N-gram models

- Unsmoothed n-gram models (review)
- Smoothing
  - Add-one (Laplacian)
  - Good-Turing
- Unknown words
- Evaluating n-gram models
- Combining estimators
  - (Deleted) interpolation
  - Backoff

Goals

- Determine the next word in a sequence
  - Probability distribution across all words in the language
  - $P(w_n \mid w_1 w_2 \ldots w_{n-1})$
- Determine the probability of a sequence of words
  - $P(w_1 w_2 \ldots w_{n-1} w_n)$

Probability of a word sequence

- $P(w_1 w_2 \ldots w_{n-1} w_n)$

$$P(w_1^n) = P(w_1)\ P(w_2|w_1)\ P(w_3|w_1^2)\ \ldots P(w_n|w_1^{n-1})$$

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

- Problem?
- Solution: *approximate* the probability of a word given all the previous words…

N-gram approximations

- Bigram model

$$P(w_n \mid w_1^{n-1}) \approx P(w_n \mid w_{n-1})$$

- Trigram model

$$P(w_n \mid w_1^{n-1}) \approx P(w_n \mid w_{n-2} w_{n-1})$$

- Probability of a word sequence

$$P(w_1^n) = P(w_1)\ P(w_2|w_1)\ P(w_3|w_1^2)\ \ldots P(w_n|w_1^{n-1})$$

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

- General form

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

Training N-gram models

- N-gram models can be trained by counting and normalizing
  - Bigrams

$$P(w_n \mid w_{n-1}) = \frac{count(w_{n-1}w_n)}{count(w_{n-1})}$$

  - General case

$$P(w_n \mid w_{n-N+1}^{n-1}) = \frac{count(w_{n-N+1}^{n-1}w_n)}{count(w_{n-N+1}^{n-1})}$$

  - An example of Maximum Likelihood Estimation (MLE)
    - » Resulting parameter set is one in which the likelihood of the training set T given the model M (i.e. P(T|M)) is maximized.

Bigram counts

	I	want	to	eat	Chinese	food	lunch
I	8	1087	0	13	0	0	0
want	3	0	786	0	6	8	6
to	3	0	10	860	3	0	12
eat	0	0	2	0	19	2	52
Chinese	2	0	0	0	0	120	1
food	19	0	17	0	0	0	0
lunch	4	0	0	0	0	1	0

- Note the number of 0's…

N-gram models

- Unsmoothed n-gram models (review)
- Smoothing
  - Add-one (Laplacian)
  - Good-Turing
- Unknown words
- Evaluating n-gram models
- Combining estimators
  - (Deleted) interpolation
  - Backoff

Smoothing

- Need better estimators than MLE for rare events
- Approach
  - Somewhat decrease the probability of previously seen events, so that there is a little bit of probability mass left over for previously unseen events
    - » Smoothing
    - » Discounting methods

Add-one smoothing

- Add one to all of the counts before normalizing into probabilities
- MLE unigram probabilities

$$P(w_x) = \frac{count(w_x)}{N}$$

corpus length in word tokens

- Smoothed unigram probabilities

$$P(w_x) = \frac{count(w_x) + 1}{N + V}$$

vocab size (# word types)

- Adjusted counts (unigrams)

$$c_i^* = (c_i + 1)\frac{N}{N + V}$$

Add-one smoothing: bigrams

Add-one bigram counts

- Original counts

	I	want	to	eat	Chinese	food	lunch
I	8	1087	0	13	0	0	0
want	3	0	786	0	6	8	6
to	3	0	10	860	3	0	12
eat	0	0	2	0	19	2	52
Chinese	2	0	0	0	0	120	1
food	19	0	17	0	0	0	0
lunch	4	0	0	0	0	1	0

- New counts

	I	want	to	eat	Chinese	food	lunch
I	9	1088	1	14	1	1	1
want	4	1	787	1	7	9	7
to	4	1	11	861	4	1	13
eat	1	1	3	1	20	3	53
Chinese	3	1	1	1	1	121	2
food	20	1	18	1	1	1	1
lunch	5	1	1	1	1	2	1

Add-one smoothed bigram probabilites

- Original

	I	want	to	eat	Chinese	food	lunch
I	.0023	.32	0	.0038	0	0	0
want	.0025	0	.65	0	.0049	.0066	.0049
to	.00092	0	.0031	.26	.00092	0	.0037
eat	0	0	.0021	0	.020	.0021	.055
Chinese	.0094	0	0	0	0	.56	.0047
food	.013	0	.011	0	0	0	0
lunch	.0087	0	0	0	0	.0022	0

- Add-one smoothing

	I	want	to	eat	Chinese	food	lunch
I	.0018	.22	.00020	.0028	.00020	.00020	.00020
want	.0014	.00035	.28	.00035	.0025	.0032	.0025
to	.00082	.00021	.0023	.18	.00082	.00021	.0027
eat	.00039	.00039	.0012	.00039	.0078	.0012	.021
Chinese	.0016	.00055	.00055	.00055	.00055	.066	.0011
food	.0064	.00032	.0058	.00032	.00032	.00032	.00032
lunch	.0024	.00048	.00048	.00048	.00048	.00096	.00048