

SZÁMÍTÓGÉPES SZEMANTIKA

VECTOR SEMANTICS AND EMBEDDINGS

Ferenczi Zsanett

2020. április 24.

1. Bevezetés
2. Lexikális szemantika
3. Szavak és vektorok
4. Vektorok hasonlósága
5. tf-idf
6. PPMI
7. Skip-gram
8. A szóbeágyazások szemantikai jellemzői
9. Vektoros modellek kiértékelése

Bevezetés

- a szavakat ábrázolhatjuk atomi elemekként, de így elveszítjük a kapcsolatokat közöttük
 - szavak hasonlósága fontos bizonyos feladatok esetén
 - ötlet: a szavakat próbáljuk **jelentésük** alapján kódolni, ezáltal a hasonlóság is megragadható lesz két szó között
 - 1950-es évek: **disztribúciós hipotézis**: a szinonimák gyakran hasonló környezetben fordulnak elő
 - pl. *oculist* és *eye-doctor* gyakran állnak együtt az *eye*, *examined* szavakkal
- a hasonló környezetben álló szavak hasonló jelentéssel bírnak

Tesgüino?

A bottle of **tesgüino** is on the table

Everybody likes **tesgüino**

Tesgüino makes you drunk

We make **tesgüino** out of corn.

"You shall know a word by the company it keeps!" (Firth (1957))

Lexikális szemantika

egér (főnév)

1. nagy szemű és fülű, hegyes orrú rágcsáló...
2. számítógép kézi vezérlőeszköze...

- lemma (címszó): egér
- szóalakok: egér, egerek, egérnek, stb.
- jelentés (word sense): itt a két definíció adja meg az egyes jelentéseket
- poliszémia: egy szónak több jelentése van → jelentés egyértelműsítés (WSD)

- **szinonímia**: egyik szó jelentése közel azonos egy másik szó jelentésével
- **hasonlóság (word similarity)**: szavak között állhat fenn, pl. *kutya* és *macska*
- **rokonság (word relatedness)**: szavak közötti egyéb kapcsolat pl. *kórház* és *sebész*
ilyen még: **hiperonímia**, **antonímia**, **meronímia** (ld. 19. fejezet)
- **szemantikai keret, szerepek**: olyan szavak, szereplők halmaza, melyek egy eseményhez köthetők
- **konnotáció**: pl. negatív konnotáció: *szomorú*, pozitív konnotáció: *boldog*

Vector Semantics

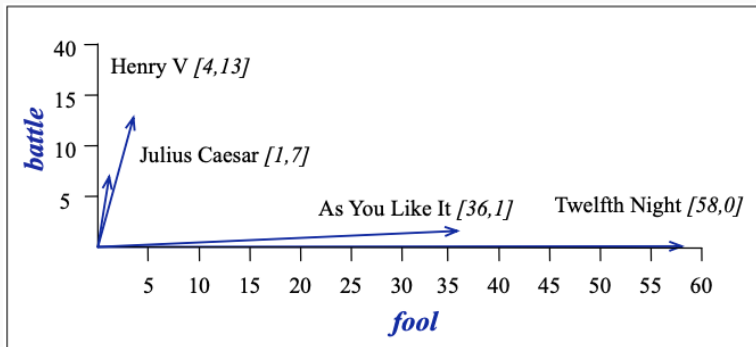
	kellemesség	intenzitás	dominancia
music	7.67	5.57	6.5
heartbreak	2.45	5.65	3.58
life	6.68	5.59	5.89

- egy szó jelentése ábrázolható pontként a térben (Osgood et al. (1957))
- hasonló környezetben előforduló szavak hasonló jelentésűek (egy szó ábrázolása a körülötte előforduló szavak számlálásával)
- ezen két megfigyelést ötvözi a vector semantics
- **szóbeágyazás**: olyan vektor, amely egy szót reprezentál

Szavak és vektorok

- a vektorok általában együttes előfordulási mátrixon alapulnak (co-occurence matrix)
- pl. **term-document matrix**: minden sor egy szót jelöl, minden oszlop egy dokumentumot
- V. Henriket a [13, 89, 4, 3] vektorral lehetne ábrázolni
- a vektortér dimenziója ebben az esetben 4

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	0	7	13
good	114	80	62	89
fool	36	58	1	4
wit	20	15	2	3



- Shakespeare 4 darabja két dimenzióan ábrázolva (4 dokumentumvektor)
- egy term-document mátrix annyi dimenziós lenne, ahány type van a dokumentumokban ($|V|$ sor)

Term-term mátrix

- **term-term (word-word vagy term-context) mátrix:**
sorok és oszlopok is szavakat jelölnek
- a dimenziója: $|V| \times |V|$ (a $|V|$ általában 10 000 és 50 000 közötti)
- sor: **célszó** (target word), oszlop: **kontextus (szavak)**
- az egyes cellák azt jelölik, hogy a célszó hányszor fordult elő a kontextusszó környezetében
- a környezet lehet egy dokumentum, de lehet kisebb egység is, pl. a szó körüli **ablak** (a célszótól jobbra és balra 4-4 szó)

	aardvark	...	computer	data	result	pie	sugar	...
cherry	0	...	2	8	9	442	25	
strawberry	0	...	0	0	1	60	19	
digital	0	...	1670	1683	85	5	4	
information	0	...	3325	3982	378	5	13	

Vektorok hasonlósága

Vektorok hasonlósága

- két vektor közötti hasonlóság mérése → **skaláris szorzattal**
- nagy lesz, ha két vektor ugyanazon dimenzióinak értékei nagyok
- 0 pedig, ha egyáltalán nem hasonlók

$$\text{dot product}(\mathbf{v}, \mathbf{w}) = \mathbf{v} \cdot \mathbf{w} = \sum_{i=1}^N v_i w_i = v_1 w_1 + v_2 w_2 + \dots + v_N w_N$$

Vektorok hasonlósága

- a skaláris szorzat előnyben részesíti a hosszabb vektorokat, így a gyakoribb szavak magasabb értékeket kapnak, míg a kevésbé gyakori szavakhoz nehéz hasonlót találni → ez probléma
- egy megoldás: elosztjuk a vektorok hosszával → **a bezárt szög koszinusza** (0 és 1 közötti szám)
- vektor hossza:

$$|v| = \sqrt{\sum_{i=1}^N v_i^2}$$

$$\frac{a \cdot b}{|a||b|} = \cos\theta$$

	pie	data	computer
cherry	442	8	2
digital	5	1683	1670
information	5	3982	3325

$$\cos(\text{cherry}, \text{information}) = \frac{442 * 5 + 8 * 3982 + 2 * 3325}{\sqrt{442^2 + 8^2 + 2^2} \sqrt{5^2 + 3982^2 + 3325^2}} = .017$$

$$\cos(\text{digital}, \text{information}) = \frac{5 * 5 + 1683 * 3982 + 1670 * 3325}{\sqrt{5^2 + 1683^2 + 1670^2} \sqrt{5^2 + 3982^2 + 3325^2}} = .996$$

tf-idf

- a sokszor együtt előforduló szavak fontosabbak, mint a csak néhányszor előfordulók
- de a túl gyakori szavak nem lényegesek: a `good` minden dokumentumban nagyjából ugyanannyiszor fordul elő (ld. 7. dia)
- ezt egyensúlyozni kell → **tf-idf algoritmus**
- **term frequency (tf)**: $tf_{t,d}$ = t szó gyakorisága d dokumentumban

$$tf_{t,d} = \log_{10}(\text{count}(t, d) + 1)$$

$$tf_{t,d} = \frac{\text{count}(t, d)}{|D|}$$

	Collection Frequency	Document Frequency
Romeo	113	1
action	113	31

- **document frequency (df)**: df_t = hány dokumentumban fordul elő t szó
- **collection frequency**: t szó hányszor fordul elő összesen a dokumentumokban
- az olyan szavak, mint a Romeo kihangsúlyozása **idf**-fel
- **inverse document frequency (idf)**: a kevesebb dokumentumban előforduló szavak előnyben részesítése (N = dokumentumok száma)

$$idf_t = \log_{10}\left(\frac{N}{df_t}\right)$$

- a **tf-idf** ezek szorzata:

$$w_{t,d} = tf_{t,d} \times idf_t$$

- **tf-idf model**:
 - célszó vektorként való ábrázolása
 - annyi dimenzióval, ahány szó van a dokumentumban
 - minden dimenzión azt jelölve, hányszor fordult elő az adott szóval
 - **tf-idf**-fel súlyozva
 - szavak hasonlósága: **tf-idf** vektorok koszinusza

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	0.074	0	0.22	0.28
good	0	0	0	0
fool	0.019	0.021	0.0036	0.0083
wit	0.049	0.044	0.018	0.022

PPMI

- **PMI (pointwise mutual information)**: annak mértéke, hogy két esemény milyen gyakran következik be, azzal összevetve, hogy mire számítanánk, ha egymástól függetlenek lennének
- w - célszó, c - kontextus(szó)

$$PMI(w, c) = \log_2 \frac{P(w, c)}{P(w)P(c)}$$

- ez $-\infty$ és $+\infty$ közötti eredményt ad
- ahhoz, hogy valami kevesebbszer forduljon elő, mint várnánk, hatalmas korpusz kellene → **Positive PMI**
- minden negatív értéket 0-ra cserélünk

$$PPMI(w, c) = \max(\log_2 \frac{P(w, c)}{P(w)P(c)}, 0)$$

Skip-gram

Skip-gram

- a vektorok eddig hosszúak és ritkák (sok elem 0) voltak
- másik metódus: rövidebb (50-1000) és sűrűbb vektorok → **skip-gram with negative sampling**
- ez a **word2vec** egyik algoritmus
- ennek módja:
 - a célszót és a környezetét kezeljük pozitív mintaként
 - válasszunk random szavakat a lexikonból (negatív minták)
 - logisztikus regresszió használata a betanításhoz
 - regressziós súlyok használata szóbeágyazásokként
- a skip-gram probablisztikus osztályozója: τ célszó és k méretű ablak esetén $c_{1:k}$ kontextus mennyire hasonlít a τ szóhoz

$$P(+|t, c) = \frac{1}{1 + e^{-t \cdot c}}$$

$$P(-|t, c) = 1 - P(+|t, c)$$

$$P(+|t, c_{1:k}) = \prod_{i=1}^k \frac{1}{1 + e^{-t \cdot c_i}}$$

Skip-gram szóbeágyazások tanulása

... lemon, a [tablespoon of apricot jam, a] pinch ...

c1

c2

t

c3

c4

positive examples +

t

c

apricot tablespoon

apricot of

apricot jam

apricot a

negative examples -

t

c

t

c

apricot aardvark apricot seven

apricot my apricot forever

apricot where apricot dear

apricot coaxial apricot if

- kétszer annyi negatív mintát használ, mint pozitívat
- kell egy kiinduló szóbeágyazás-halmaz
- ezek segítségével maximalizálni kell a pozitív minták hasonlóságát és minimalizálni a negatív minták hasonlóságát

$$L(\theta) = \sum_{(t,c) \in +} \log P(+|t, c) + \sum_{(t,c) \in -} \log P(-|t, c)$$

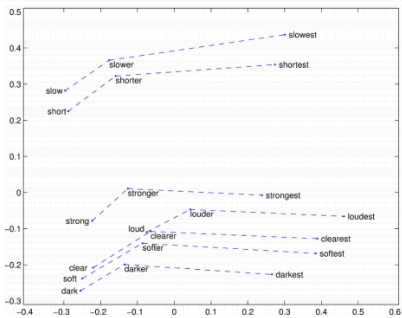
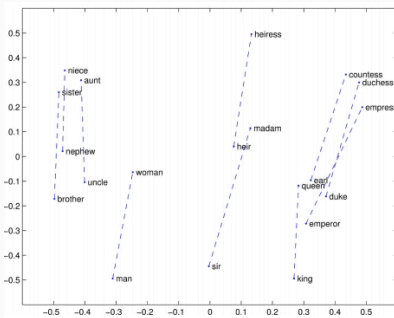
- minden szóhoz két különböző vektort tanul meg: amikor t célszó, vagy amikor c kontextusszó
- két mátrixban vannak ezek tárolva: T `target matrix`, és C `context matrix`
- 3 lehetőség:
 - csak a T -t tartjuk meg
 - összeadunk minden (d -dimenziós) vektort \rightarrow új d -dimenziós vektor
 - konkaténáljuk őket \rightarrow új $2d$ -dimenziós vektor
- az L ablak mérete befolyásolja a teljesítményt: devseten lehet finomhangolni

A szóbeágyazások szemantikai jellemzői

A szóbeágyazások szemantikai jellemzői

- **az ablak mérete**: általában 3 és 20 közötti (1-10 mindkét oldalon)
 - ha kisebb, inkább szintaktikai hasonlóságot mutatnak (pl. szófaj megegyezik)
 - ha nagyobb, témában hasonlóak
- pl. **Hogwarts**hoz leghasonlóbb szavak (Levy és Goldberg (2014))
 - ± 2 ablakkal: Evernight, Sunnydale, Collinwood, stb.
 - ± 5 -össel: Dumbledore, half-blood, Malfoy, stb.
- **first-order co-occurrence**: tipikusan közel állnak egymáshoz, pl. *wrote, book*
- **second-order co-occurrence**: hasonló szomszédjaik vannak, pl. *wrote, said*

A szóbeágyazások szemantikai jellemzői



- **analógia**: az egyes vektorok közötti eltolások mintha valamilyen jelentéssel bírnának
- **történeti szemantika**: hogyan változott a jelentése egy szónak

Vektoros modellek kiértékelése

- **extrinsic** (beépítve más NLP feladatokba)
- **intrinsic**
 - hasonlóság mérése, összevetve egy gold standarddal
 - kontextus nélkül:
 - **WordSim-353** (0-10-es skálán 353 főnévpárt osztályoztak)
 - **SimLex-999** (melléknevek, igék, főnevek)
 - **TOEFL dataset** (80 kérdés, 4 lehetséges válasszal)
 - kontextussal:
 - **Stanford Contextual Word Similarity (SCWS)** dataset (2 003 szópár mondatba illesztve)
 - **Word-in-Context dataset** (egy célszó két kontextusban való megadása, el kell dönten, hogy azonos jelentésben szerepel-e)

- **GloVe**: valószínűségek arányát használja fel
- **fasttext**:
 - a word2vec kiterjesztése
 - kezeli az ismeretlen szavakat és ezáltal a gazdag morfológiájú nyelveket
 - minden szó saját maga és a benne megtalálható n-gramok alapján van reprezentálva

John R Firth. A synopsis of linguistic theory, 1930-1955. *Studies in linguistic analysis*, 1957.

Daniel Jurafsky and James Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*, volume 3.

Omer Levy and Yoav Goldberg. Dependency-based word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 302–308, 2014.

Charles Egerton Osgood, George J Suci, and Percy H Tannenbaum. *The measurement of meaning*. Number 47. University of Illinois press, 1957.