The goal in this chapter is to assign a probability to a sentence. → applications → Machine translation
→ speech recognition
→ summary generation. etc

goal: compute the probability of a sentence (or seq. of words)

$$P(w) = P(w_1, w_2, \ldots w_n)$$

related task: compute the probability of the next word.

→ A model that computes either of these is called a language model

$P(w)$?

$$P(w) = P(w_1, w_2, \ldots w_n)$$

[chain rule $P(A,B) = P(A) \times P(B|A)$]

$$P(x_1) \times P(x_2, \ldots x_n | x_1)$$

applying chain rule recursively on $P(w)$:

$$P(w) = P(w_1)\, P(w_2|w_1)\, P(w_3|w_1, w_2) \ldots P(w_n|w_1, \ldots w_{n-1})$$

$$P(w_1, w_2, \ldots w_n) = \prod P(w_i \mid w_1, \ldots w_{i-1})$$

let's say we have M words in a .

$$P(w_1, w_2, \ldots w_M) = p(w_1)\, p(w_2 \mid w_1) \ldots P(w_M \mid w_{M-1} \ldots w_1)$$

to compute the probability

$$p(w_M \mid w_{M-1} \ldots w_1)$$

I

we need to model $V^m$ contents.

this is not possible with finite sample data.

To solve this problem, n-gram models, which make a simplifying approximation to condition on only the past (n-1) words

i.e,

$$p(w_m \mid w_1, \ldots w_{m-1}) \approx p(w_m \mid w_{m-1}, \ldots w_{m-n+1})$$

simplest case:

unigram model

$$P(w_1 \ldots w_n) \approx \prod_i P(w_i)$$

bigram model:

$$p(w_1, \ldots w_n) \approx \prod_i P(w_i \mid w_{i-1})$$

→ can extend to trigrams, 4-grams, 5-grams, ...

in general, this is an insufficient model of language.

because language has long distance dependencies.

estimating bi-gram probabilities

$$p(w_i|w_{i-1}) : \frac{count(w_{i-1}, w_i)}{count(w_{i-1})}$$

Ug for n-grams

$$p(w_n|w_{n-1} \cdots w_{n-n+1}) : \frac{count(w_n, w_{n-1} \cdots w_{n-n+1})}{count(w_{n-1} \cdots w_{n-n+1})}$$

Evaluating language models:

extrinsic evaluation:

intrinsic evaluation: test whether our model prefers "good sentences" to "bad ones".

Perplexity: $w_1 \dots w_M$ → held-out data for validating

$$l(w) = \sum_{m=1}^{M} \log P(w_m | w_{m-1} \dots w_1)$$

$$ppl(w) = 2^{-\frac{l(w)}{M}}$$

→ minimizing perplexity is the same as maximizing probability.

There might be a chance that probability $p(w_m | w_{m-1} \dots w_1)$ will be zero, so to avoid this we need to smoothing

Smoothing

! intuition: steal some probability mass & distribute among the zero-probability ones

laplace smoothing:

$$p(w_i | w_{i-1}) = \frac{C(w_{i-1}, w_i) + 1}{C(w_{i-1}) + V}$$

$$C^*(w_{i-1} w_i) = \frac{(C(w_{i-1}, w_i) + 1) \cdot C(w_{i-1})}{C(w_{i-1}) + V}$$

discount $d_c = \frac{c^*}{c}$

Interpolation: mix unigram, bigram, trigram statistics

$$P(w_i | w_{i-1}, w_{i-2}) : \lambda_1 P(w_i | w_{i-1}, w_{i-2}) +$$
$$\lambda_2 P(w_i | w_{i-1}) + \lambda_3 P(w_i)$$

Dealing with unseen words:

↳ create a token <UNK>