To appear in *Proceedings of the 34th Annual Meeting of the ACL*, June 1996

An Empirical Study of Smoothing Techniques for Language Modeling

Stanley F. Chen
Harvard University
Aiken Computation Laboratory
33 Oxford St.
Cambridge, MA 02138
sfc@eecs.harvard.edu

Joshua Goodman
Harvard University
Aiken Computation Laboratory
33 Oxford St.
Cambridge, MA 02138
goodman@eecs.harvard.edu

Abstract

We present an extensive empirical comparison of several smoothing techniques in the domain of language modeling, including those described by Jelinek and Mercer (1980), Katz (1987), and Church and Gale (1991). We investigate for the first time how factors such as training data size, corpus (*e.g.*, Brown versus Wall Street Journal), and n-gram order (bigram versus trigram) affect the relative performance of these methods, which we measure through the cross-entropy of test data. In addition, we introduce two novel smoothing techniques, one a variation of Jelinek-Mercer smoothing and one a very simple linear interpolation technique, both of which outperform existing methods.

1 Introduction

Smoothing is a technique essential in the construction of n-gram language models, a staple in speech recognition (Bahl, Jelinek, and Mercer, 1983) as well as many other domains (Church, 1988; Brown et al., 1990; Kernighan, Church, and Gale, 1990). A *language model* is a probability distribution over strings $P(s)$ that attempts to reflect the frequency with which each string s occurs as a sentence in natural text. Language models are used in speech recognition to resolve acoustically ambiguous utterances. For example, if we have that $P(\textit{it takes two}) \gg P(\textit{it takes too})$, then we know *ceteris paribus* to prefer the former transcription over the latter.

While smoothing is a central issue in language modeling, the literature lacks a definitive comparison between the many existing techniques. Previous studies (Nadas, 1984; Katz, 1987; Church and Gale, 1991; MacKay and Peto, 1995) only compare a small number of methods (typically two) on a single corpus and using a single training data size. As a result, it is currently difficult for a researcher to intelligently choose between smoothing schemes.

In this work, we carry out an extensive empirical comparison of the most widely used smoothing techniques, including those described by Jelinek and Mercer (1980), Katz (1987), and Church and Gale (1991). We carry out experiments over many training data sizes on varied corpora using both bigram and trigram models. We demonstrate that the relative performance of techniques depends greatly on training data size and n-gram order. For example, for bigram models produced from large training sets Church-Gale smoothing has superior performance, while Katz smoothing performs best on bigram models produced from smaller data. For the methods with parameters that can be tuned to improve performance, we perform an automated search for optimal values and show that sub-optimal parameter selection can significantly decrease performance. To our knowledge, this is the first smoothing work that systematically investigates any of these issues.

In addition, we introduce two novel smoothing techniques: the first belonging to the class of smoothing models described by Jelinek and Mercer, the second a very simple linear interpolation method. While being relatively simple to implement, we show that these methods yield good performance in bigram models and superior performance in trigram models.

We take the performance of a method m to be its cross-entropy on test data

$$\frac{1}{N_T} \sum_{i=1}^{l_T} -\log_2 P_m(t_i)$$

where $P_m(t_i)$ denotes the language model produced with method m and where the test data T is composed of sentences $(t_1, \ldots, t_{l_T})$ and contains a total of N_T words. The entropy is inversely related to the average probability a model assigns to sentences in the test data, and it is generally assumed that lower entropy correlates with better performance in applications.

1.1 Smoothing n-gram Models

In n-gram language modeling, the probability of a string $P(s)$ is expressed as the product of the probabilities of the words that compose the string, with each word probability conditional on the identity of the last $n-1$ words, *i.e.*, if $s = w_1 \cdots w_l$ we have

$$P(s) = \prod_{i=1}^{l} P(w_i|w_1^{i-1}) \approx \prod_{i=1}^{l} P(w_i|w_{i-n+1}^{i-1}) \qquad (1)$$

where w_i^j denotes the words $w_i \cdots w_j$. Typically, n is taken to be two or three, corresponding to a *bigram* or *trigram* model, respectively.[1]

Consider the case $n = 2$. To estimate the probabilities $P(w_i|w_{i-1})$ in equation (1), one can acquire a large corpus of text, which we refer to as *training data*, and take

$$\begin{aligned} P_{\mathrm{ML}}(w_i|w_{i-1}) &= \frac{P(w_{i-1}w_i)}{P(w_{i-1})} \\ &= \frac{c(w_{i-1}w_i)/N_S}{c(w_{i-1})/N_S} \\ &= \frac{c(w_{i-1}w_i)}{c(w_{i-1})} \end{aligned}$$

where $c(\alpha)$ denotes the number of times the string α occurs in the text and N_S denotes the total number of words. This is called the *maximum likelihood* (ML) estimate for $P(w_i|w_{i-1})$.

While intuitive, the maximum likelihood estimate is a poor one when the amount of training data is small compared to the size of the model being built, as is generally the case in language modeling. For example, consider the situation where a pair of words, or *bigram*, say *burnish the*, doesn't occur in the training data. Then, we have $P_{\mathrm{ML}}(\textit{the}|\textit{burnish}) = 0$, which is clearly inaccurate as this probability should be larger than zero. A zero bigram probability can lead to errors in speech recognition, as it disallows the bigram regardless of how informative the acoustic signal is. The term *smoothing* describes techniques for adjusting the maximum likelihood estimate to hopefully produce more accurate probabilities.

As an example, one simple smoothing technique is to pretend each bigram occurs once more than it actually did (Lidstone, 1920; Johnson, 1932; Jeffreys, 1948), yielding

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

where V is the vocabulary, the set of all words being considered. This has the desirable quality of preventing zero bigram probabilities. However, this scheme has the flaw of assigning the same probability to say, *burnish the* and *burnish thou* (assuming neither occurred in the training data), even though intuitively the former seems more likely because the word *the* is much more common than *thou*.

To address this, another smoothing technique is to *interpolate* the bigram model with a unigram model $P_{\mathrm{ML}}(w_i) = c(w_i)/N_S$, a model that reflects how often each word occurs in the training data. For example, we can take

$$P_{\mathrm{interp}}(w_i|w_{i-1}) = \lambda P_{\mathrm{ML}}(w_i|w_{i-1}) + (1-\lambda) P_{\mathrm{ML}}(w_i)$$

getting the behavior that bigrams involving common words are assigned higher probabilities (Jelinek and Mercer, 1980).

2 Previous Work

The simplest type of smoothing used in practice is *additive* smoothing (Lidstone, 1920; Johnson, 1932; Jeffreys, 1948), where we take

$$P_{\mathrm{add}}(w_i|w_{i-n+1}^{i-1}) = \frac{c(w_{i-n+1}^{i}) + \delta}{c(w_{i-n+1}^{i-1}) + \delta|V|} \qquad (2)$$

and where Lidstone and Jeffreys advocate $\delta = 1$. Gale and Church (1990; 1994) have argued that this method generally performs poorly.

The Good-Turing estimate (Good, 1953) is central to many smoothing techniques. It is not used directly for n-gram smoothing because, like additive smoothing, it does not perform the interpolation of lower- and higher-order models essential for good performance. Good-Turing states that an n-gram that occurs r times should be treated as if it had occurred r^* times, where

$$r^* = (r+1)\frac{n_{r+1}}{n_r}$$

and where n_r is the number of n-grams that occur exactly r times in the training data.

Katz smoothing (1987) extends the intuitions of Good-Turing by adding the interpolation of higher-order models with lower-order models. It is perhaps the most widely used smoothing technique in speech recognition.

Church and Gale (1991) describe a smoothing method that combines the Good-Turing estimate with *bucketing*, the technique of partitioning a set of n-grams into disjoint groups, where each group is characterized independently through a set of parameters. Like Katz, models are defined recursively in terms of lower-order models. Each n-gram is assigned to one of several buckets based on its frequency predicted from lower-order models. Each bucket is treated as a separate distribution and Good-Turing estimation is performed within each, giving corrected counts that are normalized to yield probabilities.

[1] To make the term $P(w_i|w_{i-n+1}^{i-1})$ meaningful for $i < n$, one can pad the beginning of the string with a distinguished token. In this work, we assume there are $n-1$ such distinguished tokens preceding each sentence.

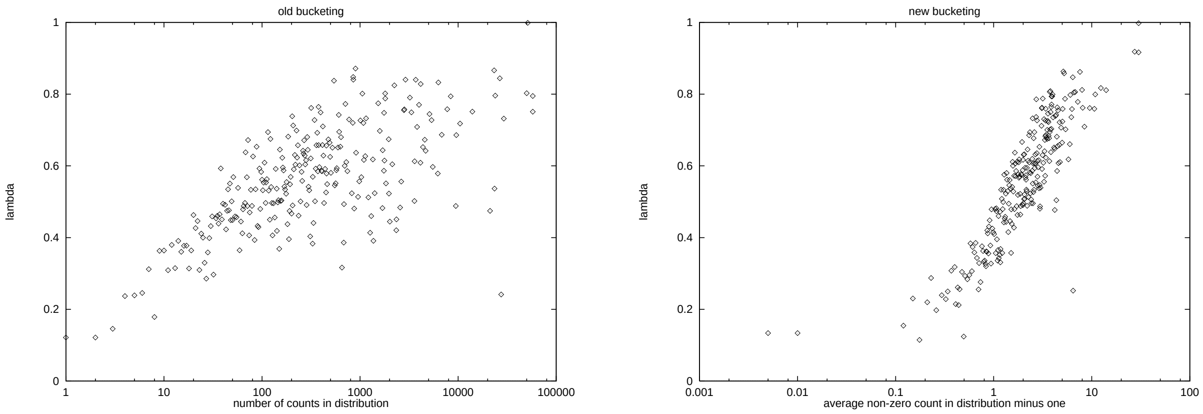

Figure 1: λ values for old and new bucketing schemes for Jelinek-Mercer smoothing; each point represents a single bucket

The other smoothing technique besides Katz smoothing widely used in speech recognition is due to Jelinek and Mercer (1980). They present a class of smoothing models that involve linear interpolation, *e.g.*, Brown et al. (1992) take

$$\begin{aligned} P_{\text{interp}}(w_i|w_{i-n+1}^{i-1}) = \\ \lambda_{w_{i-n+1}^{i-1}} P_{\text{ML}}(w_i|w_{i-n+1}^{i-1}) + \\ (1-\lambda_{w_{i-n+1}^{i-1}})\, P_{\text{interp}}(w_i|w_{i-n+2}^{i-1}) \end{aligned} \tag{3}$$

That is, the maximum likelihood estimate is interpolated with the smoothed lower-order distribution, which is defined analogously. Training a distinct $\lambda_{w_{i-n+1}^{i-1}}$ for each w_{i-n+1}^{i-1} is not generally felicitous; Bahl, Jelinek, and Mercer (1983) suggest partitioning the $\lambda_{w_{i-n+1}^{i-1}}$ into buckets according to $c(w_{i-n+1}^{i-1})$, where all $\lambda_{w_{i-n+1}^{i-1}}$ in the same bucket are constrained to have the same value.

To yield meaningful results, the data used to estimate the $\lambda_{w_{i-n+1}^{i-1}}$ need to be disjoint from the data used to calculate P_{ML}.[2] In *held-out interpolation*, one reserves a section of the training data for this purpose. Alternatively, Jelinek and Mercer describe a technique called *deleted interpolation* where different parts of the training data rotate in training either P_{ML} or the $\lambda_{w_{i-n+1}^{i-1}}$; the results are then averaged.

Several smoothing techniques are motivated within a Bayesian framework, including work by Nadas (1984) and MacKay and Peto (1995).

3 Novel Smoothing Techniques

Of the great many novel methods that we have tried, two techniques have performed especially well.

[2]When the same data is used to estimate both, setting all $\lambda_{w_{i-n+1}^{i-1}}$ to one yields the optimal result.

3.1 Method *average-count*

This scheme is an instance of Jelinek-Mercer smoothing. Referring to equation (3), recall that Bahl et al. suggest bucketing the $\lambda_{w_{i-n+1}^{i-1}}$ according to $c(w_{i-n+1}^{i-1})$. We have found that partitioning the $\lambda_{w_{i-n+1}^{i-1}}$ according to the average number of counts per non-zero element $\frac{c(w_{i-n+1}^{i-1})}{|w_i : c(w_{i-n+1}^{i})>0|}$ yields better results.

Intuitively, the less sparse the data for estimating $P_{\text{ML}}(w_i|w_{i-n+1}^{i-1})$, the larger $\lambda_{w_{i-n+1}^{i-1}}$ should be. While larger $c(w_{i-n+1}^{i-1})$ generally correspond to less sparse distributions, this quantity ignores the allocation of counts between words. For example, we would consider a distribution with ten counts distributed evenly among ten words to be much more sparse than a distribution with ten counts all on a single word. The average number of counts per word seems to more directly express the concept of sparseness.

In Figure 1, we graph the value of λ assigned to each bucket under the original and new bucketing schemes on identical data. Notice that the new bucketing scheme results in a much tighter plot, indicating that it is better at grouping together distributions with similar behavior.

3.2 Method *one-count*

This technique combines two intuitions. First, MacKay and Peto (1995) argue that a reasonable form for a smoothed distribution is

$$P_{\text{one}}(w_i|w_{i-n+1}^{i-1}) = \frac{c(w_{i-n+1}^{i}) + \alpha P_{\text{one}}(w_i|w_{i-n+2}^{i-1})}{c(w_{i-n+1}^{i-1}) + \alpha}$$

The parameter α can be thought of as the number of counts being added to the given distribution, where the new counts are distributed as in the lower-order distribution. Secondly, the Good-Turing estimate can be interpreted as stating that the number of these extra counts should be proportional to the number of words with exactly one count in the given distribution. We have found that taking

$$\alpha = \gamma \left[n_1(w_{i-n+1}^{i-1}) + \beta \right] \quad (4)$$

works well, where

$$n_1(w_{i-n+1}^{i-1}) = |w_i : c(w_{i-n+1}^{i}) = 1|$$

is the number of words with one count, and where β and γ are constants.

4 Experimental Methodology

4.1 Data

We used the Penn treebank and TIPSTER corpora distributed by the Linguistic Data Consortium. From the treebank, we extracted text from the tagged Brown corpus, yielding about one million words. From TIPSTER, we used the Associated Press (AP), Wall Street Journal (WSJ), and San Jose Mercury News (SJM) data, yielding 123, 84, and 43 million words respectively. We created two distinct vocabularies, one for the Brown corpus and one for the TIPSTER data. The former vocabulary contains all 53,850 words occurring in Brown; the latter vocabulary consists of the 65,173 words occurring at least 70 times in TIPSTER.

For each experiment, we selected three segments of held-out data along with the segment of training data. One held-out segment was used as the test data for performance evaluation, and the other two were used as development test data for optimizing the parameters of each smoothing method. Each piece of held-out data was chosen to be roughly 50,000 words. This decision does not reflect practice very well, as when the training data size is less than 50,000 words it is not realistic to have so much development test data available. However, we made this decision to prevent us having to optimize the training versus held-out data tradeoff for each data size. In addition, the development test data is used to optimize typically very few parameters, so in practice small held-out sets are generally adequate, and perhaps can be avoided altogether with techniques such as deleted estimation.

4.2 Smoothing Implementations

In this section, we discuss the details of our implementations of various smoothing techniques. Due to space limitations, these descriptions are not comprehensive; a more complete discussion is presented in Chen (1996). The titles of the following sections include the mnemonic we use to refer to the implementations in later sections. Unless otherwise specified, for those smoothing models defined recursively in terms of lower-order models, we end the recursion by taking the $n = 0$ distribution to be the uniform distribution $P_{\text{unif}}(w_i) = 1/|V|$. For each method, we highlight the parameters (*e.g.*, λ_n and δ below) that can be tuned to optimize performance. Parameter values are determined through training on held-out data.

4.2.1 Baseline Smoothing (`interp-baseline`)

For our baseline smoothing method, we use an instance of Jelinek-Mercer smoothing where we constrain all $\lambda_{w_{i-n+1}^{i-1}}$ to be equal to a single value λ_n for each n, *i.e.*,

$$\begin{aligned} P_{\text{base}}(w_i|w_{i-n+1}^{i-1}) &= \lambda_n \, P_{\text{ML}}(w_i|w_{i-n+1}^{i-1}) + \\ &\quad (1-\lambda_n) \, P_{\text{base}}(w_i|w_{i-n+2}^{i-1}) \end{aligned}$$

4.2.2 Additive Smoothing (`plus-one` and `plus-delta`)

We consider two versions of additive smoothing. Referring to equation (2), we fix $\delta = 1$ in `plus-one` smoothing. In `plus-delta`, we consider any δ.

4.2.3 Katz Smoothing (`katz`)

While the original paper (Katz, 1987) uses a single parameter k, we instead use a different k for each $n > 1$, k_n. We smooth the unigram distribution using additive smoothing with parameter δ.

4.2.4 Church-Gale Smoothing (`church-gale`)

To smooth the counts n_r needed for the Good-Turing estimate, we use the technique described by Gale and Sampson (1995). We smooth the unigram distribution using Good-Turing without any bucketing.

Instead of the bucketing scheme described in the original paper, we use a scheme analogous to the one described by Bahl, Jelinek, and Mercer (1983). We make the assumption that whether a bucket is large enough for accurate Good-Turing estimation depends on how many n-grams with non-zero counts occur in it. Thus, instead of partitioning the space of $P(w_{i-1})P(w_i)$ values in some uniform way as was done by Church and Gale, we partition the space so that at least $c_{\min}$ non-zero n-grams fall in each bucket.

Finally, the original paper describes only bigram smoothing in detail; extending this method to trigram smoothing is ambiguous. In particular, it is unclear whether to bucket trigrams according to $P(w_{i-2}^{i-1})P(w_i)$ or $P(w_{i-2}^{i-1})P(w_i|w_{i-1})$. We chose the former; while the latter may yield better performance, our belief is that it is much more difficult to implement and that it requires a great deal more computation.

4.2.5 Jelinek-Mercer Smoothing (`interp-held-out` and `interp-del-int`)

We implemented two versions of Jelinek-Mercer smoothing differing only in what data is used to train the λ's. We bucket the $\lambda_{w_{i-n+1}^{i-1}}$ according to $c(w_{i-n+1}^{i-1})$ as suggested by Bahl et al. Similar to our Church-Gale implementation, we choose buckets to ensure that at least $c_{\min}$ words in the data used to train the λ's fall in each bucket.

In `interp-held-out`, the λ's are trained using held-out interpolation on one of the development test sets. In `interp-del-int`, the λ's are trained using the *relaxed deleted interpolation* technique described by Jelinek and Mercer, where one word is deleted at a time. In `interp-del-int`, we bucket an n-gram according to its count before deletion, as this turned out to significantly improve performance.

4.2.6 Novel Smoothing Methods (`new-avg-count` and `new-one-count`)

The implementation `new-avg-count`, corresponding to smoothing method *average-count*, is identical to `interp-held-out` except that we use the novel bucketing scheme described in section 3.1. In the implementation `new-one-count`, we have different parameters β_n and γ_n in equation (4) for each n.

5 Results

In Figure 2, we display the performance of the `interp-baseline` method for bigram and trigram models on TIPSTER, Brown, and the WSJ subset of TIPSTER. In Figures 3–6, we display the relative performance of various smoothing techniques with respect to the baseline method on these corpora, as measured by difference in entropy. In the graphs on the left of Figures 2–4, each point represents an average over ten runs; the error bars represent the empirical standard deviation over these runs. Due to resource limitations, we only performed multiple runs for data sets of 50,000 sentences or less. Each point on the graphs on the right represents a single run, but we consider sizes up to the amount of data available. The graphs on the bottom of Figures 3–4 are close-ups of the graphs above, focusing on those algorithms that perform better than the baseline. To give an idea of how these cross-entropy differences translate to perplexity, each 0.014 bits correspond roughly to a 1% change in perplexity.

In each run except as noted below, optimal values for the parameters of the given technique were searched for using Powell's search algorithm as realized in *Numerical Recipes in C* (Press et al., 1988, pp. 309–317). Parameters were chosen to optimize the cross-entropy of one of the development test sets associated with the given training set. To constrain the search, we searched only those parameters that were found to affect performance significantly, as verified through preliminary experiments over several data sizes. For `katz` and `church-gale`, we did not perform the parameter search for training sets over 50,000 sentences due to resource constraints, and instead manually extrapolated parameter values from optimal values found on smaller data sizes. We ran `interp-del-int` only on sizes up to 50,000 sentences due to time constraints.

Method	Lines
`interp-baseline`[3]	400
`plus-one`	40
`plus-delta`	40
`katz`	300
`church-gale`	1000
`interp-held-out`	400
`interp-del-int`	400
`new-avg-count`	400
`new-one-count`	50

Table 1: Implementation difficulty of various methods in terms of lines of C++ code

From these graphs, we see that additive smoothing performs poorly and that methods `katz` and `interp-held-out` consistently perform well. Our implementation `church-gale` performs poorly except on large bigram training sets, where it performs the best. The novel methods `new-avg-count` and `new-one-count` perform well uniformly across training data sizes, and are superior for trigram models. Notice that while performance is relatively consistent across corpora, it varies widely with respect to training set size and n-gram order.

The method `interp-del-int` performs significantly worse than `interp-held-out`, though they differ only in the data used to train the λ's. However, we delete one word at a time in `interp-del-int`; we hypothesize that deleting larger chunks would lead to more similar performance.

In Figure 7, we show how the values of the parameters δ and $c_{\min}$ affect the performance of methods `katz` and `new-avg-count`, respectively, over several training data sizes. Notice that poor parameter setting can lead to very significant losses in performance, and that optimal parameter settings depend on training set size.

To give an informal estimate of the difficulty of implementation of each method, in Table 1 we display the number of lines of C++ code in each implementation excluding the core code common across techniques.

6 Discussion

To our knowledge, this is the first empirical comparison of smoothing techniques in language modeling of such scope: no other study has used multiple training data sizes, corpora, or has performed parameter optimization. We show that in order to completely

[3] To implement the baseline method, we just used the `interp-held-out` code as it is a special case. Written anew, it probably would have been about 50 lines.

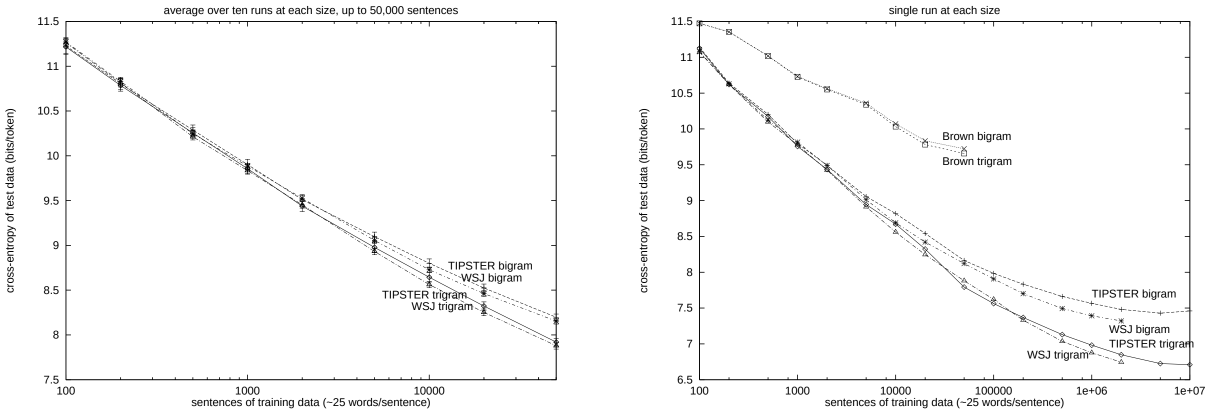

Figure 2: Baseline cross-entropy on test data; graph on left displays averages over ten runs for training sets up to 50,000 sentences, graph on right displays single runs for training sets up to 10,000,000 sentences

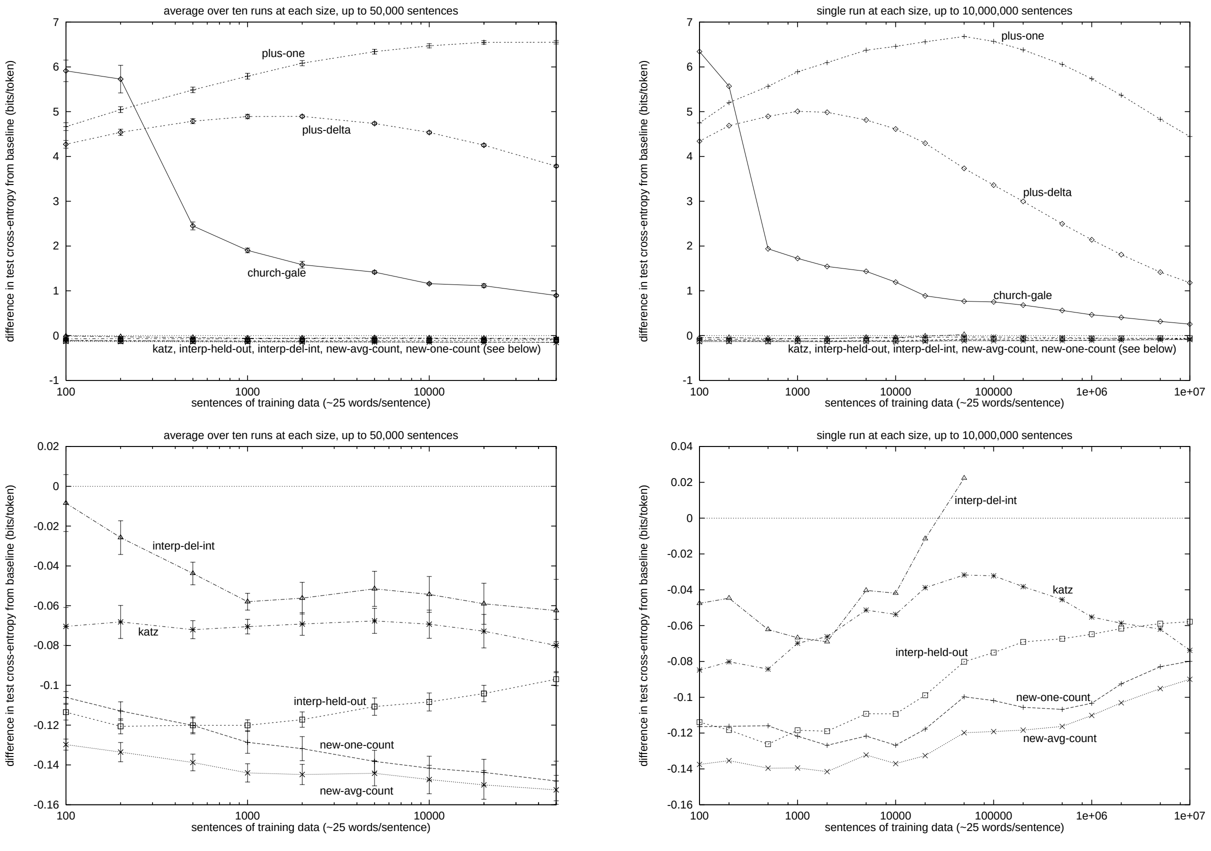

Figure 3: Trigram model on TIPSTER data; relative performance of various methods with respect to baseline; graphs on left display averages over ten runs for training sets up to 50,000 sentences, graphs on right display single runs for training sets up to 10,000,000 sentences; top graphs show all algorithms, bottom graphs zoom in on those methods that perform better than the baseline method

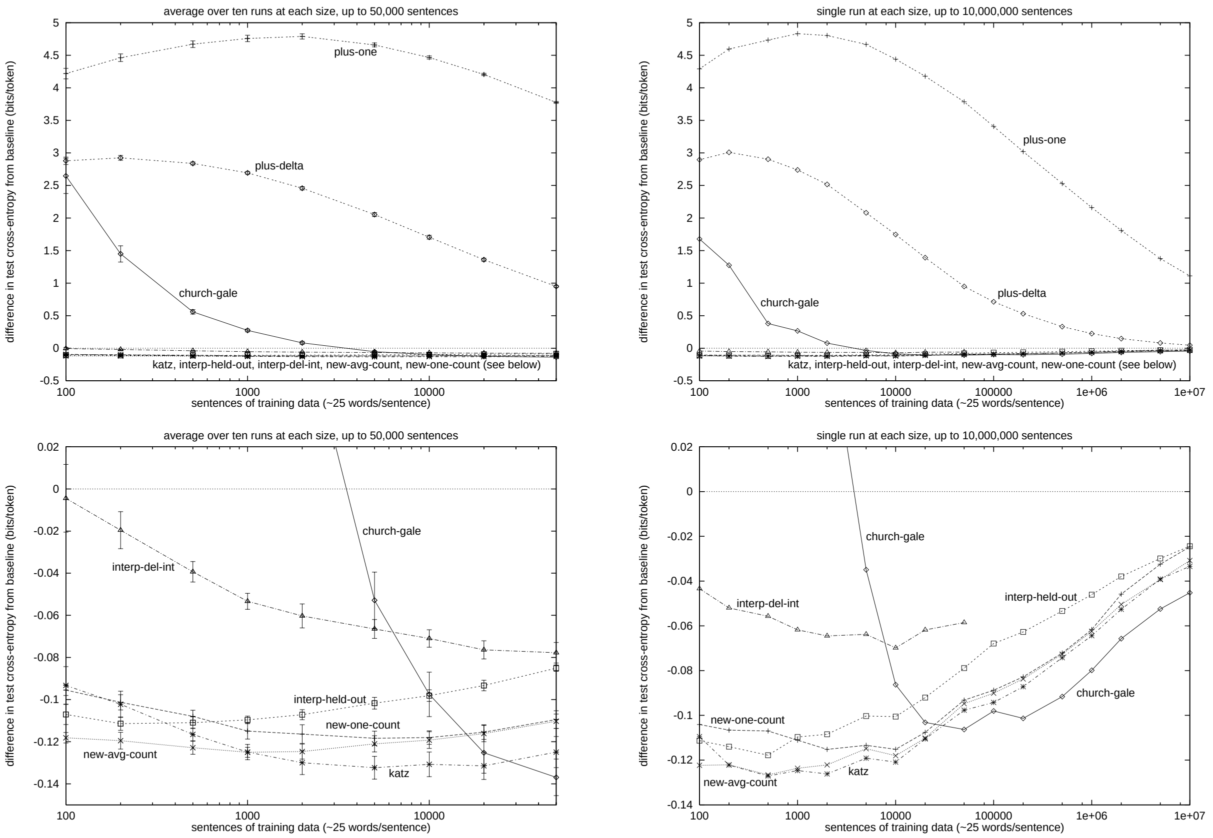

Figure 4: Bigram model on TIPSTER data; relative performance of various methods with respect to baseline; graphs on left display averages over ten runs for training sets up to 50,000 sentences, graphs on right display single runs for training sets up to 10,000,000 sentences; top graphs show all algorithms, bottom graphs zoom in on those methods that perform better than the baseline method

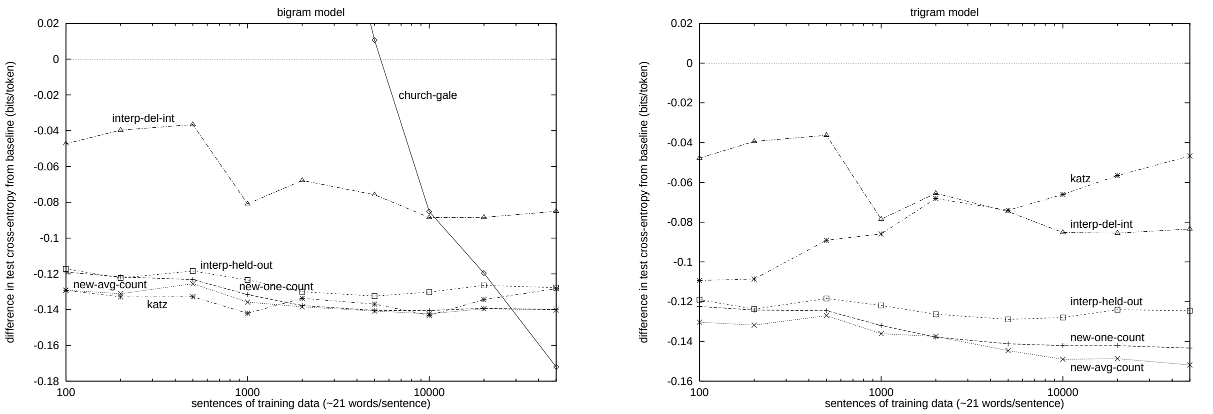

Figure 5: Bigram and trigram models on Brown corpus; relative performance of various methods with respect to baseline

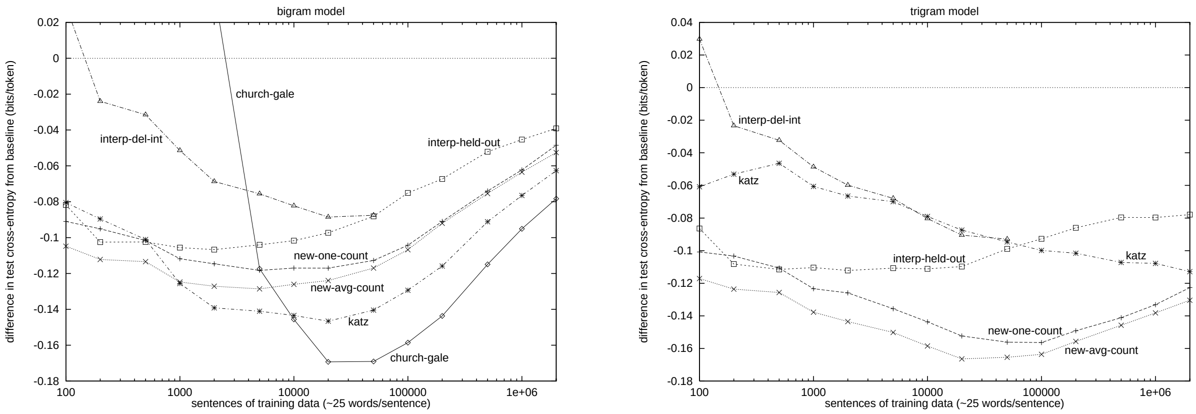

Figure 6: Bigram and trigram models on Wall Street Journal corpus; relative performance of various methods with respect to baseline

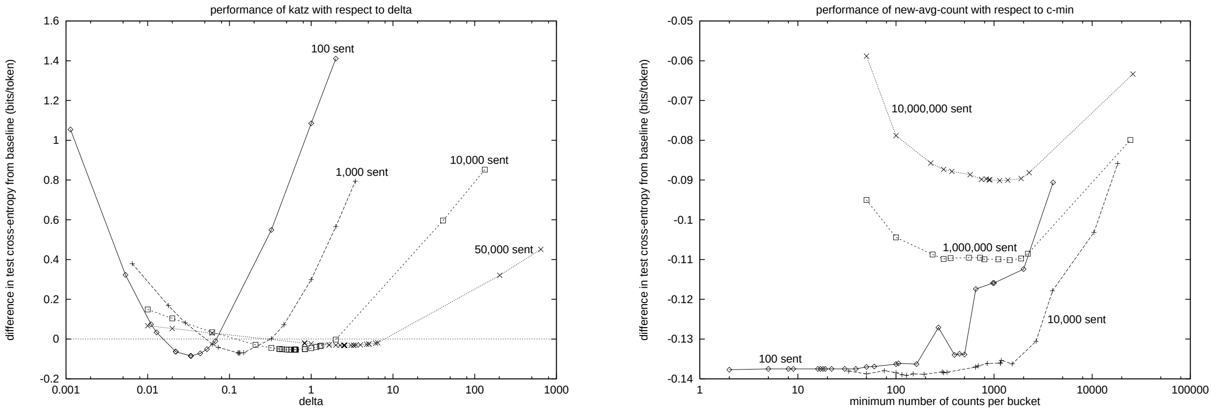

Figure 7: Performance of `katz` and `new-avg-count` with respect to parameters δ and $c_{\min}$, respectively

characterize the relative performance of two techniques, it is necessary to consider multiple training set sizes and to try both bigram and trigram models. Multiple runs should be performed whenever possible to discover whether any calculated differences are statistically significant. Furthermore, we show that sub-optimal parameter selection can also significantly affect relative performance.

We find that the two most widely used techniques, Katz smoothing and Jelinek-Mercer smoothing, perform consistently well across training set sizes for both bigram and trigram models, with Katz smoothing performing better on trigram models produced from large training sets and on bigram models in general. These results question the generality of the previous reference result concerning Katz smoothing: Katz (1987) reported that his method slightly outperforms an unspecified version of Jelinek-Mercer smoothing on a single training set of 750,000 words. Furthermore, we show that Church-Gale smoothing, which previously had not been compared with common smoothing techniques, outperforms all existing methods on bigram models produced from large training sets. Finally, we find that our novel methods *average-count* and *one-count* are superior to existing methods for trigram models and perform well on bigram models; method *one-count* yields marginally worse performance but is extremely easy to implement.

In this study, we measure performance solely through the cross-entropy of test data; it would be interesting to see how these cross-entropy differences correlate with performance in end applications such as speech recognition. In addition, it would be interesting to see whether these results extend to fields other than language modeling where smoothing is used, such as prepositional phrase attachment (Collins and Brooks, 1995), part-of-speech tagging (Church, 1988), and stochastic parsing (Magerman, 1994).

Acknowledgements

The authors would like to thank Stuart Shieber and the anonymous reviewers for their comments on previous versions of this paper. We would also like to thank William Gale and Geoffrey Sampson for supplying us with code for "Good-Turing frequency estimation without tears." This research was supported by the National Science Foundation under Grant No. IRI-93-50192 and Grant No. CDA-94-01024. The second author was also supported by a National Science Foundation Graduate Student Fellowship.

References

Bahl, Lalit R., Frederick Jelinek, and Robert L. Mercer. 1983. A maximum likelihood approach to continuous speech recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-5(2):179–190, March.

Brown, Peter F., John Cocke, Stephen A. DellaPietra, Vincent J. DellaPietra, Frederick Jelinek, John D. Lafferty, Robert L. Mercer, and Paul S. Roossin. 1990. A statistical approach to machine translation. *Computational Linguistics*, 16(2):79–85, June.

Brown, Peter F., Stephen A. DellaPietra, Vincent J. DellaPietra, Jennifer C. Lai, and Robert L. Mercer. 1992. An estimate of an upper bound for the entropy of English. *Computational Linguistics*, 18(1):31–40, March.

Chen, Stanley F. 1996. *Building Probabilistic Models for Natural Language*. Ph.D. thesis, Harvard University. In preparation.

Church, Kenneth. 1988. A stochastic parts program and noun phrase parser for unrestricted text. In *Proceedings of the Second Conference on Applied Natural Language Processing*, pages 136–143.

Church, Kenneth W. and William 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, Michael and James Brooks. 1995. Prepositional phrase attachment through a backed-off model. In David Yarowsky and Kenneth Church, editors, *Proceedings of the Third Workshop on Very Large Corpora*, pages 27–38, Cambridge, MA, June.

Gale, William A. and Kenneth W. Church. 1990. Estimation procedures for language context: poor estimates are worse than none. In *COMPSTAT, Proceedings in Computational Statistics, 9th Symposium*, pages 69–74, Dubrovnik, Yugoslavia, September.

Gale, William A. and Kenneth W. Church. 1994. What's wrong with adding one? In N. Oostdijk and P. de Haan, editors, *Corpus-Based Research into Language*. Rodolpi, Amsterdam.

Gale, William A. and Geoffrey Sampson. 1995. Good-Turing frequency estimation without tears. *Journal of Quantitative Linguistics*, 2(3). To appear.

Good, I.J. 1953. The population frequencies of species and the estimation of population parameters. *Biometrika*, 40(3 and 4):237–264.

Jeffreys, H. 1948. *Theory of Probability*. Clarendon Press, Oxford, second edition.

Jelinek, Frederick and Robert L. Mercer. 1980. Interpolated estimation of Markov source parameters from sparse data. In *Proceedings of the Workshop on Pattern Recognition in Practice*, Amsterdam, The Netherlands: North-Holland, May.

Johnson, W.E. 1932. Probability: deductive and inductive problems. *Mind*, 41:421–423.

Katz, Slava M. 1987. 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, March.

Kernighan, M.D., K.W. Church, and W.A. Gale. 1990. A spelling correction program based on a noisy channel model. In *Proceedings of the Thirteenth International Conference on Computational Linguistics*, pages 205–210.

Lidstone, G.J. 1920. Note on the general case of the Bayes-Laplace formula for inductive or *a posteriori* probabilities. *Transactions of the Faculty of Actuaries*, 8:182–192.

MacKay, David J. C. and Linda C. Peto. 1995. A hierarchical Dirichlet language model. *Natural Language Engineering*, 1(3):1–19.

Magerman, David M. 1994. *Natural Language Parsing as Statistical Pattern Recognition*. Ph.D. thesis, Stanford University, February.

Nadas, Arthur. 1984. Estimation of probabilities in the language model of the IBM speech recognition system. *IEEE Transactions on Acoustics, Speech and Signal Processing*, ASSP-32(4):859–861, August.

Press, W.H., B.P. Flannery, S.A. Teukolsky, and W.T. Vetterling. 1988. *Numerical Recipes in C*. Cambridge University Press, Cambridge.