



COMPUTER SCIENCE
UNIVERSITY OF MARYLAND

N-gram Language Models

CMSC 470

Marine Carpuat

Slides credit: Jurasky & Martin

Roadmap

- Language Models
 - Our first example of modeling sequences
- n-gram language models
- How to estimate them?
- How to evaluate them?
- Neural models

Probabilistic Language Models

- Goal: assign a probability to a sentence
- Why?
 - Machine Translation:
 - $P(\text{high winds tonite}) > P(\text{large winds tonite})$
 - Spell Correction
 - The office is about fifteen **minuets** from my house
 - $P(\text{about fifteen minutes from}) > P(\text{about fifteen minuets from})$
 - Speech Recognition
 - $P(\text{I saw a van}) \gg P(\text{eyes awe of an})$
 - + Summarization, question-answering, etc., etc.!!

Probabilistic Language Modeling

- Goal: compute the probability of a sentence or sequence of words

$$P(W) = P(w_1, w_2, w_3, w_4, w_5 \dots w_n)$$

- Related task: probability of an upcoming word

$$P(w_5 | w_1, w_2, w_3, w_4)$$

- A model that computes either of these:

$$P(W) \quad \text{or} \quad P(w_n | w_1, w_2 \dots w_{n-1})$$

is called a **language model**.

How to compute $P(W)$

- How to compute this joint probability:
 - $P(\text{its, water, is, so, transparent, that})$
- Intuition: let's rely on the Chain Rule of Probability

Recall: Zipf's Law

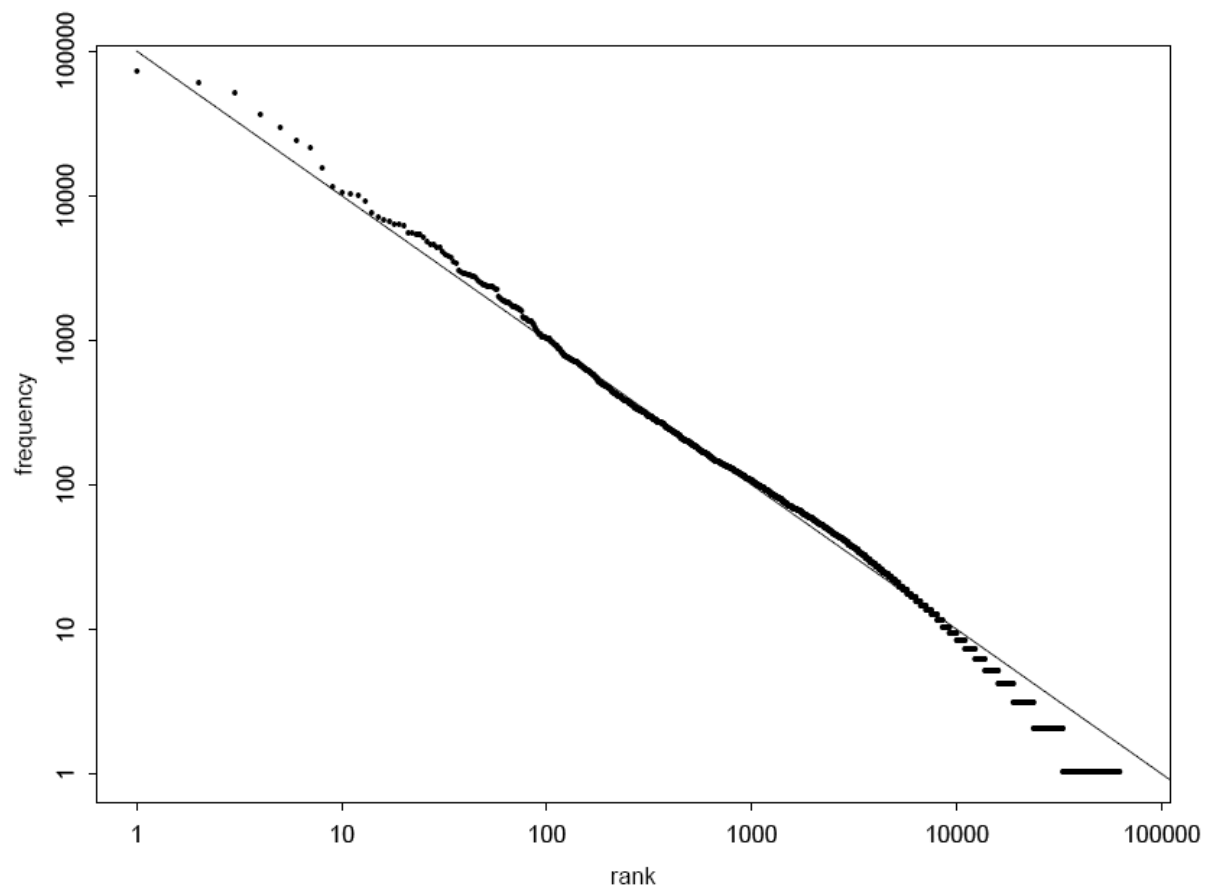
- George Kingsley Zipf (1902-1950) observed the following relation between frequency and rank

$$f \cdot r = c \quad \text{or} \quad f = \frac{c}{r}$$

f = frequency
 r = rank
 c = constant

- Example
 - the 50th most common word should occur three times more often than the 150th most common word

Recall: Zipf's Law



Graph illustrating Zipf's Law for the Brown corpus

Reminder: The Chain Rule

- Recall the definition of conditional probabilities

$$p(\mathbf{B}|\mathbf{A}) = \mathbf{P}(\mathbf{A},\mathbf{B})/\mathbf{P}(\mathbf{A}) \quad \text{Rewriting: } \mathbf{P}(\mathbf{A},\mathbf{B}) = \mathbf{P}(\mathbf{A})\mathbf{P}(\mathbf{B}|\mathbf{A})$$

- More variables:

$$\mathbf{P}(\mathbf{A},\mathbf{B},\mathbf{C},\mathbf{D}) = \mathbf{P}(\mathbf{A})\mathbf{P}(\mathbf{B}|\mathbf{A})\mathbf{P}(\mathbf{C}|\mathbf{A},\mathbf{B})\mathbf{P}(\mathbf{D}|\mathbf{A},\mathbf{B},\mathbf{C})$$

- The Chain Rule in General

$$\mathbf{P}(x_1, x_2, x_3, \dots, x_n) = \mathbf{P}(x_1)\mathbf{P}(x_2|x_1)\mathbf{P}(x_3|x_1, x_2)\dots\mathbf{P}(x_n|x_1, \dots, x_{n-1})$$

The Chain Rule applied to compute joint probability of words in sentence

$$P(w_1 w_2 \dots w_n) = \prod_i P(w_i | w_1 w_2 \dots w_{i-1})$$

$P(\text{"its water is so transparent"}) =$

$P(\text{its}) \times P(\text{water} | \text{its}) \times P(\text{is} | \text{its water})$

$\times P(\text{so} | \text{its water is})$

$\times P(\text{transparent} | \text{its water is so})$

How to estimate these probabilities

- Could we just count and divide?

$$P(\text{the } | \text{its water is so transparent that}) = \frac{\textit{Count}(\text{its water is so transparent that the})}{\textit{Count}(\text{its water is so transparent that})}$$

- No! Too many possible sentences!
- We'll never see enough data for estimating these

Markov Assumption

- Simplifying assumption:

$P(\text{the } | \text{ its water is so transparent that}) \gg P(\text{the } | \text{ that})$

- Or maybe

$P(\text{the } | \text{ its water is so transparent that}) \gg P(\text{the } | \text{ transparent that})$



Andrei Markov

Markov Assumption

$$P(w_1 w_2 \dots w_n) \approx \prod_i P(w_i | w_{i-k} \dots w_{i-1})$$

- In other words, we approximate each component in the product

$$P(w_i | w_1 w_2 \dots w_{i-1}) \approx P(w_i | w_{i-k} \dots w_{i-1})$$

Unigram model (1-gram)

$$P(w_1 w_2 \dots w_n) \gg \prod_i P(w_i)$$

Some automatically generated sentences from a unigram model

fifth, an, of, futures, the, an, incorporated, a,
a, the, inflation, most, dollars, quarter, in, is,
mass

thrift, did, eighty, said, hard, 'm, july, bullish

that, or, limited, the

Bigram model (2-gram)

- Condition on the previous word:

$$P(w_i | w_1 w_2 \dots w_{i-1}) \gg P(w_i | w_{i-1})$$

texaco, rose, one, in, this, issue, is, pursuing, growth, in,
a, boiler, house, said, mr., gurria, mexico, 's, motion,
control, proposal, without, permission, from, five, hundred,
fifty, five, yen

outside, new, car, parking, lot, of, the, agreement, reached

this, would, be, a, record, november

N-gram models

- We can extend to 3-grams (“trigrams”), 4-grams, 5-grams
- In general this is an insufficient model of language
 - because language has **long-distance dependencies**:
“The computer which I had just put into the machine room on the ground floor crashed.”
- But we can often get away with N-gram models

Estimating bigram probabilities

- The Maximum Likelihood Estimate

$$P(w_i | w_{i-1}) = \frac{\textit{count}(w_{i-1}, w_i)}{\textit{count}(w_{i-1})}$$

$$P(w_i | w_{i-1}) = \frac{c(w_{i-1}, w_i)}{c(w_{i-1})}$$

Example 1: Estimating bigram probabilities on toy corpus

<s> I am Sam </s>

<s> Sam I am </s>

<s> I do not like green eggs and ham </s>

$$P(w_i | w_{i-1}) = \frac{c(w_{i-1}, w_i)}{c(w_{i-1})}$$

$$P(\mathbf{I} | \langle \mathbf{s} \rangle) = \frac{2}{3} = .67$$

$$P(\mathbf{Sam} | \langle \mathbf{s} \rangle) = \frac{1}{3} = .33$$

$$P(\mathbf{am} | \mathbf{I}) = \frac{2}{3} = .67$$

$$P(\langle \mathbf{s} \rangle | \mathbf{Sam}) = \frac{1}{2} = 0.5$$

$$P(\mathbf{Sam} | \mathbf{am}) = \frac{1}{2} = .5$$

$$P(\mathbf{do} | \mathbf{I}) = \frac{1}{3} = .33$$

Example 2: Estimating bigram probabilities on Berkeley Restaurant Project sentences

9222 sentences in total

Examples

- can you tell me about any good cantonese restaurants close by
- mid priced thai food is what i'm looking for
- tell me about chez panisse
- can you give me a listing of the kinds of food that are available
- i'm looking for a good place to eat breakfast
- when is caffe venezia open during the day

Raw bigram counts

- Out of 9222 sentences

	i	want	to	eat	chinese	food	lunch	spend
i	5	827	0	9	0	0	0	2
want	2	0	608	1	6	6	5	1
to	2	0	4	686	2	0	6	211
eat	0	0	2	0	16	2	42	0
chinese	1	0	0	0	0	82	1	0
food	15	0	15	0	1	4	0	0
lunch	2	0	0	0	0	1	0	0
spend	1	0	1	0	0	0	0	0

Raw bigram probabilities

- Normalize by unigrams:

i	want	to	eat	chinese	food	lunch	spend
2533	927	2417	746	158	1093	341	278

- Result:

	i	want	to	eat	chinese	food	lunch	spend
i	0.002	0.33	0	0.0036	0	0	0	0.00079
want	0.0022	0	0.66	0.0011	0.0065	0.0065	0.0054	0.0011
to	0.00083	0	0.0017	0.28	0.00083	0	0.0025	0.087
eat	0	0	0.0027	0	0.021	0.0027	0.056	0
chinese	0.0063	0	0	0	0	0.52	0.0063	0
food	0.014	0	0.014	0	0.00092	0.0037	0	0
lunch	0.0059	0	0	0	0	0.0029	0	0
spend	0.0036	0	0.0036	0	0	0	0	0

Using bigram model to compute sentence probabilities

$P(\langle s \rangle \text{ I want english food } \langle /s \rangle) =$

$P(\text{I} | \langle s \rangle)$

$\times P(\text{want} | \text{I})$

$\times P(\text{english} | \text{want})$

$\times P(\text{food} | \text{english})$

$\times P(\langle /s \rangle | \text{food})$

$= .000031$

What kinds of knowledge?

- $P(\text{english} | \text{want}) = .0011$
- $P(\text{chinese} | \text{want}) = .0065$
- $P(\text{to} | \text{want}) = .66$
- $P(\text{eat} | \text{to}) = .28$
- $P(\text{food} | \text{to}) = 0$
- $P(\text{want} | \text{spend}) = 0$
- $P(i | \langle s \rangle) = .25$

Google N-Gram Release, August 2006

AUG

3

All Our N-gram are Belong to You

Posted by Alex Franz and Thorsten Brants, Google Machine Translation Team

Here at Google Research we have been using word [n-gram models](#) for a variety of R&D projects,

...

That's why we decided to share this enormous dataset with everyone. We processed 1,024,908,267,229 words of running text and are publishing the counts for all 1,176,470,663 five-word sequences that appear at least 40 times. There are 13,588,391 unique words, after discarding words that appear less than 200 times.

Google N-Gram Release

- serve as the incoming 92
- serve as the incubator 99
- serve as the independent 794
- serve as the index 223
- serve as the indication 72
- serve as the indicator 120
- serve as the indicators 45
- serve as the indispensable 111
- serve as the indispensable 40
- serve as the individual 234

<http://googleresearch.blogspot.com/2006/08/all-our-n-gram-are-belong-to-you.html>

Problem: Zeros

- Training set:
 - ... denied the allegations
 - ... denied the reports
 - ... denied the claims
 - ... denied the request

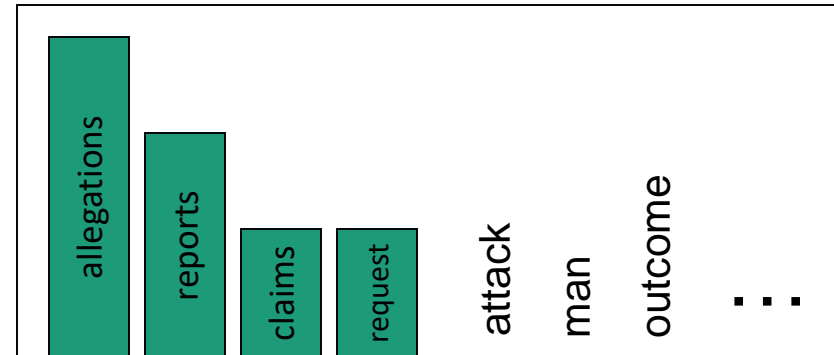
$$P(\text{"offer"} \mid \text{denied the}) = 0$$

- Test set
 - ... denied the offer
 - ... denied the loan

Smoothing: the intuition

- When we have sparse statistics:

$P(w \mid \text{denied the})$
3 allegations
2 reports
1 claims
1 request
7 total



- Steal probability mass to generalize better

$P(w \mid \text{denied the})$
2.5 allegations
1.5 reports
0.5 claims
0.5 request
2 other
7 total



Add-one estimation

- Also called Laplace smoothing
- Pretend we saw each word one more time than we did (i.e. just add one to all the counts)

- MLE estimate:

$$P_{MLE}(w_i | w_{i-1}) = \frac{c(w_{i-1}, w_i)}{c(w_{i-1})}$$

- Add-1 estimate:

$$P_{Add-1}(w_i | w_{i-1}) = \frac{c(w_{i-1}, w_i) + 1}{c(w_{i-1}) + V}$$

Berkeley Restaurant Corpus: Laplace smoothed bigram counts

	i	want	to	eat	chinese	food	lunch	spend
i	6	828	1	10	1	1	1	3
want	3	1	609	2	7	7	6	2
to	3	1	5	687	3	1	7	212
eat	1	1	3	1	17	3	43	1
chinese	2	1	1	1	1	83	2	1
food	16	1	16	1	2	5	1	1
lunch	3	1	1	1	1	2	1	1
spend	2	1	2	1	1	1	1	1

Laplace-smoothed bigrams

$$P^*(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n) + 1}{C(w_{n-1}) + V}$$

	i	want	to	eat	chinese	food	lunch	spend
i	0.0015	0.21	0.00025	0.0025	0.00025	0.00025	0.00025	0.00075
want	0.0013	0.00042	0.26	0.00084	0.0029	0.0029	0.0025	0.00084
to	0.00078	0.00026	0.0013	0.18	0.00078	0.00026	0.0018	0.055
eat	0.00046	0.00046	0.0014	0.00046	0.0078	0.0014	0.02	0.00046
chinese	0.0012	0.00062	0.00062	0.00062	0.00062	0.052	0.0012	0.00062
food	0.0063	0.00039	0.0063	0.00039	0.00079	0.002	0.00039	0.00039
lunch	0.0017	0.00056	0.00056	0.00056	0.00056	0.0011	0.00056	0.00056
spend	0.0012	0.00058	0.0012	0.00058	0.00058	0.00058	0.00058	0.00058

Reconstituted counts

$$c^*(w_{n-1}w_n) = \frac{[C(w_{n-1}w_n) + 1] \times C(w_{n-1})}{C(w_{n-1}) + V}$$

	i	want	to	eat	chinese	food	lunch	spend
i	3.8	527	0.64	6.4	0.64	0.64	0.64	1.9
want	1.2	0.39	238	0.78	2.7	2.7	2.3	0.78
to	1.9	0.63	3.1	430	1.9	0.63	4.4	133
eat	0.34	0.34	1	0.34	5.8	1	15	0.34
chinese	0.2	0.098	0.098	0.098	0.098	8.2	0.2	0.098
food	6.9	0.43	6.9	0.43	0.86	2.2	0.43	0.43
lunch	0.57	0.19	0.19	0.19	0.19	0.38	0.19	0.19
spend	0.32	0.16	0.32	0.16	0.16	0.16	0.16	0.16

Reconstituted vs. raw bigram counts

	i	want	to	eat	chinese	food	lunch	spend
i	5	827	0	9	0	0	0	2
want	2	0	608	1	6	6	5	1
to	2	0	4	686	2	0	6	211
eat	0	0	2	0	16	2	42	0
chinese	1	0	0	0	0	82	1	0
food	15	0	15	0	1	4	0	0
lunch	2	0	0	0	0	1	0	0
spend	1	0	1	0	0	0	0	0

	i	want	to	eat	chinese	food	lunch	spend
i	3.8	527	0.64	6.4	0.64	0.64	0.64	1.9
want	1.2	0.39	238	0.78	2.7	2.7	2.3	0.78
to	1.9	0.63	3.1	430	1.9	0.63	4.4	133
eat	0.34	0.34	1	0.34	5.8	1	15	0.34
chinese	0.2	0.098	0.098	0.098	0.098	8.2	0.2	0.098
food	6.9	0.43	6.9	0.43	0.86	2.2	0.43	0.43
lunch	0.57	0.19	0.19	0.19	0.19	0.38	0.19	0.19
spend	0.32	0.16	0.32	0.16	0.16	0.16	0.16	0.16

Add-1 estimation is a blunt instrument

- So add-1 isn't used for N-grams
 - Typically use back-off and interpolation instead
- But add-1 is used to smooth other NLP models
 - E.g., Naïve Bayes for text classification
 - in domains where the number of zeros isn't so huge.

Backoff

- Sometimes it helps to use **less** context
 - Condition on less context for contexts you haven't learned much about
- **Backoff:**
 - use trigram if you have good evidence,
 - otherwise bigram, otherwise unigram

Smoothing for web-scale N-grams

- “Stupid backoff” (Brants *et al.* 2007)
- No discounting, just use relative frequencies

$$S(w_i | w_{i-k+1}^{i-1}) = \begin{cases} \frac{\text{count}(w_{i-k+1}^i)}{\text{count}(w_{i-k+1}^{i-1})} & \text{if } \text{count}(w_{i-k+1}^i) > 0 \\ 0.4S(w_i | w_{i-k+2}^{i-1}) & \text{otherwise} \end{cases}$$

$$S(w_i) = \frac{\text{count}(w_i)}{N}$$

Unknown words: Open vocabulary vs. closed vocabulary tasks

- If we know all the words in advanced
 - Vocabulary V is fixed
 - Closed vocabulary task
- Often we don't know this
 - **Out Of Vocabulary** = OOV words
 - Open vocabulary task

Unknown words: Open vocabulary model with UNK token

- Define an unknown word token <UNK>
- Training of <UNK> probabilities
 - Create a fixed lexicon L of size V
 - Any training word not in L changed to <UNK>
 - Train language model probabilities as if <UNK> were a normal word
- At decoding time
 - Use <UNK> probabilities for any word not in training

Language Modeling Toolkits

- SRILM

- <http://www.speech.sri.com/projects/srilm/>

- KenLM

- <https://kheafield.com/code/kenlm/>

Roadmap

- Language Models
 - Our first example of modeling sequences
- n-gram language models
- How to estimate them?
- How to evaluate them?
- Neural models