

N-GRAMM NYELVI MODELLEK

Ferenczi Zsanett

2020. 02. 14.

1. Bevezetés
2. N-gramm modellek
3. Ismeretlen szavak
4. Simítás
5. A perplexitás és az entrópia kapcsolata
6. Egy kis történelem...
7. Összegzés

Bevezetés

Mi ez és mire jó?

- annak megjóslása, hogy mi lesz egy adott szó sor folytatása

Please turn your homework ...

- valószínűség rendelése egy adott szó sorhoz

on guys all I of notice sidewalk three a sudden
standing the

- beszéd felismerés, helyesírás-ellenőrzés, gépi fordítás...

- **nyelvi modellek** (LMs): valószínűség hozzárendelése szavak szekvenciájához
- legegyszerűbb modell: n-gramm modell
- **n-gramm**: n szó, karakter, stb. sorozata
- pl. bigramm ($n = 2$), trigramm ($n = 3$), stb.
- $P(\mathbf{w}|\mathbf{h}) \rightarrow \mathbf{w}$ szó valószínűsége \mathbf{h} előzmény esetén

$$P(w|h) = \frac{C(h+w)}{C(h)}$$

- relatív gyakoriság: abszolút elemszám osztva a minta elemszámával
- $w_1^n = w_1, w_2, \dots, w_n$
- chain rule of probability:

$$P(X_1 \dots X_n) = P(X_1)P(X_2|X_1) \dots P(X_n|X_1^{n-1})$$

$$P(w_1^n) = \prod_{k=1}^n P(w_k|w_1^{k-1})$$

N-gramm modellek

- nem tudjuk megmondani egy hosszú szólánc gyakoriságát, hogy hányszor fordul elő egy korpuszban, mert a nyelv kreatív
- ehelyett nem a teljes feltétel valószínűségét, hanem csak az **n megelőző szó** valószínűségét számítjuk
- **Markov-modell**: adott jelenbeli állapot mellett egy jövőbeli állapot nem függ múltbeliektől
- n-gramm esetén n-1 szót veszünk figyelembe a "múltból"
- bigramm modell:

$$P(w_n|w_{n-1})$$

- **Maximum likelihood estimate** (MLE): korpuszból számok kinyerése, majd ezek normalizálása, hogy a szám 0 és 1 közé essen

- tehát:

$$P(w_n | w_{n-N+1}^{n-1}) = \frac{C(w_{n-N+1}^{n-1} w_n)}{C(w_{n-N+1}^{n-1})}$$

- általában trigramm jobb eredményt ad, mint bigramm

- **extrinsic evaluation:** egy applikációba beépítve, és csak azt a komponenst kicserélve, amit épp kiértékelünk, lemérni a teljesítményt (ez nagyon költséges)
- **intrinsic evaluation:** applikációktól függetlenül mérhetünk, tanítóadat, teszthalmaz és fejlesztési halmaz (**devset**, a fejlesztési szakaszban) kell hozzá
 - a teszthalmaz nem lehet benne a tanítóanyagban különben túl nagy valószínűséget kapunk
 - **perplexitás:** NLP-ben bonyolultság mérésére használják
- általában 80% tanítóanyag, 10% devset és 10% teszthalmaz

Perplexitás

- egy nyelv vagy modell bonyolultságának mérésére szolgál
- a teszhalmaz fordított valószínűsége, normalizálva a szavak számával
- teszhalmaz $W = w_1w_2...w_N$

$$PP(W) = P(w_1w_2...w_N)^{-\frac{1}{N}}$$

$$PP(W) = \sqrt[N]{\prod_{i=1}^N \frac{1}{P(w_i|w_1...w_{i-1})}}$$

- az egyes szavak után átlagosan hány szó következhet

- a teszt- és a tanítóhalmaz műfaja lehetőleg egyezzen meg
- dialektus is (főleg angolban...)
- hiányzó n-grammok, "zero probability n-grams"

denied the allegations: 5

denied the speculation: 2

denied the rumors: 1

denied the report: 1

De hiányzik:

denied the offer

denied the loan

- ha a teszhalmazon bármely szó **valószínűsége 0**:
 - akkor az egész teszhalmaz valószínűsége 0
(mivel szorzatról van szó)
 - perplexitást nem tudunk számolni (0-val nem osztunk)

Ismeretlen szavak

- **closed dictionary:** a teszthalmazban csak olyan szavak fordulnak elő, amik szerepelnek a tanítóadatok között (pl. gépi fordítás, beszédfelismerés)
- máskor számolnunk kell ismeretlen szavakkal: OOV (out of vocabulary)
- **OOV rate:** teszthalmazban található ismeretlen szavak aránya
- **open vocabulary:** olyan rendszer, amelyben a lehetséges ismeretlen szavakat <UNK> pseudo-szóval jelöljük

- 1. módszer:
 1. kiválasztunk egy adott szóanyagot
 2. a tanítóadatokból mindent, ami nincs ebben benne, <UNK>-ká alakítunk
 3. kiszámítjuk az <UNK> valószínűségét is, a többi szóval együtt
- 2. módszer:
 - az n-nél kevesebbszer előforduló szavakat <UNK>-ra cseréljük, vagy a gyakoriság szerint sorbarendezett szavak közül az első V szót megtartjuk, a többit <UNK>-ra cseréljük
 - kiszámítjuk az <UNK> valószínűségét is, a többi szóval együtt

Simítás

- akkor is kaphatunk **0 valószínűséget**, ha egy ismert szó eddig nem látott környezetben fordul elő
- ezt kiküszöbölendő léteznek simítási módszerek
- típusai:
 - laplace smoothing
 - add-k smoothing
 - Stupid backoff
 - Kneser-Ney smoothing

Laplace smoothing

- minden bigramm számához 1-et hozzáadunk, mielőtt valószínűséget számolunk belőlük
- nem valami jó a teljesítménye
- másik neve: **add-one smoothing**
- V szó, mindegyik szó számához egyet hozzáadunk, ezért a nevezőt növelni kell V -vel

$$P_{Laplace}(w_i) = \frac{c_i + 1}{N + V}$$

- adjusted counts:

$$c^*(w_{n-1}w_n) = \frac{[C(w_{n-1}w_n) + 1] \times C(w_{n-1})}{C(w_{n-1}) + V}$$

Laplace smoothing

| | i | want | to | eat | chinese | food | lunch | spend |
|---------|------|-------|-------|-------|---------|------|-------|-------|
| i | 3.8 | 527 | 0.64 | 6.4 | 0.64 | 0.64 | 0.64 | 1.9 |
| want | 1.2 | 0.39 | 238 | 0.78 | 2.7 | 2.7 | 2.3 | 0.78 |
| to | 1.9 | 0.63 | 3.1 | 430 | 1.9 | 0.63 | 4.4 | 133 |
| eat | 0.34 | 0.34 | 1 | 0.34 | 5.8 | 1 | 15 | 0.34 |
| chinese | 0.2 | 0.098 | 0.098 | 0.098 | 0.098 | 8.2 | 0.2 | 0.098 |
| food | 6.9 | 0.43 | 6.9 | 0.43 | 0.86 | 2.2 | 0.43 | 0.43 |
| lunch | 0.57 | 0.19 | 0.19 | 0.19 | 0.19 | 0.38 | 0.19 | 0.19 |
| spend | 0.32 | 0.16 | 0.32 | 0.16 | 0.16 | 0.16 | 0.16 | 0.16 |

Figure 3.7 Add-one reconstituted counts for eight words (of $V = 1446$) in the BeRP corpus of 9332 sentences. Previously-zero counts are in gray.

Add-k smoothing

- ahelyett, hogy egyet adnánk minden szó előfordulásához, k-t adunk (pl. 0,5 vagy 0,05-öt)
- k kiválasztása: a devset-en való optimalizálással
- még mindig nem túl jó teljesítmény

$$P_{Add-k}^*(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n) + k}{C(w_{n-1}) + kV}$$

- **backoff**: visszafelé lépünk: trigramm, ha nem elegendő, bigramm, ha ez sem, unigramm
- de csak akkor lépünk, ha 0 a valószínűség (**zero evidence**)
- **lineáris interpoláció**: mindegyik beleszámít, λ kombinációs súlyok, ahol $\sum_i \lambda_i = 1$

$$\begin{aligned}\hat{P}(w_n|w_{n-2}w_{n-1}) &= \lambda_1 P(w_n|w_{n-2}w_{n-1}) \\ &\quad + \lambda_2 P(w_n|w_{n-1}) \\ &\quad + \lambda_3 P(w_n)\end{aligned}$$

- ahhoz, hogy a valószínűség ne legyen 1-nél nagyobb a modellben, a magasabb rendű n -grammoktól az alacsonyabb rendűekhez teszünk át valószínűségi mennyiséget
- α -függvény szétosztja a valószínűségi mennyiséget
- P^* -ra támaszkodunk (csökkentett valószínűség = **discounted probability**), ha már láttuk az n -grammot korábban, különben kisebb előzményt veszünk $(N-1)$ -gramm formájában
- ezt általában **Good-Turing backoff** módszerrel kombinálják

Kneser-Ney smoothing

- azon szavak, amelyek több környezetben előfordultak, nagyobb eséllyel fordulnak elő újabb környezetben, mint azok, amelyek kevesebb fajta kontextusban tűntek fel eddig

$$P_{CONTINUATION}(w) = \frac{|\{v : C(vw) > 0\}|}{\sum_{w'} |\{v : C(vw') > 0\}|}$$

- interpolated Kneser-Ney smoothing bigrammokra:

$$P_{KN}(w_i | w_{i-1}) = \frac{\max(C(w_{i-1}w_i) - d, 0)}{C(w_{i-1})} + \lambda(w_{i-1})P_{CONTINUATION}(w_i)$$

- ez adja a legjobb eredményt

A web és a stupid backoff

- hatalmas szövegmennyiségek megjelenése
→ hatalmas nyelvi modellek építése
- 2006 Google Ngram Viewer (az 1 024 908 267 229 tokennyi szövegben legalább 40-szer előforduló n-grammok)
- nagyon nagy mennyiségű szöveg esetén egy egyszerűbb algoritmus is jó eredményt adhat: **stupid backoff**

- ha nem találunk példát egy magasabb rendű n-grammra, egyet visszalépünk (n-1), és ezt súlyozzuk egy fix (kontextus-független) értékkel

$$S(w_i | w_{i-k+1}^{i-1}) = \begin{cases} \frac{\text{count}(w_{i-k+1}^i)}{\text{count}(w_{i-k+1}^{i-1})} & \text{if } \text{count}(w_{i-k+1}^i) > 0 \\ \lambda S(w_i | w_{i-k+2}^{i-1}) & \text{otherwise} \end{cases}$$

A perplexitás és az entrópia kapcsolata

- az információ mérésére szolgál
- X random változó, χ halmaz felett
- $p(x)$ valószínűség függvény

$$H(X) = - \sum_{x \in \chi} p(x) \log_2 p(x)$$

A kereszt-entrópia

- **kereszt-entrópia:** ha nem tudjuk a valószínűség-eloszlást (p)
- $m = p$ modellje, megközelítése
- a $H(p, m)$ kereszt-entrópia a $H(p)$ entrópia felső korlátja
 $H(p) \leq H(p, m)$

$$H(p, m) = \lim_{n \rightarrow \infty} -\frac{1}{n} \log m(w_1, w_2, \dots, w_n)$$

$$H(W) = -\frac{1}{N} \log P(w_1 w_2 \dots w_N)$$

$$PP(W) = 2^{H(W)}$$

Egy kis történelem...

- 1913 Markov-lánc első használata
- Markov megvizsgálta *Puskin: Anyeginének* 20.000 egymás utáni betűjét, és mással-, illetve magánhangzókra osztotta őket
- bi- és trigrammokkal számolt valószínűséget, hogy a következő betű mgh vagy msh lesz-e
- **Chomsky** hatására sok nyelvész évtizedekig nem foglalkozott a statisztikai alapú modellekkel
- az n-gramm modellek újjáélesztése **Jelineknek** köszönhető (IBM)
→ beszédfelismerés trigrammokkal (1970-es évek)
- a neurális hálós modellek jóval hatékonyabbak

Összegzés

- nyelvi modellek egy mondathoz, szósorhoz valószínűséget rendelnek, és megjósolják a következő szavakat
- **MLE**: korpuszból nyerünk ki számokat, és ezeket normalizáljuk
- az n-gramm modellek bonyolultságát perplexitással szokták mérni
- a simító algoritmusok segítenek a problémák kiküszöbölésében

Daniel Jurafsky and James Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*, volume 3. URL <http://web.stanford.edu/~jurafsky/slp3/>.

Köszönöm a figyelmet!