The Sequence Memoizer

Frank Wood*
Department of Statistics
Columbia University
New York, New York
fwood@stat.columbia.edu

Jan Gasthaus
Gatsby Computational
Neuroscience Unit
University College London
London, England
j.gasthaus@gatsby.ucl.ac.uk

Cédric Archambeau
Xerox Research Centre
Europe
Grenoble, France
cedric.archambeau@xerox.com

Lancelot James
Department of Information
Systems, Business Statistics
and Operations Management
Hong Kong University of
Science and Technology
Kowloon, Hong Kong
lancelot@ust.hk

Yee Whye Teh
Gatsby Computational
Neuroscience Unit
University College London
London, England
ywteh@gatsby.ucl.ac.uk

ABSTRACT

Probabilistic models of sequences play a central role in most machine translation, automated speech recognition, lossless compression, spell-checking, and gene identification applications to name but a few. Unfortunately, real-world sequence data often exhibit long range dependencies which can only be captured by computationally challenging, complex models. Sequence data arising from natural processes also often exhibit power-law properties, yet common sequence models do not capture such properties. The sequence memoizer is a new hierarchical Bayesian model for discrete sequence data that captures long range dependencies and power-law characteristics while remaining computationally attractive. Its utility as a language model and general purpose lossless compressor is demonstrated.

1. INTRODUCTION

It is an age-old quest to predict what comes next in sequences. Fortunes have been made and lost on the success and failure of such predictions. Heads or tails? Will the stock market go up by five percent tomorrow? Is the next card drawn from the deck going to be an ace? Does a particular sequence of nucleotides appear more often then usual in a DNA sequence? In a sentence, is the word that follows the United going to be States, Arab, Parcel, Kingdom, or something else? Using a probabilistic model of sequences fit to a particular set of data is usually an effective way of answering these kinds of questions.

Consider the general task of sequence prediction. For some sequences, the true probability distribution of the next symbol does not depend on the previous symbols in the sequence. For instance, consider flipping a coin that comes up heads with probability p or tails with probability $1-p$ every time it is flipped. Subsequent flips of such a coin are completely independent and have the same distribution (in statistics, such coin flips are called *independent and identically distributed* (iid)). In particular, heads will occur with probability p irrespective of whether previous flips have come up heads or tails. Assuming we observe a sequence of such coin flips, all we need to do is to estimate p in order to fully characterize the process that generated the data.

For more interesting processes the distribution of the next symbol often depends in some complex way on previous outcomes. One example of such a process is natural language (sequences of words). In English, the distribution of words that follow the single-word context United is quite different from the distribution of words that follow Cricket and rugby are amongst the most popular sports in the United. In the first case the distribution is relatively broad (though not nearly as broad as the distribution given no context at all), giving significant probability to words such as States, Kingdom, Airlines, and so forth, whereas in the second case the distribution is almost certainly highly peaked around Kingdom. Information from distant context (Cricket and rugby) impacts the distribution of the next word profoundly. Production of natural language is but one example of such a process; the real world is replete with other examples.

Employing models that capture long range contextual dependencies will often improve one's ability to predict what comes next, as illustrated in the example above. Of course, modeling the distribution of the next symbol emitted by a process will only be improved by consideration of longer contexts if the generating mechanism actually does exhibit

The work reported in this paper originally appeared in "A Hierarchical Bayesian Language Model based on Pitman-Yor Processes," published in the Proceedings of International Conference on Computational Linguistics and the Association for Computational Linguistics, 2006; "A Stochastic Memoizer for Sequence Data," published in the Proceedings of the International Conference on Machine Learning, 2009 and "Lossless compression based on the Sequence Memoizer" published in the Proceedings of the IEEE Data Compression Conference, 2010.

*Corresponding author

long range dependencies. Unfortunately, building models that capture the information contained in longer contexts can be difficult, both statistically and computationally. The sequence memoizer captures such long range dependencies in a way that is both statistically effective and scales well computationally.

While the sequence memoizer and related models are useful for predicting the continuation of sequences, prediction is not the only application for these models. Automated speech recognition and machine translation require assessing the typicality of sequences of words (i.e. Is this sentence a probable English sentence?). Speaker or writer identification tasks require being able to distinguish typicality of phrases under word sequence models of different writers' styles. Classifying a sequence of machine instructions as malicious or not requires establishing the typicality of the sequence under each class. Models of sequences can be used for predicting the continuation of sequences, clustering or classifying sequences, detecting change points in sequences, filling in gaps, compressing data, and more.

In this article we describe the sequence memoizer in terms of general sequences over a discrete alphabet of symbols, though often we will refer to sequences of words when giving intuitive explanations.

2. PREDICTING SEQUENCES

To start, let Σ be the set of symbols that can occur in some sequence. This set can consist of dictionary entries, ASCII values, or $\{A, C, G, T\}$ in case of DNA sequences. Suppose that we are given a sequence[1] $\mathbf{x} = x_1, x_2, \ldots, x_T$ of symbols from Σ and would like to estimate the probability that the next symbol takes on a particular value.

One way to estimate the probability that the next symbol takes some value $s \in \Sigma$ is to use the relative frequency of its occurrence in $\mathbf{x}$, i.e. if s occurs frequently in $\mathbf{x}$ we expect its probability of appearing next to be high as well. Assuming that $\mathbf{x}$ is long enough, doing this will be better than giving equal probability to all symbols in Σ. Let us denote by $N(s)$ the number of occurrences of s in $\mathbf{x}$. Our estimate of the probability of s being the next symbol is then $G(s) = N(s)/T = N(s)/\sum_{s' \in \Sigma} N(s')$. The function G is a *discrete distribution* over the elements of Σ: it assigns a non-negative number $G(s)$ to each symbol s signifying the probability of observing s, with the numbers summing to one over Σ.

Of course, this approach is only reasonable if the process generating $\mathbf{x}$ has no history dependence (e.g. if $\mathbf{x}$ is produced by a sequence of tosses of a biased coin). It is highly unsatisfying if there are contextual dependencies which we can exploit. If we start accounting for context, we can quickly improve the quality of the predictions we make. For instance why not take into account the preceding symbol? Let $\mathbf{u}$ be another symbol. If the last symbol in $\mathbf{x}$ is $\mathbf{u}$, then we can estimate the probability of the next symbol being s by counting the number of times s occurs after $\mathbf{u}$ in $\mathbf{x}$. As before, we can be more precise and define

$$G_{\mathbf{u}}(s) = \frac{N(\mathbf{u}s)}{\sum_{s' \in \Sigma} N(\mathbf{u}s')} \tag{1}$$

to be the estimated probability of s occurring after $\mathbf{u}$, where $N(\mathbf{u}s)$ is the number of occurrences of the subsequence $\mathbf{u}s$ in $\mathbf{x}$. The function $G_{\mathbf{u}}$ is again a discrete distribution over the symbols in Σ, but it is now a *conditional distribution* as the probability assigned to each symbol s depends on the context $\mathbf{u}$.

In the hope of improving our predictions it is natural to extend this counting procedure to contexts of length greater than one. The extension of this procedure to longer contexts is notationally straightforward, requiring us only to re-interpret $\mathbf{u}$ as a sequence of length $n \geq 1$ (in fact, for the remainder of this article boldface type variables will indicate sequences, and we will use Σ^* to denote the set of all finite sequences). Unfortunately, using this exact procedure for estimation with long contexts leads to difficulties which we will consider next.

3. MAXIMUM LIKELIHOOD

Some readers may realize that the counting procedure described above corresponds to an ubiquitous statistical estimation technique called *maximum likelihood* (ML) estimation. The general ML estimation set-up is as follows: we observe some data $\mathbf{x}$ which is assumed to have been generated by some underlying stochastic process and wish to estimate parameters Θ for a probabilistic model of this process. A probabilistic model defines a distribution $P(\mathbf{x}|\Theta)$ over $\mathbf{x}$ parameterized by Θ, and the maximum likelihood estimator is the value of Θ maximizing $P(\mathbf{x}|\Theta)$. In our case the data consists of the observed sequence, and the parameters are the conditional distributions $G_{\mathbf{u}}$ for some set of $\mathbf{u}$'s.

In a sense maximum likelihood is an *optimistic* procedure, in that it assumes that $\mathbf{x}$ is an accurate reflection of the true underlying process that generated it, so that the maximum likelihood parameters will be an accurate estimate of the true parameters. It is this very optimism that is its Achilles heel, since it becomes overly confident about its estimates. This situation is often referred to as *overfitting*. To elaborate on this point, consider the situation in which we have long contexts. The denominator of (1) counts the number of times that the context $\mathbf{u}$ occurs in $\mathbf{x}$. Since $\mathbf{x}$ is of finite length, when $\mathbf{u}$ is reasonably long the chance that $\mathbf{u}$ never occurs at all in $\mathbf{x}$ can be quite high, so (1) becomes undefined with a zero denominator. More pernicious still is if we are "lucky" and $\mathbf{u}$ did occur once or a few times in $\mathbf{x}$. In this case (1) will assign high probability to the few symbols that just by chance did follow $\mathbf{u}$ in $\mathbf{x}$, and zero probability to other symbols. Does it mean that these are the only symbols we expect to see in the future following $\mathbf{u}$, or does it mean that the amount of data we have in $\mathbf{x}$ is insufficient to characterize the conditional distribution $G_{\mathbf{u}}$? Given a complex process with many parameters the latter is often the case, leading to ML estimates that sharpen far too much around the exact observations and don't reflect our true uncertainty.

Obviously, if one uses models that consider only short context lengths, this problem can largely be avoided if one has enough data to estimate some (relatively) smaller number of conditional distributions. This is precisely what is typically done: one makes a *fixed-order Markov assumption* and restricts oneself to estimating collections of distributions conditioned on short contexts (for instance an n^{th}-order Markov model, or an m-gram language model). The consequence of doing this is that maximum likelihood estimation becomes feasible, but longer-range dependencies are discarded. By assumption and design they cannot be accounted for by such

[1]It is straightforward to consider multiple sequences in our setting, we consider being given only one sequence in this paper for simplicity.

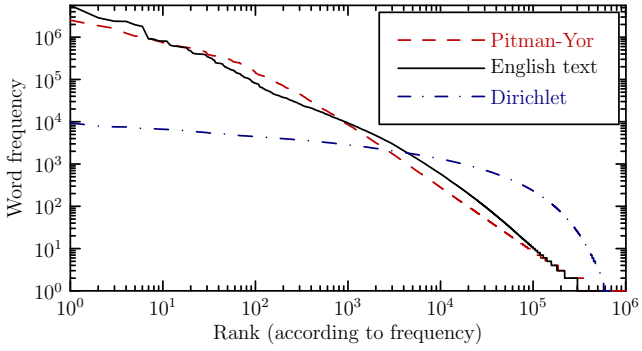

Figure 1: Illustration of the power-law scaling of word frequencies in English text. The relative word frequency (estimated from a large corpus of written text) is plotted against each word's rank when ordered according to frequency. One can see that there are a few very common words and a large number of relatively rare words; in fact, the 200 most common words account for over 50% of the observed text. The rank/frequency relationship is very close to a pure power law relationship which would be a perfectly straight line on this log-log plot. Also plotted are samples drawn from a Pitman-Yor process (in blue) and a Dirichlet distribution (in red) fitted to the data. The Pitman-Yor captures the power-law statistics of the English text much better than the Dirichlet.

restrictive models.

Even having imposed such a restriction, overfitting often remains an issue. This has led to the development of creative approaches to its avoidance. The language modeling and text compression communities have generally called these *smoothing* or *back-off* methodologies (see [3] and references therein). In the following we will propose a *Bayesian* approach that retains uncertainty in parameter estimation and thus avoids over-confident estimates.

4. BAYESIAN MODELING

As opposed to maximum likelihood, the Bayesian approach is inherently *conservative*. Rather than trusting the data fully, Bayesian parameter estimation incorporates both evidence from the data as well as from prior knowledge of the underlying process. Furthermore, uncertainty in estimation is taken into account by treating the parameters Θ as random, endowed with a *prior distribution* $P(\Theta)$ reflecting the prior knowledge we have about the true data generating process. The prior distribution is then combined with the likelihood $P(\mathbf{x}|\Theta)$ to yield, via *Bayes' Theorem* (the namesake of the approach), the *posterior distribution* $P(\Theta|\mathbf{x}) = P(\Theta)P(\mathbf{x}|\Theta)/P(\mathbf{x})$ which specifies the belief about the parameter Θ after combining both sources of information. Computations such as prediction are then done taking into account the *a posteriori* uncertainty about the underlying parameters.

What kinds of prior knowledge about natural sequence data might we wish to employ? We make use of two: that natural sequence data often exhibits power-law properties, and that conditional distributions of similar contexts tend to be similar themselves, particularly in the sense that recency matters. We will consider each of these in turn in the rest of this section.

4.1 Power-Law Scaling

As with many other natural phenomena like social networks and earthquakes, occurrences of words in a language follow a *power-law scaling* [23]. This means that there are a small number of words that occur disproportionately frequently (e.g. the, to, of), and a very large number of rare words that, although each occurs rarely, when taken together make up a large proportion of the language. The power-law scaling in written English is illustrated in Figure 1. In this subsection we will describe how to incorporate prior knowledge about power-law scaling in the true generative process into our Bayesian approach. To keep the exposition simple, we will start by ignoring contextual dependencies and instead focus only on one way of estimating probability distributions that exhibit power-law scaling.

To see why it is important to incorporate knowledge about power-law scaling, consider again the maximum likelihood estimate given by the relative frequency of symbol occurrences $G(s) = N(s)/\sum_{s' \in \Sigma} N(s')$. For the frequently occurring symbols, their corresponding probabilities will be well estimated since they are based on many observations of the symbols. On the other hand, our estimates of the rare symbol probabilities will not be good at all. In particular, if a rare symbol did not occur in our sequence (which is likely), our estimate of its probability will be zero, while the probability of a rare symbol that did occur just by chance in our sequence will be overestimated. Since most symbols in Σ will occur quite rarely under a power-law, our estimates of $G(s)$ will often be inaccurate.

To encode our prior knowledge about power-law scaling, we use a prior distribution called the Pitman-Yor process (PYP) [15], which is a distribution over the discrete probability distribution $G = \{G(s)\}_{s \in \Sigma}$. It has three parameters: a *base distribution* $G_0 = \{G_0(s)\}_{s \in \Sigma}$ which is the mean of the PYP and reflects our prior belief about the frequencies of each symbol, a *discount* parameter α between 0 and 1 which governs the exponent of the power-law, and a *concentration* parameter c which governs the variability around the mean G_0. When $\alpha = 0$ the Pitman-Yor process loses its power-law properties and reduces to the more well-known Dirichlet process. In this paper we assume $c = 0$ instead for simplicity; see [6] for the more general case when c is allowed to be positive. When we write $G \sim \mathcal{PY}(\alpha, G_0)$ it means that G has a prior given by a Pitman-Yor process with the given parameters. Figure 1 illustrates the power-law scaling produced by Pitman-Yor processes.

To convey more intuition about the Pitman-Yor process we can consider how using it affects our estimate of symbol frequencies. Note that in our Bayesian framework G is random, and one of the standard steps in a procedure called *inference* is to estimate a posterior distribution $P(G|\mathbf{x})$ from data. The probability that symbol $s \in \Sigma$ occurs next is then:

$$P(x_{T+1} = s|\mathbf{x}) = \int P(x_{T+1} = s|G)P(G|\mathbf{x})dG = \mathbb{E}[G(s)], \tag{2}$$

where $\mathbb{E}$ in this case stands for expectation with respect to the posterior distribution $P(G|\mathbf{x})$. This integral is a standard Bayesian computation that sometimes has an analytic

solution but often does not. When it does not, like in this situation, it is often necessary to turn to numerical integration approaches, including sampling and Monte Carlo integration [16].

In the case of the Pitman-Yor process, $\mathbb{E}[G(s)]$ can be computed as described at a high level in the following way. In addition to the counts $\{N(s')\}_{s' \in \Sigma}$, assume there is another set of random "counts" $\{M(s')\}_{s' \in \Sigma}$ satisfying $1 \leq M(s') \leq N(s')$ if $N(s') > 0$ and $M(s') = 0$ otherwise. The probability of symbol $s \in \Sigma$ occurring next is then given by:

$$\mathbb{E}[G(s)] = \mathbb{E}\left[\frac{N(s) - \alpha M(s) + \sum_{s' \in \Sigma} \alpha M(s') G_0(s)}{\sum_{s' \in \Sigma} N(s')}\right]. \quad (3)$$

Given this, it is natural to ask what purpose these $M(s)$'s serve? By studying (3) it can be seen that each $M(s)$ reduces the count $N(s)$ by $\alpha M(s)$ and that the total amount subtracted is then redistributed across all symbols in Σ proportionally according to the symbols' probability under the base distribution G_0. Thus non-zero counts are usually reduced, with larger counts typically reduced by a larger amount. Doing this mitigates the overestimation of probabilities of rare symbols that happen to appear by chance. On the other hand, for symbols that did not appear at all, the estimates of their probabilities are pulled upwards from zero, mitigating underestimation of their probability. We describe this effect as "stealing from the rich and giving to the poor." This is precisely how the Pitman-Yor process manifests a power-law characteristic. If one thinks of the $M(s)$'s and α as parameters then one could imagine ways to set them to best describe the data. Intuitively this is not at all far from what is done, except that the $M(s)$'s and α are themselves treated in a Bayesian way, i.e. we average over them under the posterior distribution in (3).

4.2 Context Trees

We now return to making use of the contextual dependencies in $\mathbf{x}$ and to estimating all of the conditional distributions $G_{\mathbf{u}}$ relevant to predicting symbols following a general context $\mathbf{u}$. The assumption we make is that if two contexts are similar, then the corresponding conditional distributions over the symbols that follow those contexts will tend to be similar as well. A simple and natural way of defining similarity between contexts is that of overlapping contextual suffixes. This is easy to see in a concrete example from language modeling. Consider the distribution over words that would follow $\mathbf{u} =$ in the United States of. The assumption we make is that this distribution will be similar to the distribution following the shorter context, the United States of, which we in turn expect to be similar to the distribution following United States of. These contexts all share the same length three suffix.

In this section and the following one, we will discuss how this assumption can be codified using a *hierarchical Bayesian model* [11, 8]. To start we will only consider fixed, finite length contexts. When we do this we say that we are making an n^{th} order Markov assumption. This means that each symbol only depends on the last n observed symbols. Note that this assumption dictates that distributions are not only similar but equal among contexts whose suffixes overlap in their last n symbols. This equality constraint is a strong assumption that we will relax in Section 5.

We can visualize the similarity assumption we make by constructing a *context tree*: Arrange the contexts $\mathbf{u}$ (and the associated distributions $G_{\mathbf{u}}$) in a tree where the parent of a node $\mathbf{u}$, denoted $\sigma(\mathbf{u})$, is given by its *longest proper suffix* (i.e. $\mathbf{u}$ with its first symbol from the left removed). Figure 2 gives an example of a context tree with $n = 3$ and $\Sigma = \{0, 1\}$. Since for now we are making an n^{th} order Markov assumption, it is sufficient to consider only the contexts $\mathbf{u} \in \Sigma_n^* = \{\mathbf{u}' \in \Sigma^* : |\mathbf{u}'| \leq n\}$ of length at most n. The resulting context tree has height $n+1$ and the total number of nodes in the tree grows exponentially in n. The memory complexity of models built on such context trees usually grows too large too quickly for reasonable values of n and $|\Sigma|$. This makes it nearly impossible to estimate *all* of the distributions $G_{\mathbf{u}}$ in the naïve way described in Section 2. This estimation problem led us to hierarchical Bayesian modeling using Pitman-Yor processes.

4.3 Hierarchical Pitman-Yor Processes

Having defined a context tree and shown that the Pitman-Yor prior over distributions exhibits power-law characteristics, it remains to integrate the two.

Recall that $G \sim \mathcal{PY}(\alpha, G_0)$ means that G is a random distribution with a Pitman-Yor process prior parameterized by a discount parameter α and a base distribution G_0. The expected value of G under repeated draws from the Pitman-Yor process is the base distribution G_0. Because of this fact we can use this process to encode any assumption that states that on average G should be similar to G_0. To be clear, this is just a prior assumption. As always, observing data may lead to a change in our belief. We can use this mechanism to formalize the context tree notion of similarity. In particular, to encode the belief that $G_{\mathbf{u}}$ should be similar to $G_{\sigma(\mathbf{u})}$, we can use a Pitman-Yor process prior for $G_{\mathbf{u}}$ with base distribution $G_{\sigma(\mathbf{u})}$. We can apply the same mechanism at each node of the context tree, leading to the following model specification:

$$\begin{aligned} G_\varepsilon &\sim \mathcal{PY}(\alpha_0, G_0) && (4)\\ G_{\mathbf{u}} | G_{\sigma(\mathbf{u})} &\sim \mathcal{PY}(\alpha_{|\mathbf{u}|}, G_{\sigma(\mathbf{u})}) && \text{for all } \mathbf{u} \in \Sigma_n^* \backslash \varepsilon \\ x_i | \mathbf{x}_{i-n:i-1} = \mathbf{u}, G_{\mathbf{u}} &\sim G_{\mathbf{u}} && \text{for } i = 1, \ldots, T \end{aligned}$$

The second line says that a priori the conditional distribution $G_{\mathbf{u}}$ should be similar to $G_{\sigma(\mathbf{u})}$, its parent in the context tree. The variation of $G_{\mathbf{u}}$ around its mean $G_{\sigma(\mathbf{u})}$ is described by a Pitman-Yor process with a context length-dependent discount parameter $\alpha_{|\mathbf{u}|}$. At the top of the tree the distribution G_ε for the empty context ε is similar to an overall base distribution G_0, which specifies our prior belief that each symbol s will appear with probability $G_0(s)$. The third line describes the n^{th} order Markov model for $\mathbf{x}$: It says that the distribution over each symbol x_i in $\mathbf{x}$, given that its context consisting of the previous n symbols $x_{i-n:i-1}$ is $\mathbf{u}$, is simply $G_{\mathbf{u}}$.

The hierarchical Bayesian model in (4) is called the *hierarchical Pitman-Yor process* [18]. It formally encodes our context tree similarity assumption about the conditional distributions using dependence among them induced by the hierarchy, with more similar distributions being more dependent. It is this dependence which allows the model to share information across the different contexts, and subsequently improve the estimation of all conditional distributions. It is worth noting that there is a well known connection between the hierarchical PYP and a type of smoothing for m-gram language models called interpolated Kneser-Ney [10, 18].

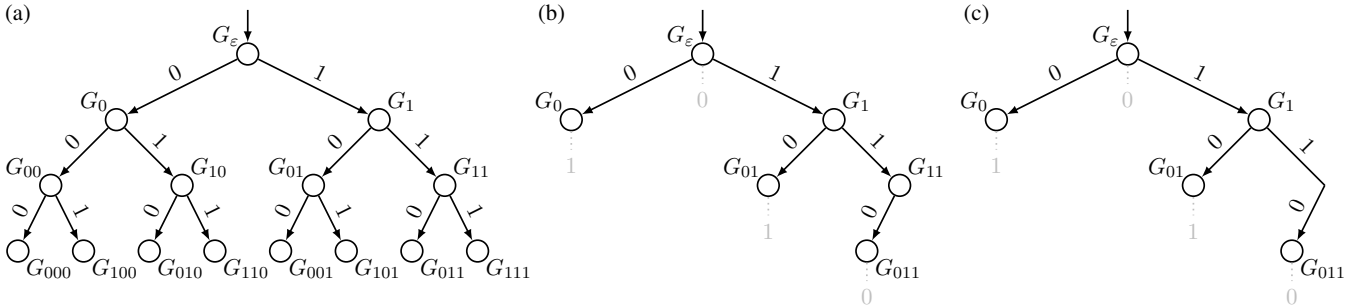

Figure 2: (a) Full context tree containing all contexts up to length 3 over symbol set $\Sigma = \{0, 1\}$. (b) Context tree actually needed for the string `0110`. Observations in the context in which they were observed are denoted in gray below the corresponding context. (c) Compact context tree for the same string, with non-branching chains marginalized out.

5. SEQUENCE MEMOIZER

The name *sequence memoizer* (SM) refers to both an extension of the hierarchical PYP model presented in the previous section, as well as to the set of techniques required to make practical use of the extended model. We first describe how the SM model extends the hierarchical PYP model and then discuss how to reduce the complexity of the model to make it computationally tractable. Finally, we sketch how inference is done in the SM.

5.1 The Model

The SM model is a notationally subtle but important extension to the hierarchical PYP model (4) described in the previous section. Instead of limiting the context lengths to n, the model is extended to include the set of distributions in all contexts of any (finite) length. This means that the distribution over each symbol is now conditioned on all previous symbols, not just the previous n.

Formally, the sequence memoizer model is defined exactly as the hierarchical PYP model in equation (4), but with two differences. First, the contexts range over all finite non-empty strings, $\mathbf{u} \in \Sigma^* \backslash \varepsilon$. Second, in the third line of (4), instead of conditioning only on the previous n symbols, we condition on all previous symbols, so that $x_i | \mathbf{x}_{1:i-1} = \mathbf{u}, G_{\mathbf{u}} \sim G_{\mathbf{u}}$. The assumptions embodied in the resulting model remain the same as that for the hierarchical PYP model: power-law scaling and similarity between related contexts.

The sequence memoizer model can be interpreted as the limit of a hierarchical PYP model as the Markov order n tends to infinity. One's first impression of such a model might be that it would be impossible to handle, both statistically because of overfitting and other problems, and computationally because the model as described so far cannot even be represented in a computer with finite memory! Fortunately the Bayesian approach, where we compute the posterior distribution and marginalize over the parameters as in (2) to obtain estimators of interest, prevents overfitting. Additionally, the techniques we develop in the next subsection make computation in this model practical.

5.2 Compacting the Context Tree

While lifting the finite restriction on context lengths seems very desirable from a modeling perspective, the resulting SM model is a prior over an infinite number of parameters (conditional distributions) $\{G_{\mathbf{u}}\}_{\mathbf{u} \in \Sigma^*}$. In order to compute in this model, the number of conditional distributions that are accessed must be reduced to a finite number. The key to realizing that this is possible is that given a *finite length* sequence of symbols $\mathbf{x}$ we only need access to a finite number of conditional distributions. In particular, we only need $G_{\mathbf{x}_{1:i}}$ where $i = 0 \ldots T$ and all the ancestors of each $G_{\mathbf{x}_{1:i}}$ in the context tree. The ancestors are needed because each $G_{\mathbf{u}}$ has a prior that depends on its parent $G_{\sigma(\mathbf{u})}$. The resulting set of conditional distributions that the sequence $\mathbf{x}$ actually depends on consists of $G_{\mathbf{u}}$ where $\mathbf{u}$ ranges over all contiguous substrings of $\mathbf{x}$, a finite set of $\mathcal{O}(T^2)$ contexts. All other contexts in the tree can effectively be ignored. We denote this subtree of the context tree that $\mathbf{x}$ actually depends on by $\mathcal{T}(\mathbf{x})$; Figure 2b shows an example with $\mathbf{x} = \texttt{0110}$.

Computation with such a quadratically sized context tree is possible but inefficient, especially if the sequence length is large. A second step reduces the tree down to a linear number of nodes. The key observation underlying this reduction is that many of the contexts that appear in $\mathcal{T}(\mathbf{x})$ only appear in non-branching chains, i.e. each node on the chain only has one child in $\mathcal{T}(\mathbf{x})$. For example, in Figure 2b, the context `11` only occurs as a suffix of the longer context `011`, and is part of the non-branching chain $G_1 \xrightarrow{1} G_{11} \xrightarrow{0} G_{011}$. In such a situation, G_{11} serves no purpose except to relate G_{011} with G_1. If we can directly express the prior of G_{011} in terms of G_1, then we can effectively ignore G_{11} and *marginalize* it out from the model.

Fortunately, a remarkable property related to an operation on Pitman-Yor processes called *coagulation* allows us to perform this marginalization exactly [14]. Specifically in the case of $G_{11}|G_1 \sim \mathcal{PY}(\alpha_2, G_1)$ and $G_{011}|G_{11} \sim \mathcal{PY}(\alpha_3, G_{11})$, the property states simply that $G_{011}|G_1 \sim \mathcal{PY}(\alpha_2\alpha_3, G_1)$ where G_{11} has been marginalized out. In other words, the prior for G_{011} is another Pitman-Yor process whose discount parameter is simply the product of the discount parameters along the chain leading into it on the tree $\mathcal{T}(\mathbf{x})$, while the base distribution is simply the head of the chain G_1.

In general, applying the same marginalization procedure to all the non-branching chains of $\mathcal{T}(\mathbf{x})$, we obtain a *compact context tree* $\hat{\mathcal{T}}(\mathbf{x})$ where all internal nodes have at least two children (all others have been integrated out). This is illustrated in Figure 2c, where each chain is replaced by an edge labeled by the sequence of symbols on the original edges of the chain (in the example only $G_1 \xrightarrow{1} G_{11} \xrightarrow{0} G_{011}$ is replaced

by $G_1 \overset{01}{\rightarrow} G_{011}$). One can easily show that the number nodes in the compact context tree $\hat{\mathcal{T}}(\mathbf{x})$ is at most twice the length of the sequence $\mathbf{x}$ (independent of $|\Sigma|$).

At this point some readers may notice that the compact context tree has a structure reminiscent of a data structure for efficient string operations called a suffix tree [9]. In fact the structure of the compact context tree is given by the suffix tree for the *reverse* sequence $x_T, x_{T-1}, \ldots, x_1$. Similar extensions from fixed-length to unbounded-length contexts, followed by reductions in the context trees have also been developed in the compression literature [4, 19].

5.3 Inference and Prediction

As a consequence of the two marginalization steps described in the previous subsection, inference in the full sequence memoizer model with an infinite number of parameters is equivalent to inference in the compact context tree $\hat{\mathcal{T}}(\mathbf{x})$ with a linear number of parameters. Further, the prior over the conditional distributions on $\hat{\mathcal{T}}(\mathbf{x})$ still retains the form of a hierarchical Pitman-Yor process: each node still has a Pitman-Yor process prior with its parent as the base distribution. This means that inference algorithms developed for the finite-order hierarchical PYP model can be easily adapted to the sequence memoizer. We will briefly describe the inference algorithms we employ.

In the SM model we are mainly interested in the predictive distribution of the next symbol being some $s \in \Sigma$ given some context $\mathbf{u}$, conditioned on an observed sequence $\mathbf{x}$. As in (2), this predictive distribution is expressed as an expectation $\mathbb{E}[G_{\mathbf{u}}(s)]$ over the posterior distribution of $\{G_{\mathbf{u}'}\}_{\mathbf{u}' \in \hat{\mathcal{T}}(\mathbf{x})}$. Just as in (3) as well, it is possible to express $\mathbb{E}[G_{\mathbf{u}}(s)]$ as an expectation over a set of random counts $\{N(\mathbf{u}'s'), M(\mathbf{u}'s')\}_{\mathbf{u}' \in \hat{\mathcal{T}}(\mathbf{x}), s' \in \Sigma}$:

$$\mathbb{E}[G_{\mathbf{u}}(s)] \tag{5}$$
$$= \mathbb{E}\left[\frac{N(\mathbf{u}s) - \alpha_{\mathbf{u}} M(\mathbf{u}s) + \sum_{s' \in \Sigma} \alpha_{\mathbf{u}} M(\mathbf{u}s') G_{\sigma(\mathbf{u})}(s)}{\sum_{s' \in \Sigma} N(\mathbf{u}s')}\right]$$

Again, the first term in the numerator can be interpreted as a count of the number of times s occurs in the context $\mathbf{u}$, the second term is the reduction applied to the count, while the third term spreads the total reduction across Σ according to the base distribution $G_{\sigma(\mathbf{u})}(s)$. Each context $\mathbf{u}$ now has its own discount parameter $\alpha_{\mathbf{u}}$ which is the product of discounts on the non-branching chain leading to $\mathbf{u}$ on $\mathcal{T}(\mathbf{x})$, while the parent $\sigma(\mathbf{u})$ is the head of the chain. Notice that (5) is defined recursively, with the predictive distribution $G_{\mathbf{u}}$ in context $\mathbf{u}$ being a function of the same in the parent $\sigma(\mathbf{u})$ and so on up the tree.

The astute reader might notice that the above does not quite work if the context $\mathbf{u}$ does not occur in the compact context tree $\hat{\mathcal{T}}(\mathbf{x})$. Fortunately the properties of the hierarchical PYP work out in our favor, and the predictive distribution is simply the one given by the longest suffix of $\mathbf{u}$ that is in $\mathcal{T}(\mathbf{x})$. If this is still not in $\hat{\mathcal{T}}(\mathbf{x})$, then a converse of the coagulation property (called *fragmentation*) allows us to re-introduce the node back into the tree.

To evaluate the expectation (5), we use stochastic (Monte Carlo) approximations where the expectation is approximated using *samples* from the posterior distribution. The samples are obtained using Gibbs sampling [16] as in [18, 21], which repeatedly makes local changes to the counts, and using sequential Monte Carlo [5] as in [7], which iterates through the sequence $x_1, x_2, \ldots, x_T$, keeping track of a set of samples at each step, and updating the samples as each symbol x_i is incorporated into the model.

Source	Perplexity
Bengio et al. [2]	109.0
Mnih et al. [13]	83.9
4-gram Interpolated Kneser-Ney [3, 18]	106.1
4-gram Modified Kneser-Ney [3, 18]	102.4
4-gram Hierarchical PYP [18]	101.9
Sequence Memoizer [21]	96.9

Table 1: Language modeling performance for a number of models on an Associated Press news corpus (lower perplexity is better). Interpolated and modified Kneser-Ney are state-of-the-art language models. Along with hierarchical PYP and the sequence memoizer, these models do not model relationships among words in the vocabulary. Provided for comparison are the results for the models of Bengio et al. and Mnih et al. which belong to a different class of models that learn word representations from data.

6. DEMONSTRATION

We now consider two target applications: language modelling and data compression. It is demonstrated that the SM model is able achieve better performance than most state-of-the-art techniques by capturing long-range dependencies.

6.1 Language Modeling

Language modeling is the task of fitting probabilistic models to sentences (sequences of words), which can then be used to judge the plausibility of new sequences being sentences in the language. For instance, "God save the Queen" should be given a higher score than "Queen the God save" and certainly more than "glad slave the spleen" under any model of English. Language models are mainly used as building blocks in natural language processing applications such as statistical machine translation and automatic speech recognition. In the former, for example, a translation algorithm might propose multiple sequences of English words, at least one of which is hoped to correspond to a good translation of a foreign language source. Usually only one or a few of these suggested sequences are plausible English language constructions. The role of the language model is to judge which English construction is best. Better language models generally lead to better translators.

Language model performances are reported in terms of a standard measure called *perplexity*. This is defined as $2^{\ell(\mathbf{x})}$ where $\ell(\mathbf{x}) = -\frac{1}{|\mathbf{x}|}\sum_{i=1}^{|\mathbf{x}|} \log_2 P(x_i|\mathbf{x}_{1:i-1})$ is the average *log-loss* on a sequence $\mathbf{x}$ and the average number of bits per word required to encode the sequence using an optimal code. Another interpretation of perplexity is that it is the average number of guesses the model would have to make before it guessed each word correctly (if it makes these guesses by drawing samples from its estimate of the conditional distribution). For the SM model $P(x_i|\mathbf{x}_{1:i-1})$ is computed as in (5). Both lower log-loss and lower perplexity are better.

Figure 3 compares the SM model against n^{th} order Markov models with hierarchical PYP priors, for various values of

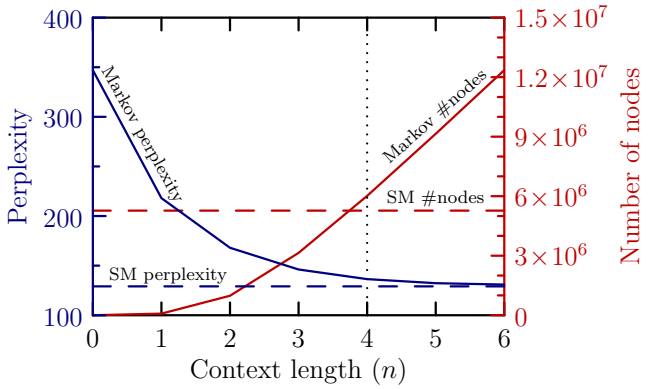

Figure 3: In blue is the performance of the SM model (dashed line) versus n^{th} order Markov models with hierarchical PYP priors (solid line) as n varies (test data perplexity, lower is better). In red is the computational complexity of the SM model (dashed line) versus the Markov models (solid line) in terms of the number of nodes in the context tree/trie. For this four million word New York Times corpus, as n passes 4, the memory complexity of the Markov models grows larger than that of the sequence memoizer, yet, the SM model yields modeling performance that is better than all Markov models regardless of their order. This suggests that for $n \geq 4$ the SM model is to be preferred: it requires less space to store yet results in a comparable if not better model.

n, on a four million word New York Times corpus[3]. Table 1 compares the hierarchical PYP Markov model and the SM model against other state-of-the-art models, on a fourteen million word Associated Press news article corpus. The AP corpus is a benchmark corpus for which the performance of many models is available. It is important to note that it was processed by [2] to remove low frequency words. Note that this is to the detriment of the sequence memoizer, which is explicitly designed to improve modeling of low word frequencies due to power-law scaling. It is also a relatively small corpus, limiting the benefits of the SM model at capturing longer range dependencies.

Our results show that the SM model is a competitive language model. However, perplexity results alone do not tell the whole story. As more data is used to estimate a language model, typically its performance improves. This means that computational considerations such as memory and runtime must enter into the discussion about what constitutes a good language model. In many applications fast prediction is imperative, in others, particularly in online settings, incorporation of new data into the model must be fast. In comparison to more complex language models, prediction in the sequence memoizer has real-world time complexity that is essentially the same as that of a smoothed finite-order Markov model, while its memory complexity is linear in the amount of data. The computational complexity of Markov models theoretically does not depend on the amount of data but is exponential in the Markov order, rendering straightforward

[3]Note that an n^{th} order Markov model is an m-gram model where $m = n + 1$.

Model	SM	PPM	CTW	bzip2	gzip
Average bits / byte	**1.89**	1.93	1.99	2.11	2.61

Table 2: Compression performance in terms of weighted average log-loss (average bits per byte under optimal entropy encoding, lower is better) for the Calgary corpus, a standard benchmark collection of diverse filetypes. The results for unbounded-length context PPM is from [4]. The results for CTW is from [20]. The bzip2 and gzip results come from running the corresponding standard unix command line tools with no extra arguments.

extensions to higher orders impractical. The SM model directly fixes this problem while remaining computationally tractable. Constant space, constant time extensions to the SM model [1] have been developed which show great promise for language modeling and other applications.

6.2 Compression

Shannon's celebrated results in information theory [17] has led to lossless compression technology that, given a coding distribution, nearly optimally achieves the theoretical lower limit (given by the log-loss) on the number of bits needed to encode a sequence. Lossless compression is closely related to sequence modeling: an incrementally constructed probabilistic sequence model such as the sequence memoizer can be used to adaptively construct coding distributions which can then be directly used for compression based on entropy coding.

We demonstrate the theoretical performance of a lossless compressor based on the SM model on a number of standard compression corpora. Table 2 summarizes a comparison of our lossless compressor against other state-of-the-art compressors on the Calgary corpus, a well-known compression benchmark consisting of fourteen files of different types and varying lengths.

In addition to the experiments on the Calgary corpus, SM compression performance was also evaluated on a 100 MB excerpt of the English version of Wikipedia (XML text dump) [12]. On this excerpt, the SM model achieved a log-loss of 1.66 bits/symbol amounting to a compressed file size of 20.80 MB. While this is worse than 16.23 MB achieved by the best demonstrated Wikipedia compressor, it demonstrates that the SM model can scale to sequences of this length. We have also explored the performance of the SM model when using a larger symbol set (Σ). In particular, we used the SM model to compress UTF-16 encoded Chinese text using a 16-bit alphabet. On a representative text file, the Chinese Union version of the bible, we achieved a log-loss of 4.91 bits per Chinese character, which is significantly better than the best results in the literature (5.44 bits) [22].

7. CONCLUSIONS

The sequence memoizer achieves improved compression and language modeling performance. These application specific performance improvements are arguably by themselves worthwhile scientific achievements. Both have the potential to be tremendously useful, and may yield practical consequences of societal and commercial value.

We encourage the reader, however, not to mentally cate-

gorize the sequence memoizer as a compressor or language model. Nature is replete with discrete sequence data that exhibit long-range dependencies and power-law characteristics. The need to model the processes that generate such data is likely to grow in prevalence. The sequence memoizer is a general purpose model for discrete sequence data that remains computationally tractable despite its power and despite the fact that it makes only very general assumptions about the data generating process.

Our aim in communicating the sequence memoizer is also to encourage readers to explore the fields of probabilistic and Bayesian modeling in greater detail. Expanding computational capacity along with significant increases in the amount and variety of data to be analyzed across many scientific and engineering disciplines is rapidly enlarging the class of probabilistic models that one can imagine and employ. Over the coming decades hierarchical Bayesian models are likely to become increasingly prevalent in data analysis oriented fields like applied statistics, machine learning, and computer science. It is our belief that the sequence memoizer and its ilk will come to be seen as relatively simple building blocks for the enormous and powerful hierarchical models of tomorrow.

Source code and example usages of the sequence memoizer are available at `http://www.sequencememoizer.com/`. A lossless compressor built using the sequence memoizer can be explored at `http://www.deplump.com/`.

8. ACKNOWLEDGMENTS

We wish to thank the Gatsby Charitable Foundation and Columbia University for funding.

9. REFERENCES

[1] N. Bartlett, D. Pfau, and F. Wood. Forgetting counts: Constant memory inference for a dependent hierarchical Pitman-Yor process. In *27th International Conference on Machine Learning*, to appear, 2010.

[2] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155, 2003.

[3] S. F. Chen and J. Goodman. An empirical study of smoothing techniques for language modeling. *Computer Speech and Language*, 13(4):359–394, 1999.

[4] J. G. Cleary and W. J. Teahan. Unbounded length contexts for PPM. *The Computer Journal*, 40:67–75, 1997.

[5] A. Doucet, N. de Freitas, and N. J. Gordon. *Sequential Monte Carlo Methods in Practice*. Statistics for Engineering and Information Science. New York: Springer-Verlag, May 2001.

[6] J. Gasthaus and Y. W. Teh. Improvements to the sequence memoizer. In *Advances in Neural Information Processing Systems 23*, to appear, 2010.

[7] J. Gasthaus, F. Wood, and Y. W. Teh. Lossless compression based on the Sequence Memoizer. In J. A. Storer and M. W. Marcellin, editors, *Data Compression Conference 2010*, pages 337–345, Los Alamitos, CA, USA, 2010. IEEE Computer Society.

[8] A. Gelman, J. B. Carlin, H. S. Stern, and D. B. Rubin. *Bayesian data analysis*. Chapman & Hall/CRC, 2nd edition, 2004.

[9] R. Giegerich and S. Kurtz. From Ukkonen to McCreight and Weiner: A unifying view of linear-time suffix tree construction. *Algorithmica*, 19(3):331–353, 1997.

[10] S. Goldwater, T. L. Griffiths, and M. Johnson. Interpolating between types and tokens by estimating power law generators. In *Advances in Neural Information Processing Systems 18*, pages 459–466. MIT Press, 2006.

[11] D. J. C. MacKay and L. B. Peto. A hierarchical Dirichlet language model. *Natural language engineering*, 1(2):289–307, 1995.

[12] M. Mahoney. Large text compression benchmark. URL: `http://www.mattmahoney.net/text/text.html`, 2009.

[13] A. Mnih, Z. Yuecheng, and G. Hinton. Improving a statistical language model through non-linear prediction. *Neurocomputing*, 72(7-9):1414 – 1418, 2009.

[14] J. Pitman. Coalescents with multiple collisions. *Annals of Probability*, 27:1870–1902, 1999.

[15] J. Pitman and M. Yor. The two-parameter Poisson-Dirichlet distribution derived from a stable subordinator. *Annals of Probability*, 25:855–900, 1997.

[16] C. P. Robert and G. Casella. *Monte Carlo statistical methods*. Springer Verlag, 2004.

[17] C. E. Shannon. A mathematical theory of communication. *Bell System Technical Journal (reprinted in ACM SIGMOBILE Mobile Computing and Communications Review 2001)*, 1948.

[18] Y. W. Teh. A hierarchical Bayesian language model based on Pitman-Yor processes. In *Proceedings of the Association for Computational Linguistics*, pages 985–992, 2006.

[19] F. M. J. Willems. The context-tree weighting method: Extensions. *IEEE Transactions on Information Theory*, 44(2):792–798, 1998.

[20] F. M. J. Willems. CTW website. URL: `http://www.ele.tue.nl/ctw/`, 2009.

[21] F. Wood, C. Archambeau, J. Gasthaus, L. James, and Y. W. Teh. A stochastic memoizer for sequence data. In *26th International Conference on Machine Learning*, pages 1129–1136, 2009.

[22] P. Wu and W. J. Teahan. A new PPM variant for Chinese text compression. *Natural Language Engineering*, 14(3):417–430, 2007.

[23] G. Zipf. *Selective Studies and the Principle of Relative Frequency in Language*. Harvard University Press, Cambridge, MA, 1932.