

N-gram Language Modeling

- Placing distributions over sentences

$$P(\bar{w}) = P(w_1, w_2 \dots w_m) = P(w_1) P(w_2 | w_1) P(w_3 | w_2 w_1) \dots$$

- N-gram LM

$$P(\bar{w}) = \prod_{i=1}^m P(w_i | \underbrace{w_{i-n+1} \dots w_{i-1}}_{\text{previous } n-1 \text{ words}})$$

- 2-gram LM: $P(w_1 | \langle s \rangle) P(w_2 | w_1) P(w_3 | w_2)$
* w_3 independent of w_1, w_2

* N-gram LM \longleftrightarrow $n-1$ order Markov Model

- 3-gram LM: $P(w_1 | \langle s \rangle \langle s \rangle) P(w_2 | \langle s \rangle w_1) P(w_3 | w_2 w_1) \dots$

- Multinomial Distributions (2-gram LM example):

V = size of vocab $|V| \times |V|$ parameters

$P(w | \text{"the"}) =$

0.001	Hovae
0.0005	Dog
0.0005	Cat

Very Flat distribution

Parameter estimation: MLE from large corpus

MLE \curvearrowright $P(\text{dog} | \text{the}) = \frac{\text{count}(\text{the}, \text{dog})}{\text{count}(\text{the})}$

Does not tell you what words come up next. Only gives rough probability dist. of what words are likely to come next.

Why: ① Generation: Machine translation
(applications) ② Grammatical Error Correction
③ Way to build "word2vec"++

Smoothing in N-gram LM's

- 5-gram models work well!

$$(2) P(w|to) \quad (3) P(w|go to) \quad (5) P(w|hate to go to)$$

$$(6) P(w|want to go to)$$

class

Austin

$$P(\text{Austin} | to) > 0 \rightarrow \text{seen in data}$$

$$P(\text{Austin} | want to go to) = 0 \rightarrow \text{if corpus isn't huge}$$

* How do we give some portion of probability to these novel instances?

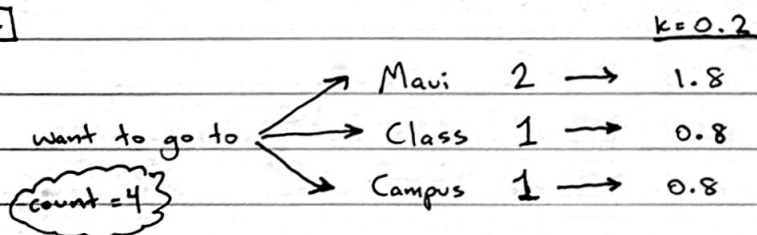
- Absolute Discounting : Reserve mass from seen 5-grams to allocate to unseen 5-grams

$$P_{\text{AD}}(\text{Austin} | want to go to) = \frac{\text{count}(want to go to Austin) - K}{\text{count}(want to go to)} + \lambda P_{\text{AD}}(\text{Austin} | to go to)$$

4-gram

* λ set to make this normalized

Ex.



$$\lambda = \frac{0.6}{4}$$

word types
seen in this
context times
 K

* Can do this recursively!

$$P_{\text{AD}}(\text{Austin} | to go to) = \frac{\text{count}(to go to Austin)}{\text{count}(to go to)} + \lambda P_{\text{AD}}(\text{Austin} | go to)$$

3-gram

keep unrolling, and you
eventually end up with
 $P(\text{Austin})$, which
is > 0 if Austin is in
corpus