A Stochastic Memoizer for Sequence Data

Frank Wood[⋆] FWOOD@GATSBY.UCL.AC.UK
Cédric Archambeau[†] C.ARCHAMBEAU@CS.UCL.AC.UK
Jan Gasthaus[⋆] J.GASTHAUS@GATSBY.UCL.AC.UK
Lancelot James[‡] LANCELOT@UST.HK
Yee Whye Teh[⋆] YWTEH@GATSBY.UCL.AC.UK
[⋆]Gatsby Computational Neuroscience Unit
University College London, 17 Queen Square, London, WC1N 3AR, UK
[†]Centre for Computational Statistics and Machine Learning
University College London, Gower Street, London, WC1E 6BT, UK
[‡]Department of Information and Systems Management
Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong

Abstract

We propose an unbounded-depth, hierarchical, Bayesian nonparametric model for discrete sequence data. This model can be estimated from a single training sequence, yet shares statistical strength between subsequent symbol predictive distributions in such a way that predictive performance generalizes well. The model builds on a specific parameterization of an unbounded-depth hierarchical Pitman-Yor process. We introduce analytic marginalization steps (using coagulation operators) to reduce this model to one that can be represented in time and space linear in the length of the training sequence. We show how to perform inference in such a model without truncation approximation and introduce fragmentation operators necessary to do predictive inference. We demonstrate the sequence memoizer by using it as a language model, achieving state-of-the-art results.

1. Introduction

A Markov assumption is often made when modeling sequence data. This assumption stipulates that conditioned on the present value of the sequence, the past and the future are independent. Making this assumption allows one to fully characterize a sequential process in terms of a set of conditional distributions that describe the dependence of future values on a finite history (or context) of values. The length of this context is called the order of the Markov model.

The literature provides ample evidence of the fact that making such an assumption is often reasonable in a practical sense. Even data that is clearly not Markov in nature (for instance natural language) is often well-enough described by Markov models for them to be of significant practical utility. Increasing the order of the Markov model often improves application performance. Unfortunately it is often difficult to increase the order in practice because increasing the order requires either vastly greater amounts of training data or significantly more complicated smoothing procedures.

In this work we propose a non-Markov model for stationary discrete sequence data. The model is non-Markov in the sense that the next value in a sequence is modelled as being conditionally dependent on *all* previous values in the sequence. It is immediately clear that such a model must have a very large number of latent variables. To constrain the learning of these latent variables, we employ a hierarchical Bayesian prior based on Pitman-Yor processes which promotes sharing of statistical strength between subsequent symbol predictive distributions for equivalent contexts of different lengths (Teh, 2006). We find that we can analytically marginalize out most latent variables, leaving a number that is linear in the size of the input sequence. We demonstrate that inference in the resulting collapsed model is tractable and efficient.

Posterior inference in the model can be understood as stochastically "memoizing" (Michie, 1968) con-

Appearing in *Proceedings of the 26th International Conference on Machine Learning*, Montreal, Canada, 2009.

text/observation pairs. While memoization refers to deterministic caching of function outputs given inputs, what we mean by stochastic memoization is exactly that used by (Goodman et al., 2008): calling a function multiple times with the same arguments may return an instance from a set of previous return values, but also may return a new value. We call our contribution a stochastic memoizer for sequence data (sequence memoizer (SM) for short) because given a context (the argument) it will either return a symbol that was already generated in that full context, a symbol that was returned given a context that is shorter by one symbol, or, at the recursion base, potentially something entirely novel. The stochastic memoizer for sequence data consists of a model and efficient algorithms for model construction and inference.

In the next section we formalize what we mean by *non-Markov* model and define the prior we use in the sequence memoizer. In Section 3 we explain how a posterior sampler for the sequence memoizer given a finite sequence of observations can be constructed and represented in linear space and time. In Section 4 we explain the marginalization operations necessary to achieve such a representation. In Section 5 we discuss sequence memoizer inference, particularly the novel steps necessary to perform predictive inference in contexts that do not occur in the training data. Finally, in Section 6 we use the sequence memoizer for language modelling and demonstrate promising empirical results.

2. Non-Markov Model

Consider a sequence of discrete random variables $x_{1:T} = (x_1 x_2 \cdots x_T)$ of arbitrary length T, each taking values in a symbol set Σ. The joint distribution over the sequence is

$$P(x_{1:T}) = \prod_{i=1}^{T} P(x_i | x_{1:i-1}), \tag{1}$$

where each factor on the right hand side is the predictive probability of x_i given a context consisting of all preceding variables $x_{1:i-1}$. When one makes a n^{th} order Markov approximation to (1) it is assumed that only the values taken by at most the preceding n variables matter for predicting the value of the next variable in the sequence, i.e. $P(x_i|x_{1:i-1}) = P(x_i|x_{i-n:i-1})$ for all i. If the context is not truncated to some fixed context length, we say the model is non-Markovian.

When learning such a model from data, a vector of predictive probabilities for the next symbol given each possible context must be learned. Let $\mathbf{s} \in \Sigma^*$ be a finite sequence of symbols (of arbitrary length). Let $G_{[\mathbf{s}]}(v)$ be the probability of the following variable taking value v given the context $\mathbf{s}$. Denote by $G_{[\mathbf{s}]}$ the vector of probabilities (parameters) with one element for each $v \in \Sigma$. Estimating parameters that generalize well to unseen contexts given a single training sequence might seem a priori unreasonable. For example, if our training sequence were $x_{1:T} = \mathbf{s}$, it is easy to see that there is only a single observation $x_i = s_i$ in the context $x_{1:i-1} = s_{1:i-1}$ for every prefix $s_{1:i-1}$. In most cases this single observation clearly will not be sufficient to estimate a whole parameter vector $G_{[s_{1:i-1}]}$ that generalizes in any reasonable way. In the following we describe a prior that hierarchically ties together the vector of predictive probabilities in a particular context to vectors of probabilities in related, shorter contexts. By doing this we are able to use observations that occur in very long contexts to recursively inform the estimation of the predictive probabilities for related shorter contexts and vice versa.

The way we do this is to place a hierarchical Bayesian prior over the set of probability vectors $\{G_{[\mathbf{s}]}\}_{\mathbf{s}\in\Sigma^*}$. On the root node we place a Pitman-Yor prior (Pitman & Yor, 1997; Ishwaran & James, 2001) on the probability vector $G_{[]}$ corresponding to the empty context $[]$:

$$G_{[]}|d_0, c_0, H \sim \mathcal{PY}(d_0, c_0, H), \tag{2}$$

where d_0 is the discount parameter, c_0 the concentration parameter and H the base distribution.[1] For simplicity we take H to be the uniform distribution over the (assumed) finite symbol set Σ. At the first level, the random measures $\{G_{[s]}\}_{s\in\Sigma}$ are conditionally independent given $G_{[]}$, with distributions given by Pitman-Yor processes with discount parameter d_1, concentration parameter c_1 and base distribution $G_{[]}$:

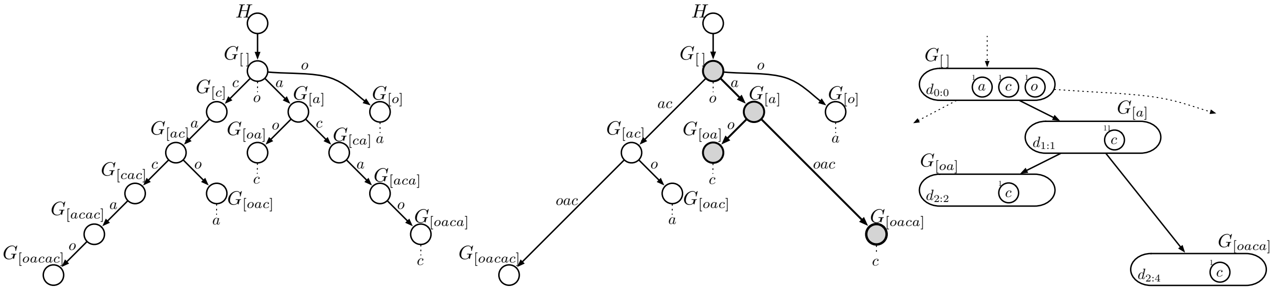
$$G_{[s]}|d_1, c_1, G_{[]} \sim \mathcal{PY}(d_1, c_1, G_{[]}). \tag{3}$$

The hierarchy is defined recursively for any number of levels. For each non-empty finite sequence of symbols $\mathbf{s}$, we have

$$G_{[\mathbf{s}]}|d_{|\mathbf{s}|}, c_{|\mathbf{s}|}, G_{[\mathbf{s}']} \sim \mathcal{PY}(d_{|\mathbf{s}|}, c_{|\mathbf{s}|}, G_{[\mathbf{s}']}), \tag{4}$$

where $[\mathbf{s}] = [s\mathbf{s}']$ for some symbol $s \in \Sigma$, that is, $\mathbf{s}'$ is $\mathbf{s}$ with the first contextual symbol removed and $|\mathbf{r}|$ is the length of string $\mathbf{r}$. The resulting graphical model can be infinitely deep and is tree-structured, with a random probability vector on each node. The number

[1] In the statistics literature the discount parameter is typically denoted by α and the concentration parameter by θ. In the machine learning literature α is often used to denote the concentration parameter instead. We use different symbols here to avoid confusion.



(a) Prefix trie for *oacac*. (b) Prefix tree for *oacac*. (c) Initialisation.

Figure 1. (a) prefix trie and (b) corresponding prefix tree for the string *oacac*. Note that (a) and (b) correspond to the suffix trie and the suffix tree of *cacao*. (c) Chinese restaurant franchise sampler representation of subtree highlighted in (b).

of branches descending from each node is given by the number of elements in Σ.

The hierarchical Pitman-Yor process (HPYP) with finite depth has been applied to language models (Teh, 2006), producing state-of-the-art results. It has also been applied to unsupervised image segmentation (Sudderth & Jordan, 2009). Defining an HPYP of unbounded depth is straightforward given the recursive nature of the HPYP formulation. One contribution of this paper to make inference in such a model tractable and efficient.

A well known special case of the HPYP is the hierarchical Dirichlet process (Teh et al., 2006), which arises from setting $d_n = 0$ for $n \geq 0$. Here, we will use a less-well-known special case where $c_n = 0$ for $n \geq 0$. In this parameter setting the Pitman-Yor process specializes to a normalized stable process (Perman, 1990). We use this particular prior because, as we shall see, it makes it possible to construct representations of the posterior of this model in time and space linear in the length of a training observation sequence. The trade-off between this particular parameterization of the Pitman-Yor process and one in which non-zero concentrations are allowed is studied in Section 6 and shown to be inconsequential in the language modelling domain. This is largely due to the fact that the discount parameter and the concentration both add mass to the base distribution in the Pitman-Yor process. This notwithstanding, the potential detriment of using a less expressive prior is often outweighed when gains in computational efficiency mean that more data can be modelled albeit using a slightly less expressive prior.

3. Representing the Infinite Model

Given a sequence of observations $x_{1:T}$ we are interested in the posterior distribution over $\{G_{[\mathbf{s}]}\}_{\mathbf{s}\in\Sigma^*}$, and ultimately in the predictive distribution for a continuation of the original sequence (or a new sequence of observations $y_{1:\tau}$), conditioned on having already observed $x_{1:T}$. Inference in the sequence memoizer as described is computationally intractable because it contains an infinite number of latent variables $\{G_{[\mathbf{s}]}\}_{\mathbf{s}\in\Sigma^*}$. In this section we describe two steps that can be taken to reduce the number of these variables such that inference becomes feasible (and efficient).

First, consider a single, finite training sequence $\mathbf{s}$ consisting of T symbols. The only variables that will have observations associated with them are the ones that correspond to contexts that are prefixes of $\mathbf{s}$, i.e. $\{G_{[\pi]}\}_{\pi\in\{\mathbf{s}_{1:i}|0\leq i<T\}}$. These nodes depend only on their ancestors in the graphical model, which correspond to the suffixes of the contexts π. Thus, the only variables that we need perform inference on are precisely all those corresponding to contexts which are contiguous subsequences of $\mathbf{s}$, i.e. $\{G_{[\mathbf{s}_{j:i}]}\}_{1\leq j\leq i<T}$.

This reduces the effective number of variables to $\mathcal{O}(T^2)$. The structure of the remaining graphical model for the sequence $\mathbf{s} = oacac$ is given in Figure 1(a). This structure corresponds to what is known as a *prefix trie*, which can be constructed from an input string in $\mathcal{O}(T^2)$ time and space (Ukkonen, 1995).

The second marginalization step is more involved and requires a two step explanation. We start by highlighting a marginalization transformation of this prefix trie graphical model that results in a graphical model with fewer nodes. In the next section we describe how such analytic marginalization operations can be done for the Pitman-Yor parameterization ($c_n = 0 \;\forall\; n$) we have chosen.

Consider a transformation of the branch of the graphical model trie in Figure 1(a) that starts with a. The transformation of interest will involve marginalizing out variables like $G_{[ca]}$ and $G_{[aca]}$. In general we are interested in marginalizing out all variables that correspond to non-branching interior nodes in the trie. Assume for now that we can in fact marginalize out such variables. What remains is to efficiently identify those variables that can be marginalized out. However, just building a prefix trie is of $\mathcal{O}(T^2)$ time and space complexity so using the trie to identify such nodes is infeasible for long observation sequences.

Interestingly, the collapsed graphical model in Figure 1(b) has a structure called a *prefix tree* that can be built directly from an input string in $\mathcal{O}(T)$ time and space complexity (Weiner, 1973; Ukkonen, 1995).[2] The resulting prefix tree retains precisely the nodes (variables) of interest, eliminating all non-branching nodes in the trie by allowing each edge label to be a sequence of symbols (or meta-symbol), rather than a single symbol. The marginalization results of the next section are used to determine the correct Pitman-Yor conditional distributions for remaining nodes.

4. Marginalization

Now that we can identify the variables that can be marginalized out it remains to show how to do this. When we perform these marginalizations we would like to ensure that the required marginalization operations result in a model whose conditional distributions remain tractable and preferably stay in the same Pitman-Yor family. In this section we show that this is the case. We establish this fact by reviewing *coagulation* and *fragmentation operators* (Pitman, 1999; Ho et al., 2006).

For the rest of this section we shall consider a single path in the graphical model, say $G_1 \rightarrow G_2 \rightarrow G_3$, with G_2 having no children other than G_3. Recall that many marginalizations of this type will be performed during the construction of the tree. Marginalizing out G_2 leaves $G_1 \rightarrow G_3$, and the following result shows that the conditional distribution of G_3 given G_1 stays within the same Pitman-Yor family:

Theorem 1. *If $G_2|G_1 \sim \mathcal{PY}(d_1, 0, G_1)$ and $G_3|G_2 \sim \mathcal{PY}(d_2, 0, G_2)$ then $G_3|G_1 \sim \mathcal{PY}(d_1 d_2, 0, G_1)$ with G_2 marginalized out.*

Clearly, Theorem 1 can be applied recursively if G_1 or G_3 have just one child as well. The resulting prefix tree graphical model has conditional distributions that are also Pitman-Yor distributed, with discounts obtained by multiplying the discounts along the paths of the uncollapsed prefix trie model. This is the key operation we use to build the prefix tree graphical model.

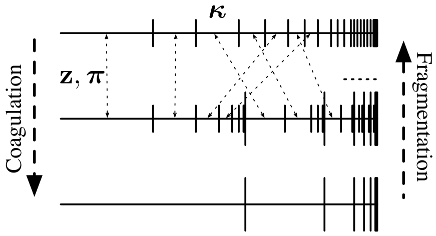


Figure 2. Depiction of coagulation and fragmentation.

Theorem 1 follows from results on *coagulation operators*. In the following we shall outline coagulation operators as well as their inverses, fragmentation operators. This will set the stage for Theorem 2 from which Theorem 1 follows. Consider the stick-breaking construction of $G_2|G_1$ and $G_3|G_2$. The weights or “sticks” are distributed according to two-parameter GEM distributions[3] with concentration parameters equal to 0:

$$G_2 = \sum_{i=1}^{\infty} \pi_i \delta_{\phi_i}, \qquad \phi_i \overset{\text{iid}}{\sim} G_1, \qquad \boldsymbol{\pi} \sim \text{GEM}(d_1, 0),$$
$$G_3 = \sum_{j=1}^{\infty} \kappa_j \delta_{\psi_j}, \qquad \psi_j \overset{\text{iid}}{\sim} G_2, \qquad \boldsymbol{\kappa} \sim \text{GEM}(d_2, 0),$$

where δ_θ is a point mass located at θ. The child measure G_3 necessarily has the same support as its parent G_2. Hence, we can sum up (i.e. *coagulate*) the sticks associated with each subset of the point masses of G_3 that correspond to a point mass of G_2, such that

$$G_3 = \sum_{i=1}^{\infty} \tau_i \delta_{\phi_i}, \qquad \tau_i = \sum_{j=1}^{\infty} \kappa_j I(z_j = i). \qquad (5)$$

Here, $I(\cdot)$ is the indicator function and $z_j = i$ if $\psi_j = \phi_i$. Note that $z_j \sim_{\text{iid}} \boldsymbol{\pi}$. The above describes a coagulation of $\boldsymbol{\kappa}$ by $\text{GEM}(d_1, 0)$. In general, we shall write $(\boldsymbol{\tau}, \boldsymbol{\pi}, \mathbf{z}) \sim \text{COAG}_{\text{GEM}(d,c)}(\boldsymbol{\kappa})$ if $\boldsymbol{\pi} \sim \text{GEM}(d, c)$, each $z_j \sim_{\text{iid}} \boldsymbol{\pi}$ and $\boldsymbol{\tau}$ is as given in Eq. (5). Intuitively, the z_j's define a partitioning of $\boldsymbol{\kappa}$, and the elements of each partition are subsequently summed up to obtain the coagulated sticks $\boldsymbol{\tau}$. The coagulation operator is the downward operation shown in Figure 2.

The reverse operation is called *fragmentation*. It takes each stick τ_i, breaks it into an infinite number of sticks, and reorders the resulting shorter sticks by *size-biased*

[2] For purposes of clarity it should be pointed out that the literature for constructing these data structures is focused entirely on suffix rather than prefix tree construction. Conveniently, however, the prefixes of a string are the suffixes of its reverse so any algorithm for building a suffix tree can be used to construct a prefix tree by simply giving it the reverse of the input sequence as input.

[3] We say that $\boldsymbol{\rho} = (\rho_k)_{k=1}^{\infty}$ is jointly $\text{GEM}(d, c)$ if $\rho_k = b_k \prod_{l=1}^{k-1}(1 - b_l)$ and $b_k \sim \text{Beta}(1 - d, c + kd)$ for all k.

permutation. The size-biased permutation of a set of positive numbers is obtained by iteratively picking (without replacement) entries with probabilities proportional to their sizes.

To be more precise, we define a fragmentation of $\boldsymbol{\tau}$ by $\mathrm{GEM}(d, c)$ as follows. For each stick τ_i, draw $\boldsymbol{\rho}_i \sim \mathrm{GEM}(d, c)$, define $\tilde{\kappa}_{ik} = \tau_i \rho_{ik}$ for all k and let $\boldsymbol{\kappa} = (\kappa_j)_{j=1}^{\infty}$ be the size-biased permutation of $(\tilde{\kappa}_{ik})_{ik=1}^{\infty}$. The fragmentation operation corresponds to the upward operation shown in Figure 2. We also require the fragmentation operation to return $\boldsymbol{\pi} = (\pi_i)_{i=1}^{\infty}$. These sticks are directly extracted from the reversal of the size-biased permutation, which maps each κ_j to some τ_i. We set $z_j = i$ in this case and define π_i as the asymptotic proportion of z_j's that take the value i:

$$\pi_i = \lim_{j \to \infty} \frac{1}{j} \sum_{l=1}^{j} I(z_l = i). \tag{6}$$

We write $(\boldsymbol{\kappa}, \boldsymbol{\pi}, \mathbf{z}) \sim \mathrm{FRAG}_{\mathrm{GEM}(d,c)}(\boldsymbol{\tau})$ if $\boldsymbol{\kappa}$, $\boldsymbol{\pi}$ and $\mathbf{z}$ are as constructed above.

Theorem 2. *The following statements are equivalent:* *(1)* $\boldsymbol{\kappa} \sim \mathrm{GEM}(d_2, c)$, $(\boldsymbol{\tau}, \boldsymbol{\pi}, \mathbf{z}) \sim \mathrm{COAG}_{\mathrm{GEM}(d_1, c/d_2)}(\boldsymbol{\kappa})$*;* *(2)* $\boldsymbol{\tau} \sim \mathrm{GEM}(d_1 d_2, c)$, $(\boldsymbol{\kappa}, \boldsymbol{\pi}, \mathbf{z}) \sim \mathrm{FRAG}_{\mathrm{GEM}(d_2, -d_1 d_2)}(\boldsymbol{\tau})$.

The above theorem was proven by (Pitman, 1999; Ho et al., 2006). While coagulation is important for constructing the collapsed model, fragmentation is important for reinstantiating nodes in the graphical model corresponding to contexts in test sequences that did not occur in the training data. For instance, we might need to perform inference in the context corresponding to node G_2 given only G_1 and G_3 in the collapsed representation (see Section 5). This is formalized in the following:

Corollary 1. *Suppose* $G_3 | G_1 \sim \mathcal{PY}(d_1 d_2, 0, G_1)$*, with stick-breaking representation* $G_3 = \sum_{i=1}^{\infty} \tau_i \delta_{\phi_i}$ *where* $\boldsymbol{\tau} \sim \mathrm{GEM}(d_1 d_2, 0)$ *and* $\phi_i \sim_{\mathrm{iid}} G_1$*. Let* $(\boldsymbol{\kappa}, \boldsymbol{\pi}, \mathbf{z}) \sim \mathrm{FRAG}_{\mathrm{GEM}(d_2, -d_1 d_2)}(\boldsymbol{\tau})$*. Then* $G_2 = \sum_{i=1}^{\infty} \pi_i \delta_{\phi_i}$ *is a draw from* $G_2 | G_1, G_3$ *and we can equivalently write* $G_3 = \sum_{j=1}^{\infty} \kappa_j \delta_{\psi_j}$ *where* $\psi_j = \phi_{z_j}$*.*

5. Inference and Prediction

Once the collapsed graphical model representation has been built, inference proceeds as it would for any other hierarchical Pitman-Yor process model. In this work we used Gibbs sampling in the Chinese restaurant franchise representation, and refer the reader to (Teh, 2006) for further details. Figure 1(c) depicts a valid seating arrangement for the restaurants in the Chinese restaurant franchise representation corresponding to each bold node in Figure 1(b).

At test time we may need to compute the predictive probability of a symbol v given a context $\mathbf{s}$ that is not in the training set. It is easy to see that the predictive probability is simply $\mathbb{E}[G_{[\mathbf{s}]}(v)] = \mathbb{E}[G_{[\mathbf{s}']}(v)]$, where $\mathbf{s}'$ is the longest suffix of $\mathbf{s}$ that occurs in the prefix *trie* and the expectations are taken over the posterior. $\mathbb{E}[G_{[\mathbf{s}']}(v)]$ can be estimated by averaging over the seating arrangements of the restaurant corresponding to $\mathbf{s}'$. However $\mathbf{s}'$ itself may not appear in the prefix *tree*, in which case we will have to *reinstantiate* the corresponding restaurant.

For concreteness, in the rest of this section we consider $\mathbf{s} = [oca]$ in Figure 1(b). The longest suffix in the prefix trie is $\mathbf{s}' = [ca]$, but this does not appear in the prefix tree as $G_{[ca]}$ has been marginalized out. Thus for predictive inference we need to reinstantiate $G_{[ca]}$ (or rather, its Chinese restaurant representation). We do this by fragmenting $G_{[oaca]} | G_{[a]}$ into $G_{[ca]} | G_{[a]}$ and $G_{[oaca]} | G_{[ca]}$. Using the equivalence between Chinese restaurant processes and stick-breaking constructions for Pitman-Yor processes, we can translate the fragmentation operation in Corollary 1 into a fragmentation operation on the Chinese restaurant representation of $G_{[oaca]} | G_{[a]}$ instead. This results in the procedure given in the next paragraph for reinstantiating the $G_{[ca]} | G_{[a]}$ restaurant.

Suppose there are K tables in the $G_{[oaca]} | G_{[a]}$ restaurant, table k having n_k customers. Independently for each table k, sample a partition of n_k customers in a restaurant corresponding to a Pitman-Yor process with discount parameter $d_3 d_4$ and concentration parameter $-d_2 d_3 d_4$. Say this results in J_k tables, with numbers of customers being n_{kj}, with $\sum_{j=1}^{J_k} n_{kj} = n_k$. The n_k customers in the original table are now seated at J_k tables in the $G_{[oaca]} | G_{[ca]}$ restaurant with table j having n_{kj} customers. Each of these tables sends a customer to the $G_{[ca]} | G_{[a]}$ restaurant; these customers are all seated at the same table. There was one customer in the $G_{[a]} | G_{[]}$ restaurant corresponding to the original table in $G_{[oaca]} | G_{[a]}$ with n_k customers. There is still one customer in $G_{[a]} | G_{[]}$ corresponding to the new table in $G_{[ca]} | G_{[a]}$, thus this restaurant's seating arrangement needs not be altered.

6. Experiments

We are interested in understanding the potential impact of using a sequence memoizer in place of a Markov model in general modelling contexts. To start we explore two issues: first, whether using prefix trees instead of prefix tries empirically gives the computational savings that is expected under worst-case analysis; and second, whether the predictive performance of the sequence memoizer compares favorably to a Markov model with similar complexity.

To provide concrete answers to these questions we turn to n-gram language modeling. Applying the sequence memoizer in this application domain is equivalent to letting $n \to \infty$ in an n-gram HPYP language model. For this reason we will refer to the sequence memoizer as an ∞-*gram* HPYP in language modeling contexts. For comparison, we used n-gram HPYPs with finite n as state-of-the-art baselines (Teh, 2006). The sequence of observations used as training data will be referred to as the training corpus and the predictive power of the models will be measured in terms of test corpus perplexity.

The datasets used in our experiments were an excerpt from the New York Times (NYT) corpus and the entire Associated Press (AP) corpus. The latter corpus is exactly the same as that used in (Bengio et al., 2003; Teh, 2006; Mnih & Hinton, 2009), allowing us to compare perplexity scores against other state-of-the-art models. The AP training corpus (with 1 million word validation set folded in) consisted of a total of 15 million words while the AP test corpus consisted of 1 million words. The AP dataset was preprocessed to replace low frequency words (< 5 appearances) with a single "unknown word" symbol, resulting in 17964 unique word types. This preprocessing is semi-adversarial for the ∞-gram model because the number of unique prefixes in the data is lowered, resulting in less computational savings for using the prefix tree relative to the trie. The NYT training corpus consisted of approximately 13 million words and had a 150,000 word vocabulary. The NYT test corpus consisted of approximately 200,000 words. In this more realistic dataset no preprocessing was done to replace low frequency words. For this reason we used the NYT dataset to characterize the computational savings of using the prefix tree.

We used the CRF sampler outlined in Section 5 with the addition of Metropolis-Hastings updates for the discount parameters (Wood & Teh, 2009). The discounts in the collapsed node restaurants are products of subsets of discount parameters making other approaches difficult. We use distinct discount parameters for each of the first four levels of the trie, while levels below use a single shared discount parameter. Theoretically the model can use different discounts for every depth or node in the trie. Our choice in this regard was somewhat arbitrary and warrants more experimentation. The discounts were initialized to $d_{[0,1,2,\ldots]} = (.62, .69, .74, .80, .95, .95, \ldots)$. We used extremely short burn-in (10 iterations) and collected only 5 samples. We found that this produced the same perplexity scores as a sampler using 125 burn-in iterations and 175 samples (Teh, 2006), which indicates that the posterior structure is simple and efficiently traversed by our sampler. The CRF states were initialized such that all customers of the same type in each restaurant were seated at a single table. This initial configuration corresponds to interpolated Kneser-Ney (Teh, 2006).

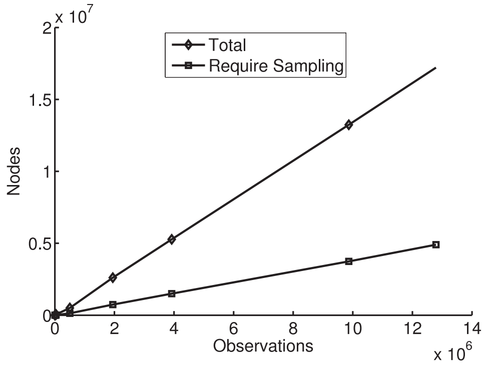


Figure 3. Total number of nodes in the tree and number of nodes that have to be sampled as a function of number of NYT observations. The number of nodes in the corresponding trie scales quadratically in the number of observations and is not shown. For reference the number of nodes in the trie corresponding to the rightmost data point is 8.2×10^{13}.

Figure 3 shows the number of nodes in prefix trees growing linearly in the corpus size. We found that the total number of nodes in the trie indeed grows quadratically in the corpus size. We do not show this quadratic growth in the figure because its scale is so extreme. Instead, the figure also shows the number of nodes that have to be sampled in the ∞-gram model, which also grows linearly in the corpus size, albeit at a lower rate. In the tree CRF representation none of the leaf nodes need to be sampled because they all will contain a single customer sitting at a single table, thus the number of nodes that have to be sampled is approximately the number of internal nodes in the prefix tree.

While the growth rate of the trie graphical model is quadratic, n-gram HPYP models do not instantiate more nodes than are necessary to represent the unique contexts in the training corpora. Figure 4 explores the numbers of nodes created in n-gram and ∞-gram HPYP language models, for differently sized NYT corpora and for different values of n. The figure uncovers two interesting fact. First, in all n-gram models the growth in the number of nodes is intially quadratic and then becomes linear. If the plot is extended to the right

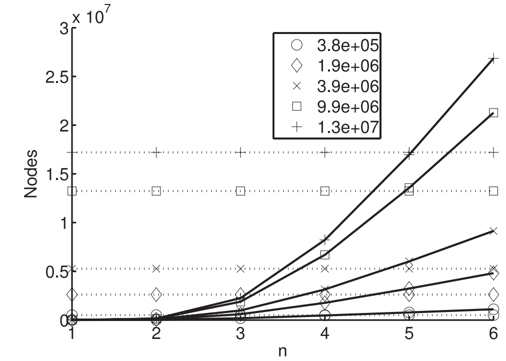


Figure 4. Nodes in prefix tries and trees as a function of n (of n-gram) and for different NYT corpora sizes. Horizontal lines are prefix tree node counts. Curved lines are prefix trie node counts. Sizes of corpora are given in the legend.

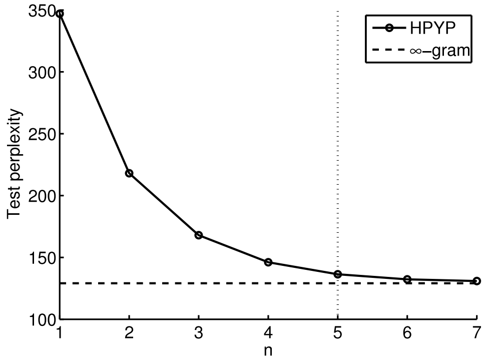


Figure 5. NYT test perplexity for n-gram and ∞-gram HPYP language models given a 4 million word subset of the NYT training corpus. The dotted line indicates the first n-gram model that has more nodes than the ∞-gram model.

significantly beyond $n = 6$ we observe that this linear growth continues for a long time. This transition between quadratic and linear growth can be explained by observing that virtually all branch points of the trie occur above a certain depth, and below this depth only linear paths remain. Also, at $n = 5$ the number of nodes in the n-gram trie is roughly the same (greater in all but one case) as the number of nodes in the ∞-gram.

Questions of model size automatically lead to questions of the expressive power of the models. Figure 5 compares the expressive power of the n-gram HPYP language model against the ∞-gram model, using the test set perplexity as a proxy. We see that the predictive performance of the n-gram HPYP asymptotically approaching that of the ∞-gram. While the performance gain over the n-gram model is modest, and certainly goes to zero as n increases, remember that the

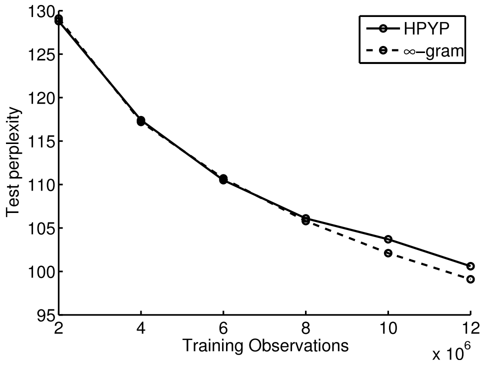


Figure 6. AP test perplexity vs. AP training corpus size for the 5-gram HPYP language model vs. the ∞-gram

Table 1. Reported AP test perplexities.

Source	Perplexity
(Mnih & Hinton, 2009)	112.1
(Bengio et al., 2003)	109.0
4-gram Modified Kneser-Ney (Teh, 2006)	102.4
4-gram HPYP (Teh, 2006)	101.9
∞-gram (Sequence Memoizer)	96.9

computational cost associated with the n-gram surpasses the ∞-gram after $n = 5$. This indicates that there is no reason, computational or statistical, for preferring n-gram models over the ∞-gram when $n \geq 5$.

In the next set of experiments we switch to using the AP corpus instead. Figure 6 shows the test perplexities of both the 5-gram HPYP and the ∞-gram fit to AP corpora of increasing size. For small training corpora its performance is indistinguishable from that of the ∞-gram. Furthermore, as the corpus size grows, enough evidence about meaningful long contexts begins to accrue to give the ∞-gram a slight relative advantage. It bears repeating here that the AP corpus is preprocessed in a way that will minimize this advantage.

Finally, despite the AP corpus being semi-adversarially preprocessed, the ∞-gram achieves the best test perplexity of any language model we know of that has been applied to the AP corpus. Comparative results are given in Table 1. This is somewhat surprising and worth further study. Remember the trade-off between using the ∞-gram vs. the n-gram HPYP: the n-gram HPYP allows for non-zero concentrations at all levels, whereas the ∞-gram requires all concentrations to be set to zero. Conversely, the advantage of the ∞-gram is that it can utilize arbitrarily long contexts whereas the

n-gram model cannot. That the ∞-gram produces better results than the HPYP can thus be explained by the fact that at least some long contexts in (English) language must sharpen predictive performance. Further, the advantage of having free concentration parameters must be minimal.

7. Discussion

The sequence memoizer is a model for sequence data that allows one to model extremely long contextual dependencies. Using hierarchical Bayesian sharing of priors over conditional distributions with similar contexts we are able to learn such a model from a single long sequence and still produce good generalization results. We show that the model can be efficiently constructed using suffix trees and coagulation operators. We provide a sampler for the model and introduce the fragmentation operations necessary to perform predictive inference in novel contexts.

The sequence memoizer (or ∞-gram) performs well as a language model. That it achieves the best known test perplexity on a well studied corpus may be of interest to a large community of natural language researchers. Diving deeper to uncover the precise reason for the improvement (deeper than acknowledging that language is non-Markov and long contexts do matter at least in some instances) is worthy of additional research. In a similar respect, a direct comparison of the sequence memoizer as a language model to existing variable order Markov models might be of interest (Mochihashi & Sumita, 2008).

In the authors' view the language modelling success is merely a validating consequence of the primary contribution of the paper: the sequence memoizer itself. We emphasize that there are other potential applications for our model for instance text compression using character level sequential models (Cleary & Teahan, 1997).

References

Bengio, Y., Ducharme, R., Vincent, P., & Jauvin, C. (2003). A neural probabilistic language model. *Journal of Machine Learning Research*, *3*, 1137–1155.

Cleary, J. G. & Teahan, W. J. (1997). Unbounded length contexts for PPM. *The Computer Journal*, *40*, 67–75.

Goodman, N. D., Mansinghka, V. K., Roy, D., Bonawitz, K., & Tenenbaum, J. B. (2008). Church: a language for generative models. In *Uncertainty and Artificial Intelligence*. to appear.

Ho, M. W., James, L. F., & Lau, J. W. (2006). Coagulation fragmentation laws induced by general coagulations of two-parameter Poisson-Dirichlet processes. http://arxiv.org/abs/math.PR/0601608.

Ishwaran, H. & James, L. F. (2001). Gibbs sampling methods for stick-breaking priors. *Journal of American Statistical Association*, *96*(453), 161–173.

Michie, D. (1968). Memo functions and machine learning. *Nature*, *218*, 19–22.

Mnih, A. & Hinton, G. (2009). A scalable hierarchical distributed language model. In *Neural Information Processing Systems 22*. to appear.

Mochihashi, D. & Sumita, E. (2008). The infinite Markov model. In *Advances in Neural Information Processing Systems 20*, (pp. 1017–1024).

Perman, M. (1990). *Random Discrete Distributions Derived from Subordinators*. PhD thesis, Department of Statistics, University of California at Berkeley.

Pitman, J. (1999). Coalescents with multiple collisions. *Annals of Probability*, *27*, 1870–1902.

Pitman, J. & Yor, M. (1997). The two-parameter Poisson-Dirichlet distribution derived from a stable subordinator. *Annals of Probability*, *25*, 855–900.

Sudderth, E. B. & Jordan, M. I. (2009). Shared segmentation of natural scenes using dependent pitman-yor processes. In *Neural Information Processing Systems 22*. to appear.

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

Teh, Y. W., Jordan, M. I., Beal, M. J., & Blei, D. M. (2006). Hierarchical Dirichlet processes. *Journal of the American Statistical Association*, *101*(476), 1566–1581.

Ukkonen, E. (1995). On-line construction of suffix trees. *Algorithmica*, *14*, 249–260.

Weiner, P. (1973). Linear pattern matching algorithms. In *IEEE 14th Annual Symposium on Switching and Automata Theory*, (pp. 1–11).

Wood, F. & Teh, Y. W. (2009). A hierarchical nonparametric Bayesian approach to statistical language model domain adaptation. In *Journal of Machine Learning, Workshop and Conference Proceedings: Artificial Intelligence in Statistics 2009*, volume 5, (pp. 607–614).