Using held out estimation on the test data

So long as the frequency of an n-gram $C(w_1 \cdots w_n)$ is the only thing that we are using to predict its future frequency in text, then we can use held out estimation performed on the test set to provide the correct answer of what the discounted estimates of probabilities should be in order to maximize the probability of the test set data. Doing this empirically measures how often n-grams that were seen r times in the training data actually do occur in the test text. The empirical estimates $f_{\text{empirical}}$ in table 6.4 were found by randomly dividing the 44 million bigrams in the whole AP corpus into equal-sized training and test sets, counting frequencies in the 22 million word training set and then doing held out estimation using the test set. Whereas other estimates are calculated only from the 22 million words of training data, this estimate can be regarded as an empirically determined gold standard, achieved by allowing access to the test data.

6.2.4 Cross-validation (deleted estimation)

The $f_{\text{empirical}}$ estimates discussed immediately above were constructed by looking at what actually happened in the test data. But the idea of held out estimation is that we can achieve the same effect by dividing the training data into two parts. We build initial estimates by doing counts on one part, and then we use the other pool of held out data to refine those estimates. The only cost of this approach is that our initial training data is now less, and so our probability estimates will be less reliable.

Rather than using some of the training data only for frequency counts and some only for smoothing probability estimates, more efficient schemes are possible where each part of the training data is used both as initial training data and as held out data. In general, such methods in statistics go under the name ***cross-validation.***

CROSS-VALIDATION

Jelinek and Mercer (1985) use a form of two-way cross-validation that they call ***deleted estimation.*** Suppose we let N_r^a be the number of n-grams occurring r times in the a^{th} part of the training data, and T_r^{ab} be the total occurrences of those bigrams from part a in the b^{th} part. Now depending on which part is viewed as the basic training data, standard held out estimates would be either:

DELETED ESTIMATION

$$P_{\text{ho}}(w_1 \cdots w_n) = \frac{T_r^{01}}{N_r^0 N} \text{ or } \frac{T_r^{10}}{N_r^1 N} \qquad \text{where } C(w_1 \ldots w_n) = r$$

The more efficient deleted interpolation estimate does counts and smoothing on both halves and then does a weighted average of the two according to the proportion of words in N_r^0 versus N_r^1:

$$P_{\text{del}}(w_1 \cdots w_n) = \frac{T_r^{01} + T_r^{10}}{N(N_r^0 + N_r^1)} \quad \text{where } C(w_1 \cdots w_n) = r \tag{6.11}$$

On large training corpora, doing deleted estimation on the training data works better than doing held-out estimation using just the training data, and indeed table 6.4 shows that it produces results that are quite close to the empirical gold standard.[10] It is nevertheless still some way off for low frequency events. It overestimates the expected frequency of unseen objects, while underestimating the expected frequency of objects that were seen once in the training data. By dividing the text into two parts like this, one estimates the probability of an object by how many times it was seen in a sample of size $\frac{N}{2}$, assuming that the probability of a token seen r times in a sample of size $\frac{N}{2}$ is double that of a token seen r times in a sample of size N. However, it is generally true that as the size of the training corpus increases, the percentage of unseen n-grams that one encounters in held out data, and hence one's probability estimate for unseen n-grams, decreases (while never becoming negligible). It is for this reason that collecting counts on a smaller training corpus has the effect of overestimating the probability of unseen n-grams.

LEAVING-ONE-OUT

There are other ways of doing cross-validation. In particular Ney et al. (1997) explore a method that they call Leaving-One-Out where the primary training corpus is of size N − 1 tokens, while 1 token is used as held out data for a sort of simulated testing. This process is repeated N times so that each piece of data is left out in turn. The advantage of this training regime is that it explores the effect of how the model changes if any particular piece of data had not been observed, and Ney et al. show strong connections between the resulting formulas and the widely-used Good-Turing method to which we turn next.[11]

10. Remember that, although the empirical gold standard was derived by held out estimation, it was held out estimation based on looking at the test data! Chen and Goodman (1998) find in their study that for smaller training corpora, held out estimation outperforms deleted estimation.

11. However, Chen and Goodman (1996: 314) suggest that leaving one word out at a time is problematic, and that using larger deleted chunks in deleted interpolation is to be preferred.

6.2.5 Good-Turing estimation

The Good-Turing estimator

Good (1953) attributes to Turing a method for determining frequency or probability estimates of items, on the assumption that their distribution is binomial. This method is suitable for large numbers of observations of data drawn from a large vocabulary, and works well for n-grams, despite the fact that words and n-grams do not have a binomial distribution. The probability estimate in Good-Turing estimation is of the form $P_{GT} = r^*/N$ where r^* can be thought of as an adjusted frequency. The theorem underlying Good-Turing methods gives that for previously observed items:

$$r^* = (r+1)\frac{E(N_{r+1})}{E(N_r)} \tag{6.12}$$

where E denotes the expectation of a random variable (see (Church and Gale 1991a; Gale and Sampson 1995) for discussion of the derivation of this formula). The total probability mass reserved for unseen objects is then $E(N_1)/N$ (*see* exercise 6.5).

Using our empirical estimates, we can hope to substitute the observed N_r for $E(N_r)$. However, we cannot do this uniformly, since these empirical estimates will be very unreliable for high values of r. In particular, the most frequent n-gram would be estimated to have probability zero, since the number of n-grams with frequency one greater than it is zero! In practice, one of two solutions is employed. One is to use Good-Turing reestimation only for frequencies $r < k$ for some constant k (*e.g.,* 10). Low frequency words are numerous, so substitution of the observed frequency of frequencies for the expectation is quite accurate, while the MLE estimates of high frequency words will also be quite accurate and so one doesn't need to discount them. The other is to fit some function S through the observed values of (Y, N_r) and to use the smoothed values S(r) for the expectation (this leads to a family of possibilities depending on exactly which method of curve fitting is employed - Good (1953) discusses several smoothing methods). The probability mass $\frac{N_1}{N}$ given to unseen items can either be divided among them uniformly, or by some more sophisticated method (see under Combining Estimators, below). So using this method with a uniform estimate for unseen events, we have:

Good-Turing Estimator: If $C(w_1 \ldots w_n) = r > 0$,

$$P_{GT}(w_1 \cdots w_n) = \frac{r^*}{N} \quad \text{where } r^* = \frac{(r+1)S(r+1)}{S(r)} \tag{6.13}$$

If $C(w_1 \ldots w_n) = 0$,

$$P_{GT}(w_1 \cdots w_n) = \frac{1 - \sum_{r=1}^{\infty} N_r \frac{r^*}{N}}{N_0} \approx \frac{N_1}{N_0 N} \tag{6.14}$$

Gale and Sampson (1995) present a simple and effective approach, Simple Good-Turing, which effectively combines these two approaches. As a smoothing curve they simply use a power curve $N_r = ar^b$ (with $b < -1$ to give the appropriate hyperbolic relationship), and estimate A and b by simple linear regression on the logarithmic form of this equation $\log N_r = a + b \log r$ (linear regression is covered in section 15.4.1, or in all introductory statistics books). However, they suggest that such a simple curve is probably only appropriate for high values of r. For low values of r, they use the measured N_r directly. Working up through frequencies, these direct estimates are used until for one of them there isn't a significant difference between r^* values calculated directly or via the smoothing function, and then smoothed estimates are used for all higher frequencies.[12] Simple Good-Turing can give exceedingly good estimators, as can be seen by comparing the Good-Turing column f_{GT} in table 6.4 with the empirical gold standard.

RENORMALIZATION

Under any of these approaches, it is necessary to *renormalize* all the estimates to ensure that a proper probability distribution results. This can be done either by adjusting the amount of probability mass given to unseen items (as in equation (6.14)) or, perhaps better, by keeping the estimate of the probability mass for unseen items as $\frac{N_1}{N}$ and renormalizing all the estimates for previously seen items (as Gale and Sampson (1995) propose).

Frequencies of frequencies in Austen

COUNT-COUNTS

To do Good-Turing, the first step is to calculate the frequencies of different frequencies (also known as count-counts). Table 6.7 shows extracts from the resulting list of frequencies of frequencies for bigrams and trigrams. (The numbers are reminiscent of the Zipfian distributions of

12. An estimate of r^* is deemed significantly different if the difference exceeds 1.65 times the standard deviation of the Good-Turing estimate, which is given by:

$$\sqrt{(r+1)^2 \frac{N_{r+1}}{N_r^2} \left(1 + \frac{N_{r+1}}{N_r}\right)}$$

Bigrams				Trigrams			
r	N_r	r	N_r	r	N_r	r	N_r
1	138741	28	90	1	404211	28	35
2	25413	29	120	2	32514	29	32
3	10531	30	86	3	10056	30	25
4	5997	31	98	4	4780	31	18
5	3565	32	99	5	2491	32	19
6	2486		...	6	1571		..
7	1754	1264	1	7	1088	189	1
8	1342	1366	1	8	749	202	1
9	1106	1917	1	9	582	214	1
10	896	2233	1	10	432	366	1
	...	2507	1		...	378	1

Table 6.7 Extracts from the frequencies of frequencies distribution for bigrams and trigrams in the Austen corpus.

section 1.4.3 but different in the details of construction, and more exaggerated because they count sequences of words.) Table 6.8 then shows the reestimated counts r^* and corresponding probabilities for bigrams.

For the bigrams, the mass reserved for unseen bigrams, N_1/N = 138741/617091 = 0.2248. The space of bigrams is the vocabulary squared, and we saw 199,252 bigrams, so using uniform estimates, the probability estimate for each unseen bigram is: $0.2248/(14585^2 - 199252) = 1.058 \times 10^{-9}$. If we now wish to work out conditional probability estimates for a bigram model by using Good-Turing estimates for bigram probability estimates, and MLE estimates directly for unigrams, then we begin as follows:

$$P(she|person) = \frac{f_{GT}(person\ she)}{C(person)} = \frac{1.228}{223} = 0.0055$$

Continuing in this way gives the results in table 6.9, which can be compared with the bigram estimates in table 6.3. The estimates in general seem quite reasonable. Multiplying these numbers, we come up with a probability estimate for the clause of 1.278×10^{-17}. This is at least much higher than the ELE estimate, but still suffers from assuming a uniform distribution over unseen bigrams.

r	r^*	$P_{GT}(\cdot)$
0	0.0007	1.058×10^{-9}
1	0.3663	5.982 x10-7
2	1.228	2.004×10^{-6}
3	2.122	3.465×10^{-6}
4	3.058	4.993×10^{-6}
5	4.015	6.555×10^{-6}
6	4.984	8.138×10^{-6}
7	5.96	9.733×10^{-6}
8	6.942	1.134×10^{-5}
9	7.928	1.294×10^{-5}
10	8.916	1.456×10^{-5}
...		
28	26.84	4.383 x10-5
29	27.84	4.546×10^{-5}
30	28.84	4.709×10^{-5}
31	29.84	4.872×10^{-5}
32	30.84	5.035×10^{-5}
...		
1264	1263	0.002062
1366	1365	0.002228
1917	1916	0.003128
2233	2232	0.003644
2507	2506	0.004092

Table 6.8 Good-Turing estimates for bigrams: Adjusted frequencies and probabilities. Smoothed using the software on the website.

$P(she\|person)$	0.0055
$P(was\|she)$	0.1217
$P(inferior\|was)$	6.9×10^{-8}
$P(to\|inferior)$	0.1806
$P(both\|to)$	0.0003956
$P(sisters\|both)$	0.003874

Table 6.9 Good-Turing bigram frequency estimates for the clause from *Persuasion.*

6.2.6 Briefly noted

Ney and Essen (1993) and Ney et al. (1994) propose two discounting models: in the absolute discounting model, all non-zero MLE frequencies are discounted by a small constant amount δ and the frequency so gained is uniformly distributed over unseen events:

Absolute discounting: If $C(w_1 \ldots w_n) = r$,

$$P_{\text{abs}}(w_1 \cdots w_n) = \begin{cases} (r-\delta)/N & \text{if } r > 0 \\ \frac{(B-N_0)\delta}{N_0 N} & \text{otherwise} \end{cases} \tag{6.15}$$

(Recall that B is the number of bins.) In the linear discounting method, the non-zero MLE frequencies are scaled by a constant slightly less than one, and the remaining probability mass is again distributed across novel events:

Linear discounting: If $C(w_1 \ldots w_n) = r$,

$$P(w_1 \cdots w_n) = \begin{cases} (1-\alpha)r/N & \text{if } r > 0 \\ \alpha/N_0 & \text{otherwise} \end{cases} \tag{6.16}$$

These estimates are equivalent to the frequent engineering move of making the probability of unseen events some small number ϵ instead of zero and then rescaling the other probabilities so that they still sum to one – the choice between them depending on whether the other probabilities are scaled by subtracting or multiplying by a constant. Looking again at the figures in table 6.4 indicates that absolute discounting seems like it could provide a good estimate. Examining the $f_{\text{empirical}}$ figures there, it seems that a discount of $\delta \approx 0.77$ would work well except for bigrams that have only been seen once previously (which would be underestimated). In general, we could use held out data to estimate a good value for 6. Extensions of the absolute discounting approach are very successful, as we discuss below. It is hard to justify linear discounting. In general, the higher the frequency of an item in the training text, the more accurate an unadjusted MLE estimate is, but the linear discounting method does not even approximate this observation.

A shortcoming of Lidstone's law is that it depends on the number of bins in the model. While some empty bins result from sparse data problems, many more may be principled gaps. Good-Turing estimation is one

method where the estimates of previously seen items do not depend on the number of bins. Ristad (1995) explores the hypothesis that natural sequences use only a subset of the possible bins. He derives various forms for a *Natural Law of Succession*, including the following probability estimate for an n-gram with observed frequency $C(w_1 \cdots w_n) = r$:

NATURAL LAW OF SUCCESSION

$$P_{\text{NLS}}(w_1 \cdots w_n) = \begin{cases} \frac{r+1}{N+B} & \text{if } N_0 = 0 \\ \frac{(r+1)(N+1+N_0-B)}{N^2+N+2(B-N_0)} & \text{if } N_0 > 0 \text{ and } r > 0 \\ \frac{(B-N_0)(B-N_0+1)}{N_0(N^2+N+2(B-N_0))} & \text{otherwise} \end{cases} \tag{6.17}$$

The central features of this law are: (i) it reduces to Laplace's law if something has been seen in every bin, (ii) the amount of probability mass assigned to unseen events decreases quadratically in the number N of trials, and (iii) the total probability mass assigned to unseen events is independent of the number of bins B, so there is no penalty for large vocabularies.

6.3 Combining Estimators

So far the methods we have considered have all made use of nothing but the raw frequency r of an n-gram and have tried to produce the best estimate of its probability in future text from that. But rather than giving the same estimate for all n-grams that never appeared or appeared only rarely, we could hope to produce better estimates by looking at the frequency of the (n − l)-grams found in the n-gram. If these $(n-1)$-grams are themselves rare, then we give a low estimate to the n-gram. If the (n − 1)-grams are of moderate frequency, then we give a higher probability estimate for the n-gram.[13] Church and Gale (1991a) present a detailed study of this idea, showing how probability estimates for unseen bigrams can be estimated in terms of the probabilities of the unigrams that compose them. For unseen bigrams, they calculate the joint-if-independent probability $P(w_1)P(w_2)$, and then group the bigrams into bins based on this quantity. Good-Turing estimation is then performed on each bin to give corrected counts that are normalized to yield probabilities.

13. **But** if the $(n-1)$-grams are of very high frequency, then we may actually want to lower the estimate again, because the non-appearance of the n-gram is then presumably indicative of a principled gap.

But in this section we consider the more general problem of how to combine multiple probability estimates from various different models. If we have several models of how the history predicts what comes next, then we might wish to combine them in the hope of producing an even better model. The idea behind wanting to do this may either be smoothing, or simply combining different information sources.

For n-gram models, suitably combining various models of different orders is in general the secret to success. Simply combining MLE n-gram estimates of various orders (with some allowance for unseen words) using the simple linear interpolation technique presented below results in a quite good language model (Chen and Goodman 1996). One can do better, but not by simply using the methods presented above. Rather one needs to combine the methods presented above with the methods for combining estimators presented below.

6.3.1 Simple linear interpolation

One way of solving the sparseness in a trigram model is to mix that model with bigram and unigram models that suffer less from data sparseness. In any case where there are multiple probability estimates, we can make a linear combination of them, providing only that we weight the contribution of each so that the result is another probability function. Inside Statistical NLP, this is usually called linear interpolation, but elsewhere the name (finite) mixture models is more common. When the functions being interpolated all use a subset of the conditioning information of the most discriminating function (as in the combination of trigram, bigram and unigram models), this method is often referred to as ***deleted interpolation.*** For interpolating n-gram language models, such as deleted interpolation from a trigram model, the most basic way to do this is:

LINEAR INTERPOLATION
MIXTURE MODELS

DELETED INTERPOLATION

$$P_{\text{li}}(w_n|w_{n-2},w_{n-1}) = \lambda_1 P_1(w_n) + \lambda_2 P_2(w_n|w_{n-1}) + \lambda_3 P_3(w_n|w_{n-1},w_{n-2}) \quad (6.18)$$

where $0 \leq \lambda_i \leq 1$ and $\sum_i \lambda_i = 1$.

While the weights may be set by hand, in general one wants to find the combination of weights that works best. This can be done automatically by a simple application of the Expectation Maximization (EM) algorithm, as is discussed in section 9.2.1, or by other numerical algorithms. For instance, Chen and Goodman (1996) use Powell's algorithm, as presented in (Press et al. 1988). Chen and Goodman (1996) show that this simple

model (with just slight complications to deal with previously unseen histories and to reserve some probability mass for out of vocabulary items) works quite well. They use it as the baseline model (see section 7.1.3) in their experiments.

6.32 Katz's backing-off

BACK-OFF MODELS

In ***back-off models,*** different models are consulted in order depending on their specificity. The most detailed model that is deemed to provide sufficiently reliable information about the current context is used. Again, back-off may be used to smooth or to combine information sources.

Back-off n-gram models were proposed by Katz (1987). The estimate for an n-gram is allowed to back off through progressively shorter histories:

$$P_{\text{bo}}(w_i|w_{i-n+1}\ldots w_{i-1}) = \begin{cases} (1-d_{w_{i-n+1}\cdots w_{i-1}})\frac{C(w_{i-n+1}\cdots w_i)}{C(w_{i-n+1}\cdots w_{i-1})} \\ \qquad \text{if } C(w_{i-n+1}\cdots w_i) > k \\ \alpha_{w_{i-n+1}\cdots w_{i-1}} P_{\text{bo}}(w_i|w_{i-n+2}\cdots w_{i-1}) \\ \qquad \text{otherwise} \end{cases} \tag{6.19}$$

If the n-gram of concern has appeared more than k times (k is normally set to 0 or 1), then an n-gram estimate is used, as in the first line. But the MLE estimate is discounted a certain amount (represented by the function d) so that some probability mass is reserved for unseen n-grams whose probability will be estimated by backing off. The MLE estimates need to be discounted in some manner, or else there would be no probability mass to distribute to the lower order models. One possibility for calculating the discount is the Good-Turing estimates discussed above, and this is what Katz actually used. If the n-gram did not appear or appeared k times or less in the training data, then we will use an estimate from a shorter n-gram. However, this back-off probability has to be multiplied by a normalizing factor α so that only the probability mass left over in the discounting process is distributed among n-grams that are estimated by backing off. Note that in the particular case where the $(n-1)$-gram in the immediately preceding history was unseen, the first line is inapplicable for any choice of w_i, and the back-off factor α takes on the value 1. If the second line is chosen, estimation is done recursively via an $(n-1)$-gram estimate. This recursion can continue down, so that one can start

with a four-gram model and end up estimating the next word based on unigram frequencies.

While backing off in the absence of much data is generally reasonable, it can actually work badly in some circumstances. If we have seen the bigram $w_i w_j$ many times, and w_k is a common word, but we have never seen the trigram $w_i w_j w_k$, then at some point we should actually conclude that this is significant, and perhaps represents a 'grammatical zero,' rather than routinely backing off and estimating $P(w_k|h)$ via the bigram estimate $P(w_k|w_j)$. Rosenfeld and Huang (1992) suggest a more complex back-off model that attempts to correct for this.

Back-off models are sometimes criticized because their probability estimates can change suddenly on adding more data when the back-off algorithm selects a different order of n-gram model on which to base the estimate. Nevertheless, they are simple and in practice work well.

6.3.3 General linear interpolation

In simple linear interpolation, the weights were just a single number, but one can define a more general and powerful model where the weights are a function of the history. For k probability functions P_k the general form for a linear interpolation model is:

$$P_{\text{li}}(w|h) = \sum_{i=1}^{k} \lambda_i(h) P_i(w|h) \tag{6.20}$$

where $\forall h, 0 \leq \lambda_i(h) \leq 1$ and $\sum_i \lambda_i(h) = 1$.

Linear interpolation is commonly used because it is a very general way to combine models. Randomly adding in dubious models to a linear interpolation need not do harm providing one finds a good weighting of the models using the EM algorithm. But linear interpolation can make bad use of component models, especially if there is not a careful partitioning of the histories with different weights used for different sorts of histories. For instance, if the λ_i are just constants in an interpolation of n-gram models, the unigram estimate is always combined in with the same weight regardless of whether the trigram estimate is very good (because there is a lot of data) or very poor.

In general the weights are not set according to individual histories. Training a distinct $\lambda_{w_{(i-n+1)(i-1)}}$ for each $w_{(i-n+1)(i-1)}$ is not in general felicitous, because it would worsen the sparse data problem. Rather one

wants to use some sort of equivalence classing of the histories. Bahl et al. (1983) suggest partitioning the λ into bins according to $C(w_{(i-n+1)(i-1)})$, and tying the parameters for all histories with the same frequency.

Chen and Goodman (1996) show that rather than this method of putting the A parameters into bins, a better way is to group them according to the average number of counts per non-zero element:

$$\frac{C(w_{(i-n+1)(i-1)})}{|w_i : C(w_{(i-n+1)i}) > 0|} \tag{6.21}$$

That is, we take the average count over non-zero counts for n-grams $w_{i-n+1} \cdots w_{i-1} w^x$. We presume that the reason this works is that, because of the syntax of language, there are strong structural constraints on which words are possible or normal after certain other words. While it is central to most Statistical NLP language models that any word is allowed after any other – and this lets us deal with all possible disfluencies – nevertheless in many situations there are strong constraints on what can normally be expected due to the constraints of grammar. While some n-grams have just not been seen, others are 'grammatical zeroes,' to coin a phrase, because they do not fit with the grammatical rules of the language. For instance, in our Austen training corpus, both of the bigrams great deal and of *that* occur 178 times. But of that is followed in the corpus by 115 different words, giving an average count of 1.55, reflecting the fact that any adverb, adjective, or noun can felicitously follow within a noun phrase, and any capitalized word starting a new sentence is also a possibility. There are thus fairly few grammatical zeroes (mainly just verbs and prepositions). On the other hand, great deal is followed by only 36 words giving an average count of 4.94. While a new sentence start is again a possibility, grammatical possibilities are otherwise pretty much limited to conjunctions, prepositions, and the comparative form of adjectives. In particular, the preposition of follows 38% of the time. The higher average count reflects the far greater number of grammatical zeroes following this bigram, and so it is correct to give new unseen words a much lower estimate of occurrence in this context.

Finally, note that back-off models are actually a special case of the general linear interpolation model. In back-off models, the functions $\lambda_i(h)$ are chosen so that their value is 0 for a history h except for the coefficient of the model that would have been chosen using a back-off model, which has the value 1.

6.3.4 Briefly noted

Bell et al. (1990) and Witten and Bell (1991) introduce a number of smoothing algorithms for the goal of improving text compression. Their "Method C" is normally referred to as *Witten-Bell smoothing* and has been used for smoothing speech language models. The idea is to model the probability of a previously unseen event by estimating the probability of seeing such a new (previously unseen) event at each point as one proceeds through the training corpus. In particular, this probability is worked out relative to a certain history. So to calculate the probability of seeing a new word after, say, sat *in* one is calculating from the training data how often one saw a new word after sat *in,* which is just the count of the number of trigram types seen which begin with sat *in.* It is thus an instance of generalized linear interpolation:

WITTEN-BELL SMOOTHING

$$P_{\text{WB}}(w_i|w_{(i-n+1)(i-1)}) = \lambda_{w_{(i-n+1)(i-1)}} P_{\text{MLE}}(w_i|w_{(i-n+1)(i-1)}) + (1-\lambda_{w_{(i-n+1)(i-1)}}) P_{\text{WB}}(w_i|w_{(i-n+2)(i-1)}) \tag{6.22}$$

where the probability mass given to new n-grams is given by:

$$(1-\lambda_{w_{(i-n+1)(i-1)}}) = \frac{|\{w_i : C(w_{i-n+1}\cdots w_i) > 0\}|}{|\{w_i : C(w_{i-n+1}\cdots w_i) > 0\}| + \sum_{w_i} C(w_{i-n+1}\cdots w_i)} \tag{6.23}$$

However, Chen and Goodman's (1998) results suggest that this method is not as good a smoothing technique for language models as others that we discuss in this section (performing particularly poorly when used on small training sets).

LINEAR SUCCESSIVE ABSTRACTION

Samuelsson (1996) develops *Linear Successive Abstraction,* a method of determining the parameters of deleted interpolation style models without the need for their empirical determination on held out data. Samuelsson's results suggest similar performance within a part-of-speech tagger to that resulting from conventional deleted interpolation; we are unaware of any evaluation of this technique on word n-gram models.

Another simple but quite successful smoothing method examined by Chen and Goodman (1996) is the following. MacKay and Peto (1990) argue for a smoothed distribution of the form:

$$P_{\text{MP}}(w_i|w_{i-n+1}\cdots w_{i-1}) = \frac{C(w_{i-n+1}\cdots w_i) + \alpha P_{\text{MP}}(w_i|w_{i-n+2}\cdots w_{i-1})}{C(w_{i-n+1}\cdots w_{i-1}) + \alpha} \tag{6.24}$$

where α represents the number of counts added, in the spirit of Lidstone's law, but distributed according to the lower order distribution.

Model	Cross-entropy	Perplexity
Bigram	7.98 bits	252.3
Trigram	7.90 bits	239.1
Fourgram	7.95 bits	247.0

Table 6.10 Back-off language models with Good-Turing estimation tested on *Persuasion.*

Chen and Goodman (1996) suggest that the number of added counts should be proportional to the number of words seen exactly once, and suggest taking:

(6.25) $$\alpha = \gamma(N_1(w_{i-n+1} \cdots w_{i-1}) + \beta)$$

where $N_1(w_{i-n+1} \ldots w_{i-1}) = |\{w_i : C(w_{i-n+1} \ldots w_i) = 1\}|$, and then optimizing β and γ on held out data.

Kneser and Ney (1995) develop a back-off model based on an extension of absolute discounting which provides a new more accurate way of estimating the distribution to which one backs off. Chen and Goodman (1998) find that both this method and an extension of it that they propose provide excellent smoothing performance.

6.3.5 Language models for Austen

With the introduction of interpolation and back-off, we are at last at the point where we can build first-rate language models for our Austen corpus. Using the CMU-Cambridge Statistical Language Modeling Toolkit (see the website) we built back-off language models using Good-Turing estimates, following basically the approach of Katz (1987).[14] We then calculated the cross-entropy (and perplexity) of these language models on our test set, ***Persuasion.*** The results appear in table 6.10. The estimated probabilities for each following word, and the n-gram size used to estimate it for our sample clause is then shown in table 6.11. Our probability estimates are at last pleasingly higher than the unigram estimate with which we began!

While overall the trigram model outperforms the bigram model on the test data, note that on our example clause, the bigram model actually as-

14. The version of Good-Turing smoothing that the package implements only discounts low frequencies – words that occurred fewer than 7 times.

	$P(she\|h)$	$P(was\|h)$	$P(inferior\|h)$	$P(to\|h)$	$P(both\|h)$	$P(sisters\|h)$	Product
Unigram	0.011	0.015	0.00005	0.032	0.0005	0.0003	3.96×10^{-17}
Bigram	0.00529	0.1219	0.0000159	0.183	0.000449	0.00372	3.14×10^{-15}
n used	2	2	1	2	2	2	
Trigram	0.00529	0.0741	0.0000162	0.183	0.000384	0.00323	1.44×10^{-15}
n used	2	3	1	2	2	2	

Table 6.11 Probability estimates of the test clause according to various language models. The unigram estimate is our previous MLE unigram estimate. The other two estimates are back-off language models. The last column gives the overall probability estimate given to the clause by the model.

signs a higher probability. Overall, the fourgram model performs slightly worse than the trigram model. This is expected given the small amount of training data. Back-off models are in general not perfectly successful at simply ignoring inappropriately long contexts, and the models tend to deteriorate if too large n-grams are chosen for model building relative to the amount of data available.

6.4 Conclusions

A number of smoothing methods are available which often offer similar and good performance figures. Using Good-Turing estimation and linear interpolation or back-off to circumvent the problems of sparse data represent good current practice. Chen and Goodman (1996, 1998) present extensive evaluations of different smoothing algorithms. The conclusions of (Chen and Goodman 1998) are that a variant of Kneser-Ney back-off smoothing that they develop normally gives the best performance. It is outperformed by the Good-Turing smoothing method explored by Church and Gale (1991a) when training bigram models on more than 2 million words of text, and one might hypothesize that the same would be true of trigram models trained on a couple of orders of magnitude more text. But in all other circumstances, it seems to perform as well or better than other methods. While simple smoothing methods may be appropriate for exploratory studies, they are best avoided if one is hoping to produce systems with optimal performance. Active research continues on better ways of combining probability models and dealing with sparse data.

6.5 Further Reading

Important research studies on statistical estimation in the context of language modeling include (Katz 1987), (Jelinek 1990), (Church and Gale 1991a), (Ney and Essen 1993), and (Ristad 1995). Other discussions of estimation techniques can be found in (Jelinek 1997) and (Ney et al. 1997). Gale and Church (1994) provide detailed coverage of the problems with "adding one." An approachable account of Good-Turing estimation can be found in (Gale and Sampson 1995). The extensive empirical comparison of various smoothing methods in (Chen and Goodman 1996, 1998) are particularly recommended.

The notion of maximum likelihood across the values of a parameter was first defined in (Fisher 1922). See (Ney et al. 1997) for a proof that the relative frequency really is the maximum likelihood estimate.

Recently, there has been increasing use of maximum entropy methods for combining models. We defer coverage of maximum entropy models until chapter 16. See Lau et al. (1993) and Rosenfeld (1994, 1996) for applications to language models.

The early work cited in section 6.2.2 appears in: (Lidstone 1920), (Johnson 1932), and (Jeffreys 1948). See (Ristad 1995) for discussion. Good (1979: 395-396) covers Turing's initial development of the idea of Good-Turing smoothing. This article is reprinted with amplification in (Britton 1992).

6.6 Exercises

Exercise 6.1 [★★]

Explore figures for the percentage of unseen n-grams in test data (that differs from the training data). Explore varying some or all of: (i) the order of the model (i.e., n), (ii) the size of the training data, (iii) the genre of the training data, and (iv) how similar in genre, domain, and year the test data is to the training data.

Exercise 6.2 [★]

As a smaller example of the problems with Laplace's law, work out probability estimates using Laplace's law given that 100 samples have been seen from a potential vocabulary of 1000 items, and in that sample 9 items were seen 10 times, 2 items were seen 5 times and the remaining 989 items were unseen.

Exercise 6.3 [⋆]

Show that using ELE yields a probability function, in particular that

$$\sum_{w_1 \cdots w_n} P_{\text{ELE}}(w_1 \cdots w_n) = 1$$

Exercise 6.4 [⋆]

Using the word and bigram frequencies within the Austen test corpus given below, confirm the ELE estimate for the test clause ***she was inferior to both sisters*** given in section 6.2.2 (using the fact that the word before ***she*** in the corpus was ***person).***

w	$C(w)$	$w_1 w_2$	$C(w_1 w_2)$
person	223	person she	2
she	6,917	she was	843
was	9,409	was inferior	0
inferior	33	inferior to	7
to	20,042	to both	9
both	317	both sisters	2

Exercise 6.5 [⋆]

Show that Good-Turing estimation is well-founded. I.e., you want to show:

$$\sum_{w_1 \cdots w_n} P_{\text{GT}}(w_1 \cdots w_n) = \frac{f_{\text{GT}}(w_1 \cdots w_n)}{N} = 1$$

Exercise 6.6 [⋆]

We calculated a Good-Turing probability estimate for ***she was inferior to both sisters*** using a bigram model with a uniform estimate of unseen bigrams. Make sure you can recreate these results, and then try doing the same thing using a trigram model. How well does it work?

Exercise 6.7 [⋆⋆]

Build language models for a corpus using the software pointed to on the website (or perhaps build your own). Experiment with what options give the best language model, as measured by cross-entropy.

Exercise 6.8 [⋆⋆]

Get two corpora drawn from different domains, and divide each into a training and a test set. Build language models based on the training data for each domain. Then calculate the cross-entropy figures for the test sets using both the language model trained on that domain, and the other language model. How much do the cross-entropy estimates differ?