N-gram nyelvi modellek

Ferenczi Zsanett

2020. március 23.

Vázlat

1. Bevezetés
2. N-gram 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 szósor folytatása

  `Please turn your homework ...`

- valószínűség rendelése egy adott szósorhoz

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

- beszédfelismerés, helyesírás-ellenőrzés, gépi fordítás...

Nyelvi modellek

- **nyelvi modellek** (LMs): valószínűség hozzárendelése szavak szekvenciájához
- legegyszerűbb modell: n-gram modell
- **n-gram**: n szó, karakter, stb. sorozata
- pl. bigram (n = 2), trigram (n = 3), stb.
- `P(w|h)` $\rightarrow$ `w` szó valószínűsége `h` előzmény esetén

Valószínűség

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

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

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

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

N-gram modellek

N-gram

- 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-gram esetén n-1 szót veszünk figyelembe a ”múltból”
- bigram modell:

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

Maximum likelihood esimate

- **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 trigram jobb eredményt ad, mint bigram

Kiértékelés

- **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 valószínűség reciproka, normalizálva a szavak számával
- teszthalmaz $W = w_1 w_2 ... w_N$

$$PP(W) = P(w_1 w_2 ... w_N)^{-\frac{1}{N}}$$

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

- az egyes szavak után átlagosan hány szó következhet

Általánosítás

- a teszt- és a tanítóhalmaz műfaja lehetőleg egyezzen meg
- dialektus is (főleg angolban...)
- hiányzó n-gramok, ”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

Probléma

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

Ismeretlen szavak

Ismeretlen szavak

- **closed vocabulary**: 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>` pszeudo-szóval jelöljük

Ismeretlen szavak

- 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

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 bigram 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:

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

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 és interpoláció

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

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

Katz backoff

- ahhoz, hogy a valószínűség ne legyen 1-nél nagyobb a modellben, a magasabb rendű n-gramoktól 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-gramot korábban, különben kisebb előzményt veszünk (N-1)-gram 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 bigramokra:

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

- ez adja a legjobb eredményt

A web és a stupid backoff

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

Stupid backoff

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

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

A perplexitás és az entrópia kapcsolata

Az entrópia

- 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 $H(p) \leq H(p, m)$

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

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

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

Egy kis történelem...

Történelem

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

Összegzés

Összegzés

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

Házi feladat

1. Számold ki az `"i want chinese food"` mondat valószínűségét. Két számítást végezz:
   - egyet a simítatlan, 6. oldalon található táblázat alapján (`Figure 3.2`),
   - egyet az add-one simított táblázat alapján (14. oldal, `Figure 3.6`). Ez utóbbi esetben a `P(i|<s>) = 0.19` és `P(</s>|food) = 0.40` legyen.

   Ezt követően vizsgáld meg, hogy a simítatlan vagy a simított valószínűség nagyobb-e. Indokold meg, miért.

Bibliográfia

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/`.