# ON THE RELATION BETWEEN ADDITIVE SMOOTHING AND UNIVERSAL CODING

Nikola Jevtić Alon Orlitsky

ECE Department, UCSD 9500 Gilman Drive La Jolla, CA 92093, USA

# ABSTRACT

We analyze the performance of smoothing methods for language modeling from the perspective of universal compression. We use existing asymptotic bounds on the performance of simple additive rules for compression of finite-alphabet memoryless sources to explain the empirical predictive abilities of additive smoothing techniques. We further suggest a smoothing method that overcomes some of the problems observed in previous approaches. The new method outperforms existing ones on the Wall Street Journal(WSJ) database for bigram and trigram models. We then suggest possible directions for future research.

# 1. INTRODUCTION

Language modeling is an essential component of many expert systems such as speech recognition, handwriting recognition, and language translation. The performance of these systems depends on the quality of underlying language models. Yet we do not have good ways of evaluating how close current modeling techniques are to the optimal.

Our goal is to reduce the gap between the theoretical foundations of universal compression, capable of producing tight performance bounds for underlying models, and practical techniques used in language modeling. We also build on a paper by Chen and Goodman [1] that first compared many of the smoothing techniques, and our experiments use similar methodologies.

A sequential language model is a probability distribution over sentences,  $w_0^l = w_0 \dots w_l$ . We assume that  $w_0$  is a symbol for the beginning of a sentence consisting of l - 1 words and  $w_l$  is a symbol for the end of the sentence. The most popular approach to language modeling for speech recognition applications uses the *n*gram approximation, which assumes that probability distribution for the current word depends only on the previous n - 1 words

$$p(w_1^l|w_0) = \prod_{i=1}^l p(w_i|w_0^{i-1}) \approx \prod_{i=1}^l p(w_i|w_{i-n+1}^{i-1}).$$

Under these assumptions, the problem of creating a language model is reduced to that of learning a Markov source of order n - 1. Observe that the probability for the end of sentence symbol is evaluated at the end of the sentence as if it were another word, while the beginning of the sentence is treated just as a history information.

Calling different histories *contexts*, it is common to view a Markov source as a collection of i.i.d. sources, one for each context. In creating the language model it is therefore desirable to derive optimal probability estimates for each context. The main difficulty in deriving these estimates arises from the sparsity of

the training data. Since most words are never or infrequently observed, their estimates are unreliable.

A text is a collection of sentences and its probability is the product of the individual sentence probabilities. Let  $W_T$  denote the number of words in a text T. To measure the quality of a language model on T, we use the *per-word coding length*, or *cross-entropy*,

$$H_p(T) \stackrel{\text{def}}{=} -\frac{1}{W_T} \log p(T).$$

Note that if a word appearing in a text is assigned zero probability, the resulting coding length is infinite. Similarly, in a word recognizer, words that are assigned zero probability can never be recognized, regardless of how likely they are acoustically. Hence many smoothing techniques were introduced to ensure that all words, even those that did not appear in the training data, are assigned positive probability.

Among the first approaches to this *zero frequency problem* was *additive* smoothing that dates back to Laplace [2]. Given a set of symbols V, let c(u) denote the number of times symbol  $u \in V$  was generated by an i.i.d. source whose distribution we are trying to learn. These estimators then assign to each symbol  $w \in V$  the probability

$$p_{add}(w) \stackrel{\text{def}}{=} \frac{c(w) + \delta}{\sum_{u \in V} (c(u) + \delta)}.$$
 (1)

When  $\delta = 1$ , this equation is known as the Laplace's law of succession, and presents the posterior probability of symbol w, if we assume uniform prior probability over all possible sample count ensembles, see, e.g., [3, 4]. When  $\delta \neq 1$ , Equation (1) represents Lidstone's law, often used to estimate probabilities in sparse data. These probability estimation methods are also considered in universal compression, where their performance can be precisely analyzed which will be the main focus of this paper. We interpret empirical results of additive techniques used for language modeling with these analytic tools, and provide further insight in the complex behavior of language.

In Section 2 we define the universal coding problem and state the relevant results and bounds on the performance of the additive smoothing. We also emphasize the implications of these results to the language modeling problem. In Section 3 we describe some of the standard approaches to language modeling. In Section 4 we introduce a particularly informative language model that avoids the deficiencies of additive smoothing. In Section 5 we plot performance of the language models addressed in this paper in the same fashion as used in [1]. In Section 6 we summarize the results and present some open problems.

#### 2. UNIVERSAL CODING

A coder assigns a codeword to each sequence. When the underlying source distribution p is known, an optimal coder assigns to each sequence of n symbols  $w^n$  a codeword of length  $-\log p(w^n)$ .

Universal coding refers to compression where the only available information about the source is that it belongs to a known set of distributions  $\mathcal{P}$  over some alphabet  $\mathcal{A}$ . Still we would like to assign to each sequence a codeword whose length is close to that assigned by an optimal coder, that knows the underlying distribution.

#### 2.1. Redundancy

The *redundancy* of a coder is the largest number of extra number of bits it assigns to any sequence over any encoder in  $\mathcal{P}$ , including the one corresponding to the underlying distribution.

Formally, the *maximum-likelihood probability* of a sequence  $w^n$  is

$$\hat{p}(w^n) \stackrel{\text{def}}{=} \max\{p(w^n) : p \in \mathcal{P}\}\$$

the highest probability assigned to  $w^n$  by any distribution in  $\mathcal{P}$ . The *worst* case redundancy (loss) of  $\mathcal{P}$  is therefore

$$\hat{R}_n \stackrel{\text{def}}{=} \min_{q_n} \max_{w^n} \log \frac{\hat{p}(w^n)}{q_n(w^n)}$$

and the *average* case redundancy of  $\mathcal{P}$  is

$$\bar{R}_n \stackrel{\text{def}}{=} \min_{q_n} \max_{p \in \mathcal{P}} E_p \log \frac{p(w^n)}{q_n(w^n)}$$
$$= \min_{q_n} \max_{p \in \mathcal{P}} \sum_{w^n} p(w^n) \log \frac{p(w^n)}{q_n(w^n)}$$

We say that arbitrary coder  $q_n$  is universal in the worst (average) case sense, with respect to the set of sources  $\mathcal{P}$ , if its per symbol redundancy diminishes with the block size.

We limit  $\mathcal{P}$  to be the set of all i.i.d. distributions over an alphabet of finite cardinality V. In this setting, the worst case redundancy is analyzed in [5],

$$\hat{R}_n = \frac{V-1}{2}\log\frac{n}{2\pi} + C_V + o(1)$$

and the average case redundancy, in [6] where it was shown that

$$\bar{R}_n = \frac{V-1}{2}\log\frac{n}{2\pi e} + C_V + o(1),$$

where

$$C_V = \log \frac{\Gamma(\frac{1}{2})^V}{\Gamma(\frac{V}{2})}.$$

Note that the two redundancies differ only by a constant, and the results obtained for the two measures are often very similar. In this paper we concentrate on the worst case redundancy as it is somewhat easier to analyze.

#### 2.2. Sequential coding

Achieving the redundancy of i.i.d. sources requires computationally infeasible coders that encode the whole block of symbols at once. Often it is desirable to encode the symbols one at a time. Such is the case in language modeling where we cannot wait to gather large enough blocks for each context before assigning probabilities.

To encode (assign probability to) each symbol as it arrives, suboptimal sequential schemes are evaluated and used. Some of them are shown to be asymptotically optimal as block sizes increase, but there are no bounds on any particular symbol. However, if a method predicts a block well, it must predict well most of the symbols inside.

A special case of sequential codes are *additive-rule* codes defined by Equation (1). It can be shown that all additive-rule codes correspond to setting a Dirichlet( $\underline{\delta}$ ) prior over all i.i.d. distributions, so that after observing the whole block, the aggregate probability  $m_{\delta}(w^n)$  is equal to the average source probability with respect to the given Dirichlet( $\underline{\delta}$ ) prior. Unlike Laplace's approach, we are not left to wonder if such a prior is reasonable, but instead concentrate on redundancy bounds to determine if the method is better or worse than another. In the following we present several approaches used for language model estimation and discuss their block performance bounds.

#### 2.3. Add-one rule

The add-one rule uses the Laplace's law of succession for estimating the probability of the next word. It was one of the first methods employed in language modeling, but Church and Gale [7] showed experimentally that it had poor performance. It is easy to show that the worst-case redundancy of the add-one rule satisfies

$$\hat{R}_n(m_1) = \max_{w^n} \log \frac{\hat{p}(w^n)}{m_1(w^n)} \ge (V-1)\log n + O(1),$$

hence, for the worst source, the leading term in the redundancy bound is at least twice that of the best universal predictor.

#### 2.4. Add-half rule

The add-half rule, also known as the Krichevsky-Trofimov estimator [8], is often suggested in the compression literature as the best universal sequential compression method for finite-alphabets. The average redundancy of the add-half rule asymptotically converges to the optimal value if all the symbol probabilities of the unknown source are bounded away from zero. In terms of the worst case problem formulation, the redundancy converges to the optimal if all the symbol counts grow approximately linearly with the sequence length. If that is not the case, worst case redundancy increases, and at the vertex point in probability space, it achieves [5]:

$$R_n(m_{\frac{1}{2}}, 1^n) = \log \frac{\hat{p}(1^n)}{m_{\frac{1}{2}}(1^n)} = \frac{V-1}{2} \log \frac{n}{\pi} + C_V + o(1).$$

This redundancy overhead may appear to be small, as the incurred loss is only  $\frac{V-1}{2}$  bits higher than that of the best universal code. However, if it were known that only a known subset  $\mathcal{B} \subset \mathcal{A}$  has non-zero probabilities,  $|\mathcal{B}| = b$ , the target performance would be  $\hat{R}_n(m_{\frac{1}{2}}) = \frac{b-1}{2} \log \frac{n}{2\pi} + C_b + o(1)$ , yielding the overhead of the add-half rule of  $\frac{V-b}{2} \log n + O(1)$  over a method that could determine the active alphabet. The add-half rule effectively spends too many bits on the nonexisting symbols and the impossible sequences. In speech, most of the contexts can be followed by only a small subset of the vocabulary ( $b \ll V$ ) and thus it is not reasonable to expect good performance of this method.

#### 2.5. Add-small-delta rule

The method that performed better than the add-one rule on the language modeling task is the add-small-delta rule. Using asymptotic analysis similar to [5], we can show that the block redundancy of the add- $\delta$  rule is upper-bounded by

$$\hat{R}_n(m_\delta) \leq \frac{V-1}{2} \log \frac{n}{2\pi} + C_{V,\delta} + \frac{V\delta^2 \log e}{2} \\ -(\frac{1}{2} - \delta)V \log V + o(1),$$

where

$$C_{V,\delta} = \log \frac{\Gamma(\delta)^V}{\Gamma(V\delta)}.$$

Note that the leading term in the redundancy does not increase compared to the best universal code, although the constant is somewhat changed. Hence the method is never much worse than the add-half rule.

The important property of add-delta rule was shown in [5]; if a particular symbol appears rarely, the method is able to outperform the best universal code. More rigorously, let  $T_1(w^n)$  be the number of times symbol 1 appeared in  $w^n$ . If exists  $p \in (0,1)$ s.t.  $\forall n \quad T_1(w^n) < n^p$ , then the redundancy while encoding sequence  $w^n$  with the add- $\delta$  rule can be bounded by:

$$\log \frac{\hat{p}(w^n)}{m_{\delta}(w^n)} \leq \left(\frac{V-1}{2} - \left(\frac{1}{2} - \delta\right)(1-p)\right)\log n + C_{V,\delta} + \frac{V(V+1)}{4}\log e + o(1).$$

This means that the add-small-delta rule would be able to remove log *n* terms corresponding to all nonexisting words if  $\delta$  were close enough to zero. However the limit of  $C_{V,\delta}$  when  $\delta$  approaches zero,  $\lim_{\delta \to 0} C_{V,\delta} = +\infty$ , and that prohibits us from completely removing the redundancy caused by those words.

The approach taken in language modeling was to try and find the best  $\delta$  that would maximize the probability scores on some held-out part of the data. The method effectively balanced the removal of redundancy incurred by nonexisting words and a high penalty for a newly observed word. However, due to its low flexibility it had much worse performance compared to the standard approaches in language modeling.

#### 3. POPULAR LANGUAGE MODELS

In general, add-small-delta rule and add-half rule have strong compression ability when all the words in the vocabulary have nonzero probabilities. However they perform poorly (add-small-delta less so) when there are words with diminishing probabilities. To deal with the sparse observations and low probability words, most popular language models use the concept of backing-off, using the observation that it might be wise not to treat all the unobserved words equally in the context. Instead, to redistribute the probability, the distribution from the broader context is used, which has more training data and which is referred to as the backoff context. In terms of n-gram language models, from context with memory of n-1 most recent words, they back off to a context with memory of n-2 most recent words. The recursion could end at uniform distribution, so that in the end all the words get assigned some probability, even if they were completely unobserved in the training data.

Chen and Goodman [1] distinguish two implementations of backoff, the *strict* and the *interpolated* and conclude that interpolated back-offs outperform the strict ones. In this paper we only consider the interpolated variant:

$$p(w_i|w_{i-n+1}^{i-1}) = \overline{\lambda} \cdot p_0(w_i|w_{i-n+1}^{i-1}) + \lambda \cdot p(w_i|w_{i-n+2}^{i-1}),$$

where  $\lambda$  is an interpolation parameter and  $c(w_{i-n+1}^i)$  represents how many times a string  $w_{i-n+1}^i$  has been observed in the training. There are several methods for balancing the full context distribution and its back-off, most described in [1], and we present a few of the most popular ones. This is not the full overview, as e.g. a whole class of Good-Turing based methods are not covered, but it includes the state-of-the-art models. It is interesting to mention that the Good-Turing method is a universal compression method for unknown vocabularies[9], in a problem setting different than the one considered here.

#### 3.1. Jelinek-Mercer model

Jelinek and Mercer [10] have described a general class of n-gram models that interpolate different memory Markov sources.

$$p(w_i|w_{i-n+1}^{i-1}) = \sum_{j=0}^{n-1} \lambda_j \cdot p_{ML}(w_i|w_{i-j}^{i-1}),$$

where  $\sum_{j=0}^{n-1} \lambda_j = 1$ ,  $p_{ML}(w_i|w_{i-n+1}^{i-1}) = \frac{c(w_{i-n+1}^i)}{c(w_{i-n+1}^{i-1})}$ . Training data is split into two nonoverlapping subsets, the *retained* and the *held-out* set. Maximum likelihood on the data from the retained set is used as an estimate for probabilities assigned at each level  $(p_{ML})$ , and the interpolation parameters  $(\lambda)$  are optimized to maximize probability on the held-out set. The performance of the Jelinek-Mercer smoothing was poor if the interpolation parameters were uniform over all contexts. However, separate estimation for every context was not possible due to a lack of data. A particularly useful variant defined interpolation as hierarchical [11].

$$p_{interp}(w_i|w_{i-n+1}^{i-1}) = \lambda_{w_{i-n+1}^{i-1}} p_{ML}(w_i|w_{i-n+1}^{i-1}) + (1 - \lambda_{w_{i-n+1}^{i-1}}) p_{interp}(w_i|w_{i-n+2}^{i-1}).$$

The hierarchical definition made it possible to cluster  $\lambda$ 's for several similar contexts at every level separately, and then jointly estimate the optimal values through the Expectation Maximization (EM) procedure. The original criterion used for clustering in [11] was the number of times the context was observed (total count). The assumption was that more frequent contexts would be better estimated and trusted with higher confidence. The critical parameter that had to be chosen is the number of contexts that are clustered together, or the number of free interpolation parameters that should be estimated, and it has been observed that these values depend on the size of the training data. Chen shows in his thesis [12] that it is better to use average word count as a criterion for clustering. This criterion is also much less sensitive to the cluster size change, compared to the total count criterion.

### 3.2. Linear and Absolute Discounting

Ney, Essen and Kneser [13, 14] argued that all the words in longer contexts are oversampled and that there could be two general ways

of discounting (or reducing their probabilities) to share the probability with the unobserved words through back-off: *linear* and *absolute discounting*. In the linear discounting the maximum likelihood probabilities in contexts are discounted proportionally to the probabilities (scaled) and the total discounted probability is given to the back-off context (which actually corresponds to the Jelinek-Mercer smoothing with single cluster). In the absolute discounting all words are discounted by the equal additive constant:

$$p_{abs}(w_i|w_{i-n+1}^{i-1}) = \begin{cases} \frac{c(w_{i-n+1}^{i})-D}{c(w_{i-n+1}^{i-1})}, & \text{if } c(w_{i-n+1}^{i}) > 0\\ \frac{D\cdot N_1 + (w_{i-n+1}^{i-1})}{c(w_{i-n+1}^{i-1})} p_{abs}(w_i|w_{i-n+2}^{i-1}), \text{o.w.} \end{cases}$$

$$(2)$$

It is assumed that 0 < D < 1 and  $N_{1+}(w_{i-n+1}^{i-1} \cdot)$  is the number of different words that have been observed once or more times following  $w_{i-n+1}^{i-1}$ . They showed that absolute discounting (equation 2) performs better than linear discounting. However, when context clustering is used for linear discounting, the performance is about the same.

# 3.3. Kneser-Ney model

Kneser and Ney [15] took the absolute discounting model a step further. They put a constraint on the back-off distribution, forcing the marginals of a higher order distribution to match the marginals of the training data,

$$\sum_{w_{i-n+1}} p_{KN}(w_{i-n+1}^{i}) = \frac{c(w_{i-n+2}^{i})}{N}.$$
(3)

The resulting back-off distribution was proportional, not to the number of times a word has been observed in a context, but to the number of different contexts it was observed in.

$$p_{KN}(w_i|w_{i-n+2}^{i-1}) = \frac{N_{1+}(\cdot w_{i-n+2}^{i-1})}{N_{1+}(\cdot w_{i-n+2}^{i-1})}$$

This produced big improvement in the coding lengths of unseen data, and it is not clear why did this occur. In general, a language model that has marginals that do not match the correct probabilities of the source will diverge arbitrarily over time [16]. In the given analysis however, n-gram frequencies are used instead of the correct marginals which are unknown. We feel that deeper understanding of the reasons for this effect is needed.

## 3.4. Variations of the Kneser-Ney model

Ney *et al.* [17] have suggested a variation of absolute discounting that used two discounts,  $D_1$  for symbols observed once and  $D_{2+}$  for those observed two or more times. They reported mixed results compared to the single constant case.

Chen and Goodman in [1] show that three constants  $D_1$ ,  $D_2$ and  $D_{3+}$  consistently outperform single constant model over variety of corpora and for a variety of training set sizes. The Kneser-Ney language model with the three discounting parameters is the best known model under the the n-gram constraint.

#### 4. THE INTERPOLATED ADDITIVE MODEL

In Section 2 we observed a class of additive models that share a similar flaw – they waste too many bits on nonexisting words

when the actual alphabet is smaller than expected. We propose an *interpolated additive* (IA) model that tries to address that issue by using the additive rule only to estimate the probability distribution among the observed words in the context. The assumption is that observed words would account for most of the probability and asymptotically optimal estimate among them would lead to a better probability estimation with less data.

For each context  $w_{i-n+1}^{i-1}$ , among words observed in the training data we use additive constant  $\delta \in (-1, +\infty)$  to smooth the distribution.

$$p(w_i|w_{i-n+1}^{i-1}) = \begin{cases} \frac{c(w_{i-n+1}^i)+\delta}{c(w_{i-n+1}^{i-1})+N_{1+}(w_{i-n+1}^{i-1}\cdot)\delta}, & c(w_{i-n+1}^i) > 0\\ 0, & c(w_{i-n+1}^i) = 0 \end{cases}$$

To address the probabilities of the unobserved words, the interpolated Jelinek-Mercer approach is used. The recursion ends at the uniform distribution.

$$p_{opt}(w_i|w_{i-n+1}^{i-1}) = \lambda_{w_{i-n+1}^{i-1}} p(w_i|w_{i-n+1}^{i-1}) + (1 - \lambda_{w_{i-n+1}^{i-1}}) p_{opt}(w_i|w_{i-n+2}^{i-1}).$$

This formulation allows us to use the EM algorithm to estimate  $\lambda$  parameters over different levels in the same way it is used in the Jelinek-Mercer model. The optimal additive constants are estimated as an internal part of the EM procedure as well. In the expectation step, the contribution for each held-out set sample in each n-gram level is calculated, and in the maximization step an optimization procedure is run to determine the best additive constants. As there is no closed form solution, iterative search is conducted, but fortunately EM removes the dependency among the additive constants at different n-gram levels, and instead of the Powell's method used in [1] for optimizing Kneser-Ney model much simpler one dimensional search is used. Additive constants were allowed to take both the positive and the negative values.

While all the universal methods suggest adding a constant to get the good estimate of the probability, absolute discounting achieves surprisingly good performance while subtracting a constant. At best, this result is surprising.

The IA model allows us to test whether the optimal additive strategy for smoothing probabilities among the observed words is to add or to subtract. Unlike absolute discounting and Kneser-Ney whose only way to give some probability to the the backoff is by definition through subtracting, the IA model uses other independent methods for supplying the backoff probability while leaving additive constant(s) unconstrained. Thus we can observe whether it is better to subtract or to add as our guess motivated by the previous analysis might be.

### 5. TESTING METHODOLOGY AND RESULTS

Similar to [1] we compare smoothing techniques over variety of training data sizes. However, we chose not to train language models with retained data sizes much smaller than held-out data size, which is kept fixed for all the tests. Two held-out sets and a test set size are both fixed at 2500 sentences. For each experiment, the new nonoverlapping held-out, test, and training sets were selected at random from the WSJ corpus. Segmentation was deliberately made non-contiguous to avoid short-span phenomena and to obtain the average performance, independent of set contents. The vocabulary was reduced to the 20000 most frequent words in the



Trigram performance relative to JM baseline -0.05 Jelinek-Merce Absolute discounting -0 Kneser-Ney Kneser-Ney modified IA model, std. backoff IA model, K–N backoff -0. Relative coding length -0.1 -0 -0.25 -0.3 10 10 10 Training set size (sentences)

Fig. 1. Relative coding length per word for bigram models

database. To further reduce the vocabulary (and its impact on the required memory for the system), the symbol "'s" was removed from the end of words and was considered as a separate word. This was not done in [1] but is common in speech compression. All outof-vocabulary words were mapped to a special symbol, and were treated as a single word. End-of-sentence symbol was added at the end of each sentence and its probability was also estimated. The first word of every sentence was considered to come out of the begin-of-sentence context, and no cross-sentence correlation was kept.

For methods that require clustering of the interpolation parameters (Jelinek-Mercer and Interpolated Additive model), models can be built for several cluster sizes and the model that scores best on the second held-out set can be chosen. This is a slightly inconvenient procedure because the computation of the IA model takes much more time than e.g. Jelinek-Mercer smoothing. However, the clustering results obtained through the average count are very stable over a large span of cluster sizes, and it was enough to train the Jelinek-Mercer models for several values of bigram and trigram cluster sizes and use the best results on the IA model as well.

In the typical implementations, n-gram contexts are kept in hashtables for the fast access. To be able to handle some of the experiments, in [1] the authors reduce the language model data structures only to those contexts that are relevant to the training and testing. However, the training procedures involve a relatively small held-out set. It is therefore more efficient to simply create a linked list with tokens containing only the necessary statistics for the computations involved with each word from the held-out set. The language model reduction can then be omitted. The linked-list access to data is much faster, and compared to hashtable implementation (even when the reduced set of contexts is used), we experienced over ten-fold increase in the training speed for the class of the absolute discounting models.

The results of tests are shown in Figures 1 and 2. Relative difference in average coding length of the test set was plotted against the training set size. All plots show relative performance of the

Fig. 2. Relative coding length per word for trigram models

smoothing techniques compared to the Jelinek-Mercer baseline method with a single interpolation constant for every n-gram level. We test IA technique both in configuration with the standard and the Kneser-Ney back-off. Similarly to modified Kneser-Ney model, there is no theoretical justification for using Kneser-Ney backoff. Still, the perplexity of the model is reduced significantly.

The interpolated additive model outperforms (slightly) the modified Kneser-Ney model, when in configuration with Kneser-Ney style back-off. A significance of this result increases when we observe the time needed to train these models. This time is given in the Table 1, and is represented by the average number of passes through the data during the training. <sup>1</sup> Convergence criterion used was not the standard Euclidian distance in the parameter space, often default for the Powell's method, but the improvement in the iteration round, more akin to the EM type algorithms. The common precision used for termination in all algorithms was  $10^{-4}$ bits/word.

It has been argued in [16] that the modified Kneser-Ney model takes too long to train, especially with longer-span language models. The Powell's algorithm, used in both the standard and modified Kneser-Ney algorithms, runs in time roughly quadratic in the number of estimation parameters. Thus the addition of n-gram levels leads to poor scalability. Because of this in the tests presented in [16], the authors use the standard Kneser-Ney smoothing.

The interpolated additive model promises better scalability, since the complexity does not increase with the addition of n-gram levels. Each additional additive constant is added for its individual estimation independent of other additive constants in the maximization phases of the EM.

The most interesting observation that is not shown on the plots is that the optimal bias (the additive smoothing constant) for the proposed model was always negative, as estimated by the optimization on the held-out set. This matches the assumption in the absolute discounting model that a constant amount should be sub-

<sup>&</sup>lt;sup>1</sup>The IA and the absolute discounting model have almost equivalent operations in a pass through the held-out set.

Model	Bigram	Trigram
Jelinek-Mercer	5.8	6.5
Kneser-Ney	25.17	54.5
interpolated additive	48.83	58.75
modified Kneser-Ney	110.0	201.83

Table 1. Average number of passes through the training data

tracted from all the word counts in the context. It can be shown that subtracting is asymptotically suboptimal if all the active symbols have linear lower-bounds for their count as a function of sample size (in other words if their probabilities are bounded away from zero). However, if there were elements that grow sublinearly in the sequence, subtracting strategy could outperform even the universally optimal strategy for the given alphabet. The apparent existence of such symbols suggests that language can not be represented as a static i.i.d source.

### 6. DISCUSSION

We presented the most popular techniques for language modeling, along with some early additive smoothing approaches. We also presented results obtained through asymptotic analysis of the additive methods and showed that they do not achieve optimal redundancies when the alphabet is unknown. To address this problem, we proposed the IA method that learns the optimal bias from the data and also learns the vocabulary through backing off. The results show that IA outperforms both the absolute discounting and Jelinek-Mercer model. IA with Kneser-Ney style back-off outperforms, albeit by a smaller margin, the Kneser-Ney and the modified Kneser-Ney methods.

We believe that the main contribution of this paper is the added insight to the non static nature of language. Similar evidence that language can not be modeled properly by a static model was also offered e.g. through successful use of cache based language models [18]. We have shown that only the existence of sublinear word sequences in a context would justify count discounting asymptotically. It is up to the future research to identify words responsible for those effects, if they all come from temporal phenomena, are topic dependent, etc. We believe that additional perplexity reduction can be achieved by discounting those words. Note that the current rule of thumb for improving the quality of the language model was to add more training data. Although by doing so we indeed improve the model, we inevitably observe more irrelevant words, an issue that we think should be addressed.

We also believe that the language models used in speech recognition should be evaluated dynamically, with online acquisition of training data similar to the ongoing conversation and on the fly parameter smoothing. This is also where we see the applicability of the proposed IA model where scalability and fast adaptation would play important roles.

### 7. REFERENCES

- S. F. Chen and J. Goodman. An empirical study of smoothing techniques for language modeling. *Computer Speech and Language*, 13:359–394, 1999.
- [2] P. S. de Laplace. *Essay Philosophique sur la Probabilités*. Courcier Imprimeur, Paris, 1816.

- [3] H. Jeffreys. Theory of Probability. Clarendon, Oxford, 1939.
- [4] I. H. Witten and T. C. Bell. The zero frequency problem: Estimating the probabilities of novel events in adaptive text compression. *IEEE Transactions on Information Theory*, 37(4):1085–94, July 1991.
- [5] Q. Xie and A. R. Barron. Asymptotic minimax regret for data compression, gambling and prediction. *IEEE Transactions* on *Information Theory*, 46(2):431–445, March 2000.
- [6] Q. Xie and A. R. Barron. Minimax redundancy for the class of memoryless sources. *IEEE Transactions on Information Theory*, 43(2):646–657, March 1997.
- [7] W. A. Gale and K. W. Church. What's wrong with adding one. Corpus-Based Research Into Language (Oosdijk, N. and de Haan, P., eds), 1994.
- [8] R. E. Krichevsky and V. K. Trofimov. The performance of universal encoding. *IEEE Transactions on Information The*ory, 27(2):199–207, March 1981.
- [9] A. Orlitsky, N. Santhanam, and J. Zhang. Always good turing: Asymptotically optimal probability estimation. *International Journal of Foundations of Computer Science*, to appear in October 2003.
- [10] F. Jelinek and R. L. Mercer. Interpolated estimation of markov source parameters from sparse data. *Proceedings* of the Workshop on Pattern Recognition in Practice, pages 381–397, May 1980.
- [11] P. F. Brown, S. A. Della Pietra, V. J. Della Pietra, J. C. Lai, and R. L. Mercer. An estimate of an upper bound for the entropy of english. *Computational Linguistics*, 18:31–40, 1992.
- [12] S. F. Chen. Building Probabilistic Models for Natural Language. PhD Thesis, 1996.
- [13] H. Ney and U. Essen. On smoothing techniques for bigrambased natural language modeling. *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, 2:825–829, 1991.
- [14] H. Ney, U. Essen, and R. Kneser. On structuring probabilistic dependences in stochastic language modeling. *Computer Speech and Language*, 8:1–38, 1994.
- [15] R. Kneser and H. Ney. Improved backing-off for m-gram language modeling. Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, 1:181–184, May 1995.
- [16] J. Goodman. A bit of progress in language modeling. Computer Speech and Language, pages 403–434, October 2001.
- [17] H. Ney, S. Martin, and F. Wessel. Statistical language modeling using leaving-one-out. In Corpus Based Methods in Language and Speech Processing, pages 174–207, 1997.
- [18] R. Kuhn and R De Mori. A cache-based natural language model for speech recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(6):570–583, June 1990.