CS474 Natural Language Processing

- **Last class**
  - Noisy channel model
    - » Decoding algorithm
- **Today**
  - Introduction to generative models of language
    - » What are they?
    - » Why they're important
    - » Issues for counting words
    - » Statistics of natural language

Motivation for generative models

- Word prediction
  - *Once upon a…*
  - *I'd like to make a collect…*
  - *Let's go outside and take a…*
- The need for models of word prediction in NLP has not been uncontroversial
  - But it must be recognized that the notion "probability of a sentence" is an entirely useless one, under any known interpretation of this term.   -Noam Chomsky (1969)
  - Every time I fire a linguist the recognition rate improves. -Fred Jelinek (IBM speech group, 1988)

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
  - Use preceding context to improve solutions to the subproblem of pronunciation variation
- Lexical tagging tasks
- …

Why are word prediction models important?

- Closely related to the problem of computing the probability of a sequence of words
  - Can be used to assign a probability to the next word in an incomplete sentence
  - Useful for part-of-speech tagging, probabilistic parsing

N-gram model

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

CS474 Natural Language Processing

- Last class
  - Noisy channel model
    » Decoding algorithm
- Today
  - Introduction to generative models of language
    » What are they?
    » Why they're important
    » Issues for counting words
    » Statistics of natural language

Counting words in corpora

- Ok, so how many words are in this sentence?
- Depends on whether or not we treat punctuation marks as words
  - Important for many NLP tasks
    » Grammar-checking, spelling error detection, author identification, part-of-speech tagging
- Spoken language corpora
  - Utterances don't usually have punctuation, but they do have other phenomena that we might or might not want to treat as words
    » I do uh main- mainly business data processing
  - Fragments
  - Filled pauses
    » *um* and *uh* behave more like words, so most speech recognition systems treat them as such

Counting words in corpora

- Capitalization
  - Should *They* and *they* be treated as the same word?
    » For most statistical NLP applications, they are
    » Sometimes capitalization information is maintained as a feature
      ◆ E.g. spelling error correction, part-of-speech tagging
- Inflected forms
  - Should *walks* and *walk* be treated as the same word?
    » No…for most n-gram based systems
    » based on the **wordform** (i.e. the inflected form as it appears in the corpus) rather than the **lemma** (i.e. set of lexical forms that have the same stem)

Counting words in corpora

- Need to distinguish
  - word types
    - » the number of distinct words
  - word tokens
    - » the number of running words
- Example
  - *All for one and one for all.*
  - 8 tokens (counting punctuation)
  - 6 types (assuming capitalized and uncapitalized versions of the same token are treated separately)

CS474 Natural Language Processing

- Last class
  - Noisy channel model
    - » Decoding algorithm
- Today
  - Introduction to generative models of language
    - » What are they?
    - » Why they're important
    - » Issues for counting words
    - » Statistics of natural language

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?

frequency

function words

content words

rare words

rank in frequency list
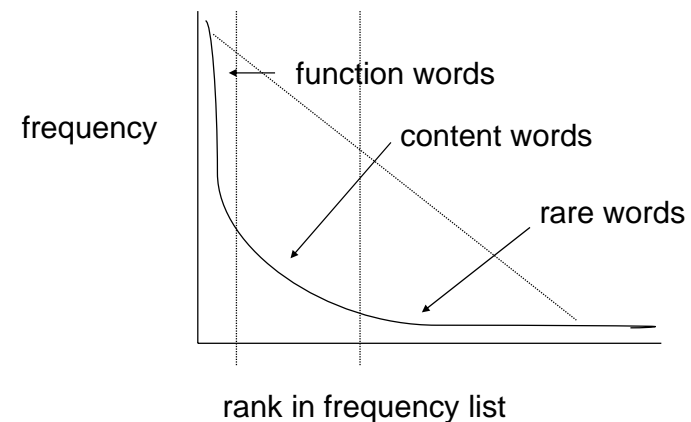
Statistical Properties of Text

- Zipf's Law relates a term's frequency to its rank
  - Frequency $\propto$ 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

Models of word sequences

- Simplest model
  - Let any word follow any other word
    » P (word1 follows word2) = 1/# words in English
- Probability distribution at least obeys actual relative word frequencies
    » P (word1 follows word2) =
      # occurrences of word1 / # words in English
- Pay attention to the preceding words
  - "Let's go outside and take a [   ]"
    » walk   very reasonable
    » break   quite reasonable
    » lion   less reasonable
  - Compute conditional probability P (walk| let's go…)