N-Gram Smoothing Techniques

Natalie Parde
UIC CS 421

Handling Words in Unseen Contexts

- Smoothing: Taking a bit of the probability mass from more frequent events and giving it to unseen events.
  - Sometimes also called “discounting”
- Many different smoothing techniques:
  - Laplace (add-one)
  - Add-k
  - Stupid backoff
  - Kneser-Ney

Bigram	Frequency
CS 421	8
CS 590	5
CS 594	2
CS 521	0 😢

Bigram	Frequency
CS 421	7
CS 590	5
CS 594	2
CS 521	1 🥰

Laplace Smoothing

- Add one to all n-gram counts before they are normalized into probabilities
- Not the highest-performing technique for language modeling, but a useful baseline
  - Practical method for other text classification tasks
- $P(w_i) = \frac{c_i}{N} \rightarrow P_{\text{Laplace}}(w_i) = \frac{c_i+1}{N+V}$

Example: Laplace Smoothing

Corpus Statistics:

Unigram	Frequency
Chicago	4
is	8
cold	6
hot	0

Bigram	Frequency
Chicago is	2
is cold	4
is hot	0
…	0

Example: Laplace Smoothing

Corpus Statistics:

Unigram	Frequency
Chicago	4
is	8
cold	6
hot	0

Bigram	Frequency
Chicago is	2
is cold	4
is hot	0
…	0

$$P(w_i) = \frac{c_i}{N}$$

Unigram	Probability
Chicago	$\frac{4}{18} = 0.22$
is	$\frac{8}{18} = 0.44$
cold	$\frac{6}{18} = 0.33$
hot	$\frac{0}{18} = 0.00$

Bigram	Probability
Chicago is	
is cold	
is hot	

Example: Laplace Smoothing

Corpus Statistics:

Unigram	Frequency
Chicago	4
is	8
cold	6
hot	0

Bigram	Frequency
Chicago is	2
is cold	4
is hot	0
…	0

$$P(w_i) = \frac{c_i}{N}$$

Unigram	Probability
Chicago	$\frac{4}{18} = 0.22$
is	$\frac{8}{18} = 0.44$
cold	$\frac{6}{18} = 0.33$
hot	$\frac{0}{18} = 0.00$

Bigram	Probability
Chicago is	$\frac{2}{4} = 0.50$
is cold	$\frac{4}{8} = 0.50$
is hot	$\frac{0}{8} = 0.00$

Example: Laplace Smoothing

Corpus Statistics:

Unigram	Frequency
Chicago	4
is	8
cold	6
hot	0

Bigram	Frequency
Chicago is	2
is cold	4
is hot	0
…	0

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

Unigram	Probability
Chicago	
is	
cold	
hot	

Bigram	Probability
Chicago is	
is cold	
is hot	

Example: Laplace Smoothing

Corpus Statistics:

Unigram	Frequency
Chicago	4+1
is	8+1
cold	6+1
hot	0+1

Bigram	Frequency
Chicago is	2+1
is cold	4+1
is hot	0+1
…	0+1

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

Unigram	Probability
Chicago	
is	
cold	
hot	

Bigram	Probability
Chicago is	
is cold	
is hot	

Example: Laplace Smoothing

Corpus Statistics:

Unigram	Frequency
Chicago	4+1
is	8+1
cold	6+1
hot	0+1

Bigram	Frequency
Chicago is	2+1
is cold	4+1
is hot	0+1
…	0+1

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

Unigram	Probability
Chicago	$\frac{5}{22} = 0.23$
is	$\frac{9}{22} = 0.41$
cold	$\frac{7}{22} = 0.32$
hot	$\frac{1}{22} = 0.05$

Bigram	Probability
Chicago is	
is cold	
is hot	

Example: Laplace Smoothing

Corpus Statistics:

Bigram	Frequency
Chicago Chicago	0+1
Chicago is	2+1
Chicago cold	0+1
Chicago hot	0+1

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

Unigram	Frequency
Chicago	4
is	8
cold	6
hot	0

Bigram	Frequency
Chicago is	2+1
is cold	4+1
is hot	0+1
…	0+1

Unigram	Probability
Chicago	$\frac{5}{22} = 0.23$
is	$\frac{9}{22} = 0.41$
cold	$\frac{7}{22} = 0.32$
hot	$\frac{1}{22} = 0.05$

Bigram	Probability
Chicago is	$\frac{3}{4+4} = \frac{3}{8} = 0.38$
is cold	$\frac{5}{8+4} = \frac{5}{12} = 0.42$
is hot	$\frac{1}{8+4} = \frac{1}{12} = 0.08$

Probabilities: Before and After

Bigram	Probability
Chicago is	$\frac{2}{4} = 0.50$
is cold	$\frac{4}{8} = 0.50$
is hot	$\frac{0}{8} = 0.00$

Bigram	Probability
Chicago is	$\frac{3}{8} = 0.38$
is cold	$\frac{5}{12} = 0.42$
is hot	$\frac{1}{12} = 0.08$

Add-K Smoothing

- Moves a bit less of the probability mass from seen to unseen events
- Rather than adding one to each count, add a fractional count
  - 0.5
  - 0.05
  - 0.01
- The value *k* can be optimized on a validation set
- $P(w_i) = \frac{c_i}{N} \rightarrow P_{\text{Add-K}}(w_i) = \frac{c_i + k}{N + kV}$
- $P(w_n|w_{n-1}) = \frac{c(w_{n-1}w_n)}{c(w_{n-1})} \rightarrow P_{\text{Add-K}}(w_n|w_{n-1}) = \frac{c(w_{n-1}w_n)+k}{c(w_{n-1})+kV}$

Add-K smoothing is useful for some tasks, but still tends to be suboptimal for language modeling.

- Other smoothing techniques?
  - **Backoff:** Use the specified n-gram size to estimate probability if its count is greater than 0; otherwise, *backoff* to a lower-order n-gram
  - **Interpolation:** Mix the probability estimates from multiple n-gram sizes, weighing and combining the n-gram counts

Interpolation

N	Weight
3	0.5
2	0.4
1	0.1

N-Gram	Probability	Value
I ❤️ 421	P(421 | I ❤️)	0.7
❤️ 421	P(421 | ❤️)	0.5
421	P(421)	0.2

$0.5 * 0.7 + 0.4 * 0.5 + 0.1 * 0.2 = 0.57$

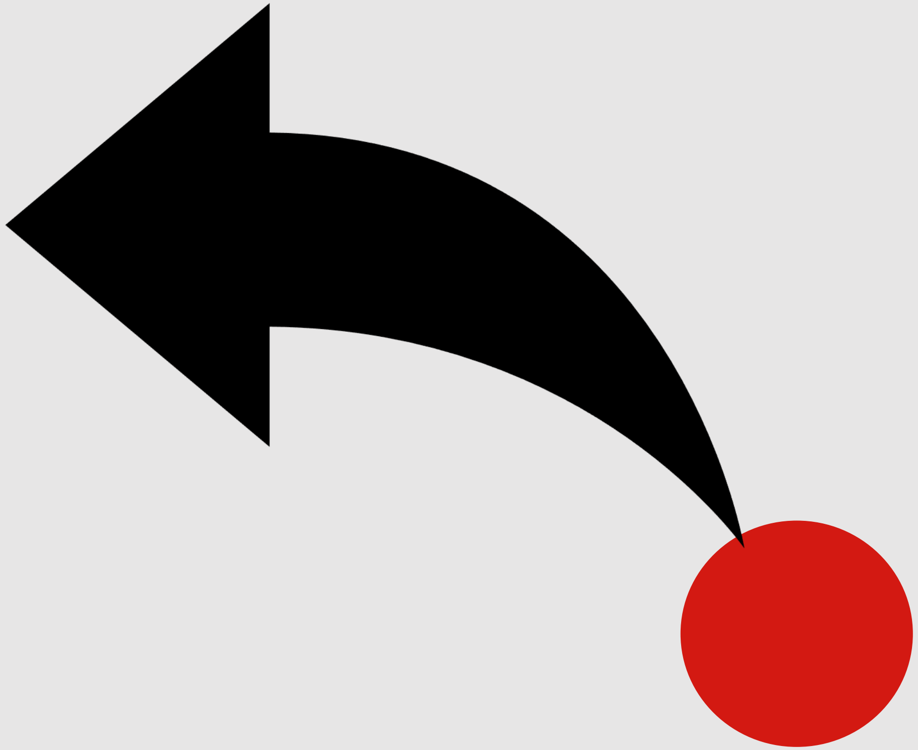
- **Linear interpolation**
  - $P'(w_n|w_{n-2}w_{n-1}) = \lambda_1 P(w_n|w_{n-2}w_{n-1}) + \lambda_2 P(w_n|w_{n-1}) + \lambda_3 P(w_n)$
    - Where $\sum_i \lambda_i = 1$
- **Conditional interpolation**
  - $P'(w_n|w_{n-2}w_{n-1}) = \lambda_1(w_{n-2}^{n-1}) P(w_n|w_{n-2}w_{n-1}) + \lambda_2(w_{n-2}^{n-1}) P(w_n|w_{n-1}) + \lambda_3(w_{n-2}^{n-1}) P(w_n)$

Context-conditioned weights

N-Gram	Probability	Value	Weight
I ❤️ 421	P(421 | I ❤️)	0.7	0.5
I 🚕 421	P(421 | I 🚕)	0.7	0.1

Backoff

- If the n-gram we need has zero counts, approximate it by backing off to the (n-1)-gram
- Continue backing off until we reach a size that has non-zero counts
- Just like with smoothing, some probability mass from higher-order n-grams needs to be redistributed to lower-order n-grams



Katz Backoff

- Incorporate a function α to distribute probability mass to lower-order n-grams
- Rely on a discounted probability P* if the n-gram has non-zero counts
- Otherwise, recursively back off to the Katz probability for the (n-1)-gram
- $P_{BO}(w_n|w_{n-N+1}^{n-1}) = \begin{cases} P^*(w_n|w_{n-N+1}^{n-1}), & \text{if } c(w_{n-N+1}^{n}) > 0 \\ \alpha(w_{n-N+1}^{n-1})P_{BO}(w_n|w_{n-N+2}^{n-1}), & \text{otherwise} \end{cases}$

Kneser-Ney Smoothing

- One of the most commonly used and best-performing n-gram smoothing methods
- Incorporates absolute discounting
  - $P_{\text{AbsoluteDiscounting}}(w_i|w_{i-1}) = \frac{C(w_{i-1}w_i)-d}{\sum_v C(w_{i-1}v)} + \lambda(w_{i-1})P(w_i)$

Discounted Bigram

Unigram with interpolation weight

Kneser-Ney Smoothing

- Objective: Capture the intuition that although some lower-order n-grams are frequent, they are mainly only frequent in specific contexts
  - tall nonfat decaf peppermint ________
    - "york" is a more frequent unigram than "mocha" (7.4 billion results vs. 135 million results on Google), but it's mainly frequent when it follows the word "new"
- Creates a unigram model that estimates the probability of seeing the word *w* as a novel continuation, in a new unseen context
  - Based on the number of different contexts in which *w* has already appeared
  - $P_{\text{Continuation}}(w) = \frac{|\{v : C(vw) > 0\}|}{|\{(u', w') : C(u'w') > 0\}|}$

Kneser-Ney Smoothing

$$P_{\mathrm{KN}}(w_i|w_{i-n+1}^{i-1}) = \frac{\max\left(c_{KN}\left(w_{i-n+1}^{i}\right) - d,\ \ 0\right)}{\sum_v c_{KN}\left(w_{i-n+1}^{i-1}v\right)} + \lambda(w_{i-n+1}^{i-1})P_{\mathrm{KN}}(w_i|w_{i-n+2}^{i-1})$$

Kneser-Ney Smoothing

$$P_{\mathrm{KN}}(w_i|w_{i-n+1}^{i-1}) = \frac{\max\left(c_{KN}\left(w_{i-n+1}^{i}\right) - d,\ \ 0\right)}{\sum_v c_{KN}\left(w_{i-n+1}^{i-1}v\right)} + \lambda(w_{i-n+1}^{i-1})P_{\mathrm{KN}}(w_i|w_{i-n+2}^{i-1})$$

Normalizing constant to distribute the probability mass that's been discounted

$$\lambda(w_{i-1}) = \frac{d}{\sum_v C(w_{i-1}v)}|\{w : c(w_{i-1}w) > 0\}|$$

Kneser-Ney Smoothing

$$P_{\mathrm{KN}}(w_i|w_{i-n+1}^{i-1}) = \frac{\max\left(c_{KN}\left(w_{i-n+1}^{i}\right) - d,\ \ 0\right)}{\sum_v c_{KN}\left(w_{i-n+1}^{i-1} v\right)} + \lambda(w_{i-n+1}^{i-1}) P_{\mathrm{KN}}(w_i|w_{i-n+2}^{i-1})$$

Normalizing constant to distribute the probability mass that's been discounted

$$\lambda(w_{i-1}) = \frac{d}{\sum_v C(w_{i-1}v)} |\{w : c(w_{i-1}w) > 0\}|$$

Normalized discount

Number of word types that can follow w_{i-1}

Kneser-Ney Smoothing

$$P_{\text{KN}}(w_i|w_{i-n+1}^{i-1}) = \frac{\max(c_{KN}(w_{i-n+1}^{i}) - d,\ \ 0)}{\sum_v c_{KN}(w_{i-n+1}^{i-1}v)} + \lambda(w_{i-n+1}^{i-1})P_{\text{KN}}(w_i|w_{i-n+2}^{i-1})$$

Normalizing constant to distribute the probability mass that's been discounted

Regular count for the highest-order n-gram, or the number of unique single word contexts for lower-order n-grams

Kneser-Ney Smoothing

$$P_{\text{KN}}(w_i|w_{i-n+1}^{i-1}) = \frac{\max(c_{KN}(w_{i-n+1}^{i}) - d,\ \ 0)}{\sum_v c_{KN}(w_{i-n+1}^{i-1}v)} + \lambda(w_{i-n+1}^{i-1})P_{\text{KN}}(w_i|w_{i-n+2}^{i-1})$$

Normalizing constant to distribute the probability mass that's been discounted

Regular count for the highest-order n-gram, or the number of unique single word contexts for lower-order n-grams

Discounted n-gram probability …when the recursion terminates, unigrams are interpolated with the uniform distribution (ε = empty string)

$$P_{KN}(w) = \frac{\max(c_{KN}(w) - d, 0)}{\sum_{w'} c_{KN}(w')} + \lambda(\varepsilon)\frac{1}{V}$$

Stupid Backoff

- Gives up the idea of trying to make the language model a true probability distribution 😌
- No discounting of higher-order probabilities
- If a higher-order n-gram has a zero count, simply backoff to a lower-order n-gram, weighted by a fixed weight
- $S(w_i|w_{i-k+1}^{i-1}) = \begin{cases} \frac{c(w_{i-k+1}^{i})}{c(w_{i-k+1}^{i-1})} & \text{if } c(w_{i-k+1}^{i}) > 0 \\ \lambda S(w_i|w_{i-k+2}^{i-1}) & \text{otherwise} \end{cases}$
    - Terminates in the unigram, which has the probability:
        - $S(w) = \frac{c(w)}{N}$

Stupid Backoff

- Gives up the idea of trying to make the language model a true probability distribution 😌
- No discounting of higher-order probabilities
- If a higher-order n-gram has a zero count, simply backoff to a lower-order n-gram, weighted by a fixed weight
- $S(w_i|w_{i-k+1}^{i-1}) = \begin{cases} \frac{c(w_{i-k+1}^{i})}{c(w_{i-k+1}^{i-1})} & \text{if } c(w_{i-k+1}^{i}) > 0 \\ \lambda S(w_i|w_{i-k+2}^{i-1}) & \text{otherwise} \end{cases}$
  - Terminates in the unigram, which has the probability:
    - $S(w) = \frac{c(w)}{N}$

Generally, 0.4 works well (Brants et al., 2007)