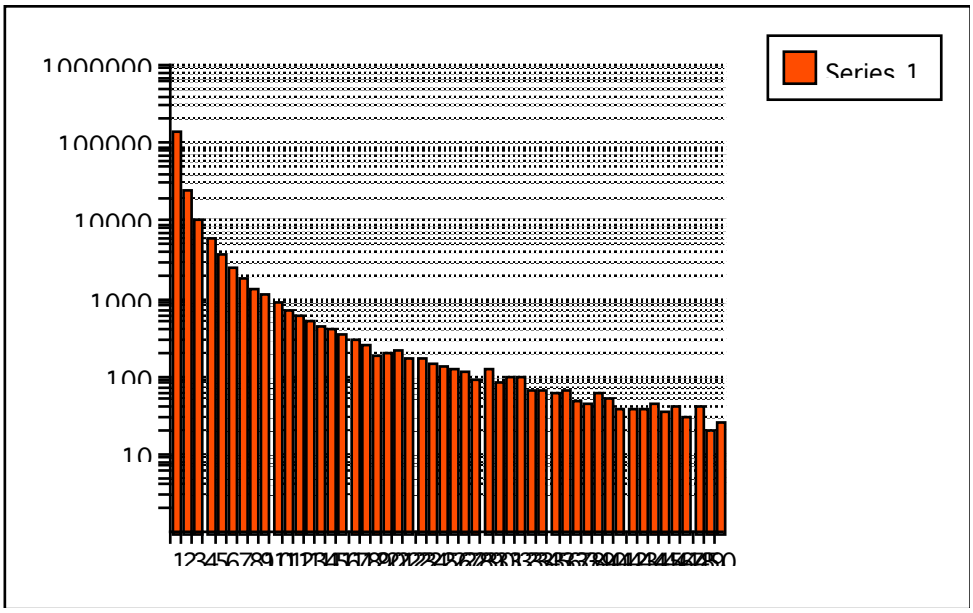
Lecture 5: More on Estimation

1. Estimations based on Smoothing Counts

In the last lecture, we saw the held-out estimation technique, which used a second corpus to estimate how accurate the estimates from the first corpora were and to adjust them accordingly. The first topic today looks at techniques to do this from a more analytic basis that does not require a second corpus, but uses some smoothing function on the size of Class(r) sets (remember, Class(r) = the set of elements that occur r times in the corpus). The main idea underlying this is that we believe that the plot of frequencies of outcomes is some well-defined function. For English word-based models, we'd expect this function to decrease as the number of occurrences rises. For instance, in a bigram model for English, we'd expect to see more bigrams that occur once than those that occur twice, and certainly more than those that occur ten times. This seems to be so, as seen in the chart from data in Manning and Schutze that plots the frequency (on a log scale) of the size of classes of ngrams that occurred once through to 60 times in 600,000 words of text from Jane Austen's novels. With a corpus of this size, we see the distribution is relatively "smooth".

So another way to smooth a probability distribution would be to smoothing the frequency



of frequency data and then convert that back into a probability distribution. Many different smoothing functions used, ranging from simple techniques such as averaging two or three values, to more complex technique of curve fitting a function to the observed data and then using the function values. For example, let's look at a simple one. Let N(r) be the number of outcomes that occur r times in the corpus (i.e., N(r) = |Class(r)|) and N_S (r) be a revised count defined by

N_S (0) = N(0)

N_S (r) = (N(r-1)+ N(r)+ N(r+1))/3, for r >0

Given we have smoothed the distribution, how do we convert those revised counts back into revised probability estimates for each class. One technique for this was developed by Good in 1953 described a technique attributed to Turing. The method is developed assuming that the true distribution of the pdf is a binomial distribution, an assumption not always valid for language processing. But the methods appears to work well in many cases, especially in applications with very large vocabularies and large training sets. Specifically, the **Good-Turing** estimate uses the ratio of the smoothed estimates of two adjacent classes. Specifically, the revised estimates for count
s r, R(r) is defined by:

$$R(r) = (r+1)\, N_S(r+1) / N_S(r)$$

This could be used with any smoothing function, although, to be well defined the smoothed values $N_S(r)$ must always be greater than 0 up to one more than the maximum size r that we need.

To complete the process, the Good-Turing Estimate would be obtained simply by normalizing these revised counts

$$P_{GT}(o) = R(r) / \sum_r N(r)*R(r), \text{ where o occurs r times in training corpus}$$

Note that since we normally expect fewer n-grams to occur r+1 times than those that occur r times (also demonstrated by the plot above), this technique typically discounts the probability of the more frequent elements (but of course boosts the probability of elements that did not occur in the original training corpus).

For instance, consider the Good-Turing estimate for our ABC example (which is really too small for this technique but serves to illustrate the technique). In this example, the corpus size N is 18, we have 1 n-gram that occurs 3 times, 3 that occur twice, 9 that occur once, and 14 that didn't occur at all. Once we derive the revised n-gram counts R(r) for each r, the normalization constant $\sum_r N(r)*R(r)$ can be computed and is 21.44. We can then compare the Good-Turing estimate with the MLE.

r	**N(r)**	**N_S(r)**	**R(r)**	**P_{GT}**	**P_{MLE}**
4	0	.33			
3	1	1.33	1	.047	.166
2	3	4.33	.92	.043	.11
1	9	8.66	1	.047	.055
0	14	14	.62	.029	0

Note in this example, we get dramatic smoothing of the distribution, which in fact is not a bad thing given it is such as small training set and so the estimates will be very uncertain.

2. Interpolation Methods

Another method for dealing with unseen events is to use a combination of probabilistic models. This especially useful when we are trying to estimate conditional probabilities and may construct a series of estimates of distributions using different contexts. For instance, to estimate the probability of a word given the context of the previous two

words, we might use a linear combination of trigram, bigram and unigram models. In particular,

$$P_{li}(w_3 \mid w_1\ w_2) = \lambda_1 P_1(w_3) + \lambda_2 P_2(w_3 \mid w_2) + \lambda_3 P_3(w_3 \mid w_1\ w_2)$$

Where P_1, P_2 and P_3 are the unigram, bigram and trigram estimates respectively. To guarantee that P_{li} is a probability distribution, we must require each λ_i be between 0 and 1, and $\sum_i \lambda_i = 1$. (You proved a version of this in assignment 1)

For example, using the same ABC corpus (repeated here for convenience) and the standard MLE technique.

A B C A B B C C A C B C A A C B C C B C

we can use MLE to estimate the bigram and unigram probabilities. The trigram probabilities were estimated in the last lecture.

Unigram	Count	MLE
A	5	.25
B	6	.3
C	9	.45

Table 1: The Unigram estimates

Bigram: $x_{i-1}\ x_i$	Pair Count	$P_{MLE}(x_i \mid x_{i-1})$
A A	1	.2
A B	2	.4
A C	2	.4
B A	0	0
B B	1	.167
B C	5	.833
C A	3	.375
C B	3	.375
C C	2	.25

Table 2: The "bigram" conditional probability

We could set the weighting factors by hand. Lets say $\lambda_1 = .05$, $\lambda_2 = .1$ and $\lambda_3 = .85$. With these weights, the linear interpolated probability function would assign the following probability to the sequence A A A.

$P_{li}(A \mid A\ A)$
$= .05 * P1(A) + .1 * P2(A \mid A) + .85 * P3(A \mid A\ A)$
$= .05 * .25 + .1 * .2 + .85 * 0$
$= .0125 + .02 = .0325$

By assigning some probability to unseen elements, this method obviously takes some probability from the ones that were seen. For instance, the most common trigram CBC is the only trigram beginning with CB. Thus $P_{MLE}(C \mid CB) = 1.0$. With linear interpolation we get

$P_{li}(C \mid CB)$
$= .85 * P3(C \mid B\ C) + .15 * P2(C \mid B) + .05 * P1(C)$
$= .85 * 1 + .1 * .833 + .05 * .45$
$= .85 + .083 + .0225$
$= .9558$

This estimate is slightly lower but still in the right ballpark because the trigram, bigram and unigram estimates all found the combination likely.

Table 3 looks at how this model works on our development corpus.

Element	$P_{MLE}(z \mid x\ y)$	$P_{MLE}(z \mid y)$	$P_{MLE}(z)$	Linear Combination
$P_{LI}(C \mid CB)$	1	.833	.45	.9558
$P_{LI}(C \mid BC)$	.5	.25	.45	.4725
$P_{LI}(A \mid CC)$	.5	.375	.25	.475
$P_{LI}(A \mid CA)$	1	.2	.25	.8825
$P_{LI}(B \mid AA)$	0	.4	.3	.055

Table 3: Calculating the likelihood of the development corpus

Multiplying these conditional probabilities together, we find that the likelihood of the development corpus with this estimate is .01. We could experiment with different weights and see what combination works best in practice. Later we will consider learning procedures to determine good values for these parameters.

Note we can also generalize this model so that we have lambda's that depend on some property of the current context or history h. This is called **generalized linear interpolation,** and the general formula given a set of probability distributions P_i

$$P_{GLI}(c \mid h) = \sum_i \lambda_i(h) P_i(w \mid h), \text{ such that for all h, } \lambda_i(h) >= 0 \text{ and } \sum_i \lambda_i(h) = 1$$

For instance, with the trigram model we might have a different set of lambda's based on how frequent the first two items in the trigram occur. If they occur frequently, then we would tend to weight the trigram probability more heavily since we have seen this context many times before. If, on the other hand, we have rarely seen the context, if ever, we would downplay the trigram model and emphasize the bigram and unigram models more. The reason why this could be a good strategy is that it captures some aspect of certainty about our estimate of the trigram model. For instance, in the ABC corpus, the bigram BA never occurs, so certainly the trigram model for BAC would also be 0, as would the bigram, and we would be left with a highly discounted unigram probability for the estimate. With the new model, we'd have a series of lambda's, one for each context

count. Instead of l_1, we'd have a function $l_1(c)$, where c is the count of the bigram context. Similarly, we have functional values for l_2 and l_3. To maximize the probability, if c=0, then we'd like $l_3(0)$ to be higher than usual, and $l_1(0)$ to be lower.

As a concrete example, let's assume the following lambda values based on context counts for the ABC corpus:

Count	λ_1	λ_2	λ_3
0	.333	.333	.333
1	.5	.3	.2
2	.7	,2	.1
3	.85	.1	.05

Table: Having the lambda's depend on context counts

Trigram	Context Count	$P_{MLE}(z \mid x\ y)$	$P_{MLE}(z \mid y)$	$P_{MLE}(z)$	$P_{GLI}(z \mid x\ y)$
P(C | B A)	0	0	P(C | A) = .4	P(A) = .25	.4 * .333 + .25 * .333 = .21
P(B | B C)	4	0	P(B | C) = .375	P(B) = .3	.05 (= .1*.833+.o5*.45)

Table: comparing estimates for two unseen trigrams

Note that while neither BAC or BCB occurred in the corpus, the context BC was quite common while BA was never seen. This means that we have reasonable evidence that B is unlikely to follow BC (since out of 4 trials it didn't occur). We have no evidence one way or the other, however, on how likely it is that C follows BA, since BA was never seen. Thus we should tend to be more conservative and smooth our estimates more heavily. The above example shows this as it estimates that C following B A is three time more likely than B following B C.

3. Back-Off Methods

The final approach can be viewed as an instance of generalized linear interpolation and uses similar intuitions. For example, if we are interested in estimating $P(W_i \mid W_{i-2}\ W_{i-1})$, we use the MLE estimate for the trigram if we feel we have a good enough estimate for the trigram. By "good enough" we mean that it occurred more than some number k in the training data. If it is not good enough, then we use the bigram estimate discounted by some factor α. In other words, our initial estimates (before normalization) would be

$$P_{BO}(W_i \mid W_{i-2}\ W_{i-1}) = \text{if } C(W_{i-2}\ W_{i-1}\ W_I) > k \text{ then } (1 - d_3) * P_{MLE}(W_i \mid W_{i-2}\ W_{i-1})$$
$$\text{else } \alpha_2 * P'_{BO}(W_i \mid W_{i-1})$$

We would compute the backoff estimate of the bigram in the same way, backing off to the unigram estimate if necessary. We would then compute the backoff probability distribution by normalizing in the usual way.

Note to use this method for a general n-gram model, we need to set *k*, the minimum number of observations to make us believe we have a good estimate, and a series of normalizing factors $\alpha_{n-1} \ldots \alpha_1$ for each possible backoff from an n-gram to an n-1-gram.

More sophisticated models can be developed that would discount the MLE estimates above and encode the remaining probability mass in the $\alpha_{\square}$'s, thus eliminating the need for renormalization. Back-off models were suggested by Katz, and his particular method including techniques for discounting is often called the **Katz Back-off Model**. There are more details in the text.