Questions that linguistics should answer

- What kinds of things do people say?
- What do these things say/ask/request about the world?

Example: *In addition to this, she insisted that women were regarded as a different existence from men unfairly.*

- Text corpora give us *data* with which to answer these questions
- They are an externalization of linguistic knowledge
- What words, rules, statistical facts do we find?
- Can we build programs that learn effectively from this data, and can then do NLP tasks?

Corpora

- A *corpus* is a *body* of naturally occurring text, normally one organized or selected in some way
  - Latin: one corpus, two corpora
- A *balanced corpus* tries to be representative across a language or other domain
- Balance is something of a chimaera: What is balanced? Who spends what percent of their time reading the sports pages?

The Brown corpus

- Famous early corpus. Made by W. Nelson Francis and Henry Kučera at Brown University in the 1960s. A balanced corpus of written American English in 1960 (except poetry!).
- 1 million words, which seemed huge at the time.
  Sorting the words to produce a word list took 17 hours of (dedicated) processing time, because the computer (an IBM 7070) had the equivalent of only about 40 kilobytes of memory, and so the sort algorithm had to store the data being sorted on tape drives.
- Its significance has increased over time, but also awareness of its limitations.
- Tagged for part of speech in the 1970s
  - The/AT General/JJ-TL Assembly/NN-TL ,/, which/WDT adjourns/VBZ today/NR ,/, has/HVZ performed/VBN

Recent corpora

- British National Corpus. 100 million words, tagged for part of speech. Balanced.
- Newswire (*NYT* or *WSJ* are most commonly used): Something like 600 million words is fairly easily available.
- Legal reports; UN or EU proceedings (parallel multilingual corpora – same text in multiple languages)
- The Web (in the billions of words, but need to filter for distinctness).
- Penn Treebank: 2 million words (1 million *WSJ*, 1 million speech) of parsed sentences (as phrase structure trees).

Common words in *Tom Sawyer* (71,370 words)

Word	Freq.	Use
the	3332	determiner (article)
and	2972	conjunction
a	1775	determiner
to	1725	preposition, verbal infinitive marker
of	1440	preposition
was	1161	auxiliary verb
it	1027	(personal/expletive) pronoun
in	906	preposition
that	877	complementizer, demonstrative
he	877	(personal) pronoun
I	783	(personal) pronoun
his	772	(possessive) pronoun
you	686	(personal) pronoun
Tom	679	proper noun
with	642	preposition

Frequencies of frequencies in *Tom Sawyer*

Word Frequency	Frequency of Frequency		
1	3993	71,730	word tokens
2	1292	8,018	word types
3	664		
4	410		
5	243		
6	199		
7	172		
8	131		
9	82		
10	91		
11–50	540		
51–100	99		
> 100	102		

Zipf's law in *Tom Sawyer*

Word	Freq. (f)	Rank (r)	$f \cdot r$
the	3332	1	3332
and	2972	2	5944
a	1775	3	5235
he	877	10	8770
but	410	20	8400
be	294	30	8820
there	222	40	8880
one	172	50	8600
about	158	60	9480
more	138	70	9660
never	124	80	9920
Oh	116	90	10440
two	104	100	10400

Zipf's law in *Tom Sawyer*

Word	Freq. (f)	Rank (r)	$f \cdot r$
turned	51	200	10200
you'll	30	300	9000
name	21	400	8400
comes	16	500	8000
group	13	600	7800
lead	11	700	7700
friends	10	800	8000
begin	9	900	8100
family	8	1000	8000
brushed	4	2000	8000
sins	2	3000	6000
Could	2	4000	8000
Applausive	1	8000	8000

Zipf's law

$$f \propto \frac{1}{r} \qquad (1)$$

There is a constant k such that

$$f \cdot r = k \qquad (2)$$

(Now frequently invoked for the web too!

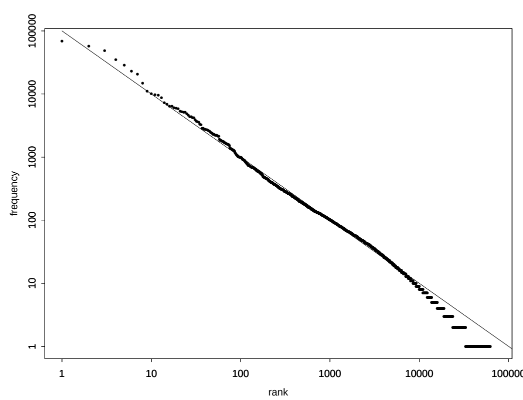
See http://linkage.rockefeller.edu/wli/zipf/)

Mandelbrot's law

$$f = P(r + \rho)^{-B} \qquad (3)$$

$$\log f = \log P - B \log(r + \rho) \qquad (4)$$

Zipf's law for the Brown corpus



NLP: Large, sparse, discrete distributions

- Both features and assigned classes regularly involve multinomial distributions over huge numbers of values (often in the tens of thousands).
- The distributions are very uneven, and have fat tails
- Enormous problems with data sparseness: much work on smoothing distributions/backoff (shrinkage), etc.
- We normally have inadequate (labeled) data to estimate probabilities
- Unknown/unseen things are usually a central problem
- Generally dealing with discrete distributions though

Sparsity

- How often does an every day word like *kick* occur in a million words of text?
  - *kick*: about 10 [depends vastly on genre, of course]
  - *wrist*: about 5
- Normally we want to know about something bigger than a single word, like how often you *kick a ball*, or how often the conative alternation *he kicked **at** the balloon* occurs.
- How often can we expect that to occur in 1 million words?
- Almost never.
- "There's no data like more data" [if of the right domain]

Probabilistic language modeling

- Assigns probability $P(t)$ to a word sequence $t = w_1 w_2 \cdots w_n$
- Chain rule and joint/conditional probabilities for text t:

$$P(t) = P(w_1 \cdots w_n) = P(w_1) \cdots P(w_n | w_1, \cdots w_{n-1}) = \prod_{i=1}^{n} P(w_i | w_1 \cdots w_{i-1})$$

where

$$P(w_k | w_1 \ldots w_{k-1}) = \frac{P(w_1 \ldots w_k)}{P(w_1 \ldots w_{k-1})} \approx \frac{C(w_1 \ldots w_k)}{C(w_1 \ldots w_{k-1})}$$

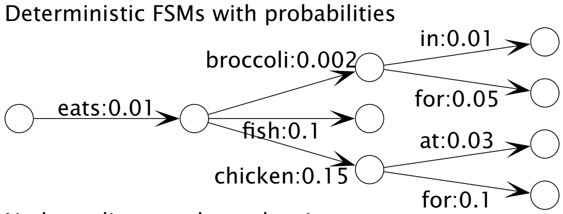
- The chain rule leads to a *history-based* model: we predict following things from past things
- But there are too many histories; we need to cluster histories into equivalence classes

n-gram models: the classic example of a statistical model of language

- Each word is predicted according to a conditional distribution based on a limited prior context
- Conditional Probability Table (CPT): $P(X|\textit{both})$
  - $P(\textit{of}|\textit{both}) = 0.066$
  - $P(\textit{to}|\textit{both}) = 0.041$
  - $P(\textit{in}|\textit{both}) = 0.038$
- From 1940s onward (or even 1910s – Markov 1913)
- a.k.a. Markov (chain) models

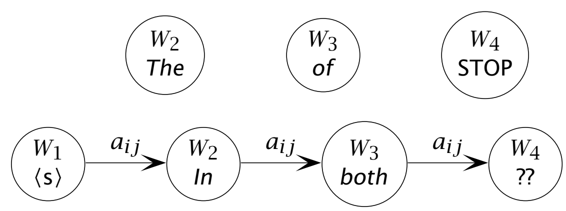
Markov models = *n*-gram models

- Deterministic FSMs with probabilities



- No long distance dependencies
  - “The future is independent of the past given the present”
- No notion of structure or syntactic dependency
- But lexical
- (And: robust, have frequency information, . . .)

Markov models = *n*-gram models



- Simplest linear graphical models
- Words are random variables, arrows are direct dependencies between them (CPTs)

n-gram models

- Core language model for the engineering task of better predicting the next word:
  - Speech recognition
  - OCR
  - Context-sensitive spelling correction
- These simple engineering models have just been amazingly successful.
- It is only recently that they have been improved on for these tasks (Chelba and Jelinek 1998; Charniak 2001).
- But linguistically, they are appalling simple and naive

n-th order Markov models

- First order Markov assumption = bigram

$$P(w_k | w_1 \ldots w_{k-1}) \approx P(w_k | w_{k-1}) = \frac{P(w_{k-1} w_k)}{P(w_{k-1})}$$

- Similarly, n-th order Markov assumption
- Most commonly, trigram (2nd order):

$$P(w_k | w_1 \ldots w_{k-1}) \approx P(w_k | w_{k-2}, w_{k-1}) = \frac{P(w_{k-2} w_{k-1} w_k)}{P(w_{k-2}, w_{k-1})}$$

Why mightn't n-gram models work?

- Relationships (say between subject and verb) can be arbitrarily distant and convoluted, as linguists love to point out:
  - The ***man*** that I was watching without pausing to look at what was happening down the street, and quite oblivious to the situation that was about to befall him confidently ***strode*** into the center of the road.

Why do they work?

- That kind of thing doesn't happen much
- Collins (1997):
  - 74% of dependencies (in the Penn Treebank – WSJ) are with an adjacent word (95% with one ≤ 5 words away), once one treats simple NPs as units:
  - Below, 4/6 = 66% based on words

The post office will hold out discounts

Why is that?

Sapir (1921: 14):

> 'When I say, for instance, "I had a good breakfast this morning," it is clear that I am not in the throes of laborious thought, that what I have to transmit is hardly more than a pleasurable memory symbolically rendered in the grooves of habitual expression. . . . It is somewhat as though a dynamo capable of generating enough power to run an elevator were operated almost exclusively to feed an electric doorbell.'

Evaluation of language models

- Best evaluation of probability model is task-based
- As substitute for evaluating one component, standardly use corpus per-word cross entropy:

$$H(X, \mathrm{p}) = -\frac{1}{n}\sum_{i=1}^{n} \log_2 P(w_i|w_1, \ldots, w_{i-1})$$

- Shannon game: try to predict next word in discourse
- Or perplexity (measure of uncertainty of predictions):

$$PP(X, \mathrm{p}) = 2^{H(X,\mathrm{p})} = \left[\prod_{i=1}^{n} P(w_i|w_1, \ldots, w_{i-1})\right]^{-1/n}$$

- *Needs to be assessed on independent, unseen, test data*

Relative frequency = Maximum Likelihood Estimate

$$P(w_2|w_1) = \frac{C(w_1, w_2)}{C(w_1)}$$

(or similarly for higher order or joint probabilities)

Makes training data as probable as possible

	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

Selected bigram counts (Berkeley Restaurant Project – J&M)

	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

Selected bigram probabilities (Berkeley Restaurant Project – J&M)

Limitations of Maximum Likelihood Estimator

Problem: We are often infinitely surprised when unseen word appears ($P(unseen) = 0$)

- Problem: this happens commonly.
- Probabilities of zero count words are too low
- Probabilities of nonzero count words are too high
- Estimates for high count words are fairly accurate
- Estimates for low count words are mostly inaccurate
- We need smoothing! (We flatten spiky distribution and give shavings to unseen items.)

Adding one = Laplace's law (1851)

$$P(w_2|w_1) = \frac{C(w_1, w_2) + 1}{C(w_1) + V}$$

- V is the vocabulary size (assume fixed, closed vocabulary)
- This is the Bayesian (MAP) estimator you get by assuming a uniform unit prior on events (= a Dirichlet prior)

	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 counts (Berkeley Restaurant Project – J&M)

	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

Add one probabilities (Berkeley Restaurant Project – J&M)

	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

	I	want	to	eat	Chinese	food	lunch
I	6	740	.68	10	.68	.68	.68
want	2	.42	331	.42	3	4	3
to	3	.69	8	594	3	.69	9
eat	.37	.37	1	.37	7.4	1	20
Chinese	.36	.12	.12	.12	.12	15	.24
food	10	.48	9	.48	.48	.48	.48
lunch	1.1	.22	.22	.22	.22	.44	.22

Original versus add-one predicted counts

Add one estimator

- Problem: gives too much probability mass to unseens.
- Not good for large vocab, comparatively little data (i.e., NLP)
- e.g 10,000 word vocab, 1,000,000 words of training data, but *comes across* occurs 10 times. Of those, 8 times next word is *as*
  - $P_{\text{MLE}}(\textit{as}|\textit{comes across}) = 0.8$
  - $P_{+1}(\textit{as}|\textit{comes across}) = \frac{8+1}{10+10000} \approx 0.0009$

Partial fixes

- Quick fix: Lidstone's law (Mitchell's (1997) "m-estimate"):

$$P(w_2|w_1) = \frac{C(w_1, w_2) + \lambda}{C(w_1) + \lambda V}$$

  for $\lambda < 1$, e.g., 1/2 or 0.05.
  - Mitchell's m-estimate sets λV to be m and subdividing it between the words
  - Doesn't correctly estimate difference between things seen 0 and 1 time
- Unigram prior
  - More likely to see next unseen words that are a priori common

$$P(w_2|w_1) = \frac{C(w_1, w_2) + \lambda P(w_2)}{C(w_1) + \lambda}$$

Absolute discounting

- Idea is that we want to discount counts of seen things a little, and reallocate this probability mass to unseens
- By subtracting a fixed count, probability estimates for commonly seen things are scarcely affected, while probabilities of rare things are greatly affected
- If the discount is around $\delta = 0.75$, then seeing something once is not so different to not having seen it at all

$$P(w_2|w_1) = (C(w_1, w_2) - \delta)/C(w_1) \quad \text{if} \quad C(w_1, w_2) > 0$$

$$P(w_2|w_1) = (V - N_0)\delta/N_0C(w_1) \quad \text{otherwise}$$

The frequency of previously unseen events

How do you know how likely you are to see a new word type in the future (in a certain context)?

- Examine some further text and find out [empirical held out estimators = validation]
- Use things you've seen once to estimate probability of unseen things:

$$P(\text{unseen}) = \frac{N_1}{N}$$

  where N_1 is number of things seen once. (Good-Turing: Church and Gale 1991; Gale and Sampson 1995)

Good-Turing smoothing

Derivation reflects leave-one out estimation (Ney et al. 1997):

- For each word token in data, call it the test set; remaining data is training set
- See how often word in test set has r counts in training set
- This will happen every time word left out has $r+1$ counts in original data
- So total count mass of r count words is assigned from mass of $r+1$ count words [$= N_{r+1} \times (r+1)$]
- Doesn't require held out data (which is good!)

Good-Turing smoothing

- r^* is corrected frequency estimate for word occurring r times
- There are N_r words with count r in the data
- $N_r \times r^* = N_{r+1} \times (r+1)$ or
- $r^* = \frac{N_{r+1} \times (r+1)}{N_r}$
- Or if w had frequency r, $P(w) = (r+1)N_{r+1}/N_rN$
- All words with same count get same probability
- This reestimation needs smoothing.
- For small r, $N_r > N_{r+1}$. But what of *the*?
- Simple Good Turing: use best-fit power law on low count counts.

Smoothing: Rest of the story (1)

- Other methods: backoff (Katz 1987), cross-validation, Witten-Bell discounting, ... (Chen and Goodman 1998; Goodman 2001)
- Simple, but surprisingly effective: Simple linear interpolation (deleted interpolation; mixture model; shrinkage):

$$\hat{P}(w_3|w_1,w_2)=\lambda_3 P_3(w_3|w_1,w_2)+\lambda_2 P_2(w_3|w_2)+\lambda_1 P_1(w_3)$$

- The λ_i can be estimated on held out data
- They can be functions of (equivalence-classed) histories
- For open vocabulary, need to handle words unseen in any context (just use UNK, spelling models, etc.)

Smoothing: Rest of the story (2)

- Recent work emphasizes constraints on the smoothed model
- Kneser and Ney (1995): Backoff n-gram counts not proportional to frequency of n-gram in training data but to expectation of how often it should occur in novel trigram – since one only uses backoff estimate when trigram not found
- (Smoothed) maximum entropy (a.k.a. loglinear) models again place constraints on the distribution (Rosenfeld 1996, 2000)

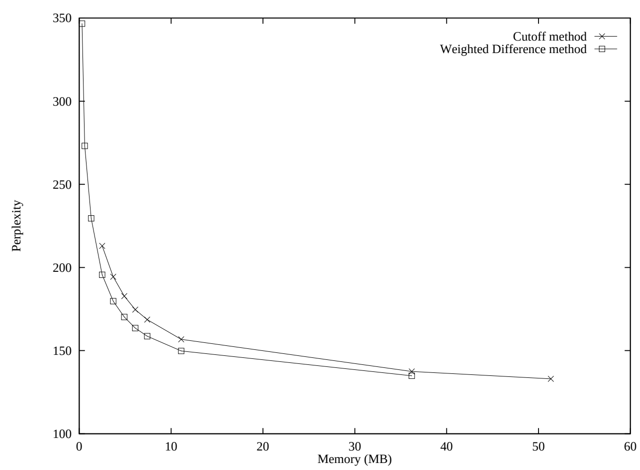
Size of language models with cutoffs

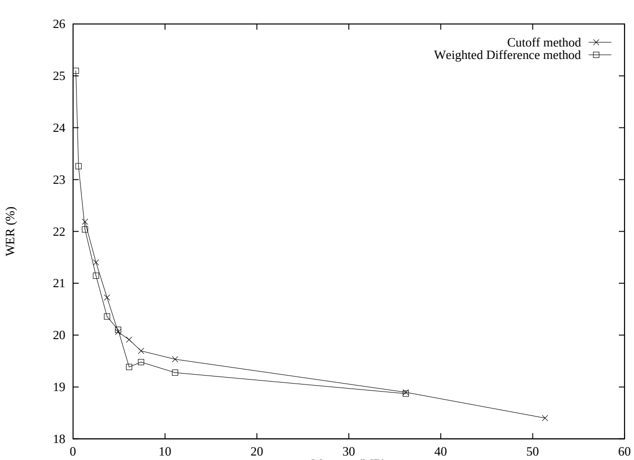
Seymore and Rosenfeld (*ICSLP*, 1996): 58,000 word dictionary, 45 M words of training data, *WSJ*, Sphinx II

Bi/Tri-gram cutoff	# Bigrams	# Trigrams	Memory (MB)
0/0	4,627,551	16,838,937	104
0/1	4,627,551	3,581,187	51
1/1	1,787,935	3,581,187	29
10/10	347,647	367,928	4

80% of unique trigrams occur only once!

- Note the possibilities for compression (if you're confident that you'll be given English text and the encoder/ decoder can use very big tables)





More LM facts

- Seymore, Chen, Eskenazi and Rosenfeld (1996)
- HUB-4: Broadcast News 51,000 word vocab, 130M words training. Katz backoff smoothing (1/1 cutoff).
- Perplexity 231
- 0/0 cutoff: 3% perplexity reduction
- 7-grams: 15% perplexity reduction
- Note the possibilities for compression, if you're confident that you'll be given English text (and the encoder/ decoder can use very big tables)