CS474 Natural Language Processing

Language Modeling

- Introduction to generative models of language
  - What are they?
  - Why they' re important
  - Issues for counting words
  - Statistics of natural language
  - Unsmoothed n-gram models

What are *generative* models of language?

- Word prediction
  - *Once upon a…*
  - *I'd like to make a collect…*
  - *Let's go outside and take a…*
- Generative models can assign probabilities to strings of words

Why are word prediction models important?

- Augmentative communication systems
  - For the disabled, to predict the next words the user wants to "speak"
- Computer-aided education
  - System that helps kids learn to read (e.g. Mostow et al. system)
- Speech recognition
- Context-sensitive spelling correction
- …

N-gram model

- Uses the previous N-1 words to predict the next word
  - 2-gram: bigram
  - 3-gram: trigram
  - 1-gram: unigram
- In speech recognition, these statistical models of word sequences are referred to as a **language model**

Next…Language Modeling

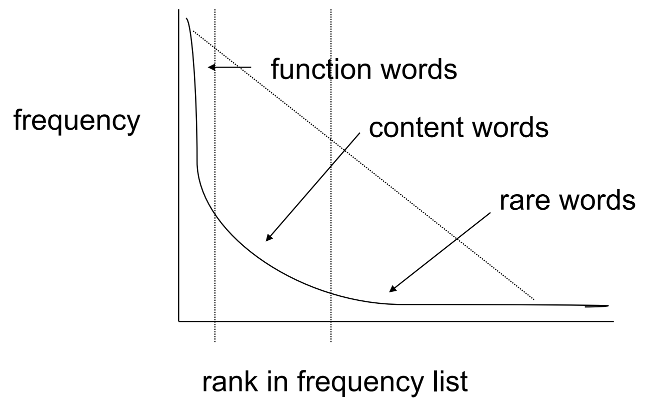
- Introduction to generative models of language
  - » What are they?
  - » Why they' re important
  - » Issues for counting words
  - » Statistics of natural language
  - » Unsmoothed n-gram models

- Introduction to generative models of language
  - » What are they?
  - » Why they' re important
  - » Issues for counting words
  - » **Statistics of natural language**
  - » Unsmoothed n-gram models

How many words are there in English?

- **Option 1: count the word entries in a dictionary**
  - OED: 600,000
  - American Heritage (3rd edition): 200,000
  - Actually counting lemmas not wordforms
- **Option 2: estimate from a corpus**
  - Switchboard: 2.4 million wordform tokens; 20,000 wordform types
  - Shakespeare' s complete works: 884,647 wordform tokens; 29,066 wordform types
  - Brown corpus: 1 million wordform tokens; 61,805 wordform types; 37,851 lemma types
  - Brown et al. 1992: 583 million wordform tokens, 293,181 wordform types

How are they distributed?



Statistical Properties of Text

- Zipf' s Law relates a term' s frequency to its rank
  - frequency ∝ 1/rank
  - There is a constant *k* such that *freq* * *rank* = *k*
- The most frequent words in one corpus may be rare words in another corpus
  - Example: "computer" in CACM vs. National Geographic
- Each corpus has a different, fairly small "working vocabulary"

These properties hold in a wide range of languages

Zipf' s Law (*Tom Sawyer*)

Word	Freq. (f)	Rank (r)	$f \cdot r$	Word	Freq. (f)	Rank (r)	$f \cdot r$
the	3332	1	3332	turned	51	200	10200
and	2972	2	5944	you'll	30	300	9000
a	1775	3	5235	name	21	400	8400
he	877	10	8770	comes	16	500	8000
but	410	20	8400	group	13	600	7800
be	294	30	8820	lead	11	700	7700
there	222	40	8880	friends	10	800	8000
one	172	50	8600	begin	9	900	8100
about	158	60	9480	family	8	1000	8000
more	138	70	9660	brushed	4	2000	8000
never	124	80	9920	sins	2	3000	6000
Oh	116	90	10440	Could	2	4000	8000
two	104	100	10400	Applausive	1	8000	8000

Manning and Schutze SNLP

Zipf' s Law

- Useful as a rough description of the frequency distribution of words in human languages
- Behavior occurs in a surprising variety of situations
  - English verb polysemy
  - References to scientific papers
  - Web page in-degrees, out-degrees
  - Royalties to pop-music composers

- Introduction to generative models of language
  - » What are they?
  - » Why they' re important
  - » Issues for counting words
  - » Statistics of natural language
  - » **Unsmoothed n-gram models**

Goals

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

Models of word sequences

- Simplest model
  - Let any word follow any other word (equally likely)
    - » P (word2 *follows* word1) = 1/# words in English = 1/# word types in corpus
- Probability distribution at least obeys actual relative word frequencies
    - » P (word2 *follows* word1) = # occurrences of word2 / # words in corpus

Models of word sequences

- Pay attention to the preceding words
  - "Let 's go outside and take a [ ]"

» walk	very reasonable
» break	quite reasonable
» shower	less reasonable

Compute conditional probability P (walk| let 's go…take a)

Probability of a word sequence

- P (w_1 w_2 … 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

- Markov assumption: probability of some future event (next word) depends only on a limited history of preceding events (previous words)
- Bigram model

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

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

- Trigram model
  - Conditions on the two preceding words
- N-gram approximation

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

Bigram grammar fragment

- Berkeley Restaurant Project

eat on	.16	eat Thai	.03
eat some	.06	eat breakfast	.03
eat lunch	.06	eat in	.02
eat dinner	.05	eat Chinese	.02
eat at	.04	eat Mexican	.02
eat a	.04	eat tomorrow	.01
eat Indian	.04	eat dessert	.007
eat today	.03	eat British	.001

- Can compute the probability of a complete string
  - P (I want to eat British food) = P(I|<s>) P(want|I) P(to|want) P(eat|to) P(British|eat) P(food|British)

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.

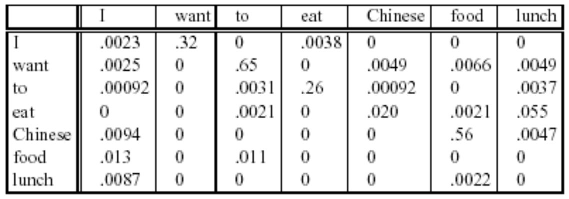
Bigram counts

	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…

Bigram probabilities

- Problem for the maximum likelihood estimates: sparse data



	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

Accuracy of N-gram models

- Accuracy increases as N increases
  - Train various N-gram models and then use each to generate random sentences.
  - Corpus: Complete works of Shakespeare
    - » **Unigram:** *Will rash been and by I the me loves gentle me not slavish page, the and hour; ill let*
    - » **Bigram:** *What means, sir. I confess she? Then all sorts, he is trim, captain.*
    - » **Trigram:** *Fly, and will rid me these news of price. Therefore the sadness of parting, as they say, 'tis done.*
    - » **Quadrigram:** *They say all lovers swear more performance than they are wont to keep obliged faith unforfeited!*

Strong dependency on training data

- Trigram model from WSJ corpus
  - *They also point to ninety nine point six billion dollars from two hundred four oh six three percent of the rates of interest stores as Mexico and Brazil on market conditions*