Nonmonotonic Extension Models[1]

Eric Sven Ristad Robert G. Thomas

Research Report CS-TR-486-95
February 1995

Abstract

We introduce three new techniques for statistical language models: extension modeling, nonmonotonic contexts, and the divergence heuristic. Together these techniques result in language models that have few states, even fewer parameters, and low message entropies. For example, our techniques achieve a message entropy of 1.97 bits/char on the Brown corpus using only 89,325 parameters. In contrast, the character 4-gram model requires more than 250 times as many parameters in order to achieve a message entropy of only 2.47 bits/char. The fact that our model performs significantly better while using vastly fewer parameters indicates that it is a better probability model of natural language text.

[1]Thanks to Andrew Appel, Carl de Marken, and Dafna Scheinvald for their critique. Both authors are partially supported by Young Investigator Award IRI-0258517 to the first author from the National Science Foundation. The second author is additionally supported by a tuition award from the Princeton University Research Board. The research was partially supported by NSF SGER IRI-9217208.

1 Introduction

Current approaches to automatic speech and handwriting transcription demand a strong language model with a small number of states and an even smaller number of parameters. If the model is weak (ie., the model entropy is high), then transcription results are abysmal. If there are too many states, then transcription becomes computationally infeasible. And if there are too many parameters, then "overfitting" occurs and the error rate increases.

In this paper we introduce three new techniques for statistical language models: extension modeling, nonmonotonic contexts, and the divergence heuristic. Together these techniques result in language models that have few states, even fewer parameters, and low message entropies. For example, our techniques achieve a message entropy of 1.97 bits/char on the Brown corpus using only 47,375 contexts and 89,325 parameters. By modestly increasing the number of model parameters in a principled manner, our techniques are able to further reduce the message entropy of the Brown Corpus to 1.91 bits/char.[2] In contrast, the character 4-gram model requires 7 times as many contexts and 250 times as many parameters in order to achieve a message entropy of only 2.47 bits/char. Given the logarithmic nature of codelengths, a savings of 0.5 bits/char is quite significant. The fact that our model performs significantly better using vastly fewer parameters argues that it is a much better probability model of natural language text.

Our first two techniques – *nonmonotonic contexts* and *extension modeling* – are generalizations of the traditional context model [3, 10, 11]. Our third technique – *the divergence heuristic* – is an incremental model selection criterion based directly on Rissanen's minimum description length (MDL) principle [9]. The MDL principle states that the best model is the simplest model that provides a compact description of the observed data.

In the traditional context model, every prefix and every suffix of a context is also a context. Three consequences follow from this property. The first consequence is that the context dictionary is unnecessarily large because most of these contexts are redundant. The second consequence is to attenuate the benefits of context blending, because most contexts are equivalent to their maximal proper suffixes. The third consequence is that the length of the longest candidate context can increase by at most one symbol at each time step, which impairs the model's ability to model complex sources. In a nonmonotonic model, this constraint is relaxed to allow compact dictionaries, discontinuous backoff, and arbitrary context switching.

[2] The only change to our model selection procedure is to replace the simplified incremental cost formula (23) with a smaller constant cost of 2 bits/extension. This small change reduces the test message entropy from 1.97 to 1.91 bits/char but it also quadruples the number of model parameters and triples the total codelength. Since our primary criterion is to minimize the total codelength, we do not wish to emphasize this result. We mention it for those who do not share our goal of minimizing the total codelength.

The traditional context model maps every history to a unique context. All symbols are predicted using that context, and those predictions are estimated using the same set of histories. In contrast, an extension model maps every history to a *set* of contexts, one for each symbol in the alphabet. Each symbol is predicted in its own context, and the model's current predictions need not be estimated using the same set of histories. This is a kind of parameter default mechanism that increases the accuracy of the model's predictions while reducing the number of free parameters in the model.

As a result of these two generalizations, nonmonotonic extension models can outperform their equivalent context models using significantly fewer parameters. For example, an order 3 n-gram (ie., the 4-gram) requires more than 51 times as many contexts and 787 times as many parameters as the order 3 nonmonotonic extension model, yet already performs worse on the Brown corpus by 0.08 bits/char.

Our third contribution is the divergence heuristic, which adds a more specific context to the model only when it reduces the codelength of the past data more than it increases the codelength of the model. In contrast, the traditional selection heuristic adds a more specific context to the model only if it's entropy is less than the entropy of the more general context [10, 11]. The traditional minimum entropy heuristic is a special case of the more effective and more powerful divergence heuristic. The divergence heuristic allows our models to generalize from the training corpus to the testing corpus, even for nonstationary sources such as the Brown corpus.

The remainder of our article is organized into three sections. In section 2, we formally define the class of extension models and present a heuristic model selection algorithm for that model class based on the divergence criterion. Next, in section 3, we demonstrate the efficacy of our techniques on the Brown Corpus, an eclectic collection of English prose containing approximately one million words of text drawn from 500 different sources and 15 different genres [5]. The brutally nonstationary nature of this corpus poses a stringent test for language models. The final section, section 4, briefly discusses related work. Appendix A contains a summary of our notation.

2 Extension Model Class

This section consists of four parts. In 2.1, we formally define the class of extension models and prove that they satisfy the axioms of probability. We also classify language models based on the structure of their context dictionary. In 2.2, we show to estimate the parameters of an extension model using Moffat's "method C" [8]. In 2.3, we provide codelength formulas for our model class, based on efficient enumerative codes. These codelength formulas will be used to match the complexity of the model to the complexity of the data. In 2.4, we present a heuristic model selection algorithm that adds parameters to an

extension model only when they reduce the codelength of the data more than they increase the codelength of the model.

2.1 Model Class Definition

A context model predicts each symbol in the same context or set of contexts. In contrast, an extension model predicts each symbol in its own context or set of contexts. Formally, an *extension model* $\phi = \langle \Sigma, D, E, \lambda \rangle$ consists of a finite alphabet Σ, $|\Sigma| = m$, a dictionary D of contexts, $D \subset \Sigma^*$, a set of available context extensions E, $E \subseteq D \times \Sigma$, and a probability function $\lambda : E \rightarrow [0, 1]$. For every context w in D, $E(w)$ is the set of symbols available in the context w and $\lambda(\sigma|w)$ is the conditional probability of the symbol σ in the context w. Note that $\sum_{\sigma \in \Sigma} \lambda(\sigma|w) \leq 1$ for all contexts w in the dictionary D.

The probability $\tilde{p}(h|\phi)$ of a string h given the model ϕ, $h \in \Sigma^n$, is calculated as a chain of conditional probabilities (1)

$$\tilde{p}(h|\phi) \doteq \tilde{p}(h_n|h_1 \ldots h_{n-1}, \phi)\tilde{p}(h_1 \ldots h_{n-1}|\phi) \tag{1}$$

while the conditional probability $\tilde{p}(\sigma|h, \phi)$ of a single symbol σ after the history h is defined as (2).

$$\tilde{p}(\sigma|h, \phi) \doteq \begin{cases} \lambda(\sigma|h) & \text{if } \langle h, \sigma \rangle \in E \\ \delta(h)\tilde{p}(\sigma|h_2 h_3 \ldots h_n, \phi) & \text{otherwise} \end{cases} \tag{2}$$

The expansion factor $\delta(h)$ ensures that $\tilde{p}(\cdot|h, \phi)$ is a probability function if $\tilde{p}(\cdot|h_2 \ldots h_n, \phi)$ is a probability function.

$$\delta(h) \doteq \frac{1 - \lambda(E(h)|h)}{1 - \tilde{p}(E(h)|h_2 \ldots h_n, \phi)} \tag{3}$$

Note that $E(h)$ represents a set of symbols, and so by a slight abuse of notation $\lambda(E(h)|h)$ denotes $\sum_{\sigma \in E(h)} \lambda(\sigma|h)$, ie., the sum of $\lambda(\sigma|h)$ over all σ in $E(h)$.

Example 1. Let $\Sigma = \{0, 1\}$, $D = \{ \epsilon$, `"0"` $\}$, $E(\epsilon) = \{0, 1\}$, $E($`"0"`$) = \{0\}$. Suppose $\lambda(0|\epsilon) = \frac{1}{2}$, $\lambda(1|\epsilon) = \frac{1}{2}$, and $\lambda(0|$`"0"`$) = \frac{3}{4}$. Then $\delta($`"0"`$) = \frac{1}{4}/\frac{1}{2} = \frac{1}{2}$ and $\tilde{p}(1|$`"0"`$,\phi) = \delta($`"0"`$)\ \lambda(1|\epsilon) = \frac{1}{4}$.

The fundamental difference between a context model and an extension model lies in the inputs to the context selection rule, not its outputs. The traditional context model includes a selection rule $s : \Sigma^* \rightarrow D$ whose only input is the history. In contrast, an extension model includes a selection rule $s : \Sigma^* \times \Sigma \rightarrow D$ whose inputs include the past history and the symbol to be predicted. This distinction is preserved even if we generalize the selection rule to select a set of candidate contexts. Under such a generalization, the context model would map every history to a set of candidate contexts, ie., $s : \Sigma^* \rightarrow 2^D$, while an extension

model would map every history and symbol to a set of candidate contexts, ie., $s : \Sigma^* \times \Sigma \rightarrow 2^D$.

Our extension selection rule $s : \Sigma^* \times \Sigma \rightarrow D$ is defined implicitly by the set E of extensions currently in the model. The recursion in (2) says that each symbol should be predicted in its longest candidate context, while the expansion factor $\delta(h)$ says that longer contexts in the model should be trusted more than shorter contexts when combining the predictions from different contexts. Due to limited training data, we should not always trust longer (ie., more specific) contexts over shorter (ie., more general) contexts! Below, we will use the MDL criterion to eliminate untrustworthy contexts from the model.

As defined here, the extension model class does not include any context interpolation or escape probabilities. Each symbol selects a single context, and the conditional probability of that symbol in that context will be estimated with respect to all histories for which that context is a candidate. We consider the problem of context mixing in other work [12].

2.1.1 Validity

Let us now introduce a sufficient condition for an extension model to induce a probabilty function on all strings Σ^n of a given length n.

An extension model ϕ is *valid* iff it satisfies the following constraints:

$$\begin{array}{ll} a. & \epsilon \in D \wedge E(\epsilon) = \Sigma \\ b. & \forall w \in D \; [\sum_{\sigma \in E(w)} \lambda(\sigma|w) \leq 1] \\ c. & \forall w \in D \; [E(w) = \Sigma \Rightarrow \sum_{\sigma \in E(w)} \lambda(\sigma|w) = 1] \end{array} \tag{4}$$

These constraints suffice to ensure that the model ϕ defines a probability function. Constraint (4a) states that every symbol has the empty string as a context. This guarantees that every symbol will always have at least one context in every history and that the recursion in (2) will terminate. Constraint (4b) states that the sum of the probabilities of the extensions $E(w)$ available in in a given context w cannot sum to more than unity. The third constraint (4c) states that the sum of the probabilities of the extensions $E(w)$ must sum exactly to unity when every symbol is available in that context (ie., when $E(w) = \Sigma$).

Lemma 2.1 $\forall y \in \Sigma^* \quad \forall \sigma \in \Sigma \quad [\; \tilde{p}(\Sigma|y) = 1 \Rightarrow \tilde{p}(\Sigma|\sigma y) = 1 \;]$

Proof. By the definition of the expansion factor $\delta(\sigma y)$.

Theorem 1 *If an extension model ϕ is valid, then* $\forall n \sum_{s \in \Sigma^n} \tilde{p}(s|\phi) = 1$.

Proof. By induction on n. For the base case, $n = 1$ and the statement is true by the definition of validity (constraints 4a and 4c). The induction step is true by lemma 2.1 and definition (1). $\Box$

2.1.2 Monotonicity

Extension models and context models both include a finite dictionary of contexts. Each context represents a subset of Σ^*, the set of all possible histories. We say a context dictionary D is *prefix monotonic* iff every prefix of a context is also a context, ie., $\forall x, y \in \Sigma^*\ [xy \in D \Rightarrow x \in D]$. Similarly, a context dictionary D is *suffix monotonic* iff every suffix of a context is also a context, ie., $\forall x, y \in \Sigma^*\ [xy \in D \Rightarrow y \in D]$. A dictionary D is *monotonic* iff it is either prefix or suffix monotonic. Conversely, a dictionary is *nonmonotonic* iff it is neither prefix nor suffix monotonic.

Most context models, such as Rissanen's context algorithms [10, 11] and PPM [3], include all strings whose frequency exceeds a fixed threshold. As a result, such models are both prefix *and* suffix monotonic. The same is true of the powerful class of interpolated Markov sources introduced by Jelinek and Mercer [6]. More recent context models, such as PPM* [4], are prefix nonmonotonic but remain suffix monotonic. To the best of our knowledge, we are the first to propose a truely nonmonotonic context model [12].

Intuitively, a prefix monotonic model is one in which the length of the longest candidate context can increase by at most one symbol at each time step. In contrast, a prefix nonmonotonic model is one in which the length of the longest candidate context may increase or decrease by an unbounded amount in a single time step. It is well known that the optimal context order depends on the history and is highly variable. For some histories, a short context gives better predictions while for other histories a long context is better. Therefore, a prefix monotonic model must at the very least contain many useless contexts, which only increase the risk of overfitting. And if the context selection rule always picks the longest candidate context, as in PPM and its relatives, then a prefix monotonic dictionary may seriously impair model performance.

A suffix monotonic model is one in which every suffix of the longest candidate context is also a candidate context. For most sources, most of these candidate contexts are redundant. That is, the divergence between the true $p(\cdot|w_1...w_n)$ in a given context $w_1 w_2 \ldots w_n$ and the true $p(\cdot|w_2...w_n)$ in its maximal proper suffix is insignificant for nearly all candidate contexts. Consequently, blending the predictions of a context with the predictions of its maximal proper suffix will have little benefit. Such blending will even become harmful if blending is used to estimate the relative probabilities of novel events — as in PPM and PPM* — because the novel events in a given context are also likely to be novel in the maximal proper suffix of that context.

In short, a monotonic context dictionary increases the risks of context selection and attenuates the benefits of context blending. By increasing the number of free parameters in the model, a monotonic dictionary also significantly increases the risks of parameter estimation (ie., overfitting). In contrast, a nonmonotonic model whose contexts are carefully chosen need not suffer from these problems. A monotonic model must have at least as many parameters as an

equivalent nonmonotonic model. And therefore, from an MDL perspective, the class of nonmonotonic models is inherently superior to the class of monotonic models.

2.2 Parameter Estimation

Let us now estimate the conditional probabilities $\lambda(\cdot|\cdot)$ required for an extension model. Traditionally, these conditional probabilities are estimated using string frequencies obtained from a training corpus. However, the fact that a string did not occur in the training corpus does not mean it will not occur in the future. This so-called "zero-frequency problem" is overcome by reserving a small amount of probability for events that did not occur in the training corpus. In particular, let $c(\sigma|w)$ be the number of times that the symbol σ followed the string w in the training corpus, and let $c(w)$ be the sum $\sum_{\sigma\in\Sigma} c(\sigma|w)$ of all its conditional frequencies. Then we might use Laplace's rule of succession to estimate $\lambda(\sigma|w)$ as (5).

$$\lambda_A(\sigma|w) = \frac{c(\sigma|w)+1}{c(w)+|\Sigma|} \tag{5}$$

Unfortunately, Laplace's rule leads to unnecessarily high model entropies when the alphabet size is large, the true entropy in a given context is small, and the training data is limited. Following Moffat [8], we first partition the conditional event space Σ in a given context w into two subevents: the symbols $q(w)$ that have previously occurred in context w and those that $\bar{q}(w)$ that have not. Formally, $q(w) \doteq \{\sigma : c(\sigma|w) > 0\}$ and $\bar{q}(w) \doteq \Sigma - q(w)$. We estimate $\lambda_C(q(w)|w)$ as $c(w)/(c(w)+\#(w))$ and $\lambda_C(\bar{q}(w)|w)$ as $\#(w)/(c(w)+\#(w))$ where $\#(w)$ is the total weight assigned to the novel events $\bar{q}(w)$ in the context w. Currently, we calculate $\#(w)$ as $\min(|q(w)|, |\bar{q}(w)|)$ so that highly variable contexts receive more flattening, but no novel symbol in $\bar{q}(w)$ receives more than unity weight. Next, $\lambda_C(\sigma|q(w), w)$ is estimated as $c(\sigma|w)/c(w)$ for the previously seen symbols $\sigma \in q(w)$ and $\lambda_C(\sigma|\bar{q}(w), w)$ is estimated uniformly as $1/|\bar{q}(w)|$ for the novel symbols $\sigma \in \bar{q}(w)$. Combining these estimates, we obtain our overall estimate (6).

$$\lambda_C(\sigma|w) = \begin{cases} \dfrac{c(\sigma|w)}{c(w)+\#(w)} & \text{if } \sigma \in q(w) \\ \dfrac{\#(w)}{|\bar{q}(w)|\,(c(w)+\#(w))} & \text{otherwise} \end{cases} \tag{6}$$

Unlike Moffat, our estimate (6) does not use escape probabilities or any other form of context blending. All novel events $\bar{q}(w)$ in the context w are assigned uniform probability. This is simpler but obviously suboptimal.

Moffat's flattening "method C" introduces another parameter for every context w in an extension model whose extensions $E(w)$ are not a superset of the

previously seen symbols $q(w)$. In such a situation, we must know $|q(w)|$ in order to calculate $\#(w)$. Since this parameter will have little effect on the total model codelength, we will henceforth ignore it.

We note that both estimates (5,6) are incorrect when used in a context model that contains contexts that are proper suffixes of each other. In such a situation, the shorter context is only used when the longer context was *not* used. Let y and xy be two distinct contexts in a context model ϕ. Then the context y will never be used when the history is $\Sigma^* xy$. Therefore, our estimate of $\lambda(\cdot|y)$ should be conditioned on the fact that the longer context xy did not occur. The interaction between candidate contexts can become quite complex, and we consider this problem in other work [12].

Parameter estimation is only a small part of the overall model estimation problem. Not only do we have to estimate the parameters for a model, we have to find the right parameters to use! To do this, we proceed in two steps. First, in section 2.3, we use the minimum description length (MDL) principle to quantify the total merit of a model with respect to a training corpus. Next, in section 2.4, we use our MDL codelengths to derive a practical model selection algorithm with which to find a good model in the vast class of all extension models.

2.3 Codelength Formulas

The goal of this section is to establish the proper tension between model complexity and data complexity, in the fundamental units of information. Although the MDL framework obliges us to propose particular encodings for the model and the data, our goal is not to actually encode the data or the model.

Given an extension model ϕ and a text corpus T, $|T| = t$, we define the total codelength $L(T, \phi|\Phi)$ relative to the model class Φ using a 2-part code.

$$L(T, \phi|\Phi) = L(\phi|\Phi) + L(T|\phi, \Phi) \tag{7}$$

Since conditioning on the model class Φ is always understood, we will henceforth suppress it in our notation.

Firstly, we will encode the text T using the probability model ϕ and an arithmetic code, obtaining the following codelength.

$$L(T|\phi) = -\log \tilde{p}(T|\phi) \tag{8}$$

Next, we will encode the model ϕ in two parts, using enumerative codes. The dictionary D of contexts will be encoded as a labeled suffix tree whose internal vertices have degrees varying between 1 and $|\Sigma|$. The function λ will be encoded as a finite set of integers, namely, as the conditional frequencies $c(\cdot|\cdot)$ used to estimate the conditional probabilities in the model.[3]

[3]Again, our goal is to establish the proper tension between model complexity and data

Context models and extension models support slightly different encoding schemes. For a context model ϕ, we have $L(\phi) = L(D) + L^c(c|D, E)$. In this encoding, we need not encode the extensions E because it contains no information – every symbol is available in every context. However, we must encode the conditional frequencies $c(\sigma|w)$ for every context in D and every symbol in Σ. For an extension model ϕ, we have $L(\phi) = L(D) + L^e(c, E|D)$. In this encoding, we must encode the extensions E as well as the conditional frequencies $c(E(w)|w)$ for every context w in D and every symbol in $E(w)$.

For both model classes, the dictionary D of contexts forms a suffix tree containing n_i vertices with branching factor i. The tree contains $n = \sum_{i=1}^{m} n_i$ internal vertices and n_0 leaf vertices. There are $(n_0+n_1+\ldots+n_m-1)!/n_0!n_1!\ldots n_m!$ such trees [7, p.587]. Accordingly, this tree may be encoded with an enumerative code using $L(D)$ bits.

$$\begin{aligned} L(D) = \; & L_Z(n) + \log \left(\begin{array}{c} n+m-1 \\ m-1 \end{array} \right) + \log \frac{(n_0+n_1+\ldots+n_m-1)!}{n_0!n_1!\ldots n_m!} \\ & + \sum_{i=1}^{m-1} n_i \log \left(\begin{array}{c} m \\ i \end{array} \right) + log(n+1) + \log \left(\begin{array}{c} n \\ |\lfloor D \rfloor| \end{array} \right) \end{aligned} \tag{9}$$

where $\lfloor D \rfloor$ is the set of all contexts in D that are proper suffixes of another context D, ie.,

$$\lfloor D \rfloor \doteq \{y : y \in D \wedge \exists x \in \Sigma^+ \; [xy \in D]\}$$

The first term encodes the number n of internal vertices using the Elias code. The second term encodes the counts $\{n_1, n_2, \ldots, n_m\}$. Given the frequencies of these internal vertices, we may calculate the number n_0 of leaf vertices as $n_0 = 1 + n_2 + 2n_3 + 3n_4 + \ldots + (m-1)n_m$. The third term encodes the actual tree (without labels) using an enumerative code. The fourth term assigns labels (ie., symbols from Σ) to the edges in the tree. At this point the decoder knows all contexts which are not proper suffixes of other contexts, ie., $D - \lfloor D \rfloor$. The fourth term encodes the magnitude of $\lfloor D \rfloor$ as an integer bounded by the number n of internal vertices in the suffix tree. The fifth term identifies the contexts $\lfloor D \rfloor$ as interior vertices in the tree that are proper suffices of another context in D.

Finally, we encode the frequencies $c(\sigma|w)$ used to estimate the model parameters. For this encoding, we use a different encoding for context models and extension models.

complexity, not to efficiently compress the training data. Encoding the conditional frequencies of every parameter $\langle w, \sigma \rangle$ in the model allows us to establish the proper tension between model complexity and data complexity, and so it is the right thing to do here even though it is obviously an inefficient way to compress the past data.

The following codelength is used for context models.

$$L^c(c|D,E) = L_Z(c(\epsilon)) + \sum_{w \in D} \log \left(\begin{array}{c} c(w) + |\lceil w \rceil| \\ c(w) \end{array} \right) + \sum_{w \in D} \log \left(\begin{array}{c} c(w) + m - 1 \\ m - 1 \end{array} \right) \tag{10}$$

where $c(\epsilon) = |T|$ is the frequency of the empty context ϵ. The first term encodes $|T|$ using the Elias code. The second term represents the recursive partition of the context frequencies $c(w)$ into the context frequencies $c(\lceil w \rceil)$ for those contexts whose maximal proper suffix in D is w. The third term, which completely dominates the total codelength, partitions $c(w)$ into $c(\sigma|w)$ for every symbol and context.

The following codelength is used for extension models. Let m_i be the number of contexts that have exactly i extensions, ie., $m_i \doteq |\{w : |E(w)| = i\}|$. Observe that $\sum_{i=1}^{m} m_i = |D|$.

$$\begin{aligned} L^e(c,E|D) = \; & L_Z(c(\epsilon)) + \sum_{w \in D} \log \left(\begin{array}{c} c(w) + |\lceil w \rceil| \\ c(w) \end{array} \right) \\ & + \sum_{w \in D} \log \left(\begin{array}{c} c(w) + |E(w)| \\ |E(w)| \end{array} \right) + \log \left(\begin{array}{c} |D| + m - 1 \\ m - 1 \end{array} \right) \\ & + \log \left(\begin{array}{c} |D| \\ \{m_i\} \end{array} \right) + \sum_{i=1}^{m} m_i \log \left(\begin{array}{c} m \\ i \end{array} \right) \end{aligned} \tag{11}$$

As in $L^c(c|D,E)$ (10), the first term encodes $|T|$ and the second term recursively partitions $c(w)$ into $c(\lceil w \rceil)$ for every context w. The third term partitions the context frequency $c(w)$ into the available extensions $c(E(w)|w)$ and the "unallocated frequency" $c(\Sigma - E(w)|w) = c(w) - c(E(w)|w)$ in the context w. The last three terms represent the encoding of E. The fourth term represents the encoding of $\{m_i\}$ while the fifth term represents the encoding $|E(w)|$ for each w in D. The sixth and final term represents the encoding of $E(w)$ as a subset of Σ for each w in D.

The formula (11) does not encode the weight $\#(w)$ necessary to estimate $\lambda_C(\sigma|w)$.

For reasons of simplicity, our encoding of the conditional frequencies $c(\cdot|\cdot)$ fails to take advantage of the fact that $c(y) \geq c_{\min}$ and that $c(\sigma|y) \geq \sum_{xy \in D} c(\sigma|xy)$. To take advantage of the latter fact, we might introduce a variable $\hat{c}(\sigma|y) = c(\sigma|y) - c(\sigma|\lceil y \rceil)$ that represents the conditional frequency of σ in the histories $\Sigma^* y - \Sigma^* \lceil y \rceil$, where $\lceil y \rceil$ denotes all contexts whose maximal proper suffix with respect to D is y, ie., $\lceil y \rceil \doteq \{xy : xy \in D \wedge \lfloor xy \rfloor = y\}$. Since $\hat{c}(\sigma|w)$ can be significantly less than $c(\sigma|w)$, we should encode $\hat{c}(\cdot|\cdot)$ instead of $c(\cdot|\cdot)$ and then calculate the true conditional frequencies $c(\cdot|\cdot)$ from $\hat{c}(\cdot|\cdot)$ [12]. To take advantage of the former fact, that $c(y) \geq c_{\min}$, we should encode $c(y) - c_{\min} \cdot \left| \lceil y \rceil^+ \right|$

instead of $c(y)$, where $\lceil y \rceil^+$ denotes all contexts xy that are not proper suffixes of any other context in D.

Note that the codelength (11) for extension models may be further reduced if we encode all of E directly as a prefix tree, so that $L(\phi) = L(E) + L^e(c|E)$. This would also result in a more efficient code for $L^e(c|E)$. We have chosen the less efficient encoding (11) for two reasons. Firstly, it results in a simpler incremental cost formula (see section 2.4). Secondly, it allows us to more clearly express the tradeoffs between context models and extension models.

2.4 Model Selection

The final component of our contribution is a model selection algorithm for the extension model class Φ. Our algorithm repeatedly refines the accuracy of our model in increasingly long contexts. Adding a new context to the model will decrease the codelength of the data and increase the codelength of the model. Accordingly, we add a new context to the model only if doing so will decrease the total codelength of the data *and* the model.

Our algorithm performs a breadth-first traversal of the prefix tree for Σ^∞ and for each history h (ie., vertex) in that tree, we compare the direct estimate $\lambda(\cdot|h)$ of the conditional probabilities in that history to the model's current approximation $\tilde{p}(\cdot|h)$ of those conditional probabilities. Adding the history h to the model would decrease the codelength of the training data T and increase the codelength of the model ϕ. Let $\Delta L_T(h)$ be the incremental reduction in the codelength of the data T due to h and let $\Delta L_\phi(h)$ be the incremental increase in the codelength of the model ϕ due to h. If the incremental benefit $\Delta L_T(h)$ of h exceeds its incremental cost $\Delta L_\phi(h)$, then we add h to the model. The design of the model class ensures that the addition of h will only change our model's performance in histories that end in h.

The incremental benefit of adding context w to a context model ϕ is defined as:

$$\Delta L_T(w) \doteq L(T|\phi) - L(T|\phi \cup \{w\}) \tag{12}$$

The incremental cost of adding context w to a context model ϕ is defined as:

$$\Delta L_\phi(w) \doteq L(\phi) - L(\phi \cup \{w\}) \tag{13}$$

The incremental costs and benefits of adding a context are different for the context model and the extension model. Let us consider each in turn.

2.4.1 Context Model

Adding w to a context model improves the accuracy of our prediction in the context w; otherwise it has no effect. Accordingly, we approximate the incremental

benefit $\Delta L_T^c(w)$ as follows.

$$\Delta L_T^c(w) = \sum_\sigma c(\sigma|w) \log \frac{\lambda(\sigma|w)}{\tilde{p}(\sigma|w,\phi)} \tag{14}$$

Note that flattening in $\lambda(\cdot|w)$ may cause the incremental benefit $\Delta L_T^c(w)$ of an infrequent context w to be negative.[4]

Keeping only the significant, monotonically nondecreasing terms, we may approximate the incremental cost $\Delta L_\phi^c(w)$ as follows.

$$\Delta L_\phi^c(w) \approx \log |D| + \log c(\lfloor w \rfloor) + \log \left(\begin{array}{c} c(w)+m-1 \\ m-1 \end{array} \right) \tag{16}$$

The first term represents the incremental increase in the size of the suffix tree encoding of D. The second term represents (an upper bound on) the incremental increase in the encoding of $c(\cdot)$. The third term represents the additional cost of encoding the conditional frequencies $c(\cdot|w)$ for the new context w. Since the third term completely dominates the codelength of model, the incremental cost may be further simplified by dropping the first and second terms.

Having quantified the incremental cost and benefit of adding a given context to a given model, let us design a heuristic estimation algorithm for the context model based on these formulas. Our Refine(D,n) procedure considers all contexts of length n, and adds the profitable ones to D. We add a context w to the model ϕ when its predictions are significantly better than those currently provided by the context $\lfloor w \rfloor$.

Our breadth-first search considers shorter contexts before longer ones, and consequently the decision to add a profitable context y may significantly decrease the benefit of a more profitable context xy, particularly when $c(xy) \approx c(y)$. For example, consider a source with two hidden states. In the first state, the source generates the alphabet $\Sigma = \{0,1,2\}$ uniformly. In the second state, the source generates the string `"012"` with certainty. With appropriate state transition probabilities, the source generates strings where $c(0) \approx c(1) \approx c(2)$, $c(2|1)/c(1) \gg c(2|\epsilon)/c(\epsilon)$, and $c(2|01)/c(01) > c(2|1)/c(1)$. In such a situation, the best context model includes the contexts `"0"` and `"01"` along with the empty context ϵ. However, the divergence heuristic will first determine that the context `"1"` is profitable relative to the empty context, and add it to the model. Now the profitability of the better context `"01"` is reduced, and the divergence heuristic may therefore not include it in the model. This problem is best solved with

[4] Note also that flattening in $\lambda(\cdot|w)$ means that (14) gives superior results than the weighted Kullback-Liebler divergence (15).

$$c(w) \sum_\sigma \lambda(\sigma|w) \log \frac{\lambda(\sigma|w)}{\tilde{p}(\sigma|w,\phi)} \tag{15}$$

.

a best first search. Our current implementation uses a breadth first search to limit the computational complexity of model selection.

In the following pseudo-code, C_n contains all candidate contexts of length n. D_n contains all contexts of length n in the model. R contains all rejected strings, which are candidates for swapping in. $n_{\max}$ is the longest context that we are willing to consider, and $c_{\min}$ is the minimum frequency that a context must have in order to be considered. Currently, we set $c_{\min}$ equal to the square root of the alphabet size m. This helps ensure that any uniformity in $\lambda(\cdot|w)$ is not solely due to the combination of flattening and insufficient training data.

```
Refine(D,n)
1.  D_n := {}; R := {};
2.  C_n := {w : w ∈ C_{n-1}Σ ∧ c(w) > c_min};
3.  if ((n > n_max) ∨ (|C_n| = 0)) then return;
4.  for w ∈ C_n
5.      if ΔL_T^c(w) > ΔL_φ^c(w)
6.        then D_n := D_n ∪ {w};
7.        else R := R ∪ {w};
8.  D := D ∪ D_n;
9.  Swap(D,R,n-1);
10. Refine(D,n+1);
```

Table 1: D_n is the set of contexts of length n that will be added to model. C_n is the set of candidate contexts, obtained from the corpus. R is the set of rejected strings of length n that are candidates for swapping in.

2.4.2 Extension Model

The principal virtue of the extension model is that it allows us to add a single parameter to the model, unlike the context model. However, given the code-length formula (11) and the definition of $\tilde{p}(\cdot|\cdot,\phi)$ in (2), the incremental cost and benefit of adding a single parameter to a given context cannot be accurately approximated in isolation from any other parameters that might be added to that context. Accordingly, the incremental cost of adding the *set* Σ' of extensions to the context w is defined as (17) while the incremental benefit is defined as (18).

$$\Delta L^e_\phi(w,\Sigma') \doteq L(\phi \cup (\{w\} \times \Sigma')) - L(\phi) \tag{17}$$

$$\Delta L^e_T(w,\Sigma') \doteq L(T|\phi) - L(T|\phi \cup (\{w\} \times \Sigma')) \tag{18}$$

Keeping only significant terms that are monotonically nondecreasing, we approximate the incremental cost $\Delta L^e_\phi(w,\Sigma')$ as

$$\Delta L^e_\phi(w,\Sigma') \approx \log|D| + \log c(\lfloor w \rfloor) + \log \left(\begin{array}{c} c(w)+|\Sigma'| \\ |\Sigma'| \end{array} \right) + \log \left(\begin{array}{c} m \\ |\Sigma'| \end{array} \right) \tag{19}$$

The first term represents the incremental increase in the size of the context dictionary D. The second term represents (an upper bound on) the cost of encoding $c(w)$. The third term represents the cost of encoding $c(\cdot|w)$ for $E(w)$. The fourth term represents the cost of encoding the candidate extensions $E(w) = \Sigma'$. Again, only the third and fourth terms are signficant.

Let us now consider the incremental benefit of adding the extensions Σ' to a given context w. The addition of a single parameter $\langle w,\sigma\rangle$ to the model ϕ will immediately change $\lambda(\sigma|w)$, by definition of the model class. Any change to $\lambda(\cdot|w)$ will also change the expansion factor $\delta(w)$ in that context, which may in turn change the conditional probabilities $\tilde{p}(\Sigma - E(w)|w,\phi)$ of symbols not available in that context. Thus the incremental benefit of adding the extensions Σ' to the context w may be calculated as

$$\Delta L^e_T(w,\Sigma') = c(\Sigma-\Sigma'|w)\log\frac{1-\lambda(\Sigma'|w)}{1-\tilde{p}(\Sigma'|w,\phi)} + \sum_{\sigma'\in\Sigma'} c(\sigma'|w)\log\frac{\lambda(\sigma'|w)}{\tilde{p}(\sigma'|w,\phi)} \tag{20}$$

The first term represents the incremental benefit (in bits) for evaluating $\Sigma-\Sigma'$ in the context w using the more accurate expansion factor $\delta(w)$. The second term represents the incremental benefit (in bits) of using the direct estimate $\lambda(\sigma'|w)$ instead of the interpolated probability $\tilde{p}(\sigma'|w,\phi)$ in the context w. Note that $\lambda(\sigma'|w)$ may be more or less than $\tilde{p}(\sigma'|w,\phi)$.

Now the incremental cost and benefit of adding a *single* extension $\langle w,\sigma\rangle$ to a model that already contains the extensions $\langle w,\Sigma'\rangle$ may be defined as (21) and (22), respectively.

$$\Delta L^e_\phi(w,\Sigma',\sigma) \doteq \Delta L^e_\phi(w,\Sigma'\cup\{\sigma\}) - \Delta L^e_\phi(w,\Sigma') \tag{21}$$

$$\Delta L_T^e(w, \Sigma', \sigma) \doteq \Delta L_T^e(w, \Sigma' \cup \{\sigma\}) - \Delta L_T^e(w, \Sigma') \tag{22}$$

These formulas (21,22) form the basis of our heuristic estimation algorithm below.

$\Delta L_\phi^e(w, \Sigma', \sigma)$ may be expressed directly as (23).

$$\Delta L_\phi^e(w, \Sigma', \sigma) = \log \frac{c(w) + |\Sigma'| + 1}{|\Sigma'| + 1} + \log \frac{m - |\Sigma'|}{|\Sigma'| + 1} \tag{23}$$

And using the benefit formula (20), $\Delta L_T^e(w, \Sigma', \sigma)$ may also be expressed directly as (24).

$$\begin{aligned} \Delta L_T^e(w, \Sigma', \sigma) = \quad & c(\sigma|w) \left[\log \frac{\lambda(\sigma|w)}{\tilde{p}(\sigma|w, \phi)} - \log \frac{1 - \lambda(\Sigma'|w)}{1 - \tilde{p}(\Sigma'|w, \phi)} \right] \\ & + c(\Sigma - \Sigma' \cup \{\sigma\}|w) \left[\log \frac{1 - \lambda(\Sigma' \cup \{\sigma\}|w)}{1 - \tilde{p}(\Sigma' \cup \{\sigma\}|w, \phi)} \right. \\ & \left. - \log \frac{1 - \lambda(\Sigma'|w)}{1 - \tilde{p}(\Sigma'|w, \phi)} \right] \end{aligned} \tag{24}$$

Recall that the addition of the parameter $\langle w, \sigma \rangle$ immediately changes $\tilde{p}(\sigma|w, \phi)$ and also changes $\tilde{p}(\Sigma - (\Sigma' \cup \{\sigma\})|w, \phi)$, by changing the expansion factor $\delta(w)$ for the context w. Thus, the first term represents the improvement to $\tilde{p}(\sigma|w, \phi)$ while the second term represents the improvement to $\tilde{p}(\Sigma - (\Sigma' \cup \{\sigma\})|w, \phi)$. Consequently, a parameter $\langle w, \sigma \rangle$ whose observed frequency is zero may nonetheless provide a positive benefit.

Let us now use our incremental cost/benefit formulas to design a simple heuristic estimation algorithm for the extension model. The algorithm consists of two subroutines. Refine(D,E,n) augments the model with all individually profitable extensions of contexts of length n. It rests on the assumption that adding a new context does not change the model's performance in the shorter contexts. Extend(w) determines all profitable extensions of the candidate context w, if any exist. Since it is not feasible to evaluate the incremental profit of every subset of Σ, Extend(w) uses a greedy heuristic that repeatedly augments the set of profitable extensions of w by the single most profitable extension until it is not longer profitable to do so.

Refine(D,E,n)

1. $D_n := \{\}; E_n := \{\};$
2. $C_n := \{w : w \in C_{n-1}\Sigma \wedge c(w) > c_{\min}\};$
3. if $((n > n_{max}) \vee (|C_n| = 0))$ then return;
4. for $w \in C_n$
5. $\quad S :=$ Extend(w);
6. $\quad$ if $|S| > 0$ then $D_n := D_n \cup \{w\}$; $E_n(w) := S$;
7. $D := D \cup D_n$; $E := E \cup E_n$;
8. Refine(D,E,$n+1$);

Table 2: C_n is the set of candidate contexts of length n, obtained from the training corpus. D_n is the set of profitable contexts of length n, while E_n is the set of profitable extensions of those contexts.

Extend(w)

1. $S := \{\};$
2. $\sigma :=$ argmax$_{\sigma \in \Sigma} \left\{ \Delta L_T^e(w, \{\sigma\}) - \Delta L_\phi^e(w, \{\sigma\}) \right\}$
3. while $(\Delta L_T^e(w, S, \sigma) > \Delta L_\phi^e(w, S, \sigma))$
4. $\quad S := S \cup \{\sigma\};$
5. $\quad \sigma :=$ argmax$_{\sigma \in \Sigma - S} \left\{ \Delta L_T^e(w, S, \sigma) - \Delta L_\phi^e(w, S, \sigma) \right\}$
6. return(S);

Table 3: The loop in lines 3-5 repeatedly finds the single most profitable symbol σ with which to augment the set S of profitable extensions. In case of ties, all maximally profitable symbols should be included in S.

In the next section 3, we examine the performance of our model class on the Brown Corpus, an eclectic collection of English prose.

3 Empirical Results

By means of the following experiments, we hope to: (i) demonstrate the utility of our context modeling techniques; (ii) illustrate the fundamental tension between model complexity and data complexity; and (iii) substantiate the correlation between minimizing the total codelength and reducing the predictive error.

All results are based on the Brown corpus [5], an eclectic collection of English prose drawn from 500 sources across 15 genres. The irregular and nonstationary nature of this corpus is known to pose an exacting test for statistical language models. We use the first 90% of each file in the corpus to estimate our models, and then use the remaining 10% of each file in the corpus to evaluate the models. Each file contains approximately 2000 words. Due to limited computational resources, we set $n_{\max} = 10$, $c_{\min} = 8$, and restrict our our alphabet size to 70 (ie., all printing ascii characters, ignoring case distinction, and combining white space).

The best reported results on the Brown Corpus are 1.75 bits/char using a large interpolated trigram word model whose parameters are estimated using over 600,000,000 words of proprietary training data [2]. The use of proprietary training data means that these results are not independently repeatable. In contrast, our results are obtained using only 900,000 words of widely available training data and may be independently verified by anyone with the inclination to do so. The amount of training data is known to be a significant factor in model performance. Given a sufficiently rich dictionary of words and a sufficiently large training corpus, a model of word sequences is likely to outperform an otherwise equivalent model of character sequences. For these three reasons – repeatability, training corpus size, and the advantage of word models over character models – the results reported in [2] are not directly comparable to those reported here.

Section 3.1 compares the statistical efficiency of the various context model classes. Next, section 3.2 anecodatally examines the complex interactions among the parameters of an extension model.

3.1 Model Class Comparison

Given the tremendous risk of overfitting, the most important property of a model class is arguably its statistical efficiency. Informally, statistical efficiency measures the effectiveness of individual parameters in a given model class. A high efficiency indicates that our model class provides a good description of the data. Conversely, a low efficiency indicates that the model class does not adequately describe the observed data.

In this section, we compare the statistical efficiency of three model classes: context models, extension models, and traditional fixed-length Markov processes (ie., uninterpolated n-grams). We include three variants of the context model: our nonmonotonic context model (NCM) as well as Rissanen's original two context models (CM_1 [10] and CM_2 [11]). The comparison is based on two criteria of statistical efficiency: bits/parameter on the test message, and bits/order on the test message.

According to the first criteria of statistical efficiency, the best model is the one that achieves the lowest test message entropy using the fewest parameters. This criteria measures the statistical efficiency of a model class according to traditional model validation methodology, tempered by a healthy concern for overfitting. Figure 1 graphs the number of model parameters required to achieve a given test message entropy for each of the three model classes. The extension model class is the overwhelming winner. (This is particularly striking when the number of parameters is plotted on a linear scale.) For example, one of our extension models saves 0.98 bits/char over the trigram while using less than 1/3 as many parameters. Given the logarithmic nature of codelength and the scarcity of training data, this is a significant improvement.

According to the second criteria of statistical efficiency, the best model is one that achieves the lowest test message entropy for a given model order. This criteria is widely used in the language modeling community, in part because model order is typically — although not necessarily — related to both the number of model parameters and the amount of computation required to estimate the model. Figure 2 compares model order to test message entropy for each of the three model classes. As the order of the models increases from 1 (ie., unigram) to 7, we naturally expect the test message entropy to approach a lower bound, which is itself bounded below by the true source entropy. By this criteria, extension models are better than context models, nonmonotonic dictionaries are better than monotonic dictionaries, and variable length contexts are better than fixed-length contexts.

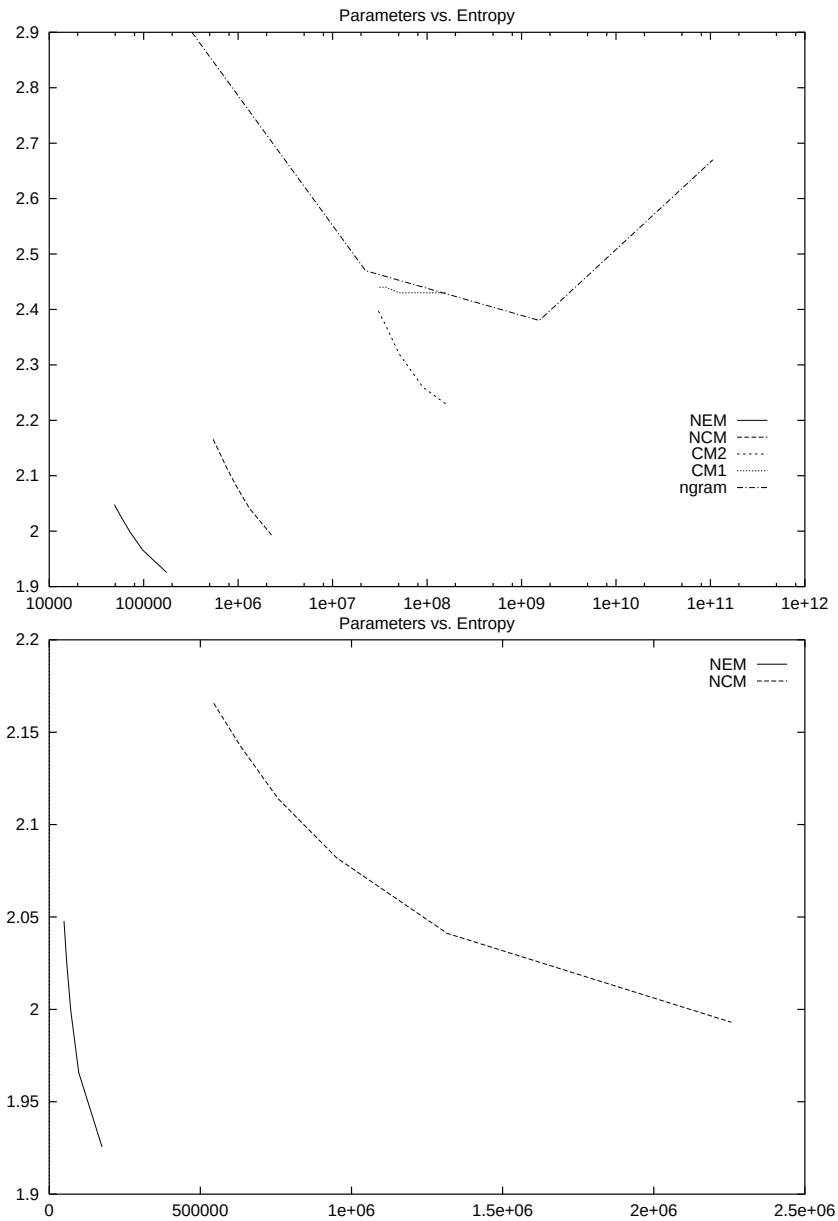

Figure 1: The relationship between the number of model parameters and test message entropy for three model classes. The number of parameters (ie., horizontal axis) is plotted on a logarithmic scale for the top graph and on a linear scale for the bottom graph. The most striking fact about this graph is the tremendous statistical efficiency of the extension model class, and the extreme inefficiency of the traditional n-gram.

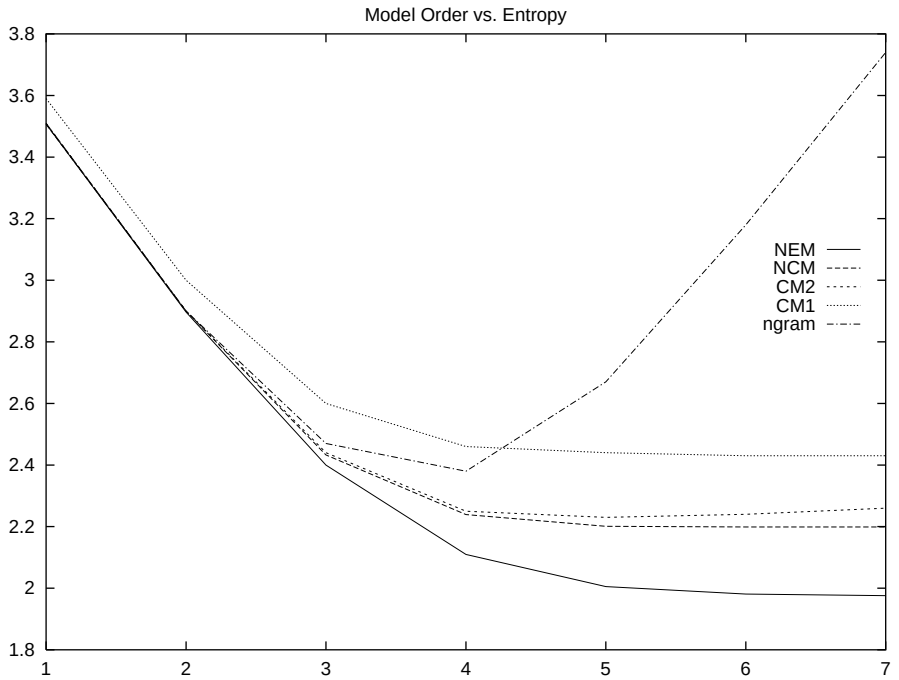

Figure 2: The relationship between model order and test message entropy for the three model classes. The context and extension models are estimated by the divergence heuristic with the simplified cost/benefit formulas. The extension model class is the clear winner by this criteria as well.

3.2 Anecdotes

It is also worthwhile to interpret the parameters of the extension model estimated from the Brown Corpus, to better understand the interaction between our model class and our heuristic model selection algorithm. Table 4 contains the four most predictable contexts and four least predictable contexts of length at least 7.

w	$E(w)$	$H(\tilde{p}(\cdot\|w,\phi))$
"␣centra"	l	0.0592
"ignifica"	n	0.0622
"ndepende"	n	0.0644
"␣struct"	u	0.0703
"father␣"	t	4.3988
"␣table␣"	/,1,2,5,7,t	4.4995
"umber␣of␣"	a,h,t,y	4.5128
"mately␣"	$,*,1,2,3,4,5,e	4.5360

Table 4: The first four contexts have the lowest entropies of any contexts of order at least 7 in our extension model for the Brown corpus. As should be obvious to any speaker of English, these are all highly predictable contexts, with the best guess included as an extension in the model. Conversely, the last four contexts have among the highest entropies, which accords with the perceived difficulty of predicting the next character in these contexts and the relatively large number of extensions in these contexts.

According to the divergence heuristic, the decision about whether or not to add an extension $\langle w, \sigma \rangle$ is made relative to that context's maximal proper suffix $\lfloor w \rfloor$ in D as well as any other extensions in the context w. An extension $\langle w, \sigma \rangle$ will be added only if the direct estimate of its conditional probability is significantly different from its conditional probability in its maximal proper suffix after scaling by the expansion factor in the context w, ie., if $\lambda(\sigma|w)$ is significantly different than $\delta(w)\tilde{p}(\sigma|\lfloor w \rfloor)$.

This is illusrated by the three contexts and six extensions shown in Table 5, where $+E(w)$ includes all symbols in $E(w)$ that are more likely in w than they were in $\lfloor w \rfloor$ and $-E(w)$ includes all symbols in $E(w)$ that are less likely in w than they were in $\lfloor w \rfloor$.

w	$+E(w)$	$-E(w)$	$H(\tilde{p}(\cdot\|w,\phi))$	$\lfloor w \rfloor$
`"blish"`	e,i,m		1.9468	`"ish"`
`"o␣establish"`	␣		1.4692	`"blish"`
`"e␣establish"`	m	e	1.2285	`"blish"`

Table 5: The six extensions shown in this table illustrate the complex interaction between longer and shorter contexts and among extensions of the same context.

The substring *blish* is most often followed by the characters 'e', 'i', and 'm', corresponding to the relatively frequent word forms *publish*{*ed,er,ing*} and *establish*{*ed,ing,ment*}. Accordingly, the context `"blish"` has three positive extensions {`e,i,m`}, of which `e` has by far the greatest probability. The context `"blish"` is the maximal proper suffix of two other contexts in the model, `"o␣establish"` and `"e␣establish"`.

The substring *o establish* occurs most frequently in the gerund *to establish*, which is nearly always followed by a space. Accordingly, the context `"o␣establish"` has a single positive extension `"␣"`. The substring *o establish* is also found before the characters 'm', 'e', and 'i' in sequences such as *to establishments*, {*who,ratio,also*} *established*, and {*to,into,also*} *establishing*. Accordingly, the context `"o␣establish"` does not have any negative extensions.

In contrast, the substring *e establish* is overwhelmingly followed by the character 'm', rarely followed by 'e', and never followed by either 'i' or space. For this reason, the context `"e␣establish"` has a single positive extension {`m`} corresponding to the great frequency of the string *the establishment*. This context also has single negative extension {`e`}, corresponding to the fact that the character 'e' is still possible in the context `"e␣establish"` but considerably less likely than in that context's maximal proper suffix `"blish"`.

Since 'i' is reasonably likely in the context `"blish"` but completely unlikely in the context `"e␣establish"`, we may well wonder why the model does not

include the negative extension 'i' in addition to 'e' or even instead of 'e'. This puzzle is explained by the expansion factor as follows. Since the substring *e establish* is only followed by 'm' and 'e', the expansion factor $\delta(\texttt{"e␣establish"})$ is essentially zero after 'm' and 'e' are added to that context, and therefore $\tilde{p}(\Sigma - \{m, e\} \mid \texttt{"e␣establish"})$ is also zero. Thus, 'i' and space are both assigned nearly zero probability in the context `"e␣establish"`, simply because 'm' and 'e' get nearly all the probability in that context.

4 Related Work

The principal inspiration for this work comes from Rissanen's work on the minimum description length principle and on context modeling. We sought to recast Rissanen's online universal data compression algorithm [10, 11] in an offline setting using the traditional MDL criterion. We thereby hoped to accomplish two goals: (i) to improve the performance of this language model by using a more general model class, a better context selection rule, and a more powerful model selection algorithm; and (ii) to obtain an offline language model suitable for use in a spoken or written language transcription system.

Our central technical contributions to Rissanen's pioneering work are to generalize the context model class and to better manage the interaction among contexts. In Rissanen's first model selection algorithm [10], contexts are added to the model if they make strong predictions, ie., they have low entropy. In Rissanen's second model selection algorithm [11], a context σw and all of its siblings Σw are added to the model if they reduce the codelength of their parent context w. In our estimation algorithm, a context is added to the model if it's conditional probabilities are significantly different than those of its maximal proper suffix. Thus, our algorithm may add a long context to the model even if its entropy is greater than the entropy of its maximal proper suffix. This is the principal reason that our models outperform Rissanen's models.

Aspects of our work are similar to (but independent of) the recent work of Ron et.al [13]. Our model class properly includes their class of prediction suffix trees, and we both use a form of the Kullback-Liebler (KL) divergence to decide whether or not to add a given context to the model. However, our work differs from theirs in its underlying philosophy and improves on theirs in a number of technical respects.

The philosophies that underly our work and theirs are different. The goal of their work is to accurately approximate the "underlying" probability function. Accordingly, they must measure the distance between the model's conditional probabilities in a given history and a direct estimate of those probabilities, and they use the KL divergence as a convenient distance measure. In contrast, our goal is to minimize the total codelength of the data and model. To do this, we measure the incremental benefit of adding a context w to the model as the number of bits required to encode σ in the history w using $\tilde{p}(\sigma|w)$ (ie., without

w in the model) minus the number of bits required to encode σ in the history w using $\lambda(\sigma|w)$ (ie., with w in the model). Coincidentally for us, this difference is related to (but not identical to) the KL divergence.

Our work improves on theirs in three technical respects. Firstly, our more general nonmonotonic extension model class improves on their prediction suffix trees by significantly reducing the number of states and free parameters in the model. Secondly, unlike their heuristic model selection algorithm, our algorithm has no hand-tunable parameters. In our work, the MDL criteria tells us exactly how to maintain the fundamental balance between model complexity and data complexity. Thirdly, their heuristic will behave badly for the following reason. Their dissimilarity measure compares the flattened estimates of the conditional probabilities in two contexts. As the total frequency of the context decreases, the entropy of its conditional probability estimates must invariably increase. Thus, their heuristic fails to distinguish between a dissimilarity due to insufficient data in the longer context and a true underlying dissimilarity between the shorter and longer contexts. They exacerbate this problem by weighting the dissimilarity between $\tilde{p}(\sigma|w)$ and $\lambda(\sigma|w)$ by the flattened estimate of the conditional probability $\lambda(\sigma|w)$ instead of the true conditional frequency $c(\sigma|w)$. In contrast, our measure of the dissimilarity between $\tilde{p}(\sigma|w)$ and $\lambda(\sigma|w)$ is weighted by the actual frequency $c(\sigma|w)$, which goes a long way to solving this problem (see p.11). They did not report any compression results in their paper, but for these reasons we are confident that our techniques will give significantly better predictions than their techniques.

5 Conclusion

Our results highlight the fundamental tension between model complexity and data complexity. If the model complexity does not match the data complexity, then both the total codelength of the past observations *and* the predictive error increase. In other words, simply increasing the number of parameters in the model does not necessarily increase predictive power of the model. Therefore, it is necessary to have a a fine-grained model along with a heuristic model selection algorithm to guide the expansion of the model in a principled manner.

References

[1] Bell, T. C., Cleary, J. G., and Witten, I. H. *Text Compression.* Prentice Hall, Englewood Cliffs, NJ, 1990.

[2] Brown, P., Pietra, V. D., Pietra, S. D., Lai, J., and Mercer, R. An estimate of an upper bound for the entropy of English. *Computational Linguistics 18* (1992), 31–40.

[3] Cleary, J., and Witten, I. Data compression using adaptive coding and partial string matching. *IEEE Trans. Comm. COM-32*, 4 (1984), 396–402.

[4] Cleary, J. G., Teahan, W., and Witten, I. H. Unboundedd length contexts for PPM. In *Proceedings of 5th Data Compression Conference* (Los Alamitos, CA, March 28–30 1995), J. Storer and M. Cohn, Eds., IEEE, IEEE Computer Society Press, pp. 52–61.

[5] Francis, W. N., and Kucera, H. *Frequency analysis of English usage: lexicon and grammar*. Houghton Mifflin, Boston, 1982.

[6] Jelinek, F., and Mercer, R. L. Interpolated estimation of Markov source parameters from sparse data. In *Pattern Recognition in Practice* (Amsterdam, May 21–23 1980), E. S. Gelsema and L. N. Kanal, Eds., North Holland, pp. 381–397.

[7] Knuth, D. E. *The Art of Computer Programming*, 1 ed., vol. 1. Addison-Wesley, Reading, MA, 1968.

[8] Moffat, A. A note on the PPM data compression algorithm. Research Report 88/7, Department of Computer Science, University of Melbourne, Parkville, Victoria, Australia, 1988.

[9] Rissanen, J. Modeling by shortest data description. *Automatica 14* (1978), 465–471.

[10] Rissanen, J. A universal data compression system. *IEEE Trans. Information Theory IT-29*, 5 (1983), 656–664.

[11] Rissanen, J. Complexity of strings in the class of Markov sources. *IEEE Trans. Information Theory IT-32*, 4 (1986), 526–532.

[12] Ristad, E. S., and Thomas, R. G. Context models in the MDL framework. In *Proceedings of 5th Data Compression Conference* (Los Alamitos, CA, March 28–30 1995), J. Storer and M. Cohn, Eds., IEEE, IEEE Computer Society Press, pp. 62–71.

[13] Ron, D., Singer, Y., and Tishby, N. The power of amnesia. Tech. rep., Institute of Computer Science, Hebrew University, Jerusalem, 1994.

A Summary of Notation

Model Class.

ϕ	–	nonmonotonic extension model $\langle \Sigma, D, E, \lambda \rangle$
Σ	–	finite alphabet Σ of size m.
D	–	finite dictionary of contexts, $D \subset \Sigma^*$.
E	–	finite set of context extensions, $E \subseteq D \times \Sigma$, where $E(w)$ is all symbols available in context w.
$\lambda(\cdot\|\cdot)$	–	conditional probability function estimated from $c(\cdot\|\cdot)$ frequencies.
$\lambda_A(\cdot\|\cdot)$	–	$\lambda(\cdot\|\cdot)$ estimated using Laplacian, ie., method A of [1].
$\lambda_C(\cdot\|\cdot)$	–	$\lambda(\cdot\|\cdot)$ estimated with $q(w)$ partition, ie., method C of [1].
$\tilde{p}(\sigma\|w, \phi)$	–	conditional probability interpolated from $\lambda(\cdot\|\cdot)$ estimates in shorter contexts.

Frequencies.

T	–	training corpus used to estimate $\lambda()$, $\|T\| = t$
$c(\sigma\|w)$	–	number of times that $w\sigma$ occurs in training corpus T.
$c(w)$	–	$\sum_\sigma c(\sigma\|w)$, ie., the number of times that w occurs in training corpus, modulo boundary effects.

Exclusions.

$q(w)$	–	all symbols that have already been seen in context w, after accounting for implicit context growth.
$\bar{q}(w)$	–	all symbols that could appear in context w but haven't.
$\#(w)$	–	total weight assigned to novel symbols $\bar{q}(w)$ in context w.

Context Operations.

$\lfloor w \rfloor$	–	maximal proper suffix of w in the context dictionary D.
$\lceil y \rceil$	–	all contexts that have y as their maximal proper suffix.

Divergence Heuristic.

$\Delta L^c_\phi(w)$	–	incremental cost (in bits) of adding context w with all extensions to context model ϕ. (p.11)
$\Delta L^c_T(w)$	–	incremental benefit (in bits) of adding context w with all extensions to context model ϕ. (p.11)
$\Delta L^e_\phi(w, \Sigma')$	–	incremental cost (in bits) of adding extensions Σ' to extension model ϕ in the context w, given that $E(w) = \{\}$. (p.13)
$\Delta L^e_T(w, \Sigma')$	–	incremental benefit (in bits) of adding extensions Σ' to extension model ϕ in the context w, given that $E(w) = \{\}$. (p.13)
$\Delta L^e_\phi(w, \Sigma', \sigma)$	–	incremental cost (in bits) of adding single parameter $\langle w, \sigma \rangle$ to extension model ϕ, given that $E(w) = \Sigma'$ already. (p.13)
$\Delta L^e_T(w, \Sigma', \sigma)$	–	incremental benefit (in bits) of adding single parameter $\langle w, \sigma \rangle$ to extension model ϕ, given that $E(w) = \Sigma'$ already. (p.14)