Why cats smooth

A discussion of the basic of smoothing with occasional LOLcat pictures

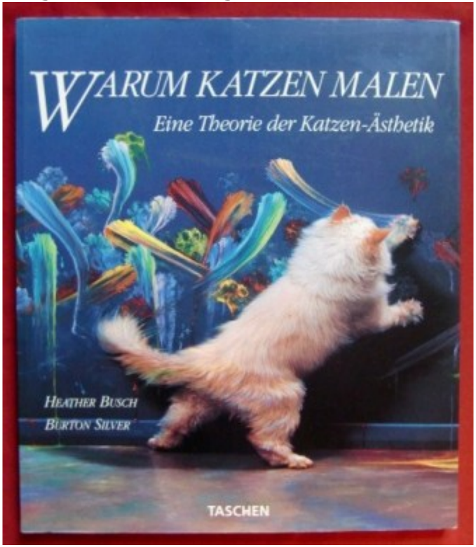
Asad Sayeed

COLI, Uni-Saarland

Q: How many of you ever recall hearing the phrase “Why cats smooth”?

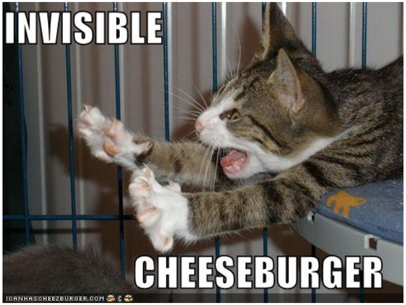
A: Probably none of you, but then, you’re all German native speakers.

A: But you may have heard of:



Q: So if your brain were a language model trained in trigram counts, what would the probability of "why cats smooth" be?

A: Probably zero, right?



N-grams

Let's check some n-gram counts.

- Google n-grams corpus.
- Has up to 5-grams.
- Includes punctuation as a token.
- Text files with counts searchable via zgrep.

A: Yep, zero.

Zero counts

But that can't possibly be right!

- I was able to say it.
- Clearly its probability in this universe is non-zero.
- As soon as these slides go up on the internet (do they do that here?), the trigram count in Google's index is going to change.

So a model that tells me that this can NEVER OCCUR—due to the zero count—is certainly wrong.

Zero counts

We know too little about the universe:

- We don't know what model *actually* generated "why cats smooth".
- We're not going to find out any time soon (or half of COLI will be out of work).
- We need a way to produce a good approximation, so that our models can at least handle some form of "novelty".

We generally call this *smoothing*. Whether it's correct or not, it's another matter.

A problem?

Just to remind you, why is this a problem in practical terms? Examples:

- Machine translation: need to translate phrases never seen before into phrases never seen before ...
- Information retrieval: make a “best guess” at what a new query intended ...
- etc...

People are creating new language all the time. N-gram models will never be large enough.

Unsmoothed probabilities

We need a place to start: maximum likelihood estimates (MLEs):

MLE for unigrams

$$P(w_i) = \frac{c_i}{N}$$

- w_i - the word
- c_i - the unsmoothed count of w_i
- N - the total number of *tokens* in the corpus

Unsmoothed probabilities

Obvious extension to bigrams

MLE for bigrams

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

Unsmoothed counts

Too hard to do a cube for now, so let's stick to unigrams and bigrams.
From Google n-grams:

	¡s¿	why	cats	smooth	paint
¡s¿	N/A	62101035	?	?	2116305
why	N/A	41664	9296	156	3496
cats	N/A	1015	30441	?	2943
smooth	N/A	?	53	?	?
paint	N/A	202	?	?	28139
UNIGRAM	95119665584	101568835	4720992	14040158	12272214

¡s¿ means "start" of sentence. (there's an end symbol as well)
TOTAL UNIGRAM COUNT: 1024908267229

Add-one smoothing

Intuition:

Instead of assuming that unseen n-grams have zero counts, let's pretend that they've been seen exactly once.

Consequences:

- Every unseen n-gram has exactly the same probability. Can't remotely be true, but whatever.

Colourless green ideas sleep furiously.

- We add 1 to everything else — fairness to what we HAVE seen.

This is also called "Laplace" smoothing.

Add-one smoothing

For unigrams:

Add-one adjusted probability

$$P^*_{\mathrm{Laplace}}(w_i) = \frac{c_i + 1}{N + V}$$

V is the vocabulary size, since we're increasing the token count by 1 per vocab item.
Alternatively, what does this do to the counts?

Add-one adjusted unigram counts

$$c_i^* = (c_i + 1)\frac{N}{N + V}$$

We don't just "add one"—we preserve the relative proportions through normalization.
Then we can divide by N to get to get the same probability.

Add-one smoothing

We can do this for bigrams by a simple extension:

Add-one adjusted bigram probability

$$P^*_{\text{Laplace}}(w_i|w_{i-1}) = \frac{C(w_{i-1}w_i)+1}{C(w_{i-1})+V}$$

And for the counts:

Add-one adjusted bigram counts

$$C^*(w_{i-1}w_i) = [C(w_{i-1}w_i)+1]\frac{C(w_{i-1})}{C(w_{i-1})+V}$$

The same structure and principle, really.

Add-one smoothing

Add-one smoothing is not typically used.

- It's very aggressive!
- Let's do a couple from our cat example.
- It "steals" too much probability mass from things we've actually seen.
- Not all *hapax legomena* are equally likely.

There are better ways to do it.

Discounting: an interlude

Before we move on... there's another way to look at add-one: in terms of a **discount**.

Add-one discount formula

$$d_c = \frac{c^*}{c}$$

This tells us how much we "stole" from a word with original count c in order to give to the unseen forms.

Good-Turing discounting

In add-one smoothing:

- We pretend we've seen unknown n-grams **once**.
- We don't take into account what effect it will have on the other n-grams.
- We compensate for it in a VERY crude manner.

We can do better:

- We can START by estimating how likely it is we're going to see something new.

Good-Turing discounting

Insight

The number of things we've never seen can be estimated from the number of things we've seen only once.

But THEN, that means that we have to steal probability from everyone else.

- How to do that fairly?
- We need to reestimate the probability of **everything** by the same principle.

Good-Turing discounting

Key concept: **frequency of frequency**.

- N_c — how often n-grams of frequency c appear in the corpus.

$$N_c = \sum_{x:\mathrm{count}(x)=c} 1$$

Then we can compute revised counts for everything.

$$c^* = (c+1)\frac{N_{c+1}}{N_c}$$

Good-Turing discounting

So how do we get the probability of missing items?

$$P^*_{GT}(\mathrm{count}(w) = 0) = \frac{N_1}{N}$$

Where N is the total number of tokens.
(Jurafsky and Martin leave the proof as an exercise for the reader, and so will we.)

Issues with Good-Turing

Some things to note:

- Assumes that the distribution of each bigram is binomial.
    - If you have a vocab V, then the number of bigrams is V^2, so what happens to OOV words?
- What happens when we don't know N_{c+1}?
    - We have to smooth out the frequency of frequency counts!
- We don't necessarily discount things where the count is big: probably reliable.
    - But everything must sum to 1!

So far we've focused mostly on bigrams.
But what about bigger "grams"?

Higher-order n-grams

What about “why cats smooth”?

- Not frequent enough to appear in Google n-grams.
- But maybe the bigrams will help us: “why cats” and “cats smooth”.

And even if bigrams don’t help us, maybe some other combination will get us a more realistic estimate.

Interpolation

So what we want to find is $P(w_n|w_{n-2}w_{n-1})$ — that's the definition of the probability of a trigram.

Linear interpolation

$$\hat{P}(w_n|w_{n-2}w_{n-1}) = \lambda_1 P(w_n|w_{n-2}w_{n-1})$$
$$+\lambda_2 P(w_n|w_{n-1}) + \lambda_3 P(w_n)$$

Then we just need to learn the λ weights (by EM or any other linear regression trick).
We can also make the weights context-dependent by making them relative to bigrams.

Backoff

An even better way: **Backoff**
For example, Katz (haha!) backoff.

$$P_{bo}(w_i|w_{i-n+1}\cdots w_{i-1}) = \begin{cases} d_{w_{i-n+1}\cdots w_i}\frac{C(w_{i-n+1}\ldots w_{i-1}w_i)}{C(w_{i-n+1}\cdots w_{i-1})} & \text{if } C(w_{i-n+1}\cdots w_i) > k \\ \alpha_{w_{i-n+1}\cdots w_{i-1}}P_{bo}(w_i|w_{i-n+2}\cdots w_{i-1}) & \text{otherwise} \end{cases}$$

$$\beta_{w_{i-n+1}\cdots w_{i-1}} = 1 - \sum_{\{w_i:C(w_{i-n+1}\cdots w_i)>k\}} d_{w_{i-n+1}\cdots w_i}\frac{C(w_{i-n+1}\ldots w_{i-1}w_i)}{C(w_{i-n+1}\cdots w_{i-1})}$$

$$\alpha_{w_{i-n+1}\cdots w_{i-1}} = \frac{\beta_{w_{i-n+1}\cdots w_{i-1}}}{\sum_{\{w_i:C(w_{i-n+1}\cdots w_i)\leq k\}} P_{bo}(w_i|w_{i-n+2}\cdots w_{i-1})}$$

Katz backoff

. . . but that's kind of ugly-looking. What it's really saying is that:

- Use the discounted weight if the count of the n-gram in question is acceptably large.
- If not, use the n-minus-1-gram's count, adjusted by a special α factor that adjust the count to include the mass you lost by excluding one word.
- You calculate THAT using all the n-minus-1-grams that involve the word you dropped.

Backoff

The whole problem is, how to make the probabilities sum to 1.

Practical considerations

Of course, we don't ever do this ourselves if we can help it.

- SRILM
- CMU toolkit.
- . . .

And, of course, we should ask ourselves if an n-gram model is really appropriate for our problem.

Not always true. . .