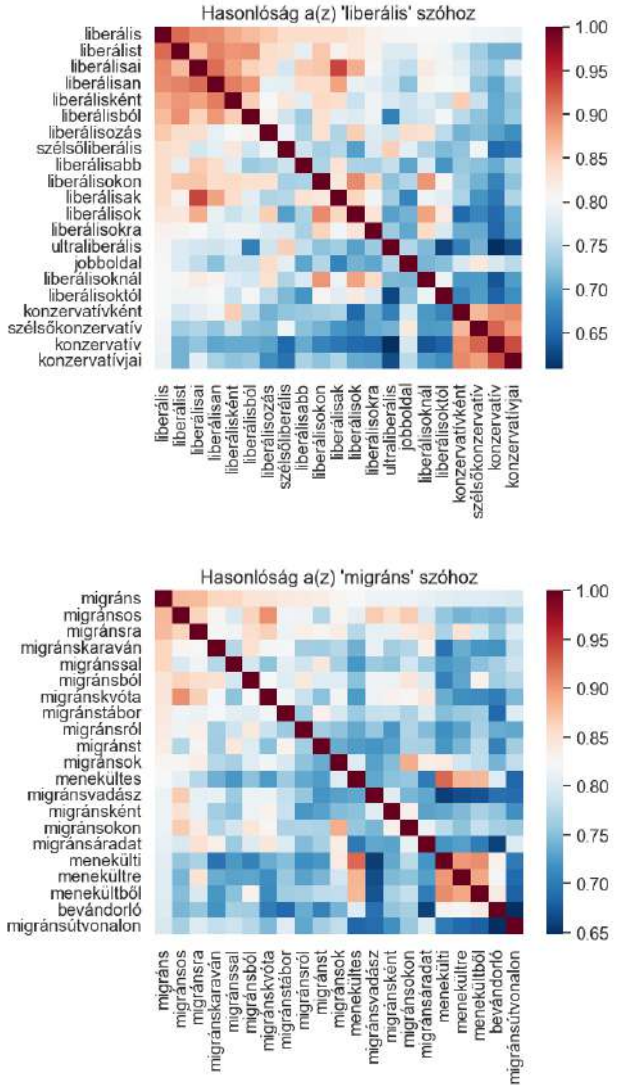
gyerekellenes és *altright* szavak vizsgálata során. Viszont a 888.hu korpuszában például a *genderqueer* és *genderlobbi* koszinusz távolsága 0,9 körüli.

A két beágyazás különbözőségeire fókuszálva a 888.hu esetén a *feminista* szónál megjelennek olyan szavak, amelyek kifejezetten jellemzőek a kormányközeli média nyelvezetére, ilyen például a *femináci*, *genderőrület* vagy *genderlobbi*. Illetve az ellenzéki, inkább baloldali sajtó szóhasználatában gyakran előforduló *altright* vagy *újjobboldal* látható a 444.hu-nál.

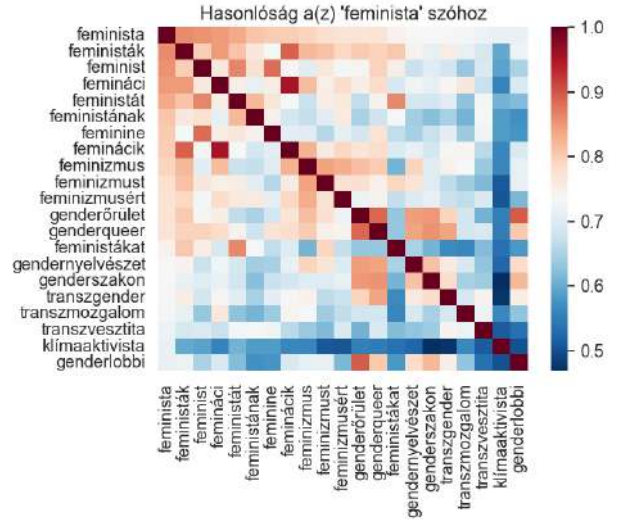
A többi szó esetén is tipikusan az adott sajtó nyelvezetére jellemző szó-eltéréseket találunk; míg a 444.hu a *migránsvadász* tokent, addig a 888.hu a *migránsbűnöző*, *illegális_bevándorló* szavakat találja a *migráns* szóhoz közelinek. Érdekes megfigyelni, hogy a 444.hu beágyazása esetén a *liberális* legközelebbi szomszédjai között szerepelnek a *konzervatív* szó különböző formái (pl.: *szélsőkonzervatív*). Ez indikálja, hogy gyakran használják hasonló vagy azonos kontextusban ezeket a szavakat.

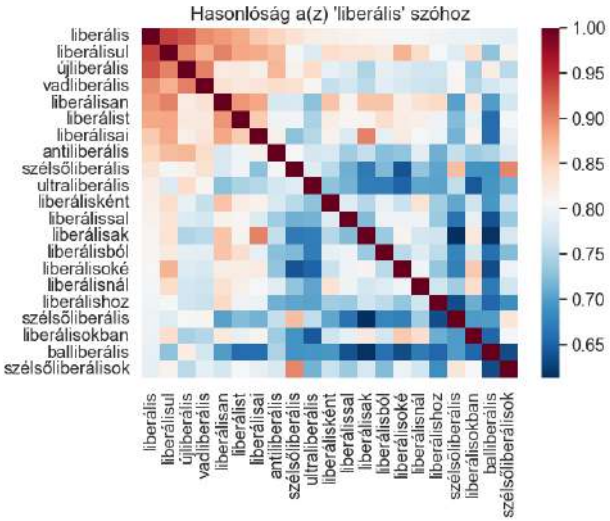
11. Ábra: Adott szavak legközelebbi szomszédai a 444.hu esetén

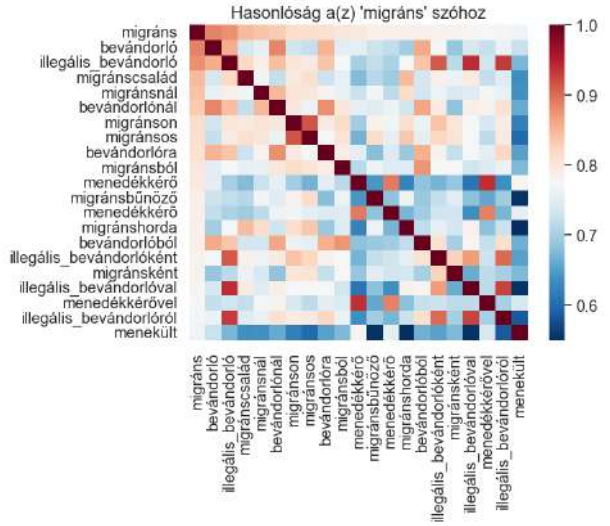




12. Ábra: Adott szavak legközelebbi szomszédai a 888.hu esetén







A teljesség igénye végett azt is megvizsgáltam, hogy 1-1 ritka szó esetén miképpen teljesítenek a modellek. A 444.hu korpuszán tanított szóbeágyazás esetén a *nyomortelepen* szó 20 legközelebbi szomszédját vizsgáltam. A beágyazás helyesen azonosítja a szó ragozott formáit, illetve szinonima szavait is (*szegénytelepen*). Továbbá közelinek azonosítja a *bérlakás, cigánytelep, hajléktalanságban* szavakat. Viszont a telep utótagból kiindulva igen magas koszinusz-távolság értéket kap például az *üdülőtelepen* vagy *ipartelepen* szó is.

8. Táblázat: A *nyomortelepen* 20 legközelebbi szomszédja a 444.hu korpuszában

nyomortelep	0,92
szegénytelepen	0,86
nyomortelepek	0,85
lakótelepen	0,81
cigánytelepen	0,80
ipartelepen	0,79
szegénytelep	0,79
telepen	0,78

Szegénytelepek	0,77
Szegénytelepeken	0,75
Szeméttelepen	0,75
Cigánytelep	0,74
Üdülőtelepen	0,73
Lakótelepe	0,73
Lakótelepeken	0,73
Cigánytelepeken	0,72

bérlakásban	0,71
nagyvásártelep	0,71

Nyomornegyedek	0,71
Hajléktalanságban	0,71

A 888.hu esetén a tortúra szót vizsgáltam. Ehhez már igen érdekes közeli szavakat rendel a koszinusz-távolság, főleg a szó morfológiai felépítése mentén. Ugyanakkor azt is látjuk, hogy a legközelebbi szó is sokkal alacsonyabb koszinusz-távolsággal rendelkezik, mint a korábban említett szavak esetében. Ez a példa remekül szemlélteti, hogy azért az n-gramokból kapott információkra épülő FastText modellnek is vannak hátrányai. Itt például a *tortúra tor* n-gramjából származik a helytelen *tortája, tornász* és *tornaszoba* szavakkal kialakított hasonlóság.

9. Táblázat: A *tortúra* 20 legközelebbi szomszédja a 888.hu korpuszában

túra	0,78
fiúra	0,75
íjász	0,70
tortája	0,69
tornász	0,66
tornaszobába	0,66
társasjáték	0,65
nagyjátékfilm	0,65
széppróza	0,65
ifjúkori	0,65
fieszta	0,65

Ökölharc	0,65
emléktúra	0,64
Cpg	0,64
osztálykirándulás	0,64
pöttyös	0,64
futballmeccs	0,64
királylány	0,63
rockopera	0,63
nagydarab	0,63

3.4. Szóbeágyazási vektorterek illesztése Prokrusztész transzformációkkal

A kiválasztott szóbeágyazások vizsgálata után végrehajtottam a vektorterek illesztését, melynek részleteit a következőkben szemléltetem.

Ahhoz, hogy a vektortereket egymásba tudjam illeszteni, először elengedhetetlen volt, hogy kiválogassam azokat az egyedi szavakat, amelyek mind a két beágyazásban megtalálhatóak. Ennélfogva létrehoztam egy közös szótárat, mely 113 838 tokent tartalmazott. Ez a szótár szolgált alapul a két szóbeágyazási modellből létrehozott szóvektor-reprezentációs mátrixhoz. Mivel 100-as dimenziójúak voltak a beágyazások, így a mátrixok mérete $113\,838 \times 100$.

Az így előállított mátrixok már alkalmas formában voltak arra, hogy egymásba illesszem őket. A 888.hu-n tanított szóbeágyazásból létrehozott mátrixot használtam referenciamátrixként a 444.hu transzformációjához. Experimentális úton jutottam erre a döntésre, alacsonyabb hibát eredményezett ez a felosztás, mind a fordítottja.

A Prokrusztész-transzformációk előtt egyfajta viszonyítási alapként végrehajtottam egy egyszerű ortogonális transzformációt. Ebben az esetben is a 444.hu korpuszából létrehozott mátrixon eszközöltem változtatást. Mivel a Prokrusztész transzformációk is Frobenius normával normalizálják a vektortereket, illetve a hibát is ez alapján határozzák meg, így ebben az esetben is így jártam el.

Mindemellett arra is kíváncsi voltam, hogy a két vektortér közelítési hibájának transzformáció előtti értéke mekkora; ez 1323,937. Amennyiben Frobenius normával normalizáljuk mátrixokat, a hiba lecsökken 1,18-ra.

Az ortogonális transzformáció végrehajtása során először mind a két mátrixot normalizáltam (N_1, N_2), majd létrehoztam a két normalizált mátrix szorzatának ($N_1^T N_2$) szinguláris értékfelbontását, amely alapján kiszámoltam a transzformációs mátrixot (T). A T és N_1 szorzatából pedig megkaptam a transzformált 444.hu korpuszából tanított szóvektor-reprezentációkat. A transzformáció közelítési hibája Frobenius normát használva 0,653.

3.4.1. A transzformációk végrehajtása és értékelése

A Prokrusztész-transzformációk értékeléséhez referenciaértékként tehát rendelkeztem a normalizált mátrixok eltérésével, illetve az ortogonálisan transzformált mátrixok közelítési hibájával.

A Prokrusztész-transzformációk végrehajtása során minden esetben normalizáltam az adatokat, viszont egyéb változtatást nem hajtottam végre. Az így kapott Frobenius norma szerinti közelítési hibákat a 10. táblázat tartalmazza.

Érdekes eredmény, hogy esetemben a transzformációs mátrixra vonatkozó megkötések nem eredményeznek alacsonyabb közelítési hibát. Különösen alkalmatlan a permutációs transzformáció a vektorterek illesztésére, de ez nem is meglepő, tekintve, hogy igen eltérő módon reprezentálják a szavakat a beágyazásaim. További izgalmas kutatás lehetne annak a vizsgálata, hogy vajon hasonló korpuszokon tanított szóbeágyazások illesztése esetén mennyiben eredményes a permutáció, mint transzformáció.

10. Táblázat: A szóvektorokat tartalmazó mátrixok transzformációjának közelítési hibája

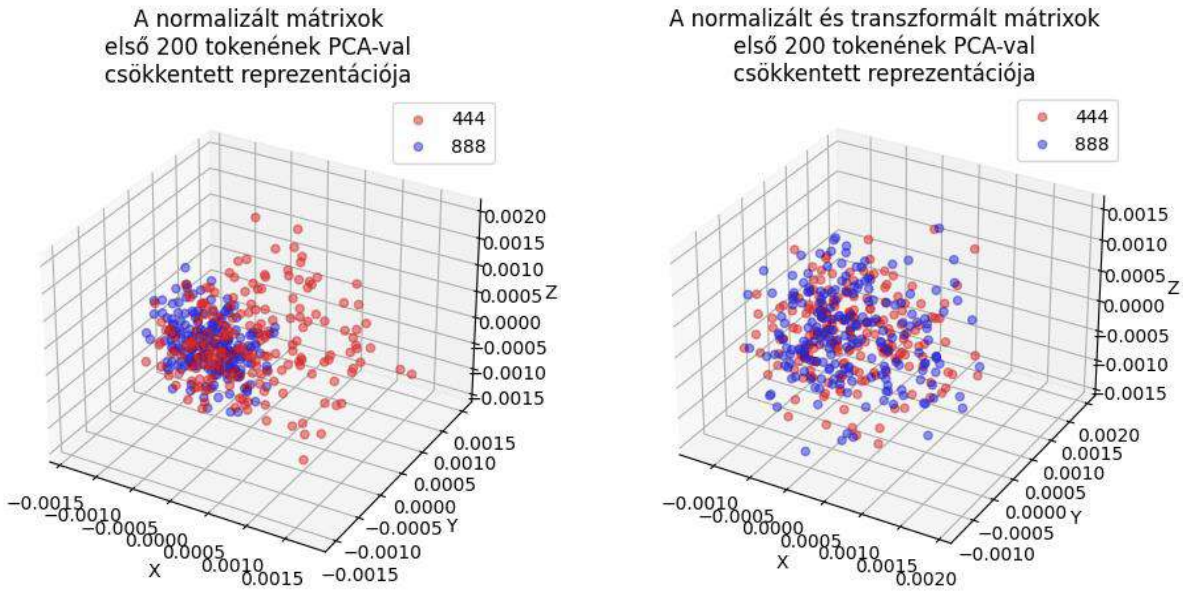
Transzformáció	**Közelítési hiba**
-	1323,937
Normalizálás	1,180
Ortogonális transzformáció normalizálással	0,653

Ortogonális Prokrusztész transzformáció normalizálással	0,426
Rotációs-ortogonális Prokrusztész transzformáció normalizálással	0,426
Permutációs Prokrusztész transzformáció normalizálással	1,246
Általános Prokrusztész transzformáció normalizálással	0,328

A közelítési hibákat alapul véve az általános Prokrusztész transzformációval kapott eredményeket fogom kiértékelni a továbbiakban.

A 12. ábrán azt szemléltetem, hogy a mátrixok első 200 elemének főkomponensanalízissel dimenziócsökkentett szóreprezentációi hogyan helyezkednek el egymáshoz képest a háromdimenziós térben. Míg az első ábrán a normalizált mátrixok elemeit látjuk, addig a második ábra az általános Prokrusztész transzformációval illesztett mátrixok pontjait szemlélteti. Látszik tehát, hogy a pontok egymásra sokkal jobban illeszkednek; ezt a későbbiekben konkrét szavak reprezentációi során is szemléltetni fogom.

13. Ábra: Az általános Prokrusztész transzformáció eredménye



3.4.2. Az illesztés elemzése, szavak kontextusának vizsgálata

A szóbeágyazási vektorterek általános Prokrusztész transzformációval történő illesztése lehetőséget kínált arra, hogy a szótárban lévő szavak két vektorreprezentációval is rendelkezzenek. Ez biztosította a reprezentációk közötti különbségek (vagy éppen hasonlóságok) vizsgálatát, melynek bemutatása jelen fejezet feladata.

Az egymásba illesztett vektorterek elemzéséhez többfajta megközelítést is alkalmaztam. Egyrészről próbáltam egy általánosabb képet adni oly módon, hogy kiszámoltam a két vektorreprezentáció koszinusz-távolságát minden szó esetén. A későbbiekben inkább téma-

specifikusabb, kvalitatív elemzést végeztem. A két reprezentáció vizsgálatához analógiákat, kapcsolatmutatókat, illetve a vizualizációhoz dimenziócsökkentést használtam.

A 11. és 12. táblázat a legalacsonyabb és legmagasabb koszinusz-távolsággal rendelkező szavakat jeleníti meg. Ebben az esetben tehát a távolság csupán az azonos szavak között vizsgáltam, vagyis a táblázatban látható például *beszélgetéséről* szó két vektora közötti eltérést néztem.

Általánosságban elmondható, hogy olyan szavak koszinusz-távolsága lett feltűnően magas, amelyek valamiféle módon a közlést kívánják kifejezni (*beszélgetésről, kijelentésről, egyeztetni, kijelentéséről*). Ez nem is meglepő, tekintve, hogy ezek a tokenek valószínűleg gyakran jelentek meg a korpuszok esetén, viszonylag neutrálisabb környezetben. Izgalmas továbbá, hogy a sport és különösen a labdarúgáshoz köthető szavak között is magasabb koszinusz távolság fedezhető fel.

11. Táblázat: Az 50 legmagasabb koszinusz távolsággal rendelkező token

Token	Koszinusz távolság
beszélgetéséről	0,957
beszélgetésről	0,953
megmozdulásokra	0,953
válogatottól	0,952
fejlesztésének	0,952
beszélgetésekről	0,951
megbeszéléseiről	0,951
fejlesztésről	0,951
meccsig	0,951
ellátásokra	0,949
vállalattal	0,949
fejlesztéséről	0,949
megbeszéléséről	0,948
továbbjutott	0,948
egyeztetésekről	0,948
válogatottbeli	0,948
látogatásáról	0,947
fejlesztése	0,947
fizetésnek	0,947
támogatására	0,946
ellátásokat	0,946
meccsünk	0,946
fejlesztésekre	0,945
világbajnokságok	0,945
kijelentéséről	0,945

Token	Koszinusz távolság
egyeztetni	0,945
válogatottba	0,944
jelentéséről	0,944
bűncselekményekkel	0,944
fejlesztésekhez	0,944
együttműködésekről	0,944
felújítására	0,944
immunbiológiai	0,944
történeteket	0,944
válogatottból	0,944
megbeszélésekről	0,944
továbbjutottak	0,944
tüntetésekről	0,944
korlátozásokra	0,944
válogatotthoz	0,944
fegyvereket	0,943
világbajnokságnak	0,943
egyeztetések	0,943
szóló	0,943
szögletre	0,943
vesztették	0,943
fegyházbüntetéssel	0,943
szállították	0,943
csökkentésére	0,942
fejlesztésekről	0,942

Ezzel szemben a legalacsonyabb koszinusz távolsággal rendelkező szavak éppen az idegen- vagy csoportnyelvi szavak (*abcúg, halal, böszme, ergo, kehi*) melyek vélhetőleg nem túl gyakran fordultak elő. Emellett kifejezetten sok tulajdonnév (*máthé, rubicon, cécile, bertha*) is megjelenik ebben a listában, feltételezhetően szintén a ritkaságuk miatt.

Tehát azokat a szavakat, melyeket maguk a szóbeágyazási modellek sem tudtak megfelelő kontextussal ellátni ritkaságukból adódóan, az illesztés sem tudta „párosítani", ennélfogva igen távol helyezkedtek el egymástól. Vagyis ezek a távolságok nem feltétlenül a két korpusz nyelvezetének különbözőségeiből fakadnak. Ez utóbbi vizsgálatára a specifikusabb szavak elemzése hatékonyabb megoldásnak bizonyult.

12. Táblázat: Az 50 legalacsonyabb koszinusz távolsággal rendelkező token

Token	**Koszinusz távolság**
rubicon	0,261
répás	0,353
ismeretes	0,357
hírei	0,368
plafon	0,374
junge	0,374
máthé	0,382
lankad	0,387
véleményrovatának	0,391
sztavrosz	0,393
anker	0,393
zoom	0,398
holm	0,398
abcúg	0,403
rakás	0,405
kuna	0,406
vivő	0,406
maxim	0,408
fuentes	0,408
techno	0,410
csakúgy	0,410
halal	0,410
kaminski	0,411
cécile	0,412
spöttle	0,413

Token	**Koszinusz távolság**
Mixet	0,416
Frekvencián	0,416
Burmai	0,418
Legfőképp	0,419
Bertha	0,420
Tereskova	0,420
Idevágó	0,422
Istálló	0,423
Tarifa	0,423
Kristály	0,425
Akác	0,426
Kehi	0,427
Pákozdi	0,429
Ergo	0,434
Hoppá	0,434
Böszme	0,434
Lagzi	0,434
Fűrész	0,437
Windisch	0,438
Czeglédi	0,440
Bede	0,443
Azad	0,444
Gődény	0,447
Giphy	0,448
Zubin	0,449

Ahogy korábban már bemutatásra került, az analógiák igen alkalmas eszközök a szavak közötti kapcsolatok vizsgálatára. Ennen eredményeképp specifikus analógiai kérdések mentén elemeztem a vektorreprezentációkat.

Először arra voltam kíváncsi, hogy a két vektortér külön-külön nem vesztett-e kontextust érintő információt az illesztés során. Így olyan analógiai feladatokat eszközöltem, melyeket korábban mind a két szóbeágyazás esetén megnéztem.

Ezért megvizsgáltam, hogy a *budapest – magyarország + oroszország* feladatra milyen megoldást talál a két különböző reprezentáció. Ebben az esetben tehát mind a négy (három megadott és egy keresendő) token vagy a 444.hu vagy a 888.hu korpuszából került kiválasztásra. Mind a két esetben helyes találat volt a *moszkva*.

A következő analógia a *budapest – magyarország + kína* volt. Korábban a 444.hu korpuszán tanított szóbeágyazás ennek a feladatnak a megoldására helytelen választ adott (*szöul*), amit ebben az esetben is láttunk. Tehát az illesztés se nem javított, se nem rontott az eredményen, hiszen a 888.hu reprezentációja helyesen azonosította *peking*-et. A *budapest – magyarország + ausztria* feladat esetén pedig helyesen felelt mind a két korpusz mind az illesztett, mind az illesztetlen esetben.

A következő feladat egy korpusz-specifikus analógiai kérdés volt; *miniszterelnök -orbán_viktor + szijjártó.* Az illesztetlen szóbeágyazási modellek helyesen feleltek erre a kérdésre, ugyanakkor a 888.hu az illesztett esetben nem találta meg helyesen a *külügyminiszter* szót. Tehát itt történt információvesztés.

Ugyanakkor arra is kíváncsi voltam, hogy amennyiben az egyik korpusz adja meg az analógiai feladatot, de a megoldást a másik korpuszból keresem ki, milyen eredményre jutok. A továbbiakban ezt szemléltetem a korábban ismertetett feladatokon, melynek eredményei a 13. táblázatban láthatók.

13. Táblázat: Analógiai feladatok eltérő feladat és megoldás korpusszal

Analógia	**Feladat: 444.hu, megoldás: 888.hu**	**Feladat: 888.hu, megoldás: 444.hu**
budapest – magyarország + oroszország	moszkva	moszkva
budapest – magyarország + kína	csungking	peking
budapest - magyarország + ausztria	strandfürdő	bécs
miniszterelnök - orbán_viktor + szijjártó	külügyminiszterrel	külügyminiszter

Annyi fejlődést látunk, hogy itt már mind a két esetben megtalálta a modell a külügyminiszer szót – még ha ragozott formában is. Ugyanakkor amennyiben a 444.hu korpuszából keressük ki a *budapest, magyarország* és *ausztria* szavakat, az analógiai kérdés megoldása *bécs* helyett *strandfürdő* lesz. Illetve *csungking*-gel annyival közelebb lettünk, hogy legalább kínai várost talált meg a modellünk.

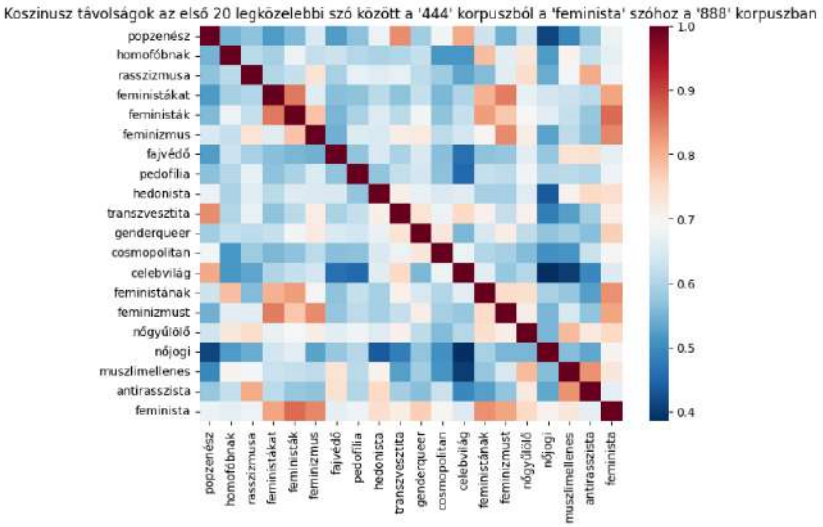
14. Táblázat: Analógia feladatok vegyes feladat és megoldás korpusszal

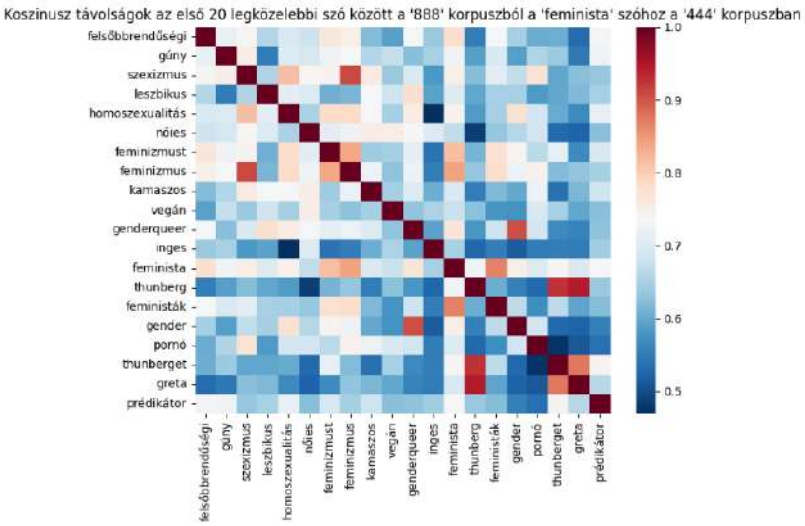
Analógia	**Feladat: vegyes, megoldás: 888.hu**	**Feladat: vegyes, megoldás: 444.hu**
budapest – magyarország + oroszország	moszkva	moszkva
budapest – magyarország + kína	jinping	peking
budapest - magyarország + ausztria	strandfürdő	bécs
miniszterelnök - orbán_viktor + szijjártó	külügyminisztériummal	külügyminiszter

A 14. táblázat pedig a vegyes felosztást szemlélteti, tehát a feladathoz szükséges szavakat vegyesen a 444.hu és a 888.hu korpuszából vesszük, a megoldást pedig a táblázat szerinti módon. Itt a *budapest – magyarország + kína* feladaton megint csak elbukik a 888.hu-ból kapott eredmény, a 13. táblázatban szemléltetett felosztáshoz képest még inkább, hiszen ez esetben nem is városnevet kapunk.

Ezek után a szóbeágyazások elemzése során már bemutatott módon megnéztem három, korábban már elemzett szó legközelebbi szomszédját, és azok egymással való koszinusz távolságát. Ezt a 13., 14. és 15. hőtérképes ábra mutatja. Ahogy az ábrák címei is szemléltetik, az adott szót a megadott korpuszból kerestem ki, viszont ennek legközelebbi szomszédjait a másik korpuszból.

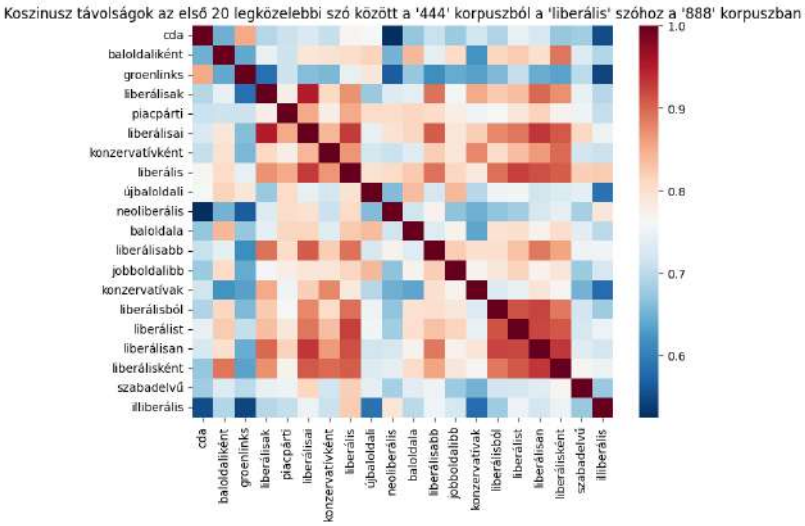
14. Ábra: Legközelebbi szomszédok a feminista szó esetén eltérő korpuszokból

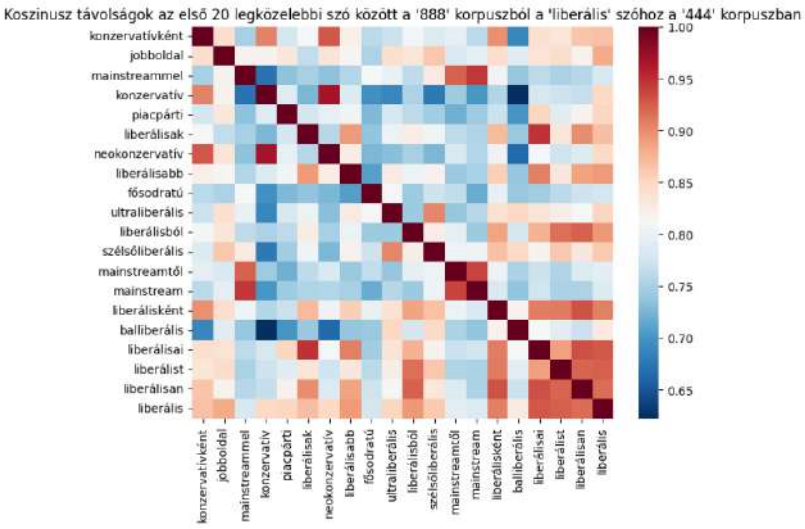




A 13. ábra alapján elmondható, hogy a *feminista* szó 444.hu-s reprezentációja a 888.hu-ban olyan tokenekhez áll a legközelebb, amelyek jelentős része igen pejoratív (*rasszizmus, homofób, fajvédő, pedofil, muszlimellenes, nőgyűlölő*), vagyis ezek a szavak hasonló kontextusban szerepelnek a 888.hu-ban, mint a 444.hu korpuszában a *feminista* szó. Ugyanakkor ezek a szavak egymással nem mutatnak jelentős hasonlóságot. Tehát vélhetőleg igen más a kontextusa a két szónak a két médiumban. Fordított esetben is jellemző az a tendencia, hogy a *feminista* tokenhez közeli szavak koszinusz-távolsága egyáltalán nem magas. Ezzel szemben a 888.hu-s reprezentációja a feminizmusnak a 444.hu-ban olyan szavakhoz áll közel, mint *felsőbbrendűségi* vagy *gúny*, illetve Greta Thunberg, klímaaktivista neve is felmerül.

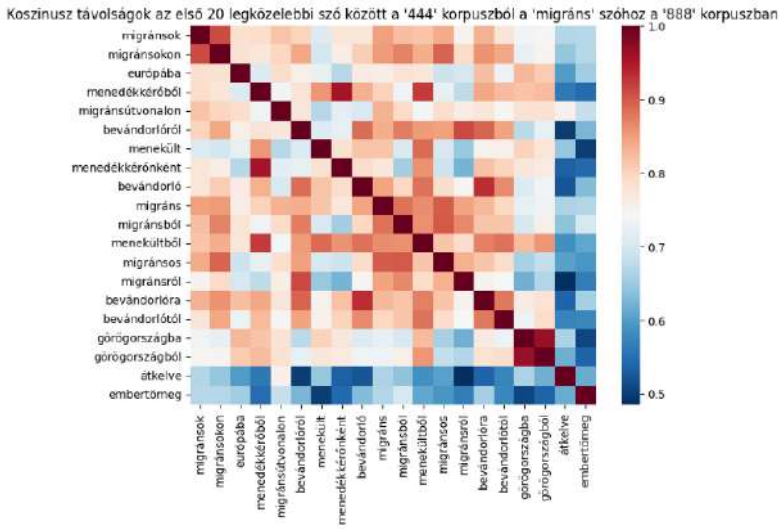
15. Ábra: Legközelebbi szomszédok a liberális szó esetén eltérő korpuszokból

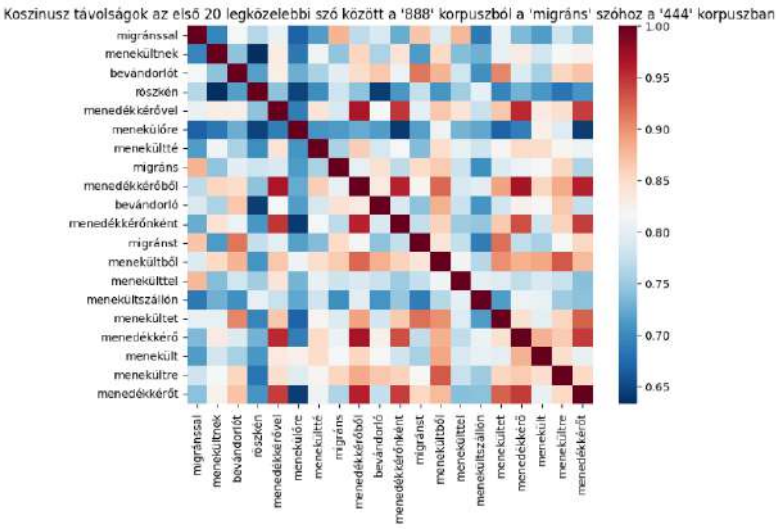




A *liberális* szó esetén sokkal hasonlóbbak az eredmények, mint a feminizmus szó esetén. Itt azért több, mindkét ábrán megjelenő tokent találunk, és a hőtérképen is megfigyelhető magasabb értékek is mutatják, hogy közelebb áll egymáshoz a két vektorreprezentáció. A 444.hu-ból érkező *liberális* vektorreprezentációja olyan szavakkal mutat magas koszinusz távolságot a 888.hu korpuszában, mint *neoliberális, piacpárti, baloldali, újbaloldali, szabadelvű.* Ezzel szemben a 888.hu-s *liberális* reprezentáció a 444.hu korpuszában olyan szavakhoz áll közel, mint *mainstream, szélsőliberális, balliberális*.

16. Ábra: Legközelebbi szomszédok a migráns szó esetén





A *migráns* szó esetében igen kicsi különbség tételezhető fel a két ábra között. Nagyon hasonló szavakkal talál hasonlóságot mind a két reprezentáció. Ugyanakkor, ha megvizsgáljuk a hőtérképet, látjuk, hogy a 444.hu korpuszából származó *migráns*-reprezentációhoz köthető 888.hu-s tokenek jóval magasabb koszinusz távolságot mutatnak egymással, mint fordított esetben.

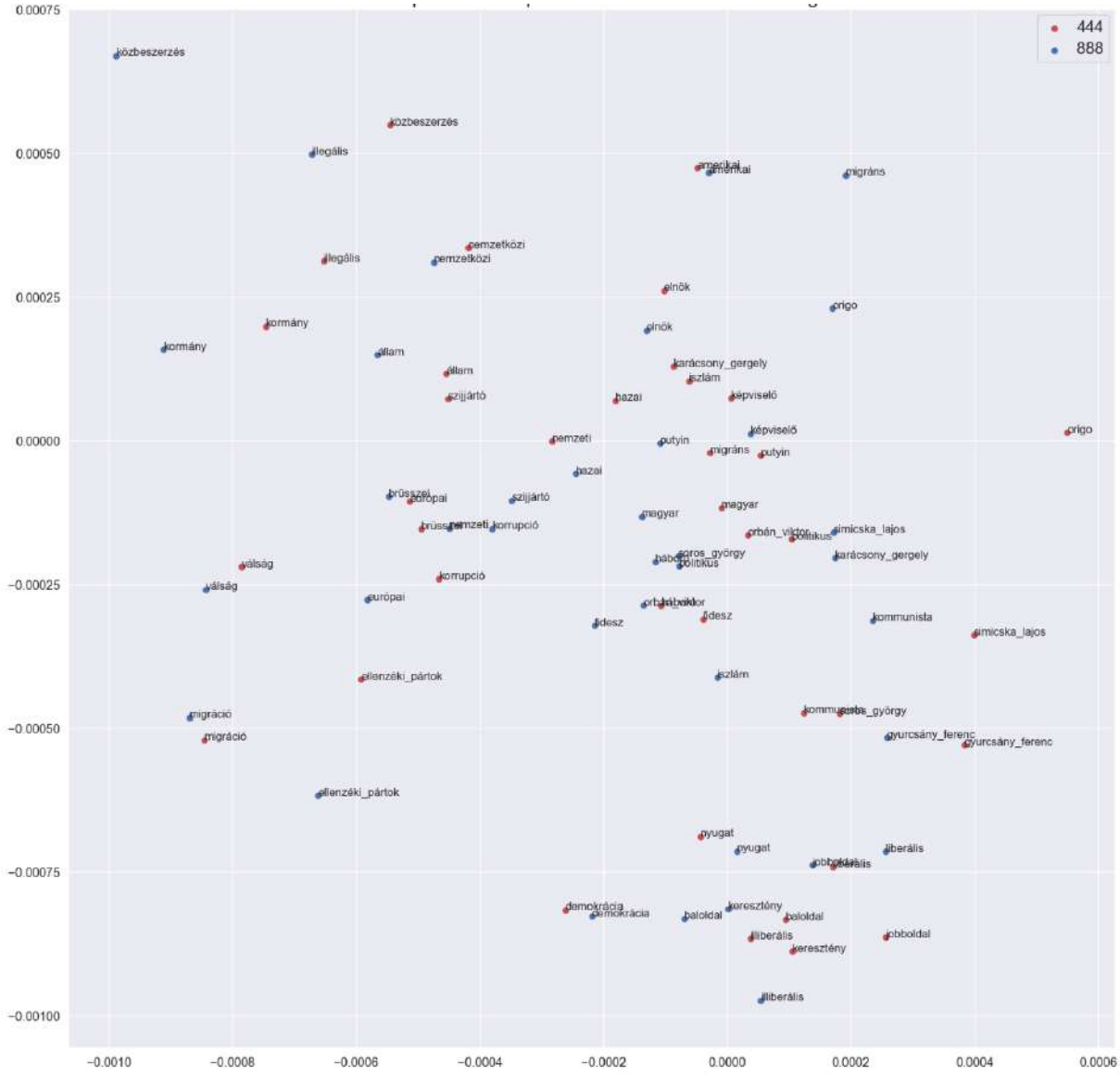
Tehát a fent bemutatott ábrák mentén igazolható, hogy téma-specifikus szavak esetén már jelentősebb módon térnek el egymástól a korpuszok, mint általános tokeneket vizsgálva. A továbbiakban is ezt szeretném bemutatni.

Az eredményesebb vizualizáció érdekében a jelenleg használt 100 dimenziós vektorokat kettőre csökkentettem. Kétfajta módszert is kipróbáltam: először a korábbiakban bemutatott T-SNE metódust alkalmaztam, azonban ez igen számításigényes volt, ennélfogva végül a jóval egyszerűbb és gyorsabb főkomponensanalízist választottam. Mindemellett további kérdéseket vetett fel, hogy mely szavakat érdemes megjeleníteni, hiszen a szótáram méretéből adódóan nem ábrázolhattam az összeset. Így kialakítottam két különböző téma-specifikus szóhalmazt, melynek eredményei a 16. és 17.-es ábrán láthatóak.

A 16. ábra olyan szavakat foglal magába, amelyek a bel- és külpolitikához kapcsolódnak. Ahogy már korábban kifejtettem, tengelyek értelmezése dimenziócsökkentés esetén abszolút nem evidens folyamat, illetve közel sem egyértelmű. Ugyanakkor a szavak jelentős többségének elhelyezkedése magyarázható a tengelyek alábbi definiálásával: az vízszintes tengely tekinthető, balról jobbra haladva, a köz – egyén (közügy-egyéni ügy) tengelyének, ahol a köz olyan fogalmakat takar, mint *kormány, válság, közbeszerzés, migráció, európai, brüsszel,*

állam, míg az egyén jobbára konkrét vagy általánosan vett egyéneket jelöl (*putyin, orbán_viktor, simicska_lajos, gyurcsány_ferenc, soros_györgy, migráns*). Az is elmondható, hogy az egyén általában pozitív értékeket kap, míg a köz, vagy közügyek jobbára negatívakat.

17. Ábra: Bel- és külpolitikához kapcsolódó szavak vizsgálata



A vertikális tengely pedig meghatározható a belügy (belpolitikai nézetek) – külügy tengelyeként. A belügyhöz az ország politikáját meghatározható irányzatok és nézetek tartoznak leginkább (*jobboldali, baloldali, liberális, illiberális, demokrácia, keresztény*), de az *ellenzéki_pártok* és a *fidesz* is ide sorolható. Ezzel szemben a külügyhöz olyan tokenek tartoznak, mint a *nemzetközi, amerikai, putyin, elnök, migráns,* és az *illegális,* ami valószínűleg a *bevándorló* és *migráns* szavak kontextusával lehet rendkívül hasonló.

Ha a szavak reprezentációinak hasonlóságát vizsgáljuk, érdemes megemlíteni, hogy a 888.hu korpuszából kapott *illegális* reprezentációja igen közel helyezkedik el a 444.hu *közbeszerzés* szavához. Amellett a *brüsszel* (888.hu) és *európai* (444.hu) szavak is egyben csoportosulnak.

Általánosságban elmondható, hogy a tulajdonnevek (*simicska_lajos, orbán_viktor, karácsony_gergely, szijjártó*) közötti távolságok jóval nagyobbak, mint a fogalmakat jelölő szavaké (*migráció, válság, demokrácia, nyugat*). Ugyanakkor igen szembetűnő, hogy a *migráns*, *iszlám, közbeszerzés* szavak kontextusa merőben eltér a két korpuszban. Illetve érdemes azt is kiemelni, hogy a 888.hu korpuszából kapott *orbán_viktor* reprezentáció szinte teljesen egyazon helyen van a 444.hu korpuszából meghatározott *háború* szóval.

A *soros_györgy* token is igen távol áll egymástól a különböző korpuszok esetén, míg a 888.hu inkább a *háború* vektorreprezentációjával azonosítja, addig a 444.hu *gyurcsány_ferenc* és *kommunista* tokenekkel.

A 17. ábra pedig az elmúlt években gyakran vitatott társadalmi nemek tudományához és a gyermekekkel kapcsolatos politikai intézkedésekhez köthető szavakat csoportosítja. Itt a tengelyek az alábbiképpen helyezik el a különböző szavakat; míg az X tengely negatív értékei jobbára kollektivista szemléletet tükröznek (*családtámogatás, gyermekvédelem, csok, demográfia, oktatás, házasság*), a társadalmat, mint egészet érintő szavak kerültek ide, addig a pozitív értékek inkább egyénközpontú szavakat foglalnak magukba (*hallgató, anya, apa, tanár*). Illetve ez utóbbiba sorolja a feminizmust, a szexuális irányultságot és nemi identitást jelölő szavakat is.

Az Y tengely pedig megjelölhető egyfajta magánélet – közélet orientációban (lentről felfelé). Míg a közéletet reprezentáló szavak, mint a *bme, egyetem, hallgató, csok* tokenekhez pozitív értékek tartoznak, addig a magánéletet reprezentáló (*házasság, családtámogatás, gyermekvállalás, szex*) szavakhoz negatívak.

Ami a szavak egymáshoz való közel és távolságát illeti, az egyik legszembetűnőbb eredménye a modellnek, hogy a 444.hu korpuszában a *nők* és *erőszak* tokenek szinte teljesen lefedik egymást, tehát a két szót a médium rendkívül hasonló kontextusban használja. De a *szexuális, zaklatás, család, apa, anya* és *gyerek* szavak is mind egymáshoz közel helyezkednek el a kétdimenziós térben mind a két korpusz esetében. Illetve maga a két korpusz is rendkívül

hasonló ezeknek a szavaknak a reprezentációját illetően. Ezzel szemben az *anyaság* és *házasság* szavak a két korpusz esetén távolabb találhatóak egymástól.

Az is elmondható az ábra alapján, hogy míg a 444.hu a *felsőbbrendű* kontextusához a *gender* kontextusát hasonlítja, a 888.hu a *patriarchálist*. Illetve a *queer* és *feminista* szavak majdnem azonos vektorreprezentációt kaptak a kétdimenziós térben a 888.hu esetén.

18. Ábra: Család- és oktatáspolitikához, a társadalmi nemek tudományához kapcsolódó szavak vizsgálata

Tehát téma-specifikus szavak mentén van néhány jelentős eltérés a két korpusz között, a vártnál több a hasonlóság is, főleg az elvont fogalmakat jelölő szavak kapcsán, ami adódhat a politikai fókuszból fakadó homogenitásból is.

Mindemellett az is megállapítható, hogy a Prokrusztész transzformációval történő illesztés jobbára alkalmas két eltérő korpuszon tanított szóbeágyazás közös térben való elhelyezésére, ám a dimenziócsökkentés tovább redukálja az amúgy is alacsony korpuszok közötti eltéréseket. Mivel már az illesztés során jelentős mértékben csökkentjük a távolságokat – melyeket később vizsgálni szeretnénk – a dimenzióredukció pedig ezt tovább csökkenti. További – e dolgozatban nem tárgyalt – kérdés lehet, hogy vajon egy szofisztikáltabb, vagy nem lineáris dimenziócsökkentési modell alkalmasabban tudja-e ezeket az apró eltéréseket identifikálni.

4. Összefoglalás, értékelés, a dolgozat korlátai

Dolgozatomban tehát megkíséreltem különböző korpuszokon tanított szóbeágyazások vektortereit egymásba illeszteni, majd a közös vektorteret elemezni, megadott analógiák és kapcsolatmutatók mentén. Emellett a kiválasztott szóbeágyazások tanításának és értékelésének vizsgálatára is sor került. Ez utóbbira azért volt szükség, hogy szemléltetésre kerüljön az alábbi probléma: különböző szóbeágyazások összehasonlítása, főként ezen összehasonlítás számszerűsítése igen fogós feladat, az eltérő dimenziójelentésből fakadóan.

Dolgozatom során úgy találtam, hogy az egyes Prokrusztész-transzformációk alkalmasak ezen probléma megoldására, ugyanakkor ehhez elengedhetetlen a szóbeágyazási modellek lehető legeredményesebb tanítása. Aprónak tűnő, de jelentős eltéréseket eredményezhet akár már az adattisztítás során eszközölt folyamatok változtatása. A dolgozatom egyik fő korlátja talán éppen ebben a problémában azonosítható; érdemes lett volna a korpuszok mélyebb tisztítása, például egy másodikkörös stopszavazás, vagy az igen ritka és jelentéktelen szavak eltávolítása, főként a 444.hu esetén, amelytől jobb eredményeket vártam a 888.hu-val szemben a szóbeágyazások tanítása során, tekintve, hogy jelentősen több tokent tartalmazott.

Továbbá érdemes azt is kiemelni, hogy az illesztés során számtalan akadályba ütköztem. Az egyik legjelentősebb abból fakadt, hogy a közös szótárral való bárminemű művelet kifejezetten számításigényes volt. Próbáltam különböző típusú tárolást is alkalmazni, ehhez Pandas és Numpy csomagot használtam, de egyik se bizonyult jelentősen gyorsabbnak a másiknál. További kutatást igényelne, hogy miképpen érhető el a számításigény redukálása.

Ahogy az a 3.4.2 alfejezetben kifejtésre került, az általam használt PCA-val történő dimenziócsökkentés jóval nehezebben ragadja meg az összeillesztéssel lecsökkentett

eltéréseket a két korpusz között, így további kutatási lehetőségként merült fel a különböző dimenziócsökkentő módszerek összehasonlítása az adott problémára.

Az előző fejezetben látható az illesztett, közös vektortér részletes elemzése, a szavak kétféle reprezentációjának összehasonlítása, mely a dolgozatom legfontosabb célja volt. Az értékelés többnyire kvalitatív módon zajlott. Ennek fő eredménye, hogy bizonyos téma-specifikus szavak valóban jelentősen eltérő reprezentációval rendelkeznek; ez különösen igaz a politikával kapcsolatba hozható tulajdonnevek, és a nemi identitással kapcsolatos fogalmak esetén. A másik jelentős korlátja a dolgozatomnak pontosan az elemzés kvalitatív mivoltából fakad, így további kutatási irányként egyértelműen azonosítható a vektorreprezentációk kvantitatív értékelése.

Mindemellett érdekes direktíva lehet az egy térbe illesztett vektorreprezentációk felhasználása arra, hogy egy „objektív” reprezentációt hozzunk létre a szavakhoz, például a vektorok átlagolásával. Ez esetleg javítani tudna az analógiai feladatokban elért teljesítményen is.

5. IRODALOMJEGYZÉK

1. Arcanum Adatbázis Kiadó. é. n. „Ki kicsoda az antik mítoszokban - Prokrusztész”. Arcanum - Kézikönyvtár (blog). https://www.arcanum.com/hu/online-kiadvanyok/Lexikonok-ki-kicsoda-az-antik-mitoszokban-F869D/p-F8BBC/prokrusztesz-F8C77/.
2. Artetxe, Mikel, Gorka Labaka, és Eneko Agirre. 2019. „Bilingual Lexicon Induction through Unsupervised Machine Translation”. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, 5002–7. https://doi.org/10.18653/v1/P19-1494.
3. Bátorfy, Attila. 2017. „Az állam foglyul ejtésétől a piac fogvatartásáig: Orbán viktor és a kormány médiamodellje 2014 után”. 18 Médiakutató (1): 7–30.
4. Bengio, Yoshua, Réjean Ducharme, Pascal Vincent, és Christian Jauvin. 2003. „A Neural Probabilistic Language Model”. Journal of Machine Learning Research 3.
5. Béni, Alexandra. 2020. „Párduc, oroszlán, gorilla?: Afrika médiareprezentációja a vezető magyar hírportálok tükrében”. Afrika Tanulmányok / Hungarian Journal of African Studies 13 (3-4.): 31–44. https://doi.org/10.15170/AT.2019.13.3-4.2.
6. Bird, Steven, Ewan Klein, és Edward Loper. 2009. Natural Language Processing with Python. Cambridge: O’Reilly.
7. Bojanowski, Piotr, Onur Celebi, Tomas Mikolov, Edouard Grave, és Armand Joulin. 2019. „Updating Pre-trained Word Vectors and Text Classifiers using Monolingual Alignment”. arXiv. http://arxiv.org/abs/1910.06241.
8. Bojanowski, Piotr, Edouard Grave, Armand Joulin, és Tomas Mikolov. 2016. „Enriching Word Vectors with Subword Information”. https://doi.org/10.48550/ARXIV.1607.04606.
9. Camacho-Collados, Jose, és Mohammad Taher Pilehvar. 2017. „On the Role of Text Preprocessing in Neural Network Architectures: An Evaluation Study on Text Categorization and Sentiment Analysis”. https://doi.org/10.48550/ARXIV.1707.01780.
10. Choi, Jaekeol, és Sang-Woong Lee. 2020. „Improving FastText with Inverse Document Frequency of Subwords”. Pattern Recognition Letters 133 (május): 165–72. https://doi.org/10.1016/j.patrec.2020.03.003.
11. Di Gennaro, Giovanni, Amedeo Buonanno, és Francesco A. N. Palmieri. 2021. „Considerations about Learning Word2Vec”. The Journal of Supercomputing 77 (11): 12320–35. https://doi.org/10.1007/s11227-021-03743-2.

12. Domonkos, Tikk. 2007. Szövegbányászat. Budapest: Typotex.
13. Doval, Yerai, Jose Camacho-Collados, Luis Espinosa-Anke, és Steven Schockaert. 2018. „Improving Cross-Lingual Word Embeddings by Meeting in the Middle”. arXiv. http://arxiv.org/abs/1808.08780.
14. Ekštein, Kamil, František Pártl, és Miloslav Konopík, szerk. 2023. Text, Speech, and Dialogue: 26th International Conference, TSD 2023, Pilsen, Czech Republic, September 4-6, 2023, Proceedings. Cham: Springer.
15. Evangelopoulos, Nicholas E. 2013. „Latent Semantic Analysis”. WIREs Cognitive Science 4 (6): 683–92. https://doi.org/10.1002/wcs.1254.
16. Evans, James A., és Pedro Aceves. 2016. „Machine Translation: Mining Text for Social Theory”. Annual Review of Sociology 42 (1): 21–50. https://doi.org/10.1146/annurev-soc-081715-074206.
17. Gower, J. C., és Garmt B. Dijksterhuis. 2004. Procrustes Problems. Oxford: Oxford University Press.
18. Gower, John C. 2010. „Procrustes Methods”. WIREs Computational Statistics 2 (4): 503–8. https://doi.org/10.1002/wics.107.
19. Gruppi, Maurício, Pin-Yu Chen, és Sibel Adali. 2021. „Fake it Till You Make it: Self-Supervised Semantic Shifts for Monolingual Word Embedding Tasks”. Proceedings of the AAAI Conference on Artificial Intelligence 35 (14): 12893–901. https://doi.org/10.1609/aaai.v35i14.17525.
20. Harris, Charles R., K. Jarrod Millman, Stéfan J. Van Der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, és mtsai. 2020. „Array Programming with NumPy”. Nature 585 (7825): 357–62. https://doi.org/10.1038/s41586-020-2649-2.
21. Harris, Zellig S. 1954. „Distributional Structure”. WORD 10 (2–3): 146–62. https://doi.org/10.1080/00437956.1954.11659520.
22. Hunter, John D. 2007. „Matplotlib: A 2D Graphics Environment”. Computing in Science & Engineering 9 (3): 90–95. https://doi.org/10.1109/MCSE.2007.55.
23. Hurley, John R., és Raymond B. Cattell. 2007. „The Procrustes Program: Producing Direct Rotation to Test a Hypothesized Factor Structure”. Behavioral Science 7 (2): 258–62. https://doi.org/10.1002/bs.3830070216.

24. Jurafsky, Daniel, és James Martin. 2023. Speech and Language Processing An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition. Third Edition draft. https://web.stanford.edu/~jurafsky/slp3/ed3book.pdf.
25. Kmetty, Zoltán. 2022. „Szóbeágyazási vektortérmodellek társadalomtudományi alkalmazása". Statisztikai Szemle 100 (2): 105–36. https://doi.org/10.20311/stat2022.2.hu0105.
26. Kubin, Emily, és Christian Von Sikorski. 2021. „The Role of (Social) Media in Political Polarization: A Systematic Review". Annals of the International Communication Association 45 (3): 188–206. https://doi.org/10.1080/23808985.2021.1976070.
27. Maaten, Laurens van der, és Geoffrey Hinton. 2008. „Viualizing data using t-SNE". Journal of Machine Learning Research 9 (2579–2605).
28. Makrai, Márton. 2015. „Comparison of distributed language models on medium-resourced languages". Előadás IV. Magyar Számítógépen Nyelvészeti Konferencia, Szeged. http://corpus.nytud.hu/efnilex-vect/pub/makrai15-lintrans.pdf.
29. Mandiner. 2015. „Elindult a kormánypárti fenegyerek portál, a 888.hu". Belföld, 2015. 0 18. https://mandiner.hu/belfold/2015/09/elindult-a-kormanyparti-fenegyerek-portal-a-888-hu.
30. McKinney, Wes. 2010. „Data Structures for Statistical Computing in Python". In , 56–61. Austin, Texas. https://doi.org/10.25080/Majora-92bf1922-00a.
31. Meng, Fanwang, Michael Richer, Alireza Tehrani, Jonathan La, Taewon David Kim, Paul W. Ayers, és Farnaz Heidar-Zadeh. 2022. „Procrustes: A Python Library to Find Transformations That Maximize the Similarity between Matrices". Computer Physics Communications 276 (július): 108334. https://doi.org/10.1016/j.cpc.2022.108334.
32. Mészáros, Evelin, és Miklós Sebők. 2018. „A szövegbányászati módszerek alkalmazásának lehetőségei a joggyakorlat-elemzésben". Forum Sententiarum Curiae, 6–12.
33. Mikolov, Tomas, Kai Chen, Greg Corrado, és Jeffrey Dean. 2013. „Efficient Estimation of Word Representations in Vector Space". https://doi.org/10.48550/ARXIV.1301.3781.
34. Mikolov, Tomas, Ilya Sutskever, Kai Chen, Greg Corrado, és Jeffrey Dean. 2013. „Distributed Representations of Words and Phrases and their Compositionality". https://doi.org/10.48550/ARXIV.1310.4546.

35. Muhi, Kristóf, és Zsolt Csaba Johanyák. 2020. „Gépi tanulásban alkalmazott dimenziószám-csökkentő módszerek”. Műszaki Tudományos Közlemények 13: 148–51. https://doi.org/10.33895/mtk-2020.13.27.
36. Naseem, Usman, Imran Razzak, Shah Khalid Khan, és Mukesh Prasad. 2021. „A Comprehensive Survey on Word Representation Models: From Classical to State-of-the-Art Word Representation Language Models”. ACM Transactions on Asian and Low-Resource Language Information Processing 20 (5): 1–35. https://doi.org/10.1145/3434237.
37. Nemeskey, Dávid Márk. 2020. „Egy emBERT próbáló feladat”. Előadás XVI. Magyar Számítógépes Nyelvészeti Konferencia, Szeged, 0 23. https://eprints.sztaki.hu/9911/1/nemeskey_2020a.pdf.
38. Németh, Renáta, Eszter Rita Katona, és Zoltán Kmetty. 2020. „Az automatizált szövegelemzés perspektívája a társadalomtudományokban”. Szociológiai szemle 30 (1): 44–62. https://doi.org/10.51624/SzocSzemle.2020.1.3.
39. Omar, Marwan, Soohyeon Choi, Daehun Nyang, és David Mohaisen. 2022. „Robust Natural Language Processing: Recent Advances, Challenges, and Future Directions”. IEEE Access 10: 86038–56. https://doi.org/10.1109/ACCESS.2022.3197769.
40. Orosz, György, Gergő Szabó, Péter Berkecz, Zsolt Szántó, és Richárd Farkas. 2023. „Advancing Hungarian Text Processing with HuSpaCy: Efficient and Accurate NLP Pipelines”. https://doi.org/10.48550/ARXIV.2308.12635.
41. Pedregosa, Fabian, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, és mtsai. 2012. „Scikit-learn: Machine Learning in Python”. https://doi.org/10.48550/ARXIV.1201.0490.
42. Popov, Alexander. 2016. „Neural Network Language Models – an Overview”. In Proceedings of The Workshop on Deep Language Processing for Quality Machine Translation (DeepLP4QMT), 2016. kiad. Sofia, Bulgaria: The Institute of Information and Communication Technologies, Bulgarian Academy of Sciences.
43. R Rehr Uv Rek és P Sojka. 2010. „Software Framework for Topic Modelling with Large Corpora”. https://doi.org/10.13140/2.1.2393.1847.
44. Rahimi, Zahra, és Mohammad Mehdi Homayounpour. 2023. „The Impact of Preprocessing on Word Embedding Quality: A Comparative Study”. Language Resources and Evaluation 57 (1): 257–91. https://doi.org/10.1007/s10579-022-09620-5.

45. Reményi, Andrea Ágnes, Csilla Sárdi, és Zsuzsa Tóth, szerk. 2016. Távlatok a mai magyar alkalmazott nyelvészetben. Budapest: Tinta Könyvkiadó.
46. Research Center for Computational Social Science. 2020. „A politikai nyilvánosság rétegei Magyarországon (2001-2020)". https://rc2s2.elte.hu/project/a-politikai-nyilvanossag-retegei-magyarorszagon-2001-2020/.
47. Rodina, Julia, és Andrey Kutuzov. 2020. „RuSemShift: a dataset of historical lexical semantic change in Russian". https://doi.org/10.48550/ARXIV.2010.06436.
48. Rodriguez, Pedro L., és Arthur Spirling. 2022. „Word Embeddings: What Works, What Doesn't, and How to Tell the Difference for Applied Research". The Journal of Politics 84 (1): 101–15. https://doi.org/10.1086/715162.
49. Rong, Xin. 2016. „word2vec Parameter Learning Explained". arXiv. http://arxiv.org/abs/1411.2738.
50. Sarkar, Dipanjan. 2016. Text Analytics with Python. Berkeley, CA: Apress. https://doi.org/10.1007/978-1-4842-2388-8.
51. Schönemann, Peter H. 1966. „A Generalized Solution of the Orthogonal Procrustes Problem". Psychometrika 31 (1): 1–10. https://doi.org/10.1007/BF02289451.
52. Srinath, K.R. 2017. „Python – The Fastest Growing Programming Language". 04 International Research Journal of Engineering and Technology (IRJET) (12): 354–57.
53. Turian, Joseph, Lev-Arie Ratinov, és Yoshua Bengio. 2010. „Word Representations: A Simple and General Method for Semi-Supervised Learning" Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (július): 384–94.
54. Turney, P. D., és P. Pantel. 2010. „From Frequency to Meaning: Vector Space Models of Semantics". Journal of Artificial Intelligence Research 37 (február): 141–88. https://doi.org/10.1613/jair.2934.
55. Wang, Bin, Angela Wang, Fenxiao Chen, Yuncheng Wang, és C.-C. Jay Kuo. 2019. „Evaluating Word Embedding Models: Methods and Experimental Results". APSIPA Transactions on Signal and Information Processing 8 (1). https://doi.org/10.1017/ATSIP.2019.12.
56. Waskom, Michael, Olga Botvinnik, Drew O'Kane, Paul Hobson, Saulius Lukauskas, David C Gemperline, Tom Augspurger, és mtsai. 2017. „mwaskom/seaborn: v0.8.1 (September 2017)". Zenodo. https://doi.org/10.5281/ZENODO.883859.

57. Werbos, Paul. 1974. „Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Science. Thesis (Ph. D.). Appl. Math.” In . Harvard University.