N-gram language model

→ $P(w|h)$

exp : $P(\text{the} \mid \text{its water is so transparent that})$

$$= \frac{C(\text{its water is so transparent that the})}{C(\text{its water is so transparent that})}$$

↗ 3.2

(1)

Avec un grand corpus on peut estimer la proba apartir de l'Equation 3.2.

même si cette methode marche bien, mais le web n'est pas assez grand pour nous donner une bonne estimation.

parceque (this is because language is creative, new sentences are created all the time and we won't always be able to count entire sentences.) Etais la plus petite modification dans de l'histoire h de $P(w|h)$ fait que le count va valoir zero (ex : Walden Pond's water is so transparant that the " instead of " its water is so transparant that the)

si on cherche la distribution jointe d'un groupe de mots il faut alors compter alors toute les possibilités pour la longueur de notre séquence et voir tous les cas favorables. Ce qui fait beaucoup de choix a estimer.

→ une façon plus intelligente d'estimer la proba $P(w|h)$

$$P(x_1, \ldots, x_n) = \prod_{k=1}^{n} P(x_k \mid x_1, \ldots, x_{k-1})$$

= Chain rule of proba

Le langage est creatif est un petit changement introduit une chose qu'on ne connaît pas.

Au lieu de compter, on se basant sur tout l'histoire avant le mot w on approximer l'histoire par quelques mots

bigram model

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

↑ Markov assumption

$$P(w_n | w_{1:n-1}) \approx P(w_n | w_{n-N+1:n-1})$$

→ How to estimate bigram or ngram probabilities.

- MLE (Maximum likelihood estimation)

$$P(w_n | w_{n-1}) = \frac{C(w_{n-1} w_n)}{\sum_w C(w_{n-1} w)}$$

$$= \frac{C(w_{n-1} w_n)}{C(w_{n-1})}$$

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

↑ relative frequency

MLE

→ end to end evaluation (extrinsic evaluation)

[19] smoothing, backoff, interpolation

21

We need the endsymbol to make a bigram a true probability distribution. Without an end symbol the sentence probabilities for all sentences of a given length would sum to one. This model would define an infinite set of probability distributions, with one distribution per sentence length. Exo 3.5

2

P(T/M)

↑

maximizes (parameter set ↔ prob)

bigram capture syntactic phenomena

like
what come after cat is noun, or an adjective
or what come after to is

→ some may be Cultural than Linguistic

→ we need to pad sentence

→ Log-probabilities
adding in log-space = multiplying in linear space

$$p_1 \times p_2 \times \dots \times p_n = \exp(\log p_1 + \dots + \log p_n)$$

- avg, perplexity, entropy

3

Avg

$$\sum P_{eval}(word) / P_{train}(word)$$

$$\sum \frac{n_{eval}(word)}{N_{eval}} \times \log(P)$$

↑ maximiser il faut que la distribution de l'unigram dans le train et test soit ressemblant.

les modèles statistiques sont inutiles si la base de l'entraînement et le test sont trop différents du point de vue de leur distribution.

$$P_{train}(unigram) = \frac{n_{train}(unigram)+k}{N_{train}+KV}$$

$$= \frac{N_{train}}{N_{train}+KV} \frac{n_{train}(unigram)}{N_{train}} + \frac{KV}{N_{train}+KV} \times \frac{1}{V}$$

Laplace

poids — unigram distribué — poids — uniform probability

Voici comment interpréter le modèle

on a combiné deux modèles avec un certain ensemble de poids pour trouver notre modèle final.

K = 0 unsmooth model

K ↗ diminue la proba des Unigrams les plus communs jusqu'au niveau où on obtient un model uniform.

Effet de l'Interpolat

Plus de test-train plus on a besoin d'interpoler (combinaison avec unigram et uniform distribution)

$\sum$ nval x log(IP)

n UNK IP(UN) + n' IP(U_n)

row = 0

N gram length

Token position in sentence

	0	1	2	3	4	5
I	1/V	n([S]I) / n([S])				
have	1/V	n(I have) / n(I)				
a	1/V	n([illegible]) / n(a)				
dream	1/V	n(a dream) / n(a)				
[END]	1/V	n([illegible]) / n(dream)				
We	1/V	[illegible]				

ngram_end = token_position + 1

$\frac{0+1}{1+1}$

2+1 = 3

'Je' 'suis' 'tm' 'pour'

Je posit 0

P(Je | [S])

ngr 1

2

3

4

5