In Gerhard Lakemeyer and Bernhard Nebel, editors,
Exploring Artificial Intelligence in the New Millenium, Morgan Kaufmann, 2002.

Learning Theory and Language Modeling

David McAllester **Robert E. Schapire**
AT&T Labs – Research
Shannon Laboratory
180 Park Avenue
Florham Park, NJ 07932
{dmac, schapire}@research.att.com

Abstract

We consider some of our recent work on Good-Turing estimators in the larger context of learning theory and language modeling. The Good-Turing estimators have played a significant role in natural language modeling for the past twenty years. We have recently shown that these particular leave-one-out estimators converge rapidly. We present these results and consider possible consequences for language modeling in general. In particular, other leave-one-out estimators, such as for the cross entropy of various forms of language models, might also be shown to be rapidly converging using proof methods similar to those used for the Good-Turing estimators. This could have broad ramification in the analysis and development of language modeling methods. We suggest that, in language modeling at least, leave-one-out estimation may be more significant than Occam's razor.

1 INTRODUCTION

How people manage to acquire language in the first few years of life is one of the great mysteries of human cognition. Computers cannot, at present, duplicate this ability. There has been considerable recent work in learning theory and one would certainly expect a mathematical theory of learning to be relevant in the study of language learning. In this paper we present some of our recent theoretical work motivated by the desire to better understand language learning. We take a statistical view of language and our results are fundamentally statistical in nature. Ultimately we expect that a proper understanding of language learning will encompass syntax and semantics. However, it seems possible that language is statistical, at least to some extent, at all levels — sentences can be grammatically ambiguous with some interpretations being more likely than others and statements can have uncertain truth values with some more likely to be true than others. It is hoped that the statistical results developed here, although not explicitly about syntax and semantics, will continue to prove their worth as our understanding of language learning evolves.

We are interested in what computational linguists call "language models," attempts to capture regularities in language by statistically modeling the probabilistic distribution of words, phrases and sentences as they occur in actual use. The first section of this paper consists of a formal definition of the notion of a language model in general and motivates a widely used formal measure of the "amount of regularity" uncovered by a given model.

The second section describes what is seemingly a very weak class of models — n-gram models. These are essentially simple Markov models of the language that do not capture any notions of grammar, meaning, etc. In spite of the intuitive weakness of these models, they have proved very effective in supporting speech recognition — more effective than models that intuitively seem more sophisticated.

The third section considers n-gram models from the point of view of learning theory. Although we believe that ultimately n-gram models will be replaced in most applications by more sophisticated forms of language models, the fundamental learning theory issues that arise in n-gram models seem likely to arise in more sophisticated models as well. A fundamental issue is the relation between n-gram models and the notion of Occam's razor as a foundation for learning theory. Informally, Occam's razor (Blumer, Ehrenfeucht, Haussler, and Warmuth 1987) states that, for learning to occur, i.e., for our model to give accurate predictions on data not seen during training, the model must be substantially "simpler" or "more compact" than the data itself. However, in general, n-gram models are not small — the model essentially memorizes the training data. So most standard theorems of learning theory that are based on Occam's razor become irrelevant; they do not provide meaningful performance guarantees for n-gram models even in the limit of infinite training data.

The fourth section introduces the Good-Turing leave-one-out estimators and discusses their relation to n-gram language models. The fundamental Good-Turing estimator is simply an estimate of the probability of seeing a word that has not been seen during training (or a word that has not been seen in a particular context). In other words, given a sample of English, Good-Turing estimators tell us how to estimate the probability of seeing a new English word in a new sample. Good-Turing estimators are relevant to n-gram models because they are used in setting certain key parameters called interpolation coefficients in the n-gram models. Empirically, setting these parameters according to Good-Turing performs better in practice than other methods such as those inspired by so-called Bayesian methods. In this section we present very recent theorems on the accuracy of the Good-Turing estimators.

A final section discusses leave-one-out error estimators in general. The leave-one-out error is computed by evaluating the expected error on a single randomly chosen training example based on an estimate obtained using the remaining training examples. It seems likely to us that the proof methods developed for the Good-Turing estimators can be used to prove rapid convergence of other leave-one-out estimators for n-gram language models. This suggests a learning procedure in which one chooses the model whose generalization error, as measured by its leave-one-out estimate, is smallest. This approach seems fundamentally different from the more standard approach of minimizing training error over a large family of models.

2 LANGUAGE MODELS IN GENERAL

We would like a computer to read a large corpus of text, perhaps several years of issues of *The New York Times*, and find regularities in the text. For example, one might hope to discover that sentences tend to contain a noun phrase followed by a verb phrase. In this section we describe a widely used mathematical notion of what it means to "find regularities".

Language can have various kinds of regularities. For example, one might find that, in *The New York Times* at least, grammatical sentences are much more common than ungrammatical ones. One might find that, among the grammatical sentences with objective truth values, true sentences are more common than false sentences — *The New York Times* is, to some extent, trustworthy. No unsupervised language learning computer can currently find these syntactic and semantic regularities, although presumably these regularities do exist in the training corpus. Computers can currently find more obvious regularities such as the statement that the word *of* tends to be followed by the word *the* or that the word *habit* is much more likely if the preceding word is *bad*. Hopefully computers will some day be able to find deeper regularities either by using better learning methods or by seeding the search with sufficient initial regularities.

In spite of the current weakness of computers in finding regularities, it is possible to define a quantitative measure of the amount of regularity that has been found. We can then at least measure the progress of our learning systems. A common measure of the amount of regularity is based on data compression — any real regularity can, in principle, be exploited in compressing English text. For instance, simply knowing the frequency of each word allows us, using standard methods from information theory (Cover and Thomas 1991) to compress the text to a fraction of its original size (where that fraction turns out to be the entropy of the distribution of words). At the opposite extreme of sophistication, if objectively true statements are much more common in *The New York Times* than objectively false statements then one could, in principle, exploit this fact for compression — if we represent each sentence by a bit string then we can use smaller bit strings for the true sentences than for the false sentences. Of course there is at present no computationally tractable method of modeling truth in a computer. Nonetheless, any regularity can in principle be exploited in compression.

Given a compression scheme one can define the quantity of regularity implicit in that scheme to be the amount of compression achieved by the scheme. Consider a coding scheme c that compresses a sentence s into a bit string code word $c(s)$. For a given probability distribution P on sentences, the average number of bits per sentence using coding scheme c, denoted $H(P||c)$, can be defined as follows where $|c(s)|$ denotes the number of bits in the code word for s:

$$H(P||c) = \sum_s P(s)|c(s)|. \tag{1}$$

Our objective is, essentially, to find a compression scheme c with a small quantity $H(P||c)$ of bits per sentence. (Actually, since sentence length varies from author to author, a more stable measure of regularity is $H(P||c)/\overline{n}$ where $\overline{n}$ is the average number of words per sentence. This is the number of bits per word in the compressed text.)

It turns out that any coding scheme for sentences corresponds to a probability distribution on sentences. We are interested in coding schemes that allow us to transmit a sequence of sentences as a sequence of code words. It is important to know where one code word ends and the next begins. This can be done if we assume that no code word is a proper prefix of any other — such a code is called prefix-free. For a prefix-free code we can interpret $2^{-|c(s)|}$ as a probability of the sentence s — it is the probability that if we generate an infinite random bit string and then decode one sentence from the front of this string we get sentence s.

Conversely, information theory gives a way of converting any probability distribution over sentences into a coding scheme. Let $\hat{P}$ be a model (such as a probabilistic grammar or an n-gram model) that defines a probability distribution over sentences where $\hat{P}(s)$ is the probability of sentence s under the model $\hat{P}$. If we use block codes — code words for large blocks of sentences rather than code words for individual sentences — then in the limit of large block size the average number of bits used to code sentence s in the code defined by $\hat{P}$ is exactly $\log_2(1/\hat{P}(s))$. So if the true probability of a sentence s is $P(s)$, then the average number of bits used to transmit a sentence under the code defined by $\hat{P}$ is given by

$$H(P||\hat{P}) = \sum_s P(s) \log \frac{1}{\hat{P}(s)}. \tag{2}$$

Now suppose that $\hat{P}(s)$ is actually defined by a coding scheme c, i.e., we have that $\hat{P}(s)$ is defined to be $2^{-|c(s)|}$. In that case we have that $\log(1/\hat{P}(s))$ equals $|c(s)|$ and so equations (1) and (2) agree. In general, coding schemes correspond to probability distributions and probability distributions correspond to (block) coding schemes. The quantity $H(P||\hat{P})$ is sometimes called the cross-entropy of P with respect to model (or coding scheme) $\hat{P}$.

If P is the true probability distribution over words then one can show that P is its own best model, i.e., the compression scheme implicit in the distribution P achieves the greatest possible compression. More formally, we have the following for any model $\hat{P}$:

$$H(P||P) \leq H(P||\hat{P}).$$

The quantity $H(P||P)$ is usually written as $H(P)$ and is the entropy of the distribution P. A widely used quantity is the Kullback-Leibler divergence, written $D(P||\hat{P})$, which is defined as follows:

$$D(P||\hat{P}) \equiv H(P||\hat{P}) - H(P||P) = \sum_s P(s) \log \frac{P(s)}{\hat{P}(s)}.$$

We can then write $H(P||\hat{P})$ as

$$H(P||\hat{P}) = H(P) + D(P||\hat{P}).$$

In practice, however, the true entropy $H(P)$ of English is unknown and the only measurable quantity is $H(P||\hat{P})$ for particular models $\hat{P}$. Each model then provides an upper bound on the true entropy of English.

We are interested in finding a probability model (coding scheme) that minimizes the average compressed length of sentences. Again, the average number of bits per word, $H(P||\hat{P})/\overline{n}$, tends to be a more stable measure of the amount of identified regularity (it is not sensitive to variations in the average sentence length). Most authors state the performance of language models by giving the perplexity which is defined to be $2^{H(P||\hat{P})/\overline{n}}$. Here, however, we will use the cross entropy per word $H(P||\hat{P})/\overline{n}$ rather than perplexity.

It has been shown that n-gram models of business news text achieve a cross-entropy of about 6.5 compressed bits per word as calculated by (2) (Chen and Goodman 1998). Models based on longer distance syntactic regularities currently only reduce this by a small fraction of a bit per word (Chelba and Jelinek 1998). But it seems plausible that significantly greater reductions are possible.

3 n-GRAM MODELS

Among the simplest and most widely used language models are the n-gram models. Essentially, these models attempt to estimate the distribution of n-grams, i.e., tuples of length n. This is roughly the same as estimating the distribution of words that will follow a sequence of $n-1$ words. For instance, such a model might capture the fact that the word following the phrase "black and" is likely to be "white." Although n-gram models do not capture long distance syntactic or semantic regularities, they have proved very useful in speech recognition systems.

One of the remarkable characteristics of n-gram models is that they include parameters estimating conditional probabilities of the form $\hat{P}(\Phi|\Psi)$ where the conjunction $\Phi \wedge \Psi$ has only occurred a single time in the training data. For instance, in the example above, $\Phi \wedge \Psi$ is the event that the entire phrase "black and white" occurs. In a large corpus, such a phrase may occur repeatedly, but there are bound to be many others that occur only once.

We will call a model parameter derived from a single training sample a *one-count* parameter. By the nature of language, the number of one-count parameters in an n-gram language model is likely to be close to the number of samples in the training data — the n-gram model essentially memorizes the training data. It is well known that the one-count parameters of an n-gram model significantly improve the model — the one-count parameters significantly reduce cross entropy of the model. More sophisticated language models, such as stochastic grammars used in open-domain parsing (Collins 1997; Charniak 2000), also involve a number of one-count parameters essentially equal to the size of the training data. The fundamental theoretical challenge is to explain why such "one-count models" do not overfit as is typical of overly complex models.

Bayesian explanations for the performance of one-count models can be given with an appropriate choice of the Bayesian model prior (Pereira and Singer 1999). The large one-count model is viewed as a posterior mixture of much smaller models. However, the validity of the Bayesian assumptions are questionable. Furthermore, the performance of Bayesian-inspired smoothing is inferior to the performance of Good-Turing inspired smoothing (smoothing is described below). Here we are interested in non-Bayesian explanations that account for the performance of Good-Turing inspired smoothing. The n-gram models provide a simple theoretical setting to explore this issue.

To simplify the discussion (and to allow provable bounds) we assume a finite fixed vocabulary of words. In speech recognition applications one might simply restrict the vocabulary to the finite set of words known to the speech recognition system — other words must be spelled out by the speaker.

An n-gram over vocabulary V is a tuple of n words. Intuitively, one can consider a probability distribution over n-grams defined by sampling a sequence of n words from a random position in a randomly selected sample of English text and replacing words not in V by an "unknown" token. Let S be a sample of m n-grams $\langle w_1^1 \cdots w_n^1 \rangle$, $\langle w_1^2 \cdots w_n^2 \rangle$, ..., $\langle w_1^m \cdots w_n^m \rangle$. In practice these n-grams would be adjacent so that $w_j^i = w_{j-1}^{i+1}$. However, considering independently sampled n-grams simplifies the theoretical analysis. For a given distribution on n-grams we can think of w_1, ..., w_n as dependent random variables. We are interested in estimating $P(w_n|w_1,\ \ldots,\ w_{n-1})$. For a given sample S of m n-grams and $1 \leq j \leq k \leq n$ we define $C(\langle w_j\ \cdots\ w_k \rangle)$ to be the number of n-grams $\langle w_1^i \cdots w_n^i \rangle$ in the sample such that $w_h^i = w_h$ for $j \leq h \leq k$. For instance, $C(\langle w_{n-1} \rangle)$ is the number of n-grams whose second to last word is w_{n-1}, and $C(\langle w_{n-1} w_n \rangle)$ is the number of n-grams ending with the pair $w_n - 1, w_n$; thus, the ratio $C(\langle w_{n-1} w_n \rangle)/C(\langle w_{n-1} \rangle)$ is the empirical probability of w_n following w_{n-1}.

It may be very difficult to estimate the distribution of words following a phrase if that phrase was only seen a small number of times during training. To handle this very common case, an n-gram model is typically "mixed" or "smoothed" or "interpolated" with an $(n-1)$-gram model, and $(n-2)$-gram model, etc. More specifically, for $0 \leq k \leq n$ we define the interpolated k-gram model $\hat{P}(w_n|w_{n-k} \ldots w_{m-1})$ derived from the sample as follows where $\lambda(\langle w_{n-k} \ldots w_{n-1} \rangle)$ is a real number in $[0, 1]$ called the interpolation coefficient for the context $\langle w_{n-k} \ldots w_{n-1} \rangle$:

$$\begin{aligned}
\hat{P}(w_n) &\equiv \lambda(\langle\rangle)\frac{C(\langle w_n \rangle)}{m} + (1-\lambda(\langle\rangle))\frac{1}{|V|} \\
\hat{P}(w_n|w_{n-1}) &\equiv \lambda(\langle w_{n-1}\rangle)\frac{C(\langle w_{n-1},\ w_n\rangle)}{C(\langle w_{n-1}\rangle)} \\
&\quad +(1-\lambda(\langle w_{n-1}\rangle))\hat{P}(w_n) \\
&\ \ \vdots \\
\hat{P}(w_n|w_{n-k}\ldots w_{m-1}) &\equiv \lambda(\langle w_{n-k}\ldots w_{n-1}\rangle)\ \frac{C(\langle w_{n-k}\ldots w_{n-1},\ w_n\rangle)}{C(\langle w_{n-k}\ldots w_{n-1}\rangle)} \\
&\quad +(1-\lambda(\langle w_{n-k}\ldots w_{n-1}\rangle))\hat{P}(w_n|w_{n-k+1}\cdots w_{n-1}).
\end{aligned}$$

The models $\hat{P}(w_n)$, $\hat{P}(w_n|w_{n-1})$ and $\hat{P}(w_n|w_{n-2}, w_{n-1})$ are called the interpolated unigram, bigram and trigram models, respectively. The interpolated trigram model interpolates between trigram count ratios and the interpolated bigram model which, in turn, interpolates between the bigram count ratios and the interpolated unigram model which, in turn, interpolates between the empirical word frequencies and the uniform distribution. At all levels a separate interpolation coefficient is used for each conditioning context — the trigram model has a separate interpolation coefficient for each bigram context which, in turn, has a separate interpolation coefficient for each unigram context which, in turn, has a single interpolation coefficient for its single empty context. If $C(\langle w_{n-k} \ldots w_{n-1} \rangle) = 0$

then we require that $\lambda(\langle w_{n-k} \ldots w_{n-1} \rangle) = 0$ so that we avoid division by zero in the count ratios. Interpolation is one form of "smoothing" where smoothing can be interpreted loosely as any method of mixing information from various empirical conditional probabilities (Chen and Goodman 1998).

Note that the empirical count ratios will typically assign zero probability to many words that in fact have nonzero probabilities. If a model assigns zero probability to an event which actually has nonzero probability then the cross entropy of that model is infinite. However, assuming all interpolation coefficients are less than 1, each interpolated k-gram model assigns nonzero probability to all words for all contexts.

Intuitively, if a context has occurred a large number of times then the count ratio should be somewhat reliable. Unfortunately, the count of the context turns out to be a poor predictor of the appropriate interpolation weight. A better analysis of the appropriate interpolation weight can be given in terms of leave-one-out estimators.

4 THE GOOD-TURING LEAVE-ONE-OUT ESTIMATORS

We will argue in Section 6 that the setting of the interpolation parameters in an interpolated n-gram model should be theoretically analyzed in terms of leave-one-out estimates of the cross-entropy of the model as a whole. Unfortunately, the theoretical analysis of leave-one-out estimators is mathematically challenging. In this section we present a theoretical analysis of the Good-Turing leave-one-out estimators. The study of these estimators can be motivated in two ways. First, these estimators have played an important role in practical methods for setting interpolation coefficients in interpolated n-gram models. Second, they provide a case study in the analysis of leave-one-out estimation, a kind of warm-up exercise for the more challenging study of leave-one-out cross-entropy estimation for complete interpolated n-gram models.

Consider the problem of setting the interpolation coefficients in an interpolated n-gram model. In particular, consider the unigram model. This has one interpolation coefficient for mixing the unigram model with the uniform model. If the sample does not contain all words in V then this interpolation coefficient should be strictly less than 1. Intuitively we would like to set the interpolation coefficient to $(1 - M_0)$ where M_0 is the probability that, when we draw a fresh n-gram, the word w_n is one that did not occur in the training sample. More generally, we would intuitively like to set $\lambda(\langle w_{n-k} \ldots w_{n-1} \rangle)$ to $(1 - M_0(\langle w_{n-k} \ldots w_{n-1} \rangle))$ where $M_0(\langle w_{n-k} \ldots w_{n-1} \rangle)$ is the probability, given context $\langle w_{n-k} \ldots w_{n-1} \rangle$, that w_n has not occurred previously with this context in the sample. The fundamental Good-Turing estimator estimates this "missing mass".

Since the publication of the Good-Turing estimators in 1953 (Good 1953), these estimators have been used extensively in language modeling applications (Chen and Goodman 1998; Church and Gale 1991; Katz 1987). According to Good (Good 2000), the Good-Turing estimators were developed by Alan Turing during World War II while breaking Enigma codes. The Enigma was an encryption device used by the German navy. The Enigma used, as part of its encryption key, a three letter sequence. These three letter sequences were selected from a book containing all such sequences in a random order. However, a person opening the book and selecting an entry was likely to select a previously used entry, say the entry on the top of a page where the binding of the book was creased. Given a sample of previously used entries, Turing wanted to estimate the likelihood that the current unknown entry was one that had been previously used, and further, to estimate the probability distribution over the previously used entries.

Although Good-Turing estimation can be motivated by n-gram models, the discussion of these estimators can be simplified by considering a process of drawing words from a single fixed distribution on words. In an n-gram model, the distribution will be the conditional distribution for some context of the model. But for the remainder of this section we consider drawing words from an arbitrary fixed distribution on words. The analysis of Good-Turing estimators discussed here does not rely on the use of a finite vocabulary so, for this section only, we allow the underlying vocabulary of words to be infinite. We simply assume an unknown probability distribution P on a countable set V and we denote the probability of word w by P_w. Although V can be any countable set we will continue to call the elements of V "words". We consider a sample S of m words drawn independently from V, each according to distribution P. For a sample S of m words and for any word $w \in V$ we define the count of w, denoted $c(w)$, to be the number of times word w occurs in the sample S. For any integer $k \geq 0$, we define S_k to be the set of words $w \in V$ such that $c(w) = k$. Note that S_0 is the set of words in V not occurring in S. We define M_k to be the probability of drawing a word in the set S_k:

$$M_k \equiv \sum_{w \in S_k} P_w.$$

Note that M_k depends on the sample, i.e., it is a random variable. The quantity M_0 is the so-called *missing mass*, i.e., the total probability mass of words not occurring in the sample.

The Good-Turing estimator of the missing mass M_0 is $G_0 \equiv |S_1|/m$, i.e., the fraction of examples seen exactly once. More generally, the Good-Turing estimator G_k of M_k is defined to be

$$G_k \equiv \frac{k+1}{m}|S_{k+1}|. \tag{3}$$

To understand these definitions, it is useful to view the Good-Turing estimators as leave-one-out estimators of the random variables M_k. We can define a general notion of a leave-one-out estimator by letting $\Phi[S,\ w]$ be any statement relating a sample S to a word w. Define $P(\Phi[S,\ w])$ to be the probability that when we draw a sample S and then a fresh word w we have that $\Phi[S,\ w]$ holds. Consider $P(w \in S_k)$. Note that $P(w \in S_k \mid S)$ equals the value of M_k for the sample S. So we have that the expected value of M_k is $\sum_S P(S)P(w \in S_k \mid S)$ which equals $P(w \in S_k)$. For any fixed sample S and element $w \in S$ define $S\backslash w$ to be the sample with the element w removed (the count of w is reduced by one). The leave-one-out estimate of $P(\Phi[S,\ w])$ is defined to be $\frac{1}{m}|\{w \in S\ :\ \Phi[S\backslash w,\ w]\}|$. In general we have that the expectation of the leave-one-out estimate of $P(\Phi[S,\ w])$ on a sample of size m equals $P(\Phi[S,\ w])$ on a sample of size $m-1$. The leave-one-out estimate of $P(w \in S_k)$ turns out to be the fraction of the sample that occurs $k+1$ times in the sample, i.e., G_k as defined in equation (3).

It is not hard to show that the expectations of G_k and M_k are close to one another. Nevertheless, this does not tell us how good an estimate G_k will be of M_k. We are therefore interested in giving a confidence interval for M_k as a function of G_k, the sample size m

and the confidence level δ. We do this by showing that, with high confidence, both G_k and M_k are near their respective expectations. In (McAllester and Schapire 2000) we prove the following where, for fixed constants, we have that (5) and (6) hold with probability at least $1 - \delta$:

$$
\begin{aligned}
|\mathrm{E}[G_k] - \mathrm{E}[M_k]| &\leq O\left(\frac{k+1}{m}\right) && (4) \\
|G_k - \mathrm{E}[G_k]| &\leq O\left((k+1)\sqrt{\frac{\ln\frac{2}{\delta}}{m}}\right) && (5) \\
|M_k - \mathrm{E}[M_k]| &\leq O\left(\left(1 + k + \ln\frac{m}{\delta}\right)\sqrt{\frac{\ln\frac{2}{\delta}}{m}}\right). && (6)
\end{aligned}
$$

Thus, together these bounds imply that as m gets large (with k fixed), the difference between G_k and M_k goes to zero; the bounds also tell us that this convergence to zero goes like $1/\sqrt{m}$. The constants in the bounds are modest but greater than one and both (5) and (6) become vacuous for $k \geq \sqrt{m}$. Bound (5) is a simple corollary of McDiarmid's theorem (given below). Bound (6) is also proved using McDiarmid's theorem but the proof is considerably more difficult and the term of $\ln(m/\delta)$ is probably an artifact of the proof method. In the next section we focus on the special case of M_0 and refer the reader to (McAllester and Schapire 2000) for the case of $k > 0$.

5 AN ANALYSIS OF G_0

Here we focus on the estimate G_0 of the missing mass. The accuracy of this estimator is covered by the above theorem for the case of $k = 0$. However, we were able to prove an upper bound M_0 that eliminates the term $\ln(m/\delta)$ in (6). More specifically, the following holds with high probability over the choice of the sample.

Theorem 1 *With probability at least $1 - \delta$ over the choice of the sample*

$$M_0 \leq G_0 + (2\sqrt{2} + \sqrt{3})\sqrt{\frac{\ln(\frac{3}{\delta})}{m}}.$$

Because theorem 1 is potentially significant for language modeling, and because it provides a case study in the analysis of a nontrivial leave-one-out estimator, we now present some of the details of its proof.

It is shown in (McAllester and Schapire 2000) that $\mathrm{E}[M_0] \leq \mathrm{E}[G_0]$. This implies that

$$M_0 \leq G_0 + (\mathrm{E}[G_0] - G_0) + (M_0 - \mathrm{E}[M_0]).$$

So it now suffices to give convergence rates of G_0 and M_0 to their respective means. To bound the difference between G_0 and its expectation, and the difference between M_0 and its expectation, we use McDiarmid's theorem. This beautiful and very useful theorem allows us to bound how fast *any* function of m independent random variables converges to its mean, provided that the function is not too sensitive to changes in individual variables.

Theorem 2 *(McDiarmid 1989) Let $X_1, \ldots, X_m$ be independent random variables taking values in a set V and let $f : V^m \to \mathbb{R}$ be such that*

$$\sup_{x_1,\ldots,x_m,x_i' \in V} |f(x_1,\ldots,x_m) - f(x_1,\ldots,x_{i-1},x_i',x_{i+1},\ldots,x_m)| \le c_i.$$

Then with probability at least $1-\delta$

$$f(X_1,\ldots,X_m) - \mathrm{E}\left[f(X_1,\ldots,X_m)\right] \le \sqrt{\frac{\ln(\frac{1}{\delta})\sum_{i=1}^m c_i^2}{2}},$$

and with probability at least $1-\delta$

$$\mathrm{E}\left[f(X_1,\ldots,X_m)\right] - f(X_1,\ldots,X_m) \le \sqrt{\frac{\ln(\frac{1}{\delta})\sum_{i=1}^m c_i^2}{2}}.$$

A natural special case is $x_i \in [0,1]$ and $f(x_1, \ldots, x_n) = \frac{1}{m}\sum_{i=1}^m x_i$. In this case, $c_i = 1/m$ and McDiarmid's theorem reduces to the Hoeffding inequalities.

We first note that a single change in the sample can change G_0 by at most $2/m$. Thus, applying McDiarmid's theorem immediately gives a bound on the difference between G_0 and its mean. In the following, $\forall^\delta S \Phi[S, \delta]$ is an alternate notation for $P(\Phi[S, \delta]) \ge 1-\delta$ so we can read $\forall^\delta S\ \Phi[S, \delta]$ as "for all but a fraction δ of the samples S we have $\Phi[S, \delta]$".

Lemma 3

$$\forall \delta > 0\ \forall^\delta S\ \ \mathrm{E}\left[G_0\right] - G_0 \le \sqrt{2}\sqrt{\frac{\ln\frac{1}{\delta}}{m}}$$

This leaves us with the more difficult problem of bounding $M_0 - \mathrm{E}\left[M_0\right]$. To bound this difference we divide M_0 into a high frequency component M_0^+ and a low frequency component M_0^- as follows:

$$\begin{aligned} M_0^+ &\equiv \sum_{w:P_w > 1/m,\ c(w)=0} P_w. \\ M_0^- &\equiv \sum_{w:P_w \le 1/m,\ c(w)=0} P_w. \end{aligned}$$

We prove the following two lemmas separately:

Lemma 4 $\forall \delta > 0\ \ \forall^\delta S\ \ \ M_0^+ \le \mathrm{E}\left[M_0^+\right] + \sqrt{\frac{3\ln(\frac{1}{\delta})}{m}}.$

Lemma 5 $\forall \delta > 0\ \ \forall^\delta S\ \ \ M_0^- \le \mathrm{E}\left[M_0^-\right] + \sqrt{\frac{2\ln(\frac{1}{\delta})}{m}}.$

Lemma 5 follows from an application of McDiarmid's theorem and the observation that a single change in the sample can change M_0^- by at most $2/m$. Lemma 4 is more involved and is proved at the end of this section. Theorem 1 now follows by applying the union bound to Lemmas 3, 4 and 5. The union bound implies that if we have

$$\begin{array}{ll} \forall\delta > 0\ \forall^{\delta} S & \Phi_1[S,\ \delta] \\ \vdots & \\ \forall\delta > 0\ \forall^{\delta} S & \Phi_k[S,\ \delta] \end{array}$$

then we have

$$\forall\delta > 0\ \forall^{k\delta} S \quad \Phi_1[S,\ \delta] \wedge \ldots \wedge \Phi_k[S,\ \delta]$$

since

$$\Pr\left[\bigvee_i \neg\Phi_i[S,\ \delta]\right] \leq \sum_i \Pr\left[\neg\Phi_i[S,\ \delta]\right].$$

It now remains only to prove Lemma 4. Let $B = \{w \in V : P_w > 1/m\}$. For each word $w \in B$, we introduce a random variable X_w which is 1 if w does *not* occur in the sample and 0 otherwise. We can then write M_0^+ as

$$M_0^+ = \sum_{w \in B} P_w X_w.$$

The counts are "contravariant," meaning that making one larger tends to make the others smaller and vice-versa. Furthermore, any system of monotonic functions of the counts is also contravariant. This allows us to prove the following lemma which generalizes an observation made in (Panconesi and Srinivasan 1997), and which will allow us to treat these dependent count variables as if they were independent.

Lemma 6 *For any finite subset B of the underlying vocabulary, and for any choice of a non-negative monotonically decreasing function f_w for each word $w \in B$, we have the following:*

$$\mathrm{E}\left[\Pi_{w\in B}\ f_w(c(w))\right] \leq \Pi_{w\in B}\mathrm{E}\left[f_w(c(w))\right]$$

Proof: See appendix. □

Lemma 6 also holds if all f_w are monotonically increasing, but we will only need the decreasing case here.

We now use lemma 6 to prove the following.

Lemma 7 *For $\lambda > 0$ and $\epsilon > 0$ we have*

$$\Pr\left[M_0^+ \geq \mathrm{E}\left[M_0^+\right] + \epsilon\right] \leq e^{F(\lambda) - \lambda\epsilon}$$

where

$$F(\lambda) \equiv \sum_{w:\ P_w > 1/m} \left(\ln(Q_w e^{\lambda P_w} + (1 - Q_w)) - \lambda P_w Q_w\right)$$

and $Q_w = (1 - P_w)^m$ is the probability that word w does not occur in the sample.

Proof:

$$\begin{aligned}
&\Pr\left[M_0^+ \geq \mathrm{E}\left[M_0^+\right] + \epsilon\right] \\
&= \Pr\left[\exp\left(\left(\lambda(M_0^+ - \mathrm{E}\left[M_0^+\right] - \epsilon)\right)\right) \geq 1\right] \\
&\leq \mathrm{E}\left[\exp\left(\lambda(M_0^+ - \mathrm{E}\left[M_0^+\right] - \epsilon)\right)\right] \\
&= e^{-\lambda(\mathrm{E}\left[M_0^+\right]+\epsilon)}\, \mathrm{E}\left[e^{\lambda M_0^+}\right] \\
&= e^{-\lambda(\mathrm{E}\left[M_0^+\right]+\epsilon)}\, \mathrm{E}\left[\Pi_{w\in B}\, e^{\lambda P_w X_w}\right] \\
&\leq e^{-\lambda(\mathrm{E}\left[M_0^+\right]+\epsilon)} \prod_{w\in B} \mathrm{E}\left[e^{\lambda P_w X_w}\right] \\
&= \exp\left(-\lambda\epsilon - \lambda\sum_{w\in B} P_w Q_w\right) \prod_{w\in B}\left(Q_w e^{\lambda P_w} + (1-Q_w)e^0\right) \\
&= \exp\left(-\lambda\epsilon - \lambda\sum_{w\in B} P_w Q_w\right) \prod_{w\in B}\left(1 + \left(e^{\lambda P_w} - 1\right) Q_w\right) \\
&= \exp\left(-\lambda\epsilon - \lambda\sum_{w\in B} P_w Q_w\right) \exp\left(\sum_{w\in B} \ln\left(1 + \left(e^{\lambda P_w} - 1\right) Q_w\right)\right) \\
&= e^{F(\lambda) - \lambda\epsilon}.
\end{aligned}$$

The first inequality uses Markov's inequality ($\mathrm{E}[X] \geq a\Pr[X \geq a]$ for X nonnegative). Lemma 6 was used in the second inequality. □

Next we prove the following bound on the function $F(\lambda)$:

Lemma 8 *For $\lambda \leq m/2$*

$$F(\lambda) \leq \frac{\lambda^2}{(e-1)m}.$$

Proof: First, note that $F(0) = 0$. Now let $F'(\lambda)$ denote the first derivative of F, i.e., $dF/d\lambda$ evaluated at λ. Then

$$F'(\lambda) = \sum_{w:\ P_w > 1/m} \frac{Q_w P_w}{(1-Q_w)e^{-\lambda P_w} + Q_w} - Q_w P_w.$$

Note that $F'(0) = 0$. Now letting $F''(\lambda)$ denote the second derivative of F we get that

$$F''(\lambda) \quad = \quad \sum_{w:\ P_w > 1/m} \frac{Q_w P_w^2 (1-Q_w) e^{-\lambda P_w}}{\left[(1-Q_w)e^{-\lambda P_w} + Q_w\right]^2}$$

$$
\begin{aligned}
&\le \sum_{w:\ P_w > 1/m} \frac{Q_w P_w^2 (1-Q_w) e^{-\lambda P_w}}{[(1-Q_w)e^{-\lambda P_w}]^2} \\
&= \sum_{w:\ P_w > 1/m} \frac{Q_w P_w^2}{(1-Q_w)e^{-\lambda P_w}} \\
&= \sum_{w:\ P_w > 1/m} P_w \frac{Q_w P_w e^{\lambda P_w}}{(1-Q_w)} \\
&\le \sum_{w:\ P_w > 1/m} P_w \frac{P_w e^{(\lambda - m)P_w}}{(1-Q_w)} \\
&\le \sum_{w:\ P_w > 1/m} P_w \frac{P_w e^{(\lambda - m)P_w}}{(1-1/e)}
\end{aligned}
$$

where the last two inequalities use the inequality $Q_w = (1-P_w)^m \le e^{-mP_w}$ which is at most $1/e$ for $P_w \ge 1/m$. For $\alpha > 0$ and $x \ge 0$ one can show, by maximizing over x, that

$$xe^{-\alpha x} \le \frac{1}{\alpha e}.$$

For $\lambda < m$, we can use this inequality with $\alpha = (m - \lambda)$ to get that

$$
\begin{aligned}
F''(\lambda) &\le \sum_{w:\ P_w > 1/m} P_w \frac{1}{(e-1)(m-\lambda)} \\
&\le \frac{1}{(e-1)(m-\lambda)}.
\end{aligned}
$$

Since $\lambda \le m/2$ we then have that

$$F''(\lambda) \le \frac{2}{(e-1)m}.$$

By Taylor's formula,

$$F(\lambda) = F(0) + \lambda F'(0) + \frac{\lambda^2}{2} F''(\lambda')$$

for some $\lambda' \in (0, \lambda)$. The lemma now follows from $F(0) = 0$, $F'(0) = 0$ and $F''(\lambda') \le 2/((e-1)m)$. □

Proof of Lemma 4: Let $\lambda = m\epsilon/2$. Lemmas 7 and 8 together imply that

$$
\begin{aligned}
\Pr\left[M_0^+ \ge \mathrm{E}\left[M_0^+\right] + \epsilon\right] &\le \exp\left(\frac{\lambda^2}{(e-1)m} - \lambda\epsilon\right) \\
&= \exp\left(\frac{m\epsilon^2}{4(e-1)} - \frac{m\epsilon^2}{2}\right) \\
&\le e^{-m\epsilon^2/3}.
\end{aligned}
$$

Lemma 4 now follows by setting this probability equal to δ and solving for ϵ. □

6 CONCLUSIONS: LEAVE-ONE-OUT MINIMIZATION

We stated in Section 3 that one of the fundamental problems in the theory of language modeling is to explain why models that memorize the training data do not overfit. There are various approaches to this problem. Bayesian model averaging memorizes the training data and can be justified with Bayesian assumptions. PAC-Bayesian model averaging is similar to Bayesian model averaging in that it is based on a prior distribution on models but, unlike Bayesian model averaging, PAC-Bayesian model averaging can be justified independent of Bayesian assumptions about the meaning of the prior (McAllester 1999). Unfortunately, neither the Bayesian approach nor the PAC-Bayesian approach justify the particular form of the smoothing methods in language modeling that work well in practice. So the real theoretical challenge is to explain the superiority of the methods that are in fact empirically best.

The Good-Turing estimate of the missing mass is fundamentally non-Bayesian — it is a direct measure of the quantity of missing mass and converges rapidly to the estimated quantity. The estimate of the missing mass is analogous to a statistical mean estimator, such as estimating the bias of a biased coin. The convergence rate guarantees that for large samples the estimate is accurate independent of Bayesian assumptions. It seems that a non-Bayesian justification — a justification not involving a prior on models — should be possible for modeling methods based on the Good-Turing estimators.

The Good-Turing estimators are leave-one-out estimators. The convergence results on the Good-Turing estimators show that, at least in some cases, leave-one-out estimators can be guaranteed to be accurate — one can give Chernoff-like confidence intervals for the true value of the estimated quantity. Recently Bousquet and Elisseeff have defined a general notion of a stable learning algorithm and have used McDiarmid's theorem to show that for any algorithm that is stable in their sense the leave-one-out estimate of the generalization loss has Chernoff-like convergence (Bousquet and Elisseeff 2001). Unfortunately their definition of stability is fairly restrictive. They require, essentially, that changing a single instance in a sample of m instances does not change the model by more than $O(1/m)$ where the distance between models is taken to be the maximum difference between the loss of the two models over all possible instances. It is interesting that several well known modeling algorithms can be shown to be stable in this very strong sense. But these stability requirements are too strong to make Bousquet and Elisseeff's results applicable to Good-Turing estimation or n-gram language models. A single change in a (very unlikely) sample can radically alter M_0^+. Our proof of a convergence rate for M_0^+ is not based on McDiarmid's theorem.

We will say that a learning algorithm is leave-one-out measurable if the leave-one-out estimate of the error of the algorithm has Chernoff-like convergence — with probability at least $1 - \delta$ the difference between the leave-one-out estimate of the generalization error and the true generalization error on a sample of size m is bounded by $O(\sqrt{\ln(1/\delta)/m})$. Bousquet and Elisseeff show that stable algorithms are leave-one-out measurable but presumably many unstable algorithms are also leave-one-out measurable. It is known that many learning algorithms are not leave-one-out measurable — the algorithm that produces the model that either always guesses 1 or always guesses 0 based on the number of 1's and 0's in the sample is not leave-one-out measurable. However, we conjecture that

n-gram language models under any of a variety of smoothing methods are leave-one-out measurable.

For any family of leave-one-out measurable algorithms we could select an algorithm by minimizing leave-one-out error over the algorithms in the family. For example, we can define a family of n-gram learning algorithms where each algorithm uses a different (large) set of interpolation parameters. In general one could combine the Chernoff-like convergence of the leave-one-out estimator with a union bound over a large class of learning algorithms to bound the generalization error of the algorithm minimizing the leave-one-out estimate. This leave-one-out minimization over a large class of algorithms seems fundamentally different from the usual empirical loss minimization over a class of models. We hope to investigate leave-one-out minimization in future work on language modeling.

References

Blumer, A., A. Ehrenfeucht, D. Haussler, and M. K. Warmuth (1987, April). Occam's razor. *Information Processing Letters 24*(6), 377–380.

Bousquet, O. and A. Elisseeff (2001). Algorithmic stability and generalization performance. In *Advances in Neural Information Processing Systems 13*.

Charniak, E. (2000). A maximum-entropy-inspired parser. In *Proceedings of the 1st Meeting of the North American Chapter of the Association for Computational Linguistics*, pp. 132–139.

Chelba, C. and F. Jelinek (1998). Exploiting syntactic structure for language modeling. In *Proceedings of the Thirty-Sixth Annual Meeting of the Association for Computational Linguistics and Seventeenth International Conference on Computational Linguistics*.

Chen, S. and J. Goodman (1998, August). An empirical study of smoothing techniques for language modeling. Technical Report TR-10-98, Harvard University.

Church, K. W. and W. A. Gale (1991). A comparison of the enhanced Good-Turing and deleted estimation methods for estimating probabilities of English bigrams. *Computer Speech and Language 5*, 19–54.

Collins, M. (1997). Three generative, lexicalised models for statistical parsing. In *Proceedings of the 35th Annual Meeting of the ACL*.

Cover, T. M. and J. A. Thomas (1991). *Elements of Information Theory*. Wiley.

Good, I. J. (1953, December). The population frequencies of species and the estimation of population parameters. *Biometrika 40*(16), 237–264.

Good, I. J. (2000). Turing's anticipation of emprical Bayes in connection with the cryptanalysis of the Naval Enigma. *Journal of Statistical Computation and Simulation 66*(2), 101–112.

Katz, S. M. (1987, March). Estimation of probabilities from sparse data for the language model component of a speech recognizer. *IEEE Transactions on Acoustics, Speech and Signal Processing ASSP-35*(3), 400–401.

McAllester, D. (1999). PAC-Bayesian model averaging. In *Proceedings of the Twelfth Annual Conference on Computational Learning Theory*.

McAllester, D. and R. Schapire (2000). On the convergence rate of Good-Turing estimators. In *Proceedings of the Thirteenth Annual Conference on Computational Learning Theory*.

McDiarmid, C. (1989). On the method of bounded differences. In *Surveys in Combinatorics 1989*, pp. 148–188. Cambridge University Press.

Panconesi, A. and A. Srinivasan (1997, April). Randomized distributed edge coloring via an extension of the Chernoff-Hoeffding bounds. *SIAM Journal of Computing 26*(2), 350–368.

Pereira, F. C. and Y. Singer (1999). An efficient extension to mixture techniques for prediction and decision trees. *Machine Learning 36*.

A PROOF OF LEMMA 6

To prove Lemma 6 we need some preliminary lemmas. For each word w fix a monotonically decreasing non-negative function f_w. For any sample S let S_B be the subset of the sample consisting of words in the set B. We start with the following lemma.

Lemma 9 *For any (possibly infinite) subset B of the vocabulary we have that*

$$\mathrm{E}\left[\Pi_{w\in B}\ f_w(c(w)) \mid |S_B| = k\right]$$

is a monotonically decreasing function of k.

Proof: For any sample S let $c(w, S)$ be the count of word w in sample S. For any pair of samples S and U we have $c(w,\ S_B \cup U_B) \geq c(w,\ S_B)$ and hence

$$\Pi_{w\in B}\ f_w(c(w,\ S_B \cup U_B)) \leq \Pi_{w\in B}\ f_w(c(w,\ S_B)).$$

So for $k_1 \leq k_2$ we have

$$\sum_{|S_B|=k_1,\ |U_B|=k_2-k_1} P(S_B \mid |S_B| = k_1)P(U_B \mid |U_B| = k_2 - k_1)\Pi_{w\in B}\ f_w(c(w,\ S_B \cup U_B))$$

$$\leq \sum_{|S_B|=k_1,\ |U_B|=k_2-k_1} P(S_B \mid |S_B| = k_1)P(U_B \mid |U_B| = k_2 - k_1)\Pi_{w\in B}\ f_w(c(w,\ S_B)).$$

The left hand side equals $\mathrm{E}\left[\Pi_{w\in B}\ f_w(c(w)) \mid |S_B| = k_2\right]$ and the right hand side equals $\mathrm{E}\left[\Pi_{w\in B}\ f_w(c(w)) \mid |S_B| = k_1\right]$. □

Lemma 10 *For $w' \notin B$ we have that $\mathrm{E}\left[\Pi_{w\in B}\ f_w(c(w)) \mid c(w') = k\right]$ is a monotonically increasing function of k.*

Proof: Let $V\backslash w'$ be the set of all words other than w'. Let g_w be f_w for $w \in B$ and the constant function 1 otherwise. We then have $\Pi_{w\in B} f_w(c(w)) = \Pi_{w\in V\backslash w'} g_w(c(w))$ which gives the following:

$$\mathrm{E}\left[\Pi_{w\in B}\ f_w(c(w)) \mid c(w') = k\right] = \mathrm{E}\left[\Pi_{w\in V\backslash w'}\ g_w(c(w)) \mid\ |S_{V\backslash w'}| = m - k\right].$$

The result now follows from Lemma 9. □

We now prove Lemma 6 by induction on the number of words in B. The result is immediate if B contains only a single word. Now assume the result holds for sets smaller than B and consider $w \in B$. Let $B\backslash w$ be the word set B minus the word w. We now have

$$\mathrm{E}\left[\Pi_{w\in B}\ f_w(c(w))\right] = \sum_{k=0}^{m} P(c(w)=k) f_w(k) \mathrm{E}\left[\Pi_{w'\in B\backslash w}\ f_{w'}(c(w')) \mid c(w)=k\right].$$

We now use the fact that for any functions f and g from reals to reals, and any distribution P on the reals, we have that if f is monotonically decreasing and g is monotonically increasing then $\mathrm{E}_{k\sim P}\left[f(k)g(k)\right] \leq \mathrm{E}_{k\sim P}\left[f(x)\right] \mathrm{E}_{k\sim P}\left[g(x)\right]$. This gives

$$\mathrm{E}\left[\Pi_{w\in B}\ f_w(c(w))\right] \leq \mathrm{E}\left[f_w(c(w)\right] \mathrm{E}\left[\Pi_{w'\in B\backslash w}\ f_{w'}(c(w'))\right].$$

Lemma 6 now follows from the induction hypothesis applied to the set $B\backslash w$.