

Lecture 3: Language models – given the text so far, what comes....



Professor Robert C. Berwick
berwick@csail.mit.edu

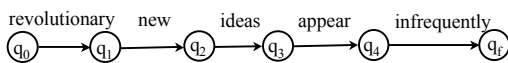
6.863J/9.611J SP11

The Menu Bar

- Administrivia
- Where do parts of speech come from? A linear model and its computational and learning implications
- Lab 2: What's a language model? (in this restricted case)
- Language models & n -grams: what is the task?
 - How to calculate?
 - What are the problems?
 - Never trust a sample under 30
 - Smoothing – the Dark Arts & the Robin Hood trick: the two key questions for Robin Hood that determine the space of possible “smoothing methods” for n -grams

6.863J/9.611J SP11

Simplest example of language ‘modeling’: beads on a string



In general: we want to find, or better, estimate from the corpus, the probability distribution for strings of tokens (letters, words, ...)

Question to keep in mind: when is ‘beads on a string model’ inappropriate???

Example: *unlockable*

6.863J/9.611J SP11

The Task

- Find a probability distribution for sentences, $p(w)$ which are strings of words
- In particular, find probability for a word in a text (utterance, etc.), given what the last n words *have been* ($n = 0, 1, 2, 3, \dots$)
- Why this is reasonable
- What the problems are: parameter estimation & conceptual issues

6.863J/9.611J SP11

Why this is reasonable

- The last few words may tell us a lot about the next word:
Collocations (“kick the bucket”)
prediction of current category:
the is followed by nouns or adjectives
semantic domain

6.863J/9.611J SP11

What is this good for?

- Spam vs. Ham
- Text categorization: like language ID
 - Topic 1 sample: In the beginning God created ...
 - Topic 2 sample: The history of all hitherto existing society is the history of class struggles. ...
- Input text: Matt’s Communist Homepage. Capitalism is unfair and has been ruining the lives of millions of people around the world. The profits from the workers’ labor ...
- Input text: And they have beat their swords to ploughshares, And their spears to pruning-hooks. Nation doth not lift up sword unto nation, neither do they learn war any more...
- Speech recognition. Only trigrams seem to work (*)
- Part of speech tagging (*the big book* – *Article Adjective* ___)

6.863J/9.611J SP11

What is this good for?

- Text categorization: Break big document or media stream into indexable chunks
- From NPR’s *All Things Considered*:
- The U. N. says its observers will stay in Liberia only as long as West African peacekeepers do, but West African states are threatening to pull out of the force unless Liberia’s militia leaders stop violating last year’s peace accord after 7 weeks of chaos in the capital, Monrovia ... Human rights groups cite peace troops as among those smuggling the arms. I’m Jennifer Ludden, reporting. Whitewater prosecution witness David Hale began serving a 28-month prison sentence today. The Arkansas judge and banker pleaded guilty two years ago to defrauding the Small Business Administration. Hale was the main witness in the Whitewater-related trial that led to the convictions ...

6.863J/9.611J SP11

Speech Recognition

Listen carefully: what am I saying?

- How do you recognize speech?
- How do you wreck a nice beach?

- Put the file in the folder
- Put the file and the folder

6.863J/9.611J SP11

Language generation

- Choose randomly among outputs:
 - Visitant which came into the place where it will be Japanese has admired that there was Mount Fuji.
- Top 10 outputs according to bigram probabilities:
 - Visitors who came in Japan admire Mount Fuji.
 - Visitors who came in Japan admires Mount Fuji.
 - Visitors who arrived in Japan admire Mount Fuji.
 - Visitors who arrived in Japan admires Mount Fuji.
 - Visitors who came to Japan admire Mount Fuji.
 - A visitor who came in Japan admire Mount Fuji.
 - The visitor who came in Japan admire Mount Fuji.
 - Visitors who came in Japan admire Mount Fuji.
 - The visitor who came in Japan admires Mount Fuji.
 - Mount Fuji is admired by a visitor who came in Japan.

6.863J/9.611J SP11

What can trigram language models do for us?

SUBJECT: Tonight

Hi everybody , A few of us are meeting up at the &NAME , &NAME , around &NUM : &NUM for dinner and drinks . All welcome - no excuses now - deadlines a month away

SUBJECT: Re : HI

Hi , I just got a 15 year fixed mortgage. I found this website where Lenders compete for your business . I thought you may want to look at it . Thanks !

Can distinguish 'spam' from 'ham'

What can trigram language models do for us?

Ivan je čitao knjigu dok je Marija gledala neki film
Answer: Croatian

Compare probabilities of generating sentence s , given that it was from language L , pick the 'most likely' one (Is this right? We'll put that to one side for now...)

That is, for a given find maximum:

$$p(\text{sent}=s \mid \text{language}=\text{english})$$

$$p(s \mid \text{language}=\text{polish})$$

...

Is this what we really want to do?

Bayes' Theorem & language models

Recall: $p(A|B) = p(A,B)/p(B)$, so $p(A|B) p(B) = p(A,B) = p(B|A) p(A)$

Therefore, dividing both red sides by $p(B)$:

$$\bullet \quad p(A \mid B) = p(B \mid A) * p(A) / p(B)$$

Or in words:

posterior probability = (likelihood * prior pr of A)/pr B

- Easy to check by removing syntactic sugar
- Use 1: Converts $p(B \mid A)$ to $p(A \mid B)$ (switch from *likelihood* to *posterior* probability)
- Use 2: Updates $p(A)$ to $p(A \mid B)$ (updates *prior* probability of A to *posterior* probability of A given information B)
- Stare at it so you'll recognize it later

Language ID

- Given a sentence s , I suggested comparing its probability in different languages – comparing *likelihoods*:
 - $p(\text{SENT}=s \mid \text{LANG}=\text{english})$
 - $p(\text{SENT}=s \mid \text{LANG}=\text{polish})$
 - $p(\text{SENT}=s \mid \text{LANG}=\text{xhosa})$
 - But surely for language ID we should compare:
 - $p(\text{LANG}=\text{english} \mid \text{SENT}=s)$
 - $p(\text{LANG}=\text{polish} \mid \text{SENT}=s)$
 - $p(\text{LANG}=\text{xhosa} \mid \text{SENT}=s)$
- These are the *posterior* probabilities

Language ID

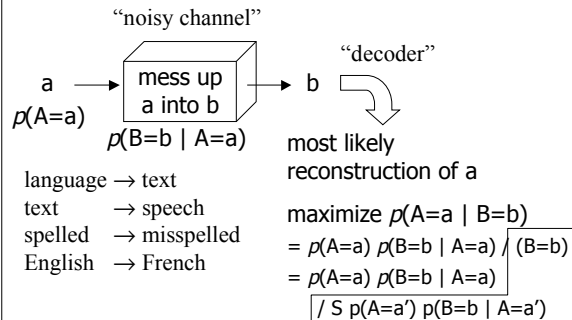
- For language ID we should compare
 - $p(\text{LANG}=\text{english} \mid \text{SENT}=s)$
 - $p(\text{LANG}=\text{polish} \mid \text{SENT}=s)$
 - $p(\text{LANG}=\text{xhosa} \mid \text{SENT}=s)$

a posteriori
- For ease, multiply by $p(\text{SENT}=s)$ (ie, over all realizations of s) and compare
 - $p(\text{LANG}=\text{english}, \text{SENT}=s)$
 - $p(\text{LANG}=\text{polish}, \text{SENT}=s)$
 - $p(\text{LANG}=\text{xhosa}, \text{SENT}=s)$

sum of these is a way to find $p(\text{SENT}=x)$; can divide back by that to get posterior probs
- Must know prior probabilities; then rewrite this as:

$p(\text{LANG}=\text{english})$	*	$p(\text{SENT}=s \mid \text{LANG}=\text{english})$
$p(\text{LANG}=\text{polish})$	*	$p(\text{SENT}=s \mid \text{LANG}=\text{polish})$
$p(\text{LANG}=\text{xhosa})$	*	$p(\text{SENT}=s \mid \text{LANG}=\text{xhosa})$
<i>a priori</i>		likelihood (what we had before)

General Case (“noisy channel”)



General Case (“noisy channel”)

- Want most likely A to have generated evidence B
 - $p(A = a1 \mid B = b)$
 - $p(A = a2 \mid B = b)$
 - $p(A = a3 \mid B = b)$

a posteriori
- For ease, multiply by $p(\text{SOUND}=x)$ and compare:
 - $p(A = a1, B = b)$
 - $p(A = a2, B = b)$
 - $p(A = a3, B = b)$
- Must know prior probabilities; then rewrite this as:

$p(A = a1)$	*	$p(B = b \mid A = a1)$
$p(A = a2)$	*	$p(B = b \mid A = a2)$
$p(A = a3)$	*	$p(B = b \mid A = a3)$
<i>a priori</i>		likelihood

Speech Recognition

- For baby speech recognition we should compare
 - $p(\text{MEANING}=\text{gimme} \mid \text{SOUND}=\text{uhh})$
 - $p(\text{MEANING}=\text{changeme} \mid \text{SOUND}=\text{uhh})$
 - $p(\text{MEANING}=\text{loveme} \mid \text{SOUND}=\text{uhh})$*a posteriori*
- For ease, multiply by $p(\text{SOUND}=\text{uhh})$ & compare
 - $p(\text{MEANING}=\text{gimme}, \text{SOUND}=\text{uhh})$
 - $p(\text{MEANING}=\text{changeme}, \text{SOUND}=\text{uhh})$
 - $p(\text{MEANING}=\text{loveme}, \text{SOUND}=\text{uhh})$
- Must know prior probabilities; then rewrite this as:

$p(\text{MEAN}=\text{gimme})$	*	$p(\text{SOUND}=\text{uhh} \mid \text{MEAN}=\text{gimme})$
$p(\text{MEAN}=\text{changeme})$	*	$p(\text{SOUND}=\text{uhh} \mid \text{MEAN}=\text{changeme})$
$p(\text{MEAN}=\text{loveme})$	*	$p(\text{SOUND}=\text{uhh} \mid \text{MEAN}=\text{loveme})$
<i>a priori</i>		likelihood

Simplest such ‘language model’: beads on a string, or *n*-grams

- But how to estimate all these probabilities?
- That’s what we’ll do for the rest of the lecture

So: Language models – review of terminology

- Simplest idea:

A probability function that assigns probabilities to word sequences, w , $p(w)$, e.g.,

$w = (\text{i}, \text{love}, \text{the}, \text{Red}, \text{Sox})$
- Note that the ‘event space’, possible word sequences, is infinite...!

$$p(w) = p(w_1, \dots, w_n)$$

$$= p(w_1)p(w_2|w_1)p(w_3|w_1, w_2) \dots p(w_n|w_1, \dots, w_{n-1})$$

$$= \prod_{i=1}^n p(w_i|w_1, \dots, w_{i-1})$$
- We let N denote the size of a corpus in words, and V denote its vocabulary (We will see that we have to tweak what V includes.)

Note: Often we’ll just let V just denote $|V|$ in what follows.

We use assumptions to limit ‘history’ length (context) in making probability estimates

Make a “Markov assumption,” or *backoff* to approximate this probability
 Either of order 1 (1-word history) or order 2 (2-word history):

$$p(w_i|w_1, \dots, w_{i-1}) \approx p(w_i|w_{i-1})$$

$$p(w_i|w_1, \dots, w_{i-1}) \approx p(w_i|w_{i-2}, w_{i-1})$$

Trigram model:
 $p(w) \approx p(it) \times p(was|it) \times p(a|it, was) \times \dots \times p(April|day, in)$

bigram or 2-gram (1-word history)
 trigram or 3-gram (2-word history)

We’ll also need to make reference to the following:

$$p(w_i) \quad \text{unigram or 1-gram (0-word history)}$$

$$p(\epsilon) \quad \text{0-gram (empty-string history)}$$

6.863J/9.611J SP11

Review of terminology

- Suppose $w = w_1 w_2 \dots w_n$ (a sequence of n words)
- Then a *trigram language model* defines:
$$p(w) \equiv p(w_1) \cdot p(w_2|w_1) \cdot p(w_3|w_1, w_2) \cdot p(w_4|w_2, w_3) \cdots p(w_n|w_{n-2}, w_{n-1})$$
 - On the assumption that the sequence was generated in the order w_1, w_2, w_3 (from left to right) with each word chosen in a way dependent on the previous two words. (Note the first word is *not* dependent on anything)
- We will often want to compare one $p(w)$ against another...and solve useful problems, as we saw
- How do we compare models? (roughly, want it to do best on predicting *unseen test data*)

6.863J/9.611J SP11

Let's check our understanding

- Use $\langle s \rangle$ and $\langle /s \rangle$ to delimit 'start' and 'stop' of sentence
- Claim: given English training data, the probability of $\langle s \rangle$ do you think the $\langle /s \rangle$ should be extremely low under any good language model
- In the case of the trigram model, which parameter(s) are responsible for this?

6.863J/9.611J SP11

Let's check our understanding

- You turn on the radio as it is broadcasting an interview. Assuming a trigram model, match up expressions (A), (B), (C) with descriptions (1), (2), (3):
(A) $p(\text{Do}) \cdot p(\text{you}|\text{Do}) \cdot p(\text{think}|\text{Do}, \text{you})$
(B) $p(\text{Do}|\langle s \rangle) \cdot p(\text{you}|\langle s \rangle, \text{Do}) \cdot p(\text{think}|\text{Do}, \text{you}) \cdot p(\langle /s \rangle|\text{you}, \text{think})$
(C) $p(\text{Do}|\langle s \rangle) \cdot p(\text{you}|\langle s \rangle, \text{Do}) \cdot p(\text{think}|\text{Do}, \text{you})$
- (1) the first complete sentence you hear is: *Do you think* (as in, "D'ya think?")
- (2) the first 3 words you hear are: *Do you think*
- (3) the first complete sentence you hear starts with: *Do you think* (Why do these distinctions matter?)

6.863J/9.611J SP11

One more standard trick

- *Force* vocabulary to be finite: assemble vocabulary by whatever means, and then add *one* special symbol OOV that represents all the other words
- New words are just various spellings of OOV
I saw the **snuffleupagus** on the TV =
I saw the OOV on the TV =
I saw [some out-of-vocabulary word] on the TV
- Isn't this probability higher than *particular* sentence w/ **snuffleupagus** ?
- Why is this not a problem when comparing models?
- How could we refine the **snuffleupagus** probability so it's not so high as OOV?

6.863J/9.611J SP11

It is now very *easy* to get bigram (or trigram) estimates

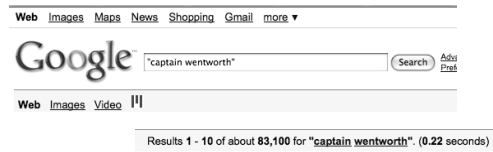
- Use Unix: an example to persuade you...5 commands or so...

```
more persuasion.wds
tail +2 persuasion.wds > persuasion.w2
paste persuasion.wds persuasion.w2 > bigrams
sort -d bigrams > bigrams.srt
uniq -c bigrams.srt > bigrams.cnt
sort -nr bigrams.cnt > bigrams.sct
430 of the
378 to be
323 in the
255 had been
227 she had
220 it was
196 captain wentworth
```

6.863J/9.611J SP11

Or everyone's favorite answer....

- Use Google search



6.863J/9.611J SP11

How to estimate probabilities

- How to estimate?
 - $p(\text{the} \mid \text{its water is so transparent that})$
- What is the "least knowledge" estimate of $p(w)$?
- Suppose V distinct vocabulary items
- Just use the uniform distribution, $1/V$ as the estimate for $p(w)$
- Call this the distribution D_Z (for 'zero knowledge')
- Can we do better? Sure...

6.863J/9.611J SP11

Using *counts* as Maximum Likelihood Estimates (MLE) of 'true' probabilities (aka, 'historical estimates' if we drew the sample many, many times)

$$p_{MLE}(w_i) = \frac{\text{count}(w_i)}{N} = \frac{C(w_i)}{N}$$
$$p_{MLE}(w_i \mid w_{i-1}) = \frac{\text{count}(w_{i-1}w_i)}{\sum_w \text{count}(w_{i-1}w)} = \frac{C(w_{i-1}w_i)}{C(w_{i-1})}$$

These distributions will be denoted D_{MLE}

6.863J/9.611J SP11

Summary of the 3 Steps for MLE: Step 1, get estimate

Let $w = w_1, w_2, \dots, w_n$ be a word sequence.

Given a training corpus, an intuitive estimate of the probability of the sequence $P(S)$, is the relative frequency of the string w_1, w_2, \dots, w_n in the corpus, the *maximum likelihood estimate* (MLE):

$$P_{MLE} = \frac{C(w_1, \dots, w_n)}{N}$$

where $C(w_1, \dots, w_n)$ is the frequency or count of the string w_1, \dots, w_n in the corpus and N is the total number of strings of length n .

Step 2, use chain rule

$$\begin{aligned} P(S) &= P(w_1, \dots, w_n) \\ &= P(w_1)P(w_2|w_1)P(w_3|w_1, w_2) \dots P(w_n|w_1, \dots, w_{n-1}) \\ &= \prod_{i=1}^n P(w_i|w_1, \dots, w_{i-1}) \\ P(S) &= P(it) \times P(was|it) \times P(a|it, was) \times P(bright|it, was, a) \times \dots \times \\ &\quad P(April|it, was, a, bright, cold, day, in) \end{aligned}$$

Step 3: use Markov assumption to limit 'history' length (context)

Make the Markov assumption, either order 1 (1 word history), or order 2 (2 word history):

$$\begin{aligned} P(w_i|w_1, \dots, w_{i-1}) &\approx P(w_i|w_{i-1}) \\ P(w_i|w_1, \dots, w_{i-1}) &\approx P(w_i|w_{i-2}, w_{i-1}) \end{aligned}$$

trigram model:

$$P(S) \approx P(it) \times P(was|it) \times P(a|it, was) \times P(bright|was, a) \times \dots \times P(April|day, in)$$

In practice, using maximum likelihood estimates are wonderful because they will maximize the probability of the training data, but they will be lousy on test data! You will see this for yourselves

Summary of MLE distribution estimate

Probability assigned *exactly* based on the n -gram count in the training corpus.

Call this distribution D_{ML}

Anything *not* in the training corpus gets probability 0.

E.g., in Brown corpus of 1 million words, "Chinese" occurs 400 times.

Estimated 1-gram probability for

$$w_i = \text{Chinese} = 400/1,000,000 = 0.0004$$

In practice, maximum likelihood estimates are *wonderful* because they will maximize the probability of the *training* data, but they will be *lousy* on *test* data!

MLE estimate assumes 'perfect knowledge' acquired from training data!

- But actually words are very rare events (even if we're not aware of that), so
- What feel like perfectly common sequences of words may be too rare to actually have in our training corpus

in a ____
 with a ____
 the last ____
 shot a ____
 open the ____
 over my ____
 President Barack ____
 keep tabs ____

6.863J/9.611J SP11

Example: Berkeley restaurant Unigram & Bigram Counts

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

- Out of 9222 sentences
- Eg. "I want" occurred 827 times

	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

Look at all the zeroes....!

Computing MLE bigram probabilities

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

- Divide bigram counts by prefix unigram counts to get probability estimates

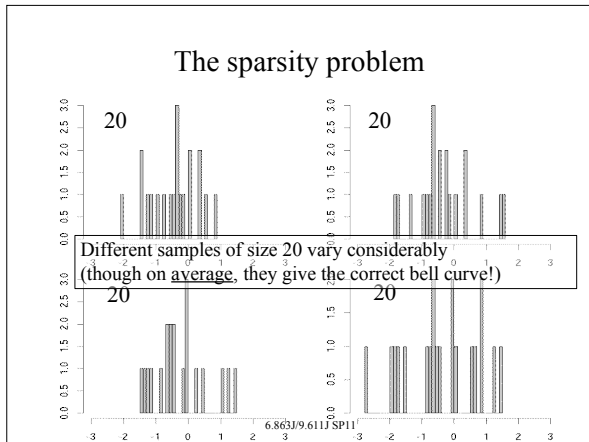
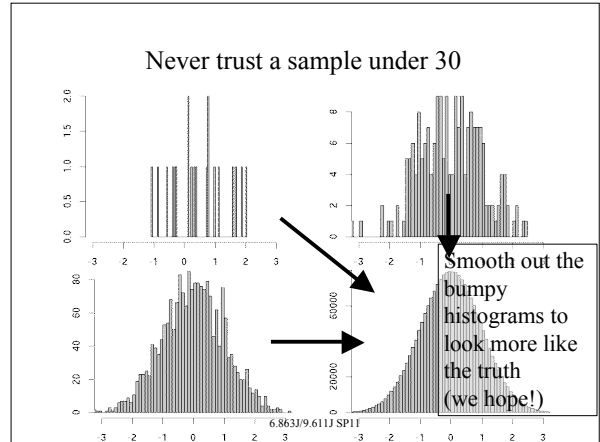
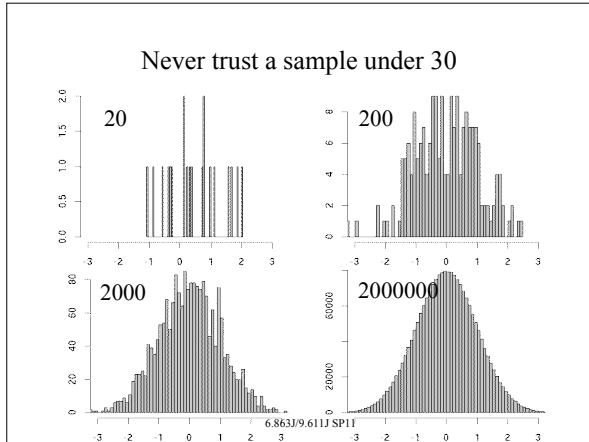
	i	want	to	eat	chinese	food	lunch	spend
i	0.002	0.33	0	0.0036	0	0	0	0.00079
want	0.0022	0	0.66	0.0011	0.0065	0.0054	0.0054	0.0011
to	0.00083	0	0.0017	0.28	0.00083	0	0.0025	0.087
eat	0	0	0.0027	0	0.021	0.0027	0.056	0
chinese	0.0063	0	0	0	0	0.52	0.0063	0
food	0.014	0	0.014	0	0.00092	0.0037	0	0
lunch	0.0059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

6.863J/9.611J SP11

Kinds of Knowledge

- As crude as they are, n -gram probabilities capture a range of interesting facts about language.
- $P(\text{english}|\text{want}) = .0011$
- $P(\text{chinese}|\text{want}) = .0065$ World knowledge
- $P(\text{to}|\text{want}) = .66$ Syntax
- $P(\text{eat} | \text{to}) = .28$
- $P(\text{food} | \text{to}) = 0$
- $P(\text{want} | \text{spend}) = 0$
- $P(i | \langle s \rangle) = .25$ Discourse

6.863J/9.611J SP11



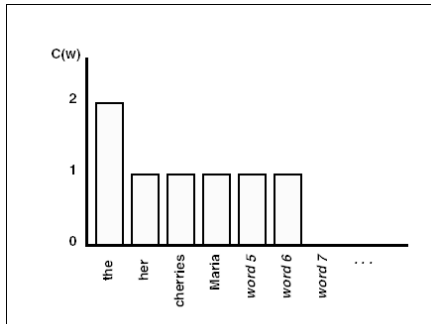
Example of poor prediction due to MLE estimate

Corpus: five Jane Austen novels

- $N = 617,091$ words (# tokens)
- $V = 14,585$ unique words (# types)
- Task: predict the next word of the trigram "inferior to _____"
- from test data, *Persuasion*: "[In person, she was] inferior to both [sisters.]"
- Question: how many possible *bigrams* are there?
 Ans: $V^2 = 212 \times 10^6$ so most trigrams will have count 0!
 And that is our MLE prediction – *which is just wrong*

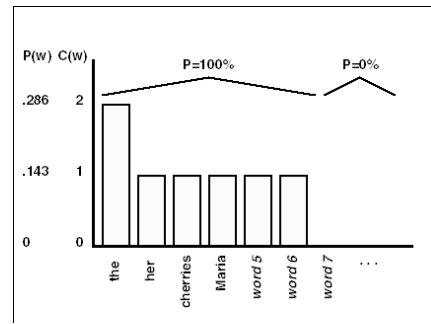
6.863J/9.611J SP11

The actual data



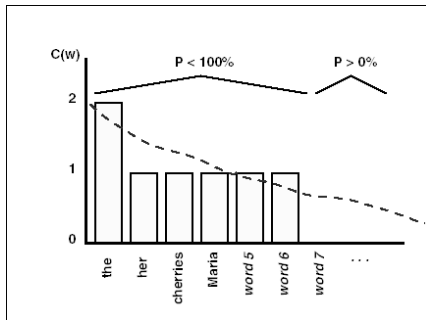
6.863J/9.611J SP11

How bad is the MLE compared to the 'truth'?



6.863J/9.611J SP11

The 'truth'



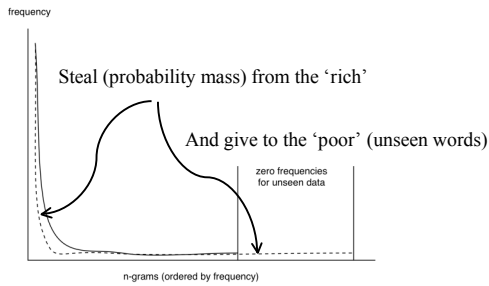
6.863J/9.611J SP11

Why not just collect more data?

- Note that the event space E is unbounded in NLP, so language model gives probability 0 to unseen words, bigrams, ..., parse trees, ...
- Moreover: if you do have enough data to estimate parameters, you probably should boost the n in your n -gram, to get a better model, and then you merely face the same problem all over again...
- So: If data sparsity is not a problem for you, then your model is too simple!

6.863J/9.611J SP11

Smoothing: the Robin Hood Answer (and the *Dark Arts*)



Sum of probability mass must still add up to 1!

6.863J/9.611J SP11

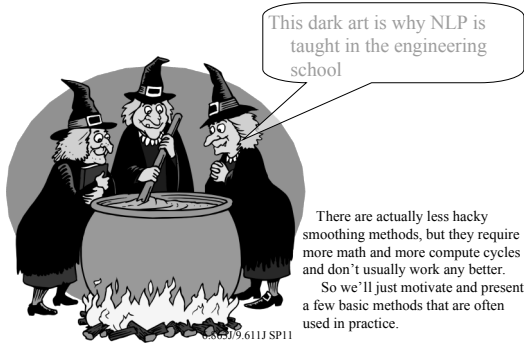
Smoothing has 2 parts: (1) discounting & (2) backoff

- Deals with events that have been observed zero times
- Generally improves accuracy of model
- Implements Robin Hood by answering 2 questions:
 1. Who do we *take the probability mass* away from, and how much? (= discounting)
 2. How do we *redistribute the probability mass*? (= backoff)

Let's cover two of the basic methods for discounting and backoff: Add-1; Witten-Bell

6.863J/9.611J SP11

Smoothing: the *Dark Side* of the force...



6.863J/9.611J SP11

To remember

- N is the total # of observed words/events in corpus
- V is the total # of distinct *total possible* vocabulary words (i.e., it is a number – the vocabulary size – don't confuse this with the *observed* vocabulary in a particular corpus...)
- What should we do if MLE estimate is 0?
(Because the count observed was 0 for an event)

6.863J/9.611J SP11

Simplest way to fix zeroes

- If MLE estimate is 0, then “back-off” to the uniform distribution estimate (our ‘base case’ of zero knowledge) – so that answers Question 1: we take from the MLE estimate and we give to the zeroes
- The second question, how to distribute?
- We just take some weighted average (interpolation of) the 2 distributions, D_Z and D_{MLE} (the zero knowledge and complete knowledge distributions)
- Define a weight λ s.t. we compute as our ‘backoff estimate’:

$$\lambda D_Z + (1-\lambda) D_{MLE} = \lambda 1/V + (1-\lambda) c(w)/N$$

How to distribute from the MLE estimate? (What should the weighting factor lambda be?)

What should the ‘weighting factor’ λ be? It should be in a weighted proportion according to both V and N

Recall the sum of the 2 probability distributions must add up to 1:

Set $\lambda = V/(V+N)$; then $(1-\lambda) = N/(V+N)$

Simplest fix

- So our estimate for w is now an interpolation (a weighted sum) between the unigram and the uniform distributions:

$$\frac{N}{V+N} \cdot \frac{C(w)}{N} + \frac{V}{V+N} \cdot \frac{1}{V} =$$

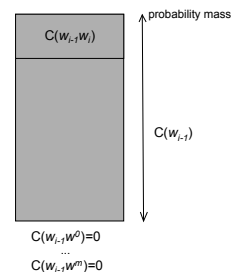
$$\frac{C(w)+1}{N+V}$$

- This method is called ‘Add-1’ because we add 1 to all the counts, and divide by V to normalize
- It’s really an interpolation (a weighted sum) of the unigram (or bigram, ...) MLE plus the uniform distribution

This idea applied to bigrams

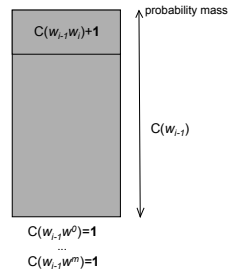
- *let’s illustrate the bigram case:*

- $w_{i-1}w_i$
- $P_{MLE}(w_i|w_{i-1}) = C(w_{i-1}w_i)/C(w_{i-1})$
- suppose there are cases $w_{i-1}w^0$ through $w_{i-1}w^m$ that don’t occur in the corpus



Smoothing and N-grams

- *add-one*
 - “give everyone 1”

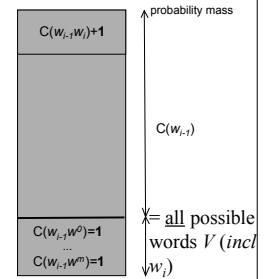


6.863J/9.611J SP11

Smoothing and N-grams

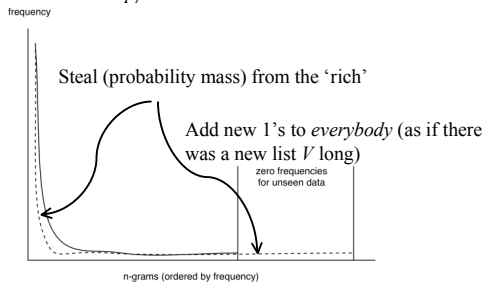
- *add-one*

- *redistribution of probability mass*
 - $p^*(w_i|w_{i-1}) = (C(w_{i-1}, w_i) + 1) / (C(w_{i-1}) + V)$
 - Must increase denominator to ensure probability mass redistributed still adds to 1



6.863J/9.611J SP11

Add-1 smoothing: the Robin Hood Answer



Problem: this adds way too much probability mass to unseen events! (It therefore robs too much from the actually observed events.) V is too big! We will see how to fix this in a bit...

6.863J/9.611J SP11

Example from restaurants: add 1

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

	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

6.863J/9.611J SP11

	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

	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

$p(\text{food} | \text{Chinese}) = 0.52$ dropped to 0.052 !!!
6.863J/9.611J SP11

Add-One Smoothing (Laplace, 1720)

xya	1	1/3	2	2/29
xyb	0	0/3	1	1/29
xyc	0	0/3	1	1/29
xyd	2	2/3	3	3/29
xye	0	0/3	1	1/29
...				
xyz	0	0/3	1	1/29
Total xy	3	3/3	29	29/29

6.863J/9.611J SP11

Add-One Smoothing

300 observations instead of 3 – better data, less smoothing

xya	100	100/300	101	101/326
xyb	0	0/300	1	1/326
xyc	0	0/300	1	1/326
xyd	200	200/300	201	201/326
xye	0	0/300	1	1/326
...				
xyz	0	0/300	1	1/326
Total xy	300	300/300	326	326/326

6.863J/9.611J SP11

Add-One Smoothing

Suppose we're considering 20,000 word types, not 26 letters

xya	1	1/3	2	2/29
xyb	0	0/3	1	1/29
xyc	0	0/3	1	1/29
xyd	2	2/3	3	3/29
xye	0	0/3	1	1/29
...				
xyz	0	0/3	1	1/29
Total xy	3	3/3	29	29/29

6.863J/9.611J SP11