

A Universal Compression Perspective of Smoothing

Nikola Jevtić Alon Orlitsky

*ECE Department, UCSD
9500 Gilman Drive
La Jolla, CA 92093-0407, USA*

Abstract

We analyze smoothing algorithms from a universal-compression perspective. Instead of evaluating their performance on an empirical sample, we analyze their performance on the most inconvenient sample possible. Consequently the performance of the algorithm can be guaranteed even on unseen data. We show that universal compression bounds can explain the empirical performance of several smoothing methods. We also describe a new *interpolated additive* smoothing algorithm, and show that it has lower training complexity and better compression performance than existing smoothing techniques.

Key words: Language modeling, universal compression, smoothing

1 Introduction

Language modeling is an essential component of many expert systems such as speech recognition, handwriting recognition, and machine translation. The performance of these systems depends on the quality of the underlying language models. Much of the difficulty in obtaining good language models lies in the sparsity of the training data. No matter how large the training data is, it is never large enough to learn the model reliably for large vocabulary tasks because the majority of possible word sequences would not appear even once. Assigning zero probability to an event would prevent its recognition no matter how strong other evidence is and is therefore undesirable. To address this problem, many *smoothing* techniques have been proposed. Their general idea is to redistribute some of the probability to unseen events and yet stay close to the empirical frequencies in the training set.

Chen and Goodman [1] compared the empirical performance of these smoothing techniques on several common data sets. They found that the relative performance of these methods was consistent on several data sets and for different data sizes. They also demonstrated approximately linear reduction in word error rate on the Broadcast News speech recognition task with the reduction of the cross-entropy of the applied language model, thus validating their methodology for comparing language models through cross-entropy.

Yet such empirical comparisons cannot tell us if a new smoothing method would outperform existing ones, and by how much. In this paper we approach language modeling from a universal-compression perspective. We compare smoothing methods not against empirical samples but based on their worst performance on any possible dataset. Specifically, the *worst-case redundancy* of a smoothing scheme is the largest possible difference between the codelength it assigns to a dataset and the lowest codelength assigned to the dataset by any distribution in the model space.

We show that the worst-case redundancies of some of the smoothing methods evaluated in [1] coincide with their empirical performance. We also present a new *interpolated additive* smoothing algorithm, motivated by the theoretical analysis. We show that it consistently outperforms previous smoothing methods on the Wall Street Journal database and that it is simpler to train than modified Kneser-Ney, previously the best known smoothing method. Finally, we show that list-based data structures can significantly reduce the training time of all the smoothing algorithms we considered.

The paper is organized as follows. In Section 2 we discuss n -gram language models. In Section 3 we formally define universal compression and state the relevant results and bounds on compression of *i.i.d.* sources. We show the implications of these results on language modeling and how they correlate with empirical results. In Section 4 we describe some of the standard approaches to language modeling. In Section 5 we introduce the *interpolated additive* language model that avoids some of the deficiencies of additive smoothing. In Section 6 we empirically compare the performances of the language models addressed in this paper with those analyzed in [1]. In Section 7 we summarize the results and present some open problems.

2 n -gram Language Models

A language model is a probability distribution over sentences, $w_0^l = w_0 \dots w_l$, where w_0 represents the beginning of a sentence, w_1, \dots, w_{l-1} are the words, and w_l represents the end of the sentence. The most popular language model used in speech recognition applications uses an *n-gram* approximation, which

assumes that the probability of the current word depends only on the previous $n - 1$ words,

$$p(w_1^l | w_0) = \prod_{i=1}^l p(w_i | w_0^{i-1}) \approx \prod_{i=1}^l p(w_i | w_{\max(0, i-n+1)}^{i-1}).$$

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. 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 . It is customary to measure the quality of a language model on T using the *per-word coding length*, or *cross-entropy*,

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

The distribution in our model space that has the smallest KL distance from the true distribution minimizes the expected coding length and consequently the distribution with smaller cross-entropy on a new sample is considered better.

Essentially all the techniques used in speech recognition construct n -gram estimators as a collection of *i.i.d.* estimators, one corresponding to each $n - 1$ word *context*. It is intuitive, and under certain assumptions can be shown formally [2], that good *i.i.d.* estimators can be combined to yield good Markov estimators. In this paper we therefore analyze *i.i.d.* estimators.

3 Universal Compression

A compression algorithm associates a binary string with every sequence of symbols emitted by a source. The objective is to minimize the number of bits used. When the source distribution p is known, an optimal compression algorithm that minimizes the expected codeword length assigns to each sequence w_1^n a codeword whose length is roughly $\log(1/p(w_1^n))$ bits.¹

In many applications the source distribution is not known except that it belongs to a known collection \mathcal{P} , such as *i.i.d.* or Markov distributions. A desirable goal in these applications is to find compression algorithms that assign to every source output a binary string whose length is close to the optimal.

¹ $\log x$ represents logarithms to the base 2, and $\ln x$ to the natural base.

3.1 Redundancy

The *redundancy* of a compression algorithm is the largest number of additional bits it assigns to a sequence over that assigned by any distribution in \mathcal{P} .

More precisely, let \mathcal{P} be a collection of distributions over a discrete alphabet \mathcal{A} . The *redundancy* of using the distribution q instead of the underlying source distribution p for compressing the sequence w_1^n is

$$R(p, q, w_1^n) = \log \frac{p(w_1^n)}{q(w_1^n)}.$$

Since the true distribution p is not known, we consider the one maximizing the redundancy,

$$\hat{R}(q, w_1^n) = \sup_{p \in \mathcal{P}} R(p, q, w_1^n) = \log \frac{\hat{p}(w_1^n)}{q(w_1^n)},$$

where $\hat{p}(w_1^n)$ is the *maximum-likelihood probability* of a sequence $w_1^n \in \mathcal{A}^n$,

$$\hat{p}(w_1^n) \stackrel{\text{def}}{=} \sup_{p \in \mathcal{P}} p(w_1^n).$$

To bound the redundancy for any sequence, we define the worst-case redundancy of a distribution q to be

$$\hat{R}_n(q) = \sup_{w_1^n \in \mathcal{A}^n} \hat{R}(q, w_1^n).$$

The worst-case redundancy of \mathcal{P} is the performance of the best compression algorithm for sequences of length n ,

$$\hat{R}_n \stackrel{\text{def}}{=} \inf_{q_n} \sup_{w_1^n \in \mathcal{A}^n} \log \frac{\hat{p}(w_1^n)}{q_n(w_1^n)},$$

where q_n is a distribution over \mathcal{A}^n and generally does not belong to \mathcal{P} .

The *average-case* redundancy of \mathcal{P} is defined similarly to be

$$\bar{R}_n \stackrel{\text{def}}{=} \inf_{q_n} \sup_{p \in \mathcal{P}} E_p \log \frac{p(w_1^n)}{q_n(w_1^n)} = \inf_{q_n} \sup_{p \in \mathcal{P}} \sum_{w_1^n} p(w_1^n) \log \frac{p(w_1^n)}{q_n(w_1^n)}.$$

Note that these redundancies are defined for blocks of size n . The block redundancy normalized by the block length is the *per-symbol* redundancy. We say that \mathcal{P} is universally compressible in the worst (average) sense if there exists a sequence of coders q_n whose per-symbol redundancies diminish with n .

In language modeling, \mathcal{A} represents a set of words that are generated by a source. As mentioned before for an n -gram model, each context corresponds to a different source, and we learn their distributions independently.

For most applications, the alphabet \mathcal{A} is restricted to the set of most frequent words, which is also used as the domain for all contexts. However, in most contexts the majority of the words would have a very low occurrence probability, and it may be better to try to learn the active alphabet instead. Thus we have two settings for universal compression, one that assumes known and finite alphabet, and another that assumes unknown and possibly infinite alphabet.

Kieffer [3] obtained a necessary and sufficient condition for a collection of sources to be universally compressible, and in particular showed that for infinite alphabets even *i.i.d.* sources have infinite redundancy. An approach taken by [4, 5] separated the description of the set of symbols emitted by the source and the description of their order, or pattern.

Orlitsky et al. [6] showed that a variant of the Good-Turing estimator can compress the patterns with worst-case redundancy of $\mathcal{O}(n^{\frac{2}{3}})$ for a block of length n , which means that patterns can be universally compressed. They also showed an algorithm with super-polynomial complexity that has a worst-case redundancy $\mathcal{O}(n^{\frac{1}{2}})$. Jevtić et al. [7] showed that the optimal compression method has a redundancy of at least $\mathcal{O}(n^{\frac{1}{3}})$ bits for a block of length n . Important questions are still open in this setting, such as the optimal redundancy of the best universal method, and which practical method could approach the optimal performance.

3.2 Universal Coding of *i.i.d.* Sources over Known Alphabet

More results exist for known and finite alphabets, which are the focus of this paper. Let an alphabet \mathcal{A} have cardinality V . An *i.i.d.* distribution over \mathcal{A} is specified by a V -dimensional parameter $\theta = (\theta_1, \dots, \theta_V)$ where each $\theta_i \geq 0$ and $\sum_{i=1}^V \theta_i = 1$. Hence the collection of all *i.i.d.* distributions over \mathcal{A} corresponds to the simplex Θ of all possible parameters θ .

For a sequence w_1^n of length n over \mathcal{A} , let $T_i \stackrel{\text{def}}{=} |\{j : w_j = i\}|$ be the number of times a symbol $i \in \mathcal{A}$ appears in the sequence. Then the probability of w_1^n is $p(w_1^n | \theta) = \prod_i \theta_i^{T_i}$. A number of researchers [8, 9, 2, 10, 11, 12, 13, 14, 15, 16] have shown that as n increases, the worst case redundancy grows as

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

and the average-case redundancy, as

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

where

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

and Γ is the Gamma function. Note that the two redundancies only differ by a constant.

In this paper we concentrate on the worst-case redundancy as it always upper bounds average redundancy and is somewhat easier to analyze. We will extensively use Feller's bounds [17] on Stirling's approximation, valid for all $x > 0$,

$$\Gamma(x) = \sqrt{2\pi} x^{x-\frac{1}{2}} e^{-x} e^{s_x}, \quad \text{where } 0 \leq s_x \leq \frac{1}{12x}. \quad (3)$$

In many applications, including language modeling it is desirable to encode the symbols sequentially. If the block length n is known in advance, sequential algorithms can compute the conditionals and achieve exactly the worst-case redundancy of the block codes, but with high computational costs. However efficient sequential methods were also developed that achieve average-case [13] and worst-case [15] redundancies asymptotically, when n is known in advance.

We evaluate the asymptotic performance of a common family of sequential algorithms used in language modeling, known as *additive-rules*. We determine their worst-case redundancy and find the sequences achieving it.

An add- δ algorithm assigns to each symbol $i \in \mathcal{A}$ the probability

$$p_\delta(w_{n+1} = i | w_1^n) \stackrel{\text{def}}{=} \frac{T_i + \delta}{\sum_{j \in \mathcal{A}} (T_j + \delta)} = \frac{T_i + \delta}{n + V\delta}. \quad (4)$$

The probability that p_δ assigns to a sequence w_1^n is therefore

$$p_\delta(w_1^n) = \frac{\prod_{i=1}^V \prod_{j=0}^{T_i-1} (j + \delta)}{\prod_{j=0}^{n-1} (j + V\delta)} = \frac{\Gamma(V\delta) \prod_i \Gamma(T_i + \delta)}{\Gamma(\delta)^V \Gamma(n + V\delta)}.$$

This shows in particular that while the conditional probabilities depend on the precise evolution of the sequence, the probability of the whole sequence depends only on the final counts T_1, \dots, T_V .

When $\delta = 1$, Equation (4) is Laplace's law of succession [18] which represents the posterior probability of the symbol j when we assume a uniform prior over all possible sample count ensembles, *e.g.*, [19, 20]. In addition to add-one, some language-modeling applications add small δ [1], and in the universal-compression literature add-half is more common, *e.g.*, [8, 2].

Another common way to derive the Laplace and other additive-rules is by assuming a Dirichlet prior over all i.i.d. distributions. For any $\delta_1, \dots, \delta_V > 0$, the Dirichlet prior is defined by

$$D_{\delta_1, \dots, \delta_V}(\theta) \stackrel{\text{def}}{=} \frac{\Gamma(\sum_i \delta_i)}{\prod_i \Gamma(\delta_i)} \prod_i \theta_i^{\delta_i - 1}.$$

Let

$$D_\delta(\theta) \stackrel{\text{def}}{=} D_{(\delta, \dots, \delta)}(\theta) = \frac{\Gamma(V\delta)}{\Gamma(\delta)^V} \prod_i \theta_i^{\delta - 1}.$$

The probability of a sequence w_1^n is then

$$p_\delta(w_1^n) \stackrel{\text{def}}{=} \int_{\Theta} p(w_1^n | \theta) D_\delta(\theta) d\theta = \int_{\Theta} \frac{\Gamma(V\delta)}{\Gamma(\delta)^V} \prod_i \theta_i^{T_i + \delta - 1} d\theta = \frac{\Gamma(V\delta)}{\Gamma(\delta)^V} \frac{\prod_i \Gamma(T_i + \delta)}{\Gamma(n + V\delta)}.$$

It follows that

$$p_\delta(w_{n+1} = i | w_1^n) = \frac{p_\delta(w_1^n, w_{n+1} = i)}{p_\delta(w_1^n)} = \frac{\frac{\Gamma(V\delta)}{\Gamma(\delta)^V} \frac{\prod_{j \neq i} \Gamma(T_j + \delta) \Gamma(T_i + 1 + \delta)}{\Gamma(n+1+V\delta)}}{\frac{\Gamma(V\delta)}{\Gamma(\delta)^V} \frac{\prod_j \Gamma(T_j + \delta)}{\Gamma(n+V\delta)}} = \frac{T_i + \delta}{n + V\delta},$$

in agreement with (4).

The redundancy of additive rules with respect to the observed sequence w_1^n is therefore

$$\hat{R}(p_\delta, w_1^n) = \max_{p \in \mathcal{P}} R(p, p_\delta, w_1^n) = \log \frac{\prod_i \left(\frac{T_i}{n}\right)^{T_i}}{\frac{\Gamma(V\delta)}{\Gamma(\delta)^V} \frac{\prod_i \Gamma(T_i + \delta)}{\Gamma(n+V\delta)}} \stackrel{\text{def}}{=} \hat{R}(p_\delta, (T_1, \dots, T_V)), \quad (5)$$

where we replaced the supremum by maximum because $R(p, q, w_1^n)$ is a continuous function in p , and \mathcal{P} is a compact domain.

Hence, just like block probabilities, worst-case redundancy depends only on the counts T_i and not on the order in which the symbols appeared, and

$$\hat{R}_n(p_\delta) = \max_{\substack{T_1, \dots, T_V \\ \sum_i T_i = n}} \hat{R}(p_\delta, (T_1, \dots, T_V)).$$

We will later show that, roughly speaking, the redundancy of an add- δ rule for a given sequence equals the sum of $\delta \log n + (\frac{1}{2} - \delta) \log(T_i + \delta) + O(1)$ over all V symbols except the most frequent. Hence each infrequent alphabet symbol contributes $\delta \log n$, and each frequent symbol, except the most frequent one, contributes $\frac{1}{2} \log n$ to the redundancy.

For $\delta \geq \frac{1}{2}$, the worst-case redundancy is achieved by sequences whose relative frequencies lie at a vertex of the simplex, namely one T_i is n , and the rest

are 0, and the redundancy is therefore roughly $(V - 1)\delta \log n$. For $\delta < \frac{1}{2}$, the worst-case redundancy is achieved by sequences whose relative frequencies lie in the center of the simplex, namely each T_i is roughly n/V , and the worst-case redundancy is therefore $\frac{V-1}{2} \log n$.

A more precise calculation, carried out in the following sections shows that

$$\hat{R}(p_\delta) = \begin{cases} (V - 1)\delta \log n + \log \frac{\Gamma(\delta)}{\Gamma(V\delta)} + o(1), & \text{if } \delta \geq \frac{1}{2}, \\ \frac{V-1}{2} \log \frac{n}{2\pi} + \log \frac{\Gamma(\delta)^V}{\Gamma(V\delta)} - (\frac{1}{2} - \delta)V \log V + o(1), & \text{if } \delta < \frac{1}{2}. \end{cases}$$

3.3 Adding $\delta \geq \frac{1}{2}$

To find the worst-case redundancy for add $\delta \geq 1/2$ rules, we first generalize results for $\delta = 1/2$ by Shtarkov [21], Suzuki [22], Freund [12], and Xie and Barron [15] to show that the redundancy is maximized at the vertices of the relative frequency simplex, namely, where one T_i is n , and the rest are 0.

Lemma 1 For $\delta \geq 1/2$, the worst-case redundancy is achieved when one T_i is n , and the rest are 0.

Proof: Assume that two counts T_1 and T_2 are positive, without loss of generality, $0 < T_1 \leq T_2$. We show that the redundancy of p_δ for a sequence with counts (T_1, T_2, \dots, T_V) is lower than that of a sequence with counts $(T_1 - 1, T_2 + 1, \dots, T_V)$. Consequently the sequence with the highest redundancy cannot have more than one non-zero count T_i .

$$\begin{aligned} \Delta R &\stackrel{\text{def}}{=} \hat{R}(p_\delta, (T_1, T_2, \dots, T_V)) - \hat{R}(p_\delta, (T_1 - 1, T_2 + 1, \dots, T_V)) & (6) \\ &= \log \frac{T_1^{T_1} T_2^{T_2} \Gamma(T_1 - 1 + \delta) \Gamma(T_2 + 1 + \delta)}{(T_1 - 1)^{T_1 - 1} (T_2 + 1)^{T_2 + 1} \Gamma(T_1 + \delta) \Gamma(T_2 + \delta)} \\ &= \log \frac{T_1^{T_1} T_2^{T_2} (T_2 + \delta)}{(T_1 - 1)^{T_1 - 1} (T_2 + 1)^{T_2 + 1} (T_1 - 1 + \delta)}. \end{aligned}$$

Generalizing a function used by Willems, Shtarkov and Tjalkens [2] and Xie and Barron [15] from half to arbitrary δ , let

$$g(x) \stackrel{\text{def}}{=} \ln \frac{x^x (x + \delta)}{(x + 1)^{x+1}}. \quad (7)$$

Then $\Delta R = (g(T_2) - g(T_1 - 1)) \log e$, and to prove the lemma it suffices to show that for $\delta \geq 1/2$ the function $g(x)$ strictly decreases with x . Taking the

derivative, we obtain

$$g'(x) = \frac{1}{x + \delta} + \ln \frac{x}{x + 1} \leq \frac{1}{x + 1/2} + \ln \frac{x}{x + 1}.$$

Substituting $x = (1 - t)/(2t)$ and using the inequalities $\ln(1 - t) < -t - t^2/2$ and $\ln(1 + t) > t - t^2/2$, we further obtain

$$g'(x) \leq 2t + \ln((1 - t)/(1 + t)) < 2t + (-t - t^2/2) - (t - t^2/2) = 0,$$

and the lemma follows. \square

This leads to the following theorem.

Theorem 1 The worst-case redundancy of additive rules when $\delta \geq \frac{1}{2}$ is

$$\hat{R}_n(p_\delta) = (V - 1)\delta \log n + \log \frac{\Gamma(\delta)}{\Gamma(V\delta)} + o(1).$$

Proof: From Lemma 1, the worst-case redundancy is obtained for sequences consisting of only one symbol, *e.g.*, a . Hence

$$\hat{R}_n(p_\delta) = \max_{w_1^n} \log \frac{\hat{p}(w_1^n)}{p_\delta(w_1^n)} = \log \frac{\hat{p}(a^n)}{p_\delta(a^n)} = \log \frac{1}{\frac{\Gamma(V\delta)}{\Gamma(\delta)} \frac{\Gamma(n+\delta)}{\Gamma(n+V\delta)}}.$$

Applying Feller's bounds (3) and the inequality $x - x^2/2 \leq \ln(1 + x) \leq x$ for $x > 0$, we obtain the theorem. \square

In particular, the worst-case redundancy for the add-one rule is,

$$\hat{R}(p_1) = (V - 1) \log n - \log \Gamma(V) + o(1),$$

and for the add-half rule,

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

where we used the fact that $\Gamma(1/2) = \sqrt{\pi}$.

Observe that the leading term in $\hat{R}_n(p_{\frac{1}{2}})$ agrees with that of \hat{R}_n , the best redundancy for *i.i.d.* sources given in Equation (1). On the other hand, the leading term in $\hat{R}_n(p_1)$ is twice as large. This discrepancy may explain experimental results in Chen and Goodman [1] showing that add-one underperformed other add-constant estimators. It may also help explain similar poor performance results of the add-one rule reported by Gale and Church [23].

Perhaps for these reasons, the add-one rule received limited attention in the data-compression literature. One exception is a paper by Rissanen[24] where

the add-one rule was used as a simple universal predictor in the context of Markov sources when all probabilities are bounded away from zero.

The add-half rule, also known as the *Krichevsky-Trofimov* estimator [8], is perhaps the most commonly used in the compression literature. It is known that in the interior of the relative frequency simplex its redundancy is equal to \hat{R}_n , see, *e.g.*, Xie and Barron [15] or Freund [12] for the binary case. The following theorem is given for completeness.

Theorem 2 If for a sequence w_1^∞ , $\lim_{n \rightarrow \infty} T_i = \infty$ for all $1 \leq i \leq V$, then

$$\hat{R}(p_{1/2}, w_1^n) = \frac{V-1}{2} \log \frac{n}{2\pi} + C_V + o(1).$$

Proof: From Equation (5),

$$\begin{aligned} \hat{R}(p_{1/2}, w_1^n) &= \log \frac{\prod_i \left(\frac{T_i}{n}\right)^{T_i}}{\frac{\Gamma(V/2)}{\Gamma(1/2)^V} \frac{\prod_i \Gamma(T_i+1/2)}{\Gamma(n+V/2)}} \\ &= C_V + \log \Gamma\left(n + \frac{V}{2}\right) - n \log n - \sum_{i=1}^V \left(\log \Gamma\left(T_i + \frac{1}{2}\right) - T_i \log T_i\right). \end{aligned}$$

Feller's bounds (3) show that

$$\log \Gamma\left(n + \frac{V}{2}\right) - n \log n = \frac{V-1}{2} \log n - n \log e + \frac{1}{2} \log 2\pi + o(1),$$

and, since by the conditions of the theorem $1/T_i = o(1)$,

$$\log \Gamma\left(T_i + \frac{1}{2}\right) - T_i \log T_i = -T_i \log e + \frac{1}{2} \log 2\pi + o(1).$$

Combined, these expressions yield the theorem. \square

While the near-optimal redundancy of the add-half rule may suggest that it would model language well, in practice it does not [1]. The reason is that in language modeling the vocabulary size, and therefore the $\frac{V-1}{2} \log n$ redundancy, are very large. By contrast, the *effective vocabulary*, the subset \mathcal{B} of the complete vocabulary that may follow a particular context, is typically a lot smaller.

Had the effective vocabulary \mathcal{B} been known ahead of time, the redundancy, achieved in particular by the add-half rule, would have been

$$\hat{R}_n(p_{\frac{1}{2}}) = \frac{|\mathcal{B}|-1}{2} \log \frac{n}{2\pi} + C_{|\mathcal{B}|} + o(1).$$

However, not knowing the effective vocabulary in advance, the add-half rule spends too many bits on nonexisting symbols, resulting in higher redundancy. This problem is partially ameliorated by adding $\delta < \frac{1}{2}$.

3.4 Adding $\delta < \frac{1}{2}$

The additive rule that performed best on language-modeling tasks [1] added a small δ . The following analysis provides two potential explanations for this improved performance. First, Theorem 3 shows that the leading term in the redundancy for all add- $\delta < 1/2$ rules is the best possible (just as it is for add-half). Second, as shown by Xie and Barron [15], and recounted in Theorem 4, the redundancy of add- $\delta < 1/2$ rules at the boundary of the relative-frequencies simplex, is lower than the worst-case redundancy of optimal estimators. Since in many large-vocabulary texts, the distribution of words is extremely skewed, add small- δ may significantly outperform add-half rules.

Theorem 3 For $\delta < \frac{1}{2}$, the worst-case redundancy is asymptotically achieved by balanced sequences and equals

$$\hat{R}_n(p_\delta) = \frac{V-1}{2} \log \frac{n}{2\pi} + C_{V,\delta} - \left(\frac{1}{2} - \delta\right) V \log V + o(1),$$

where

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

Proof: The worst-case redundancy for compressing a sequence w_1^n with p_δ can be expressed as

$$\begin{aligned} \hat{R}(p_\delta, w_1^n) &= \log \frac{\prod_i \left(\frac{T_i}{n}\right)^{T_i}}{\frac{\Gamma(V\delta)^V \prod_i \Gamma(T_i + \delta)}{\Gamma(\delta)^V \Gamma(n + V\delta)}} \\ &= C_{V,\delta} + \log \Gamma(n + V\delta) - n \log n - \sum_{i=1}^V (\log \Gamma(T_i + \delta) - T_i \log T_i) \end{aligned}$$

By using Feller's bounds and $x - x^2/2 \leq \ln(1+x) \leq x$ for all $x \geq 0$, we get

$$\log \Gamma(n + V\delta) - n \log n = \left(V\delta - \frac{1}{2}\right) \log n - n \log e + \frac{1}{2} \log 2\pi + r \log e, \quad (8)$$

where

$$-\frac{V^2\delta^2}{2n} \leq r \leq \frac{V\delta(V\delta - \frac{1}{2})}{n} + \frac{1}{12(n + V\delta)},$$

and for all $T_i \geq 0$,

$$\begin{aligned} \log \Gamma(T_i + \delta) - T_i \log T_i &= \left(\delta - \frac{1}{2}\right) \log(T_i + \delta) + T_i \log \left(1 + \frac{\delta}{T_i}\right) \\ &\quad - (T_i + \delta) \log e + \frac{1}{2} \log 2\pi + s_{T_i + \delta} \log e. \end{aligned} \quad (9)$$

Since $T_i \log(1 + \frac{\delta}{T_i}) \geq 0$, we can bound the redundancy by

$$\begin{aligned} \hat{R}(p_\delta, w_1^n) &\leq \frac{V-1}{2} \log \frac{1}{2\pi} + \left(V\delta - \frac{1}{2}\right) \log n + \left(\frac{1}{2} - \delta\right) \sum_{i=1}^V \log(T_i + \delta) \\ &\quad + C_{V,\delta} + \left(V\delta + \frac{V\delta(V\delta - \frac{1}{2})}{n} + \frac{1}{12(n + V\delta)}\right) \log e. \end{aligned} \quad (10)$$

We combine $T_i \log(1 + \frac{\delta}{T_i}) \geq (\delta - \frac{\delta^2}{2T_i}) \log e$ with Equation (9) to obtain an alternative bound

$$\begin{aligned} \hat{R}(p_\delta, w_1^n) &\leq \frac{V-1}{2} \log \frac{1}{2\pi} + \left(V\delta - \frac{1}{2}\right) \log n + \left(\frac{1}{2} - \delta\right) \sum_{i=1}^V \log(T_i + \delta) \\ &\quad + C_{V,\delta} + \left(\frac{V\delta(V\delta - \frac{1}{2})}{n} + \sum_{i=1}^V \frac{\delta^2}{2T_i} + \frac{1}{12(n + V\delta)}\right) \log e. \end{aligned} \quad (11)$$

Without loss of generality, let $T_1 = \min_i T_i$. Then if $T_1 \leq \sqrt{n}$, we have

$$\log(T_1 + \delta) \leq \log(\sqrt{n} + \delta) \leq \frac{1}{2} \log n + \frac{\delta}{\sqrt{n}} \log e,$$

and

$$\begin{aligned} \sum_{i=2}^V \log(T_i + \delta) &\leq (V-1) \log \frac{n + V\delta}{V-1} \\ &\leq (V-1) \log n - (V-1) \log(V-1) + \frac{(V-1)V\delta}{n} \log e. \end{aligned} \quad (12)$$

After combining with Equation (10) we get

$$\begin{aligned} R_1 &\stackrel{\text{def}}{=} \max_{w_1^n: \min_i T_i \leq \sqrt{n}} \hat{R}(p_\delta, w_1^n) \\ &\leq \left(\frac{V-1}{2} - \frac{1}{2}\left(\frac{1}{2} - \delta\right)\right) \log n + \frac{V-1}{2} \log \frac{1}{2\pi} - \left(\frac{1}{2} - \delta\right) (V-1) \log(V-1) \\ &\quad + C_{V,\delta} + V\delta \log e + \left(\frac{(\frac{1}{2} - \delta)\delta}{\sqrt{n}} + \frac{V\delta(\frac{V}{2} - 1 + \delta)}{n} + \frac{1}{12(n + V\delta)}\right) \log e. \end{aligned}$$

If $T_1 > \sqrt{n}$, we combine Equation (11) with

$$\sum_{i=1}^V \log(T_i + \delta) \leq V \log \frac{n + V\delta}{V} \leq V \log n - V \log V + \frac{V\delta}{n} \log e,$$

to obtain another upper bound,

$$\begin{aligned} R_2 &\stackrel{\text{def}}{=} \max_{w_1^n: \min_i T_i > \sqrt{n}} \hat{R}(p_\delta, w_1^n) \\ &\leq \frac{V-1}{2} \log \frac{n}{2\pi} + C_{V,\delta} - \left(\frac{1}{2} - \delta\right) V \log V \\ &\quad + \left(\frac{V(V-1)\delta^2}{n} + \frac{V\delta^2}{2\sqrt{n}} + \frac{1}{12(n+V\delta)} \right) \log e. \end{aligned}$$

Then,

$$\begin{aligned} \hat{R}_n(p_\delta) &= \max_{w_1^n} \hat{R}(p_\delta, w_1^n) = \max\{R_1, R_2\} \\ &\leq \frac{V-1}{2} \log \frac{n}{2\pi} + C_{V,\delta} - \left(\frac{1}{2} - \delta\right) V \log V + o(1), \end{aligned}$$

since the bound obtained for $\max_i T_i > \sqrt{n}$ dominates for large n . To show the lower bound, we observe that $\hat{R}_n(p_\delta)$ is the worst redundancy over all possible w_1^n , and substitute Equations (8) and (9),

$$\begin{aligned} \hat{R}_n(p_\delta) &\geq \hat{R}\left(p_\delta, \left(\frac{n}{V}, \dots, \frac{n}{V}\right)\right) \\ &\geq C_{V,\delta} + \left(V\delta - \frac{1}{2}\right) \log n + \frac{V-1}{2} \log \frac{1}{2\pi} + \left(\frac{1}{2} - \delta\right) V \log \frac{n + V\delta}{V} \\ &\quad - \left(\frac{V^2\delta^2}{2n} + \frac{V^2}{12(n+V\delta)}\right) \log e \\ &\geq \frac{V-1}{2} \log \frac{n}{2\pi} + C_{V,\delta} - \left(\frac{1}{2} - \delta\right) V \log V - \left(\frac{V^2\delta^2}{2n} + \frac{V^2}{12(n+V\delta)}\right) \log e \\ &= \frac{V-1}{2} \log \frac{n}{2\pi} + C_{V,\delta} - \left(\frac{1}{2} - \delta\right) V \log V + o(1). \end{aligned}$$

Since the lower and upper bounds coincide, the theorem is proven. \square

Among sequences of fixed, finite length, balanced sequences may not be the most difficult to compress by add-small-delta rules. However, when δ is sufficiently small, an argument similar to the one in Lemma 1, can be used to determine the boundary where balanced sequences become worst case. Recall the definition from Equation (6), $\Delta R = \hat{R}(T_1, T_2, \dots, T_V) - \hat{R}(T_1 - 1, T_2 + 1, \dots, T_V) = (g(T_2) - g(T_1 - 1)) \log e$, where $g(x)$ is given in Equation(7).

Lemma 2 Balanced samples yield the highest redundancies if $0 < \delta \leq 0.2$, regardless of sample size, and also when $0.2 < \delta < 1/e$ with sample size $n > (V - 1)g^{-1}(\ln \delta)$, where $g^{-1}(x)$ is the inverse function for $g(x)$.

Proof: We again assume that $0 < T_1 \leq T_2$. It is enough to show that $\Delta R \geq 0$, because then the balanced sample is an equilibrium point that can not have its redundancy increased by moving a single sample (it may remain the same if we transform to an equivalent profile).

Indeed, for all $\delta \in (0, 0.5)$ exists x_{\min} , such that for all $x \geq x_{\min}$, $g'(x) \geq 0$,

$$g'(x) = \frac{1}{x+\delta} - \ln\left(1 + \frac{1}{x}\right) \geq \frac{1}{x+\delta} - \frac{1}{x} + \frac{1}{2x^2} - \frac{1}{3x^3} = \frac{3(1-2\delta)x^2 - (2-3\delta)x - 2\delta}{6x^3(x+\delta)}.$$

By solving the quadratic equation in the numerator we obtain

$$x_{\min} = f(\delta) = \frac{2 - 3\delta + \sqrt{-39\delta^2 + 12\delta + 4}}{6(1 - 2\delta)}.$$

It can be shown that $f(\delta)$ is an increasing function in $0 < \delta < 0.5$ and that $f(0.4) = 2$. Therefore $\forall \delta \leq 0.4$ and $\forall x \geq 2$, $g'(x) \geq 0$.

We directly observe $g(0) = \ln \delta$, $g(1) = \ln((1+\delta)/4)$ and $g(2) = \ln(4(2+\delta)/27)$. By inspection, $g(1) \leq g(2)$ when $\delta \leq 5/11$ and $g(0) \leq g(1)$ when $\delta \leq 0.2$. Thus for $\delta \leq 0.2$ we have for all $0 \leq n < m$, $g(n) \leq g(m)$ which means that the only equilibrium is the balanced sample.

However when $\delta > 0.2$, we have $g(0) > g(1)$, and we no longer have a single equilibrium point. But when $\delta < 1/e$, $g(x)$ is increasing after $x = 1$, and $\lim_{x \rightarrow \infty} g(x) = -1 = \ln(1/e) > \ln \delta = g(0)$. Let $g^{-1}(x)$ be the inverse function of $g(x)$ returning values in range $[2, \infty)$. Then, $\forall \delta < 1/e$, $\forall x > g^{-1}(\ln \delta)$ we have $g(x) > g(0)$, and when the sample size n is larger than $(V - 1)g^{-1}(\ln \delta)$ we again have only one equilibrium, the balanced sample. \square

Theorem 3 shows 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. On the other hand, Xie and Barron [15] showed that if some T_i grew slower than linearly with the sample size, add-small-delta would outperform the best universal code. For completeness we recount their theorem.

Theorem 4 [15] If for some $p \in (0, 1)$ there is a word whose count $T_i < n^p$ for all n , then the leading term in the worst-case redundancy for add-small-delta rule reduces by at least $(1/2 - \delta)(1 - p) \log n$ bits.

Proof: Without loss of generality, let $T_1 < n^p$ for some $p \in (0, 1)$. Then,

$$\log(T_1 + \delta) < \log(n^p + \delta) \leq p \log n + \frac{\delta}{n^p} \log e.$$

Combining with Equations (10,12) we obtain

$$\begin{aligned} \hat{R}(p_\delta, w_1^n | T_1 \leq n^p) &< \left(\frac{V-1}{2} - \left(\frac{1}{2} - \delta \right) (1-p) \right) \log n + \frac{V-1}{2} \log \frac{1}{2\pi} \\ &+ C_{V,\delta} - \left(\frac{1}{2} - \delta \right) (V-1) \log(V-1) + V\delta \log e \\ &+ \left(\frac{(\frac{1}{2} - \delta)\delta}{n^p} + \frac{V\delta(\frac{V}{2} - 1 + \delta)}{n} + \frac{1}{12(n + V\delta)} \right) \log e, \end{aligned}$$

proving the theorem. □

This means that the add-small-delta rule would be able to remove the $\log n$ terms corresponding to all nonexisting words if δ were close enough to zero. However $\lim_{\delta \rightarrow 0} C_{V,\delta} = \infty$, 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 δ 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 language models that utilize backoff distributions.

4 Backoff Models

In general, the add-small-delta and add-half rules perform well when all the words in the vocabulary have non-negligible probabilities. When some words occur with very low probability, these rules perform poorly (add-small-delta less so). To deal with low-probability words, most popular language models *back-off*, *i.e.*, use shorter-context statistics that are less specific and thus have more training data to estimate probabilities of those words. For example, in n -gram models, from the context of the $n-1$ most recent words, they back-off to the context of $n-2$ most recent words called backoff context, and so on.

Chen and Goodman [1] distinguish two implementations of backoff, the *strict* which uses the backoff distribution only for the words unobserved in the more

specific context,

$$p(w_i|w_{i-j}^{i-1}) = \begin{cases} \lambda p_0(w_i|w_{i-j}^{i-1}), & \text{if } c(w_{i-j}^i) > 0 \\ (1 - \lambda)p(w_i|w_{i-j+1}^{i-1}), & \text{if } c(w_{i-j}^i) = 0, \end{cases}$$

and the *interpolated* that does not make such a distinction,

$$p(w_i|w_{i-j}^{i-1}) = \lambda p_0(w_i|w_{i-j}^{i-1}) + (1 - \lambda)p(w_i|w_{i-j+1}^{i-1}),$$

where p_0 is some distribution defined by the context statistics and $\lambda \in [0, 1]$. They compared several language models and concluded that interpolated back-off versions outperform the strict ones. In this paper we consider only the interpolated backoff and present a few of the most popular interpolation techniques. This is not the full overview, as *e.g.*, Good-Turing based methods such as Katz language model are not covered, but it includes the state-of-the-art models.

4.1 The Jelinek-Mercer Model

Jelinek and Mercer [25] modeled language as a general Markov source and used this model to derive an interpolation formula for the various context orders. Let $c(w_{i-j}^i)$ represent the number of times a string w_{i-j}^i has been observed in the training, and let the *maximum likelihood* estimate be

$$p_{\text{ML}}(w_i|w_{i-j}^{i-1}) \stackrel{\text{def}}{=} \frac{c(w_{i-j}^i)}{c(w_{i-j}^{i-1})}.$$

The Jelinek-Mercer model is given by

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

where $\sum_{j=0}^{n-1} \lambda_j = 1$.

To derive the maximum-likelihood estimates p_{ML} and interpolation parameters λ_j , the training data is split into two nonoverlapping subsets: *retained* and *held-out*. First, the retained set is used to derive the maximum-likelihood estimates and then the *expectation-maximization (EM)* algorithm is used to select the interpolation parameters that maximize the probability of the held-out set. Furthermore, to avoid assigning zero probability to unseen words, practical implementations typically add another interpolation level associated with the uniform distribution over the words.

A more effective variant of Jelinek-Mercer, introduced by Brown et al. [26],

used a hierarchical interpolation where for $j = 0, \dots, n - 1$,

$$p_{\text{JM}}(w_i|w_{i-j}^{i-1}) = \lambda_j p_{\text{ML}}(w_i|w_{i-j}^{i-1}) + (1 - \lambda_j) p_{\text{JM}}(w_i|w_{i-j+1}^{i-1}).$$

Again, the maximum likelihood probabilities are based on the retained set and the *backoff probabilities* λ_j 's are jointly estimated using the EM algorithm to maximize the probability of the held-out set. Note that for $j = 0$, w_i^{i-1} is an empty context and thus $p_{\text{ML}}(w_i|w_i^{i-1}) = p_{\text{ML}}(w_i)$ is the maximum likelihood estimate for the unigram probability and $p_{\text{JM}}(w_i|w_{i+1}^{i-1}) \stackrel{\text{def}}{=} 1/V$ we define to be the uniform distribution.

The main advantage of the hierarchical definition is that it allows for using different λ 's for different contexts,

$$p_{\text{JM}}(w_i|w_{i-j}^{i-1}) = \lambda_{w_{i-j}^{i-1}} p_{\text{ML}}(w_i|w_{i-j}^{i-1}) + (1 - \lambda_{w_{i-j}^{i-1}}) p_{\text{JM}}(w_i|w_{i-j+1}^{i-1}).$$

In practice, because of data sparsity, λ 's of similar contexts are clustered together. Typically, contexts are clustered together based on some prior belief that they should have similar backoff probabilities. The original criterion used in [26] was the number of times the context was observed, called the *total count*. The motivation is that estimates of contexts appearing a similar number of times can be “trusted” roughly equally and therefore should be assigned the same backoff probability. In his thesis, Chen [27] noted that better performance is achieved by normalizing this total count by the number of distinct words that follow the context in the retained set, thereby clustering the context based on the *average count*, the average number of times the context preceded various words in the retained set. It can also be shown that this criterion is less sensitive to the number of clusters used.

4.2 Linear and Absolute Discounting Models

Ney, Essen, and Kneser [28, 29] suggested a different interpolation method. They 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 backoff: *linear* and *absolute discounting*.

In linear discounting the maximum likelihood probabilities of the contexts are discounted proportionally and the total discounted probability is given to the backoff context. This model is equivalent to the Jelinek-Mercer smoothing with a single cluster.

In absolute discounting all word counts are discounted by the same additive

constant $0 < D < 1$. For $j = 0, \dots, n - 1$ let

$$N_{1+}(w_{i-j}^{i-1} \cdot) \stackrel{\text{def}}{=} |\{v : c(w_{i-j}^{i-1}v) \geq 1\}|$$

be the number of distinct words that follow w_{i-j}^{i-1} at least once. Then

$$p_{\text{abs}}(w_i | w_{i-1}^{i-1}) = \frac{\max(c(w_{i-j}^i) - D, 0)}{c(w_{i-j}^{i-1})} + \frac{DN_{1+}(w_{i-j}^{i-1} \cdot)}{c(w_{i-j}^{i-1})} p_{\text{abs}}(w_i | w_{i-j+1}^{i-1}).$$

Note that this is the interpolated form instead of the strict backoff originally introduced in [28].

The discount parameter D was originally [28] evaluated using an approximation formula that maximizes the probability in a leave-one-out procedure. More recent implementations [1] achieve better performance by using Powell search to determine the D that maximizes the likelihood of the held-out set.

While absolute discounting outperforms [28] linear discounting (Jelinek-Mercer with a single cluster), when context clustering is added to linear discounting, it outperforms absolute discounting, easily observed from the comparative plots in [1] and in Section 6. Incorporating context clustering to absolute discounting appears computationally difficult since the complexity of the Powell search increases quadratically with the number of clusters.

4.3 The Kneser-Ney Model

Kneser and Ney [30] took the absolute discounting model a step further. They kept a form similar to that of absolute discounting (again presented here in interpolated form),

$$p_{\text{KN}}(w_i | w_{i-n+1}^{i-1}) = \frac{\max(c(w_{i-n+1}^i) - D, 0)}{c(w_{i-n+1}^{i-1})} + \frac{DN_{1+}(w_{i-n+1}^{i-1} \cdot)}{c(w_{i-n+1}^{i-1})} p_{\text{KNbackoff}}(w_i | w_{i-n+2}^{i-1}),$$

but added a constraint forcing the marginals of a higher order distribution to match the marginals of the training data,

$$\sum_{w_{i-n+1}} \frac{c(w_{i-n+1}^{i-1})}{N} p_{\text{KN}}(w_i | w_{i-n+1}^{i-1}) = \frac{c(w_{i-n+2}^i)}{N}, \quad (13)$$

where N is the size of the training data.

It can be shown [28, 1] that under this condition, for any choice of D , the backoff distribution has to be 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. More precisely, let

$$N_{1+}(\cdot w_{i-n+2}^i) \stackrel{\text{def}}{=} |\{u | c(uw_{i-n+2}^i) > 0\}|,$$

and

$$\begin{aligned} N_{1+}(\cdot w_{i-n+2}^{i-1}) &= \sum_v N_{1+}(\cdot w_{i-n+2}^{i-1} v) = \sum_u N_{1+}(u w_{i-n+2}^{i-1} \cdot) \\ &= |\{(u, v) | c(uw_{i-n+2}^{i-1} v) > 0\}|. \end{aligned}$$

Then, the backoff satisfying the marginal constraint for any D is

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

The Kneser Ney model significantly improved the coding length of unseen data. This may be partially explained by the observation [31] that language models whose marginals do not match the underlying source probabilities will arbitrarily diverge over time. The constraint in (13) ensures that the marginals agree with the n -gram frequencies which in turn may be close to the actual unknown probabilities.

4.4 Variations of the Kneser-Ney Model

Ney *et al.* [32] proposed a variation of absolute discounting that uses two discount parameters, D_1 for symbols observed once and D_{2+} for those observed two or more times. They reported mixed results compared to the single parameter case, but we note that they estimated discounting parameters through leave-one-out procedure.

Chen and Goodman [1] show that three parameters D_1 , D_2 , and D_{3+} consistently outperform a single parameter over a variety of corpora and training set sizes, while optimizing parameters with the held-out set. The Kneser-Ney language model with three discount parameters is the best known n -gram model.

5 The Interpolated Additive Model

In Section 3 we observed that additive models waste too many bits on non-existing words when the actual vocabulary is smaller than expected. To address this issue, we propose an *interpolated additive (IA)* model. It uses the additive rule to estimate the probability distribution only among the observed words in the context, and interpolates these estimates between different context levels.

First, for $j = 0, \dots, n - 1$ and for each context w_{i-j}^{i-1} , we use an additive constant $\delta^{(j)} \in (-1, C)$, where C is some large positive constant, to smooth the distribution of words observed in the training data,

$$p(w_i | w_{i-j}^{i-1}) = \begin{cases} \frac{c(w_{i-j}^i) + \delta^{(j)}}{c(w_{i-j}^{i-1}) + N_1 + (w_{i-j}^{i-1} \cdot) \delta^{(j)}}, & c(w_{i-j}^i) > 0 \\ 0, & c(w_{i-j}^i) = 0. \end{cases} \quad (14)$$

Then, to address the probabilities of unobserved words, we use the Jelinek-Mercer interpolation approach,

$$p_{\text{IA}}(w_i | w_{i-j}^{i-1}) = \lambda_{w_{i-j}^{i-1}} p(w_i | w_{i-j}^{i-1}) + (1 - \lambda_{w_{i-j}^{i-1}}) p_{\text{IA}}(w_i | w_{i-j+1}^{i-1}).$$

Again, we end the recursion above with the uniform distribution by defining

$$p_{\text{IA}}(w_i | w_{i+1}^{i-1}) \stackrel{\text{def}}{=} \frac{1}{V}.$$

This formulation allows us to use the EM algorithm to estimate the λ parameters over different levels in the same way it is used in the Jelinek-Mercer model. The optimal additive constants δ are also estimated as part of the EM procedure. In the expectation step, the contribution for each held-out set sample in each n -gram level is calculated. Within the maximization step, the interpolation parameters can be determined by a closed-form formula, but there is no formula for the additive constants, and they are determined by an iterative search. Fortunately the EM algorithm 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, a much simpler one dimensional search is used for each $\delta^{(j)}$.

We also investigated more complex structures of the IA model. Similar to the modified Kneser-Ney model proposed by Chen and Goodman [1], one can use more additive constants with the IA model, *e.g.*, $\delta_1^{(j)}$, $\delta_2^{(j)}$, and $\delta_{3+}^{(j)}$, corresponding to words observed once, twice, and three or more times in the training set. Equation (14) is then modified to

$$p(w_i | w_{i-j}^{i-1}) = \begin{cases} \frac{c(w_{i-j}^i) + \delta_1^{(j)}}{\tilde{c}(w_{i-j}^{i-1})}, & c(w_{i-j}^i) = 1 \\ \frac{c(w_{i-j}^i) + \delta_2^{(j)}}{\tilde{c}(w_{i-j}^{i-1})}, & c(w_{i-j}^i) = 2 \\ \frac{c(w_{i-j}^i) + \delta_{3+}^{(j)}}{\tilde{c}(w_{i-j}^{i-1})}, & c(w_{i-j}^i) \geq 3 \\ 0, & c(w_{i-j}^i) = 0, \end{cases}$$

where the normalized counts are given by

$$\tilde{c}(w_{i-j}^{i-1}) = c(w_{i-j}^{i-1}) + N_1(w_{i-j}^{i-1} \cdot) \delta_1^{(j)} + N_2(w_{i-j}^{i-1} \cdot) \delta_2^{(j)} + N_{3+}(w_{i-j}^{i-1} \cdot) \delta_{3+}^{(j)}.$$

In this case, one-dimensional search for the optimal additive constants is not possible and Powell’s method needs to be used within each EM iteration, and for every n -gram level.

Further, it is not necessary to have a single set of additive constants for an entire n -gram level. Several sets of additive constants can be estimated in parallel, in which case we would use $\delta_k^{(w_{i-j}^i)}$ to uniquely identify each. In our tests we grouped several adjacent backoff clusters together to be represented by the same additive cluster, but other clustering strategies could be used.

In terms of training complexity, the clustering of additive parameters (apart from the problem of optimal model selection) comes for free. The data relevant to a single set of additive constants is broken into disjoint subsets when multiple additive sets are estimated, and under the assumption that the same average number of iterations is needed to optimize each of the problems, the expected computational load is the same. This will become clearer when considered in conjunction with the list training method described in the next section.

Unlike all other universal methods we analyzed that augment the counts by positive constants, absolute discounting achieves good performance while subtracting. While absolute discounting and Kneser-Ney can give some probability to the backoff only through subtracting, the IA model uses independent method for supplying the backoff probability while leaving additive constant(s) unconstrained. Thus the IA model allows us to test whether the optimal additive strategy for smoothing probabilities among the observed words is to subtract or to add.

6 Empirical Comparison of Language Models

In this Section we compare language models empirically. In Subsection 6.1 we describe the experimental setup, in Subsection 6.2 we give the comparison in perplexities of the previous smoothing methods with our IA model, and show that the IA outperforms the previous approaches. We also describe the improvements in the training complexity compared to previous language modeling techniques. In Subsection 6.3 we investigate the effects of foldback on the IA model, and also test the possibility of using the leave-one-out technique for the held-out set to avoid the need for foldback after the training, and show that leave-one-out performs poorly in this setting.

6.1 Testing Methodology

Similar to [1] we compare the various smoothing techniques using the Wall Street Journal (WSJ) database consisting of roughly 1.6 million sentences. The vocabulary was reduced to the 20,000 most frequent words, with all out-of-vocabulary words mapped to a special symbol, treated as a single word. To further reduce the vocabulary and its impact on the required memory for the system, the suffix “s”, usually used in possessive cases of singular nouns, was removed from the end of words and considered as a separate word. The 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.

The two held-out sets and the test set sizes were all fixed at 2500 sentences. The retained set size ranged from 10,000 to the full WSJ size. For each experiment, new nonoverlapping held-out, test, and retained sets were selected uniformly at random. The sentences were sampled individually, rather than in blocks, to avoid local-topic artifacts.

For methods that require clustering of the interpolation parameters (Jelinek-Mercer and Interpolated-Additive), models can be built for several cluster sizes by optimizing the parameters for the first held-out set and using the second held-out set to choose the best model. This is a slightly inconvenient procedure because the computation of the IA model takes more time than, *e.g.*, Jelinek-Mercer smoothing. However, the clustering results obtained with the average count used as similarity measure are very stable over a large span of cluster sizes, and it was enough to train the Jelinek-Mercer models with several values for bigram cluster size and then after fixing the bigram cluster size also run with several values for trigram cluster size and use the best clusters with the IA model as well.

To avoid poor-clustering effects and as a form of regularization we smoothed the interpolation parameters as well. For that we used maximum a posteriori (MAP) estimation. The standard maximum likelihood seeks parameters that satisfy

$$(\lambda, \delta) = \arg \max_{\lambda, \delta} p(T|\lambda, \delta),$$

where T is the text in the held-out set, while MAP estimate represents the mode of the posterior distribution,

$$(\lambda, \delta) = \arg \max_{\lambda, \delta} p(\lambda, \delta|T) = \arg \max_{\lambda, \delta} p(T|\lambda, \delta)p(\lambda, \delta).$$

In particular, we assumed uniform (noninformative) prior for additive param-

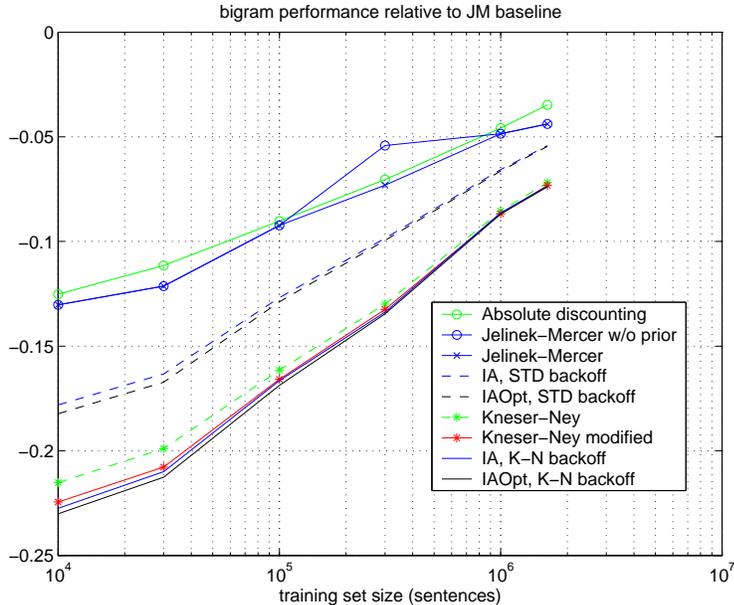


Fig. 1. Relative coding length per word for bigram models

eters over $(-1, C)$ and Dirichlet prior for the interpolation parameters,

$$p(\lambda) = \prod_i D_{3/2}(\lambda_i) = \prod_i \frac{\Gamma(3)}{\Gamma(1.5)^2} \sqrt{\lambda_i(1 - \lambda_i)},$$

where each λ_i corresponds to a different backoff cluster. The MAP estimation problem is still solved by EM, and this particular prior transforms the original EM reestimation equations for λ_i parameters to add-half rule. This is a very convenient way to smooth the backoff estimates since it is incorporated into EM and does not impair the convergence of other parameters. Details about MAP estimation can be found in *e.g.*, [33] where it is used in the context of model adaptation.

6.2 Experimental Results

We tested the various smoothing methods on bigram and trigram models. The results, shown in Figures 1 and 2, display the average test-set codelength against the training-set size. The codelength is expressed in terms of the improvement compared to the Jelinek-Mercer baseline method with a single interpolation constant for every n -gram level.

Although Kneser-Ney backoff was derived for single-constant absolute discounting, Chen and Goodman showed that it worked well with multiple-constant absolute discounting, where the original derivation does not apply exactly. Similarly, the Kneser-Ney backoff motivation does not apply to IA,

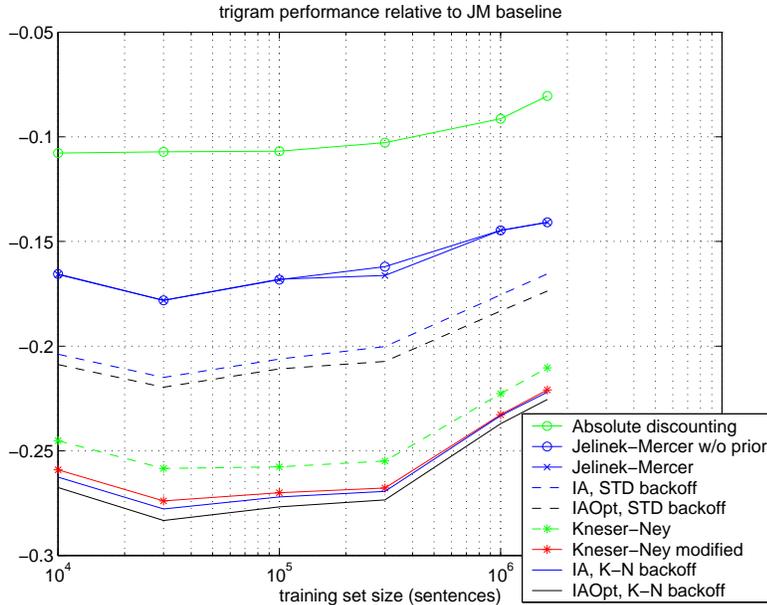


Fig. 2. Relative coding length per word for trigram models

and yet the experiments we performed showed improved performance over the IA with standard backoff model.

We present results for the basic IA model with single additive constant per n -gram level and what we call optimal interpolated additive model (IAOpt) that uses three additive constants per set and six sets for each n -gram level. We present both of these variants with standard and Kneser-Ney backoff.

IA models were improving when adding additive constants within each set, but we had difficulties finding good local optima with more than three constants. This is a standard problem with all EM type algorithms since they do not guarantee finding the global optimum. In terms of finding the optimal number of additive clusters in our initial experiments we found that for the given amount of held-out data models with three and up to twenty additive clusters had only a minor performance differences and decided not to optimize for each training data size but to use fixed number of six clusters. The existing backoff clusters were grouped in six bins, approximately of equal sizes.

Models obtained through MAP training performed almost identically to those obtained through standard ML approach (without prior) for most of the data sizes/language models/clustering parameters, but there was one instance where it prevented a severe problem. For data size 300,000 sentences, and bi-gram Jelinek-Mercer model trained through standard ML approach, one cluster had estimated backoff probability to be very close to zero. Apparently for one of the backoff clusters all the samples from the held-out set fell within the observed words, and no backoff was necessary. The same was the case with the second held-out set, and the model was chosen as the best among all the clus-

tering options tested. However, one sample in the test data was an unobserved word and scored very small probability. (These probabilities do not converge to zero, because we stop training when the improvement is very small.) On the other hand Jelinek-Mercer model obtained through described MAP training did not converge to such a small probability and scored much better on the same data. The problem was of the smaller magnitude but still observable with the trigram model, and totally invisible in any of the IA models although they use the same clustering. As can be seen on Figure 1 for bigram and Figure 2 for trigram Jelinek-Mercer models, models smoothed with our chosen Bayesian prior are more robust in these situations and identical elsewhere.

The performance differences of the various smoothing techniques for bigram models diminish with the retained set size. Bigram contexts start saturating with more data and relative frequencies in most contexts get relatively close to the true bigram probabilities. This is not the case with the trigram models where these relative performances do not change much with the data size. Trigram models are much sparser, and much more training data is needed to reach similar saturation. Because of this using better smoothing techniques is even more important with longer-span n -grams.

Directly comparing performances of these algorithms we can observe that the IA model outperforms (slightly but consistently) the modified Kneser-Ney model and IAOpt model outperforms it more clearly, when in configuration with Kneser-Ney style backoff. We further observe that, depending on the implementation, training of the interpolated additive model may be much easier.

The training complexity is given in the Table 1. It is represented by the average number of passes through the data during the training, since IA and absolute discounting have almost equivalent operations in one pass.

Model	Bigram	Trigram
Jelinek-Mercer	7.74	8.31
Kneser-Ney	25.17	54.5
IA	45.18	49.04
modified Kneser-Ney	110.0	201.83
IAOpt	184.87	214.61

Table 1

Average number of passes through the training data

We notice that while for bigram models IA and IAOpt require more iterations than the Kneser-Ney and modified Kneser-Ney, for the trigram model IA has about the same complexity as Kneser-Ney and slightly better performance than modified Kneser-Ney, and IAOpt has about the same complexity as mod-

ified Kneser-Ney and even better performance than IA. As can be observed, for both IA and IAOpt models, the training complexity does not increase significantly from bigram to trigram model. IA is essentially a two dimensional optimization problem regardless of n -gram depth, with first dimension quickly driven to convergence by the EM algorithm, and the second dimension being optimized by a line search for the basic model or a 3 dimensional Powell’s search for the IAOpt model. On the other hand, the complexity of Powell’s method used for training the Kneser-Ney and modified Kneser-Ney models is quadratic in the number of free parameters. For this reason the IA training complexity would scale much better with additional n -gram levels (four-gram, five-gram) than that of the modified Kneser-Ney model. This property of the Powell’s method was recognized in [31] where the author decided to smooth language models with standard Kneser-Ney because modified Kneser-Ney took too long to train, especially with long-span n -grams.

Convergence criterion used in our simulations 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.

To measure the training complexity performance we used the number of iterations needed to converge to the solution, but its importance depends on the implementation tradeoffs. In the typical implementation, 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 held-out and test sets. 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 linked-list access to data is faster (the tokens are accessed in sequence), but more importantly the structures can keep all the necessary statistics that would otherwise need to be computed on the fly. For IA model tokens, the master list contains all the tokens and is used for estimation of backoff parameters. Additional lists are built for each of the additive sets, but only the relevant tokens from the master list are linked in. When the optimal additive parameter (set) is estimated during the maximization phase of the EM, only the relevant samples are visited during the iteration. The total number of iterations given in Table 1 is then calculated by normalizing these iterations by the number of samples in each of the lists. For reference we ran java code under Linux on a 1.8GHz P4 (400MHz FSB) with 3GB of PC133 memory. Modified Kneser-Ney using hashtables for the trigram model and largest data size took 1987 seconds, while the implementation with lists runs in about 87.5 seconds. Further, going through 138 iterations takes only 7.5 seconds and the

remaining 80 seconds are used for creating the data structures and computing all the necessary statistics. Results are similar for interpolated models as well. Effectively, this means that the list-based training makes complexity of a language model irrelevant, since all models need approximately the same time to train, and future choices should be based only on performance.

6.3 Foldback and Leave-one-out Training

It is well known that a procedure known as *foldback* can improve the quality of the language model. After the model is trained, the data from the held-out set can be merged with the retained set, and the already estimated smoothing parameters can be applied to the complete data.

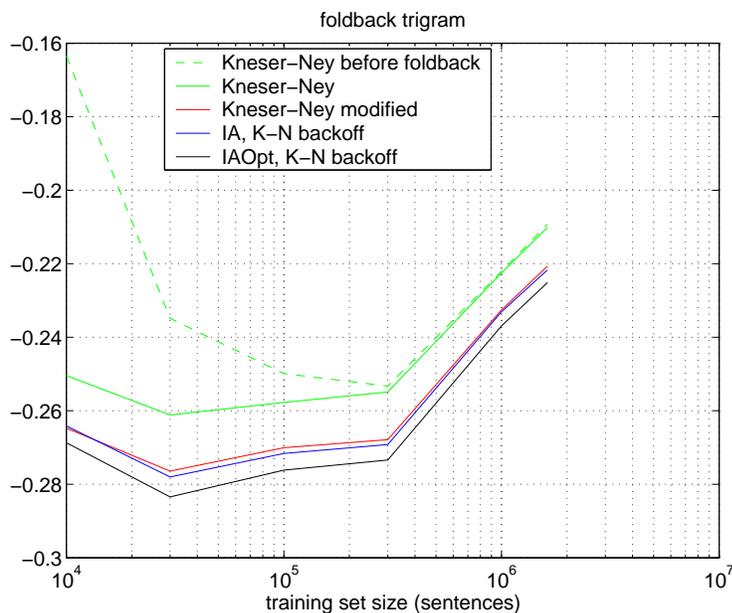


Fig. 3. Relative coding length for trigram models after foldback

Since the previous models *e.g.*, Kneser-Ney and modified Kneser-Ney are simpler than IA, we wanted to compare their robustness when foldback is applied. One of the questions was what should be done with respect to the clustering when additional samples within the context change its statistics significantly. In our experiments it was generally better to have fixed context clusters than to allow contexts to float to another cluster, more similar to its score after foldback. The only exception was for the retained set size of 10,000 sentences in which the held out set increases its size after foldback by 25%. For this data point we allowed for contexts to float to a different cluster after foldback, although this is a very noisy process.

In Figure 3 we show the effects of the foldback on our most interesting distributions, Kneser-Ney, modified Kneser-Ney, IA and IAOpt. This time results are

relative to the Jelinek-Mercer baseline distribution after foldback. To observe the effects of the foldback itself we also gave the performance of Kneser-Ney before foldback. We can observe that relative performances of these methods remain approximately the same, and that the effects of foldback almost completely vanish with the training set size of 300,000 sentences.

Another interesting possibility is to run the parameter estimation in the leave-one-out fashion. The entire training set is too large for leave-one-out estimation since even a single pass would take too much time (and force a different implementation because of the memory constraints). Instead, the same data normally used in the held-out set can be folded to the retained set prior to training and each sample can be taken out in turn for evaluation. This means that we are doing leave-one-out only for the original held-out set.

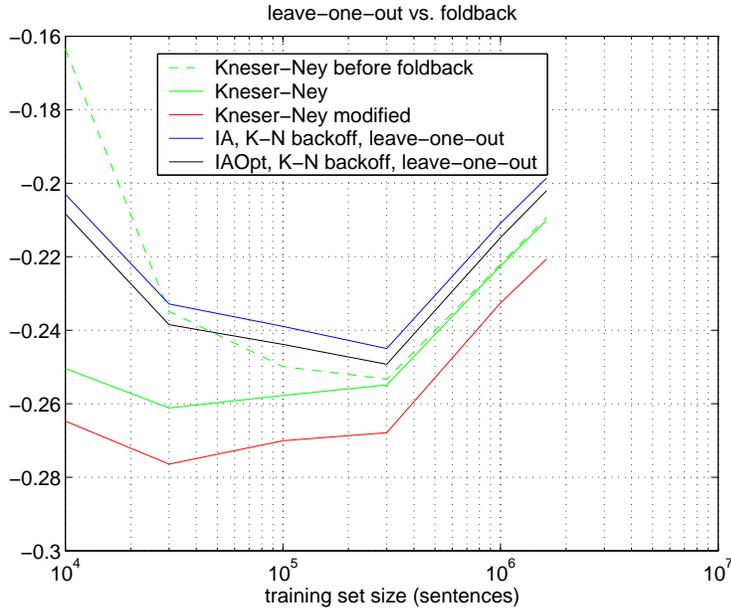


Fig. 4. leave-one-out trained IA models compared to foldback Kneser-Ney

Such leave-one-out training removes the need to use the foldback after the training, and in theory should reduce the mismatch of trained parameters with the training set after foldback. In conjunction with list-based training, additional computations (if any) occur only once in the list creation phase, and the iterative training procedure is identical.

There are again two clustering choices when a sample is removed from a context, fixed or floating. Training with floating clusters by reclustering contexts based on the snapshot with sample removed, had catastrophic results with a loss of a whole bit/word in performance. It turned out that many clusters ended up with no data to be estimated from, because most of the samples when removed would change average word count statistics significantly enough for context to jump far from the original cluster. Unfortunately even with clus-

tering fixed prior to sample removal leave-one-out failed to perform close to the foldback methods.

Figure 4 compares the performance of IA and IAOpt trained with leave-one-out strategy (fixed clusters) with that of the Kneser-Ney model variants with foldback. The performance loss while training using leave-one-out is evident when comparing results with Figure 3. Like with leave-one-out estimates for the Kneser-Ney model [32, 1], the IA model performance degrades when trained through leave-one-out procedure. While it is known that leave-one-out works well with certain type of algorithms, such as nearest neighbor, it is also known to overtrain often which seems to be the case on language modeling tasks as well.

7 Discussion

We analyzed the performance of additive rules from a universal-compression perspective. We showed that their redundancy bounds can explain the empirical performances of these techniques on common language-modeling tasks. We also showed that additive rules perform suboptimally when the vocabulary is smaller than expected. The crucial advantage for backoff models over additive rules is their ability to better determine the active vocabularies within n -gram contexts through delegating the decision to the backoff distribution.

Motivated by theoretical results stating that additive rules approach optimal performance among words with positive probabilities, and perform poorly when a large number of words appear zero or few times, we proposed the interpolated additive (IA) model. The IA model uses an additive rule among the observed words and interpolates it with the backoff distribution to achieve both fast convergence among the observed words to the true probabilities, and a better estimate of the active vocabulary through the backoff distribution.

In tests on the Wall Street Journal database, the IA model with Kneser-Ney backoff outperformed all other language models with both bigrams and trigrams. We also demonstrated that IA is easier to train than modified Kneser-Ney, previously the best-performing language model, and showed how list-based data structures can dramatically reduce the training time for all language models considered here. We also tested the leave-one-out approach for training the IA model and concluded that it generally performs poorly on language modeling tasks. Perhaps an interesting insight might be gained if this phenomenon could be explained.

Additionally, all of the optimized additive constants in the IA model were found to be negative, suggesting that there is a large number of oversampled

words. Since these additive constants mostly influence words that appear only few times in the training data it is possible that many of them do not belong in the active vocabulary. Note that the current rule of thumb for improving the quality of the language model is to add more training data. Although by doing so we indeed improve the model, we inevitably observe more irrelevant words, and a more efficient use of the data would be to try to determine those words in advance and remove their contribution directly instead of indiscriminately discounting all the words.

Potential future extensions of this work may therefore benefit from applying dynamic language models. Such models will adapt to the current topic and exclude words specific to other topics from the active vocabulary. The reported improvements in the training algorithm efficiency should be helpful for online adaptation of dynamic models as well.

References

- [1] S. F. Chen and J. Goodman. An empirical study of smoothing techniques for language modeling. *Computer Speech and Language*, 13:359–394, 1999.
- [2] F.M.J. Willems, Y.M. Shtarkov, and T.J. Tjalkens. The context-tree weighting method: Basic properties. *IEEE Transactions on Information Theory*, 41(3):653–664, May 1995.
- [3] J.C. Kieffer. A unified approach to weak universal source coding. *IEEE Transactions on Information Theory*, 24(6):674–682, Nov 1978.
- [4] J. Åberg, Y.M. Shtarkov, and B.J.M. Smeets. Multialphabet coding with separate alphabet description. In *Proceedings of compression and complexity of sequences*, 1997.
- [5] N. Jevtić, A. Orlitsky, and N. Santhanam. Universal compression of unknown alphabets. In *Proceedings of IEEE Symposium on Information Theory*, 2002.
- [6] A. Orlitsky, N. Santhanam, and J. Zhang. Always good turing: Asymptotically optimal probability estimation. *Science*, 302:427–31, October 2003.
- [7] N. Jevtić, A. Orlitsky, and N. Santhanam. A lower bound on compression of unknown alphabets. To appear in *Theoretical Computer Science*.
- [8] R. E. Krichevsky and V. K. Trofimov. The performance of universal encoding. *IEEE Transactions on Information Theory*, 27(2):199–207, March 1981.
- [9] T.M. Cover. Universal portfolios. *Mathematical Finance*, 1(1):1–29, Jan 1991.
- [10] T.M. Cover and E. Ordentlich. Universal portfolios with side information. *IEEE Transactions on Information Theory*, 42(2), Mar 1996.

- [11] J. Rissanen. Fisher information and stochastic complexity. *IEEE Transactions on Information Theory*, 42:40–47, 1996.
- [12] Y. Freund. Predicting a binary sequence almost as well as the optimal biased coin. In *Proceedings of the ninth annual conference on Computational learning theory*, pages 89–98, 1996.
- [13] 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.
- [14] W. Szpankowski. On asymptotics of certain recurrences arising in universal coding. *Problems of Information Transmission*, 34(2):142–146, 1998.
- [15] 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.
- [16] M. Drmota and W. Szpankowski. The precise minimax redundancy. In *Proceedings of IEEE Symposium on Information Theory*, 2002.
- [17] W. Feller. *An Introduction to Probability Theory*. Wiley, 1968.
- [18] P. S. de Laplace. *Essay Philosophique sur la Probabilités*. Courcier Imprimeur, Paris, 1816.
- [19] H. Jeffreys. *Theory of Probability*. Clarendon, Oxford, 1939.
- [20] 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.
- [21] Yu. M. Shtarkov. *Encoding of Discrete Sources Under Conditions of Real Restrictions and Data Compression*. PhD Thesis, 1980.
- [22] J. Suzuki. Some notes on universal noiseless coding. *IEICE Trans. Fundamentals*, E78-A(12):1840–7, December 1995.
- [23] 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.
- [24] J. Rissanen. Complexity of strings in the class of markov sources. *IEEE Transactions on Information Theory*, 32(4):526–532, July 1986.
- [25] 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.
- [26] 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.
- [27] S. F. Chen. *Building Probabilistic Models for Natural Language*. PhD Thesis, 1996.
- [28] H. Ney and U. Essen. On smoothing techniques for bigram-based natural language modeling. *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, 2:825–829, 1991.
- [29] H. Ney, U. Essen, and R. Kneser. On structuring probabilistic dependences in stochastic language modeling. *Computer Speech and Language*, 8:1–38, 1994.
- [30] R. Kneser and H. Ney. Improved backing-off for m-gram language mod-

- eling. *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, 1:181–184, May 1995.
- [31] J. Goodman. A bit of progress in language modeling. *Computer Speech and Language*, pages 403–434, October 2001.
- [32] 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.
- [33] J.-L. Gauvain and C.-H. Lee. Maximum a posteriori estimation for multivariate gaussian mixture observations of markov chains. *IEEE Transactions on Speech and Audio Processing*, 2(2):291–298, April 1994.