Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence (IJCAI 2015)

Count-Based Frequency Estimation With Bounded Memory

Marc G. Bellemare
Google DeepMind
London, United Kingdom
bellemare@google.com

Abstract

Count-based estimators are a fundamental building block of a number of powerful sequential prediction algorithms, including Context Tree Weighting and Prediction by Partial Matching. Keeping exact counts, however, typically results in a high memory overhead. In particular, when dealing with large alphabets the memory requirements of count-based estimators often become prohibitive. In this paper we propose three novel ideas for approximating count-based estimators using bounded memory. Our first contribution, of independent interest, is an extension of reservoir sampling for sampling distinct symbols from a stream of unknown length, which we call K-distinct reservoir sampling. We combine this sampling scheme with a state-of-the-art count-based estimator for memoryless sources, the Sparse Adaptive Dirichlet (SAD) estimator. The resulting algorithm, the Budget SAD, naturally guarantees a limit on its memory usage. We finally demonstrate the broader use of K-distinct reservoir sampling in nonparametric estimation by using it to restrict the branching factor of the Context Tree Weighting algorithm. We demonstrate the usefulness of our algorithms with empirical results on two sequential, large-alphabet prediction problems.

1 Introduction

Undoubtedly, *counting* is the simplest way of estimating the relative frequency of symbols in a data sequence. In the sequential prediction setting, these estimates are used to assign to each symbol a probability proportionate to its frequency. When dealing with binary alphabets the counting approach gives rise to the Krichevsky-Trofimov estimator, known to be minmax optimal [Krichevsky and Trofimov, 1981]. Count-based approaches are also useful in dealing with large alphabets, and a number of methods coexist [Katz, 1987; Tjalkens *et al.*, 1993; Friedman and Singer, 1999; Hutter, 2013], each with its own statistical assumptions and regret properties. While these simple estimators deal only with memoryless sources – whose output is context invariant – they play a critical role as building blocks for more complicated estimators [Cleary and Witten, 1984; Willems *et al.*, 1995; Tziortziotis *et al.*, 2014].

We are interested in the large alphabet setting as it naturally lends itself to the compression of language data, where symbols consist of letters, phonemes, words or even sentence fragments. This setting is also of practical interest in reinforcement learning, for example in learning dynamical models for planning [Farias *et al.*, 2010; Bellemare *et al.*, 2013]. In both contexts, the statistical efficiency of count-based estimators makes them excellent modelling candidates. However, the memory demands of such estimators typically increase linearly with the effective alphabet size M, which is problematic when M grows with the sequence length. Perhaps surprisingly, this issue is more than just a theoretical concern and often occurs in language data [Gale and Church, 1990].

Our work is also motivated by the desire to improve the Context Tree Weighting algorithm [CTW; Willems et al., 1995] to model k-Markov large alphabet sources. While CTW is a powerful modelling technique, its practical memory requirements (linear in the size of the sequence) often preclude its use in large problems. Here, even when the alphabet size is fixed and small, the number of observed k-order *contexts* typically grows with the sequence length. We are particularly interested in reducing the long term memory usage of such an estimator, for example when estimating attributes of internet packets going through a router or natural language models trained on nonstationary data.

Our solution takes its inspiration from the reservoir sampling algorithm [Knuth, 1981] and the work of Dekel *et al.* [2008] on budget perceptrons. We propose an online, randomized algorithm for sampling distinct symbols from a data stream; by choosing a sublinear reservoir size, we effectively force our statistical estimator to forget infrequent symbols. We show that this algorithm can easily be combined with a typical count-based estimator, the Sparse Adaptive Dirichlet estimator, to guarantee good statistical estimation while avoiding unbounded memory usage. As our results demonstrate, the resulting estimator is best suited to sources that emit a few frequent symbols while also producing a large pool of low-frequency "noise". We further describe a simple application of our sampling algorithm to branch pruning within the CTW algorithm.

The idea of sampling distinct symbols from a stream is

well-studied in the literature. Perhaps most related to ours is the work of Gibbons and Matias [1998], who proposed the notion of a *concise sample* to more efficiently store a "hot list" of frequent elements. Their algorithm is also based on reservoir sampling, but relies on a user-defined threshold and probabilistic counting. By contrast, our algorithm maintains a reservoir based on a random permutation of the data stream, which avoids the need for a threshold. Gibbons [2001] proposed an algorithm called *Distinct Sampling* to estimate the number of distinct values within a data stream. Similar to the hot list algorithm, Distinct Sampling employs a growing threshold, which is used to determine when infrequent symbols should be discarded. Metwally *et al.* [2005] proposed to track frequent symbols using a linked list and a hash table. Their *Space Saving* algorithm aims to find all frequently occurring items (heavy hitters), rather than provide a frequency-biased sample; a similar scheme was also proposed by Demaine *et al.* [2002]. Manku and Motwani [2002] proposed a probabilistic counting and halving approach to find all items whose frequency is above a user-defined threshold. Charikar *et al.* [2004] showed how to use a probabilistic *count sketch* in order to find frequent items in a data stream. While count sketches may seem the natural fit for count-based estimators, their memory requirements grow quadratically with the accuracy parameter ϵ; by contrast, our algorithm has only a linear dependency on the implied ϵ.

2 Background

In the sequential prediction setting we consider, an algorithm must make probabilistic predictions over a sequence of symbols observed one at a time. These symbols belong to a finite alphabet $\mathcal{X}$, with $x_{1:n} := x_1 \dots x_n \in \mathcal{X}^n$ denoting a string drawn from this alphabet. A probabilistic prediction consists in assigning a probability to a new symbol x_{n+1} given that $x_{1:n}$ has been observed.

Let $[n] := \{1, \dots, n\}$. We denote the set of finite strings by $\mathcal{X}^* := \bigcup_{i=0}^{\infty} \mathcal{X}^i$, denote the concatenation of strings s and r by sr, and use ϵ to represent the empty string. For a set $\mathcal{B} \subseteq \mathcal{X}$, we write $x_{1:n} \setminus \mathcal{B} := (x_i : i \in [n], x_i \notin \mathcal{B})$ to denote the substring produced by excising from $x_{1:n}$ the symbols in $\mathcal{B}$. We denote by $N_n(x) := N(x, x_{1:n})$ the number of occurrences of $x \in \mathcal{X}$ within $x_{1:n}$, and write $\tau_n(x) := \tau(x, x_{1:n})$ for the time of first occurrence of x in $x_{1:n}$, with $\tau(x, x_{1:n}) := n + 1$ whenever $x \notin x_{1:n}$. We denote the set of permutations of $x_{1:n}$ by $\mathcal{P}(x_{1:n})$ and the discrete uniform distribution over $\mathcal{X}$ by $\mathscr{U}_{\mathcal{X}}(\cdot)$. Finally, unless otherwise specified, we write $\log$ to mean the natural logarithm.

A coding distribution ρ is a sequence $(\rho_t : t \in \mathbb{N})$ of probability distributions $\rho_t : \mathcal{X}^t \to [0, 1]$ each respecting

1. $\sum_{x \in \mathcal{X}} \rho_{t+1}(x_{1:t}x) = \rho_t(x_{1:t})$, and
2. $\rho_0(\epsilon) = 1$.

In what follows the subscript to ρ_t is always implied from its argument; we therefore omit it. A coding distribution assigns conditional probabilities according to

$$\rho(x \mid x_{1:t}) := \frac{\rho(x_{1:t}x)}{\rho(x_{1:t})},$$

which after rearrangement yields the familiar chain rule $\rho(x_{1:n}) := \prod_{t=1}^{n} \rho(x_t \mid x_{1:t-1})$.

A *source* is a distinguished type of coding distribution which we assume generates sequences. When a coding distribution is not a source, we call it a *model*. A source $(\mu_t : t \in \mathbb{N})$ is said to be *memoryless* when $\mu_t(x \mid x_{1:t}) = \mu_t(x \mid \epsilon) =: \mu_t(x)$ for all t, all $x \in \mathcal{X}$, and all sequences $x_{1:t} \in \mathcal{X}^t$; for $\mathcal{B} \subseteq \mathcal{X}$, we then write $\mu_t(\mathcal{B}) := \sum_{x \in \mathcal{B}} \mu_t(x)$. When $\mu_t(x) := \mu_1(x)$ for all t, we say it is *stationary* and omit its subscript.

Let $x_{1:n} \in \mathcal{X}^n$ and let μ be an unknown coding distribution, for example the source that generates $x_{1:n}$. We define the redundancy of a model ρ with respect to μ as

$$\mathfrak{F}_n(\rho, \mu) := \mathfrak{F}(\rho, \mu, x_{1:n}) := -\log \frac{\rho(x_{1:n})}{\mu(x_{1:n})}.$$

The redundancy $\mathfrak{F}_n(\rho, \mu)$ can be interpreted as the excess bits (or nats) required to encode $x_{1:n}$ using ρ rather than μ. The expectation of this redundancy with respect to a random sequence $x_{1:n}$ is the *Kullback-Leibler divergence* $\mathrm{KL}_n(\mu \,\|\, \rho)$:

$$\mathrm{KL}_n(\mu \,\|\, \rho) := \mathop{\mathbf{E}}_{x_{1:n} \sim \mu} \left[\mathfrak{F}_n(\rho, \mu) \right].$$

Typically, we are interested in how well ρ compares to a class of coding distributions $\mathcal{M}$. The redundancy of ρ with respect to this class $\mathcal{M}$ is defined as

$$\mathfrak{F}_n(\rho, \mathcal{M}) := \max_{\rho' \in \mathcal{M}} \mathfrak{F}_n(\rho, \rho').$$

Intuitively, $\mathfrak{F}_n(\rho, \mathcal{M})$ measures the cost of encoding $x_{1:n}$ with ρ rather than the best model in $\mathcal{M}$.

2.1 The Sparse Adaptive Dirichlet Estimator

The Sparse Adaptive Dirichlet estimator [SAD; Hutter, 2013] is a count-based frequency estimator designed for large-alphabet sources with sparse support. Informally, the SAD predicts according to empirical frequencies in $x_{1:n}$, while reserving some escape probability γ for unseen symbols. Given $\mathcal{A}_t := \bigcup_{i=1}^{t} \{x_i\}$ the set of symbols seen up to time t and a probability distribution w_t over $\mathcal{X} \setminus \mathcal{A}_t$, the SAD estimator predicts according to

$$\rho^{\mathrm{SAD}}(x \mid x_{1:t}) := \begin{cases} \frac{N_t(x)}{t + \gamma_t} & \text{if } x \in \mathcal{A}_t, \\ \frac{\gamma_t w_t(x)}{t + \gamma_t} & \text{otherwise.} \end{cases}$$

$$\gamma_t := \gamma(t, \mathcal{A}_t) := \begin{cases} 1 & \text{if } t = 0, \\ 0 & \text{if } \mathcal{A}_t = \mathcal{X}, \\ \frac{|\mathcal{A}_t|}{\log \frac{t+1}{|\mathcal{A}_t|}} & \text{otherwise.} \end{cases}$$

Let $\mathcal{M}^{\mathrm{SAD}}$ be the class of memoryless sources generating symbols from a subalphabet $\mathcal{A} \subseteq \mathcal{X}$ of size M. The redundancy of the SAD estimator with respect to this class is bounded as

$$\mathfrak{F}_n(\rho^{\mathrm{SAD}}, \mathcal{M}^{\mathrm{SAD}}) \leq \frac{M - 1}{2} \log n + O\left(M \log \log \frac{n}{M} \right),$$

making it particularly apt at dealing with sparse-support sources.

3 Budget Sequential Estimation

The SAD estimator requires counting the occurrence of *all* encountered symbols. This is usually undesirable when the observed alphabet grows with the sequence length. Here we propose a sampling solution to this problem; this solution, K-distinct reservoir sampling, lets us impose a memory constraint on our estimator. More broadly, our sampling scheme can be combined with any algorithm with a non-parametric component; as an illustration, in Section 5.2 we apply our solution to restrict the branching factor of the Context Tree Weighting algorithm.

Let μ be an unknown source. We are given a parameter $K \in \mathbb{N}$ and seek a count-based estimator that predicts $x_{1:n} \sim \mu$ sequentially without using more than $O(K)$ memory. We call this problem the *budget, sequential, count-based prediction* setting. While similar settings have been studied extensively in the context of linear classifiers [Dekel *et al.*, 2008; Cavallanti *et al.*, 2007; Orabona *et al.*, 2008] and in relation to the 0–1 loss [Lu and Lu, 2011], to the best of our knowledge the notion of a budget count-based frequency estimator is novel. From a practical perspective, we are interested in prediction problems for which the alphabet is large but the source skewed towards a few high-frequency symbols; it is for these problems that we expect meaningful prediction to be achievable.

Our competitor class $\mathcal{M}$ is the set of budget multinomial distributions. More specifically, we consider models of the form $\rho(x\,;\,\mathcal{Z},\theta,\theta_0)$ with $\mathcal{Z} := \{x^{i_1},\dots,x^{i_K}\} \subseteq \mathcal{X}$ is a sub-alphabet of cardinality K; $\theta \in \mathbb{R}^K$; and $\theta_0 \in \mathbb{R}$. Each such model predicts according to

$$\rho(x\,;\,\mathcal{Z},\theta,\theta_0) := \begin{cases} \theta_j & \text{if } x = x^{i_j} \in \mathcal{Z} \\ \theta_0 \,/\, |\mathcal{X}\setminus\mathcal{Z}| & \text{otherwise,} \end{cases} \tag{1}$$

with the usual requirement that $\sum_{x\in\mathcal{X}} \rho(x\,;\,\mathcal{Z},\theta,\theta_0) = 1$. Let μ be a memoryless stationary source and for $\mathcal{B} \subseteq \mathcal{X}$ let

$$\mu_{\mathcal{B}}(x) = \begin{cases} 0 & \text{if } x \notin \mathcal{B}, \\ \mu(x)/\mu(\mathcal{B}) & \text{otherwise.} \end{cases}$$

Finally, arrange $\mathcal{X} := \{x^1, x^2, \dots, x^m\}$ in order of decreasing probability, i.e. such that for all i, $\mu(x^i) \geq \mu(x^{i+1})$.

Lemma 1. *The model $\rho^*_\mu \in \mathcal{M}$ which minimizes $KL_1(\mu\,\|\,\cdot)$ is*

$$\mathcal{Z}^* = \operatorname*{arg\,min}_{\mathcal{Z}:|\mathcal{Z}|=K} \left[\mu(\mathcal{X}\setminus\mathcal{Z})KL_1(\mu_{\mathcal{X}\setminus\mathcal{Z}}\,\|\,\mathscr{U}_{\mathcal{X}\setminus\mathcal{Z}})\right]$$

$$\theta^*(x) = \mu(x)\ \ \forall x \in \mathcal{Z}^* \qquad \theta^*_0 = \mu(\mathcal{X}\setminus\mathcal{Z}^*).$$

Proof (sketch). We first show that for a fixed $\mathcal{Z}$, the optimal parameters correspond to those given above. We then use the convexity property of the KL divergence, namely that for $\lambda \in [0,1]$ and pairs of probability distributions $(p_1,q_1),(p_2,q_2)$

$$\mathrm{KL}_1\big(\lambda p_1 + (1-\lambda)p_2 \,\|\, \lambda q_1 + (1-\lambda)q_2\big) \leq \lambda \mathrm{KL}_1(p_1\,\|\,q_1) + (1-\lambda)\mathrm{KL}_1(p_2\,\|\,q_2),$$

with equality when (p_1,q_1) and (p_2,q_2) have disjoint support. The result follows by setting $p_1 = q_1 = \mu_{\mathcal{Z}}$, and taking p_2, q_2 to be respectively $\mu_{\mathcal{X}\setminus\mathcal{Z}}$ and $\rho^*_{\mu,\mathcal{X}\setminus\mathcal{Z}} = \mathscr{U}_{\mathcal{X}\setminus\mathcal{Z}}$. □

The full proof is provided in the supplemental. The next result shows that the optimal $\mathcal{Z}^*$ is composed of a combination of the most and least frequent symbols according to μ.

Lemma 2. *Let $\mathcal{Z}^*$ be the subalphabet associated with the model ρ^*_μ. Then $\mathcal{Z}^* = H \cup L$, where $H, L \subseteq \mathcal{X}$ are disjoint sets such that*

- *for all $x \in H, y \in \mathcal{X}\setminus H$, $\mu(x) \geq \mu(y)$, and*
- *for all $x \in L, y \in \mathcal{X}\setminus L$, $\mu(x) \leq \mu(y)$.*

Proof (sketch). The proof follows by using the identity

$$\mathrm{KL}(X\,\|\,\mathscr{U}_{\mathcal{X}}) = \log|\mathcal{X}| - H(X)$$

for a discrete random variable X distributed over $\mathcal{X}$, with $H(X)$ its entropy [Cover and Thomas, 1991]. We then show that, for $\mathcal{Z} = \{x^i\}$, the function $\mu(\mathcal{X}\setminus\mathcal{Z})\mathrm{KL}_1(\mu_{\mathcal{X}\setminus\mathcal{Z}}\,\|\,\mathscr{U}_{\mathcal{X}\setminus\mathcal{Z}})$ is concave in i. The result is then extended to the general case $\mathcal{Z} = \{x^{i_1},\dots x^{i_K}\}$. □

When μ is known, ρ^*_μ can be constructed in time $O(K)$ by considering the different choices of H and L. Furthermore, the set $\hat{\mathcal{Z}}$ containing the K most frequent symbols achieves an expected redundancy at most $\frac{1}{e} + \frac{K}{|\mathcal{X}|}\log K$ greater than the expected redundancy of ρ^*_μ (Lemma 3, supplemental), in our experiments and for large alphabets this redundancy is negligible. In the sequel we therefore assume that $\mathcal{Z}^* = \hat{\mathcal{Z}}$.

However, we are interested in the setting where μ is unknown. In this case, our budget requirement poses a circular problem: we need good estimates of μ to find $\mathcal{Z}^*$, and simultaneously need $\mathcal{Z}^*$ to estimate μ (with bounded memory). Our solution is to maintain a time-varying set $\mathcal{Y}$ which contains (with high probability) the K most frequent symbols.

We call our algorithm *K-distinct reservoir sampling*, in relation to the reservoir sampling algorithm [Knuth, 1981]. As we shall see in Section 3.2, we can leverage K-distinct reservoir sampling to approximate the SAD estimator by estimating the frequency of symbols in $\mathcal{Y}$. We call the resulting approximation the *Budget Sparse Adaptive Dirichlet estimator*, or Budget SAD for short.

3.1 K-Distinct Reservoir Sampling

Let $x_{1:n}$ be a string of *a priori* unknown length and let $K \in \mathbb{N}$. *Reservoir sampling* [Algorithm R; Knuth, 1981] is an algorithm for sampling $\mathbf{y} := y_1 \dots y_K \in \mathcal{X}^K$ from $x_{1:n}$, without replacement and using $O(K)$ memory. It proceeds as follows:

1. Randomly assign $x_1 \dots x_K$ to $y_1 \dots y_K$.
2. For $t = K+1,\dots,n$, draw $r \sim \mathscr{U}(\{1,\dots,t\})$. If
   (a) $r \leq K$, then assign x_t to y_r;
   (b) otherwise do nothing.

We would like to use reservoir sampling to sample an approximation to $\mathcal{Z}^*$. However, Algorithm R is inadequate for our purposes: since $\mathbf{y}$ may contain duplicates, sampling K *distinct* items may require significant memory. Our solution is to modify Algorithm R to compactly represent duplicates. We begin with the following observation:

Observation 1. *Let $\tilde{x}_{1:n}$ be a random permutation of $x_{1:n}$, that is $\tilde{x}_{1:n} \sim \mathscr{U}(\mathcal{P}(x_{1:n}))$. Then $\mathbf{y} := \tilde{x}_1 \dots \tilde{x}_K$ is a sample of size K sampled without replacement from $x_{1:n}$.*

We can therefore interpret Algorithm R as maintaining the first K elements of a random permutation of $x_{1:n}$. By analogy, we define the *K-distinct operator*, which describes the first K distinct elements of a string:

Definition 1. *Let $w_{1:m} \in \mathcal{X}^m$. The* K-distinct operator $\phi_K : \mathcal{X}^* \to \bigcup_{i=0}^{K} \mathcal{X}^i$ *is recursively defined as*

$$\phi_K(w_{1:m}) := \begin{cases} \epsilon & K = 0 \text{ or } m = 0 \\ w_1 \phi_{K-1}(w_{1:m} \backslash \{w_1\}) & \text{otherwise.} \end{cases}$$

The next step is to use this operator to define a *K-distinct sample* of $x_{1:n}$. From here onwards, we assume without loss of generality that sequences contain at least K distinct symbols.

Definition 2. *Let $Y_n := Y(x_{1:n})$ be a random variable taking values in $\mathcal{X}^K$, and let $\tilde{x}_{1:n} \sim \mathscr{U}(\mathcal{P}(x_{1:n}))$. Then Y_n is a* K-distinct sample *of $x_{1:n}$ if*

$$\Pr\{Y_n = \mathbf{y}\} = \Pr\{\phi_K(\tilde{x}_{1:n}) = \mathbf{y}\},$$

In our algorithm, the simple reservoir is replaced by a *K-concise summary* which keeps track of these K first distinct symbols and their times $\tau(\cdot, w_{1:m})$ of first occurence:

Definition 3. *Let $w_{1:m} \in \mathcal{X}^m$ with $y_1 \dots y_{K+1} := \phi_{K+1}(w_{1:m})$. The* K-concise summary *of $w_{1:m}$ is a vector of K pairs $\mathcal{S} := \langle (y_i, d_i) \in \mathcal{X} \times \mathbb{N} \rangle$ such that, for all $i \le K$,*

1. *$d_i = \tau(y_{i+1}, w_{1:m}) - \tau(y_i, w_{1:m})$, or equivalently*
2. *$D_i := \sum_{j=1}^{i-1} d_j = \tau(y_i, w_{1:m}) - 1$ for $i \in [K+1]$.*

Each pair (y_i, d_i) in the K-concise summary of a string thus describes this string's i^{th} distinct symbol and the gap d_i between this distinct symbol and the next (Figure 1, bottom row).

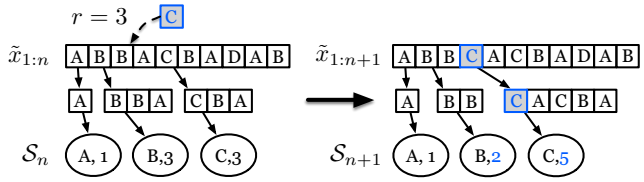

Figure 1: Insertion of 'C' at $r = 3$ into a 3-concise summary.

In the spirit of Algorithm R, K-distinct reservoir sampling incrementally generates a random permutation $\tilde{x}_{1:n}$ through a sequence of insertions, and maintains the K-concise summary of this permutation. Effectively, this summary allows us to describe the beginning of $\tilde{x}_{1:n}$ compactly (using $O(K)$ memory), while preserving the information needed for later updates.

For $x \in \mathcal{X}$, $r \in [D_{K+1}]$, and $\mathcal{S} := \langle (y_i, d_i) : i \in [K] \rangle$, define the vector and set of stored symbols

$$\mathbf{y}(\mathcal{S}) := \langle y_i : i \in [K] \rangle \qquad \mathcal{Y}(\mathcal{S}) := \{y_i : i \in [K]\}$$

as well as the indices $I : \mathcal{Y}(\mathcal{S}) \to [K]$ and $J : [D_{K+1}] \to [K]$:

$$\begin{aligned} I(x) &:= i \in [K] \text{ s.t. } y_i = x \\ J(r) &:= \max\{i \in [K+1] : r \le D_i\}. \end{aligned}$$

In K-distinct reservoir sampling (Algorithm 1), $I(x)$ and $J(r)$ are used to indicate, respectively, the index of x in $\mathcal{S}$ and the index in $\mathcal{S}$ at which a new symbol should be inserted. For example, in Figure 1 (left) $I(\text{'}B\text{'}) = 2$ and $J(3) = 3$.

Algorithm 1 K-Distinct Reservoir Sampling.

Initially: $\mathcal{S} = \langle\rangle$

SAMPLE$(K, x_{1:n})$
  for $t = 1 \dots n$ **do**
    Observe x_t
    Draw $r \sim \mathscr{U}(\{0, \dots, t-1\})$
    if $r > D_{K+1}$ **then** do nothing
    else if $x_t \notin \mathcal{Y}(\mathcal{S})$ **then** INSERT$(\mathcal{S}, x_t, r)$
    else if $I(x_t) < J(r)$ **then** $d_{J(r)-1} \leftarrow d_{J(r)-1} + 1$
    else MERGE$(\mathcal{S}, x_t)$ and INSERT$(\mathcal{S}, x_t, r)$
  Output $\mathbf{y}(S)$

INSERT$(\mathcal{S}, x_t, r)$
  Insert $(x_t, D_{J(r)} - r + 1)$ at position $J(r)$ in $\mathcal{S}$
  if $J(r) > 1$ **then** $d_{J(r)-1} \leftarrow r - D_{J(r)-1}$
  if $|\mathcal{S}| > K$ **then** remove item $K+1$ from $\mathcal{S}$

MERGE$(\mathcal{S}, x_t)$
  $d_{I(x_t)-1} \leftarrow d_{I(x_t)-1} + d_{I(x_t)}$
  Remove item $I(x_t)$ from $\mathcal{S}$

Theorem 1. *When Algorithm 1 terminates, its summary $\mathcal{S}$ is the K-concise summary of a permutation $\tilde{x}_{1:n}$ sampled uniformly at random from $\mathcal{P}(x_{1:n})$.*

Proof (sketch). Let $r_{1:n}$ be the string of random integers generated by Algorithm 1. These induce a sequence $(\tilde{x}^t_{1:t} : t \in \mathbb{N})$ of permutations of $x_{1:n}$. To show the correctness of Algorithm 1, we simply show that its operations mirror this sequence of permutations and incrementally generate the sequence of K-concise summaries of $\tilde{x}^t_{1:t}$. The different cases handled by Algorithm 1 then arise from different kinds of insertions. □

Corollary 1. *Algorithm 1 outputs a K-distinct sample of $x_{1:n}$.*

Note here the significance of Theorem 1: we can simulate, *in a fully online fashion* and in $O(K)$ memory and running time, the sampling of a permutation $\tilde{x}_{1:n} \sim \mathscr{U}(\mathcal{P}(x_{1:n}))$ and the subsequent selection of its first K distinct symbols. Interestingly enough, the use of random insertions, rather than swaps (as is done in Algorithm R), seems necessary to guarantee the correct distribution on $\mathbf{y}(\mathcal{S})$.

3.2 The Budget SAD Estimator

Naturally, we wish to use K-distinct reservoir sampling to approximate the set $\mathcal{Z}^*$ of frequently occurring symbols in $x_{1:n}$.

However, the d_i variables do not estimate the frequencies of their respective symbols. To estimate these frequencies, our Budget Sparse Adaptive Dirichlet estimator augments a K-concise summary with count information. The new summary is a vector of tuples

$$\mathcal{U} := \langle (y_i, d_i, c_i) \in \mathcal{X} \times \mathbb{N} \times \mathbb{N} : i \in [K] \rangle$$

where c_i represents the number of occurrences of y_i since its last entering the summary.

Algorithm 2 Budget SAD.

Initially: $\mathcal{U} = \langle\rangle$, $z = 0$
  for $t = 1 \dots n$ **do**
    Predict x_t according to Equation 2
    Observe x_t
    Update $\mathcal{U}$ as per Algorithm 1
    if x_t is new in $\mathcal{U}$ **then** $c_{I(x_t)} \leftarrow 1$
    else if $x_t \in \mathcal{Y}(\mathcal{U})$ **then** $c_{I(x_t)} \leftarrow c_{I(x_t)} + 1$
    else $z \leftarrow z + 1$ // discard x_t

New symbols are added to $\mathcal{U}$ with $c = 1$; then, whenever a symbol already present in $\mathcal{U}$ is observed, its count is incremented. Recall that the SAD prediction depends on $N_t(x)$, the number of occurrences of x in $x_{1:t}$, as well as the sub-alphabet $\mathcal{A}_t$ (Section 2.1). Our process yields $\mathcal{Y}_t := \mathcal{Y}(\mathcal{U}_t)$ as an approximation to $\mathcal{A}_t$, as well as an approximate count function $\hat{N}_t : \mathcal{X} \to \mathbb{R}$, with $\hat{N}_t(x) = 0$ for $x \notin \mathcal{Y}_t$. The algorithm also tracks z_t, the number of discarded symbols, and approximates the SAD prediction rule as

$$\begin{aligned}
\hat{N}_t(x) &:= (t - z_t)\frac{c_{t,I_t(x)}}{\sum_{x'} c_{t,I_t(x')}} \text{ for } x \in \mathcal{Y}_t \\
\hat{\gamma}_t &:= \gamma(t, \mathcal{Y}_t) + z_t \\
\tilde{N}_t(x) &:= \max\left\{\hat{N}_t(x), \hat{\gamma}_t w_t(x)\right\} \\
\rho^{\mathrm{B}}(x \,|\, x_{1:t}) &\propto \begin{cases} \tilde{N}_t(x) & \text{if } x \in \mathcal{Y}_t, \\ \hat{\gamma}_t w_t(x) & \text{otherwise.} \end{cases} \qquad (2)
\end{aligned}$$

where $c_{t,\cdot}$ denote the stored counts at time t, and $\sum_{x \in \mathcal{X}} \rho^{\mathrm{B}}(x \,\|\, \cdot) = 1$. Our Budget SAD is designed to overestimate the probability of frequent symbols (via the $(t - z_t)$ renormalization); the definition of $\tilde{N}_t(x)$ ensures that symbols in $\mathcal{Y}_t$ are never deemed less likely than unseen symbols. The process is summarized in Algorithm 2.

The z_t term keeps track of discarded counts; this is to correct for two types of counting errors. First, the counter $c_{t,I(x)}$ for a symbol $x \in \mathcal{Y}_t$ only equals $N_t(x)$ if 1) it was never discarded and 2) never removed from the reservoir; in general, $c_{t,I(x)}$ is an underestimate for $N_t(x)$. We also need to compensate for our lack of knowledge of $|\mathcal{A}_t|$ in computing $\hat{\gamma}_t$, the probability mass associated with unseen symbols. Effectively, z_t mitigates both issues by rectifying the stored counts so as to approximate the SAD denominator. In particular, when $\tilde{N}(x) = \hat{N}(x)$ for all $x \in \mathcal{Y}$, we have

$$\sum_{x \in \mathcal{Y}_t} \tilde{N}_t(x) + \hat{\gamma}_t = t + \gamma(t, \mathcal{Y}_t)$$

One may also increment z_t to keep tracks of counts that are discarded when a symbol leaves the reservoir; however, this does not affect our redundancy bound.

In addition to these counting errors, the Budget SAD suffers a redundancy cost from *forgetting* symbols. Clearly, this problem arises only when symbols are removed from the reservoir. As we now show, the Budget SAD achieves low redundancy with respect to the optimal memory-bounded model (Section 3) whenever the source's tail distribution is close to uniform.

4 Analysis

Let μ be a memoryless stationary source and let $x_{1:n} \sim \mu$. We compare our Budget SAD estimator ρ^{B} to the optimal memory-bounded model $\rho^*_\mu \in \mathcal{M}$ described in Section 3. In our analysis we consider the associated optimal set $\mathcal{Z}^*$ of size K^*, and derive a redundancy bound which depends on the probability mass $\mu(\mathcal{Z}^*)$ as well as the reservoir size $K > K^*$. For clarity, we take the SAD probability distribution $w_t(\cdot)$ to be uniform over the alphabet $\mathcal{X}$, noting that our result extends to the more general case.

We take a conservative approach and bound the expected redundancy $\mathbf{E}\left[\mathfrak{F}_n(\rho^{\mathrm{B}}, \rho^*_\mu)\right]$ according to the probability of a best-case scenario. In this best-case scenario, the symbols in $\mathcal{Z}^*$ are never ejected from the reservoir. We define a *good event* $G_n(x) := \left\{c_{n,I_n(x)} = N_n(x)\right\}$ which occurs when the stored count for x matches the total number of occurrences of x in $x_{1:n}$; this implies that $c_{t,I_t(x)} = N_t(x)$ for all $t \in [n]$. The instantaneous redundancy of our algorithm at time t then depends on the probability of $G_n(x_t)$. Critically, we set K so that $G_n(x)$ must occur with probability $1 - \delta$ for all $x \in \mathcal{Z}^*$. For $\mathcal{B} \subseteq \mathcal{X}$ and a memoryless stationary source μ, define

$$\upsilon(\mathcal{B}, \mu) := \min_{x \in \mathcal{B}} \mu(x).$$

We first provide a lemma stating the conditions under which $G_n(\mathcal{B}) := \bigcap_{x \in \mathcal{B}} G_n(x)$ occurs with high probability.

Lemma 3. *Let $\mathcal{B} \subseteq \mathcal{X}$, $\delta \in (0, 1)$, and $K \geq \upsilon(\mathcal{B}, \mu)^{-1} \log(|\mathcal{B}| n \delta^{-1})$. Then*

$$\Pr\{G_n(\mathcal{B})\} \geq 1 - \delta.$$

Proof (sketch). The event $G_n(x)$ occurs if x never leaves the K-concise summary after entering it. Because K-distinct reservoir sampling induces a permutation of $x_{1:n}$, we can view the corresponding K-distinct sample as generated from independent draws without replacement from μ. Computing the probability that a symbol $x \in \mathcal{B}$ is *not* drawn in K tries and taking union bounds yields the result. □

Our main theoretical result follows from this lemma:

Theorem 2. *Let $\delta = o(\log|\mathcal{X}| + n \log n)$ and $K \geq \upsilon(\mathcal{Z}^*, \mu)^{-1} \log(|\mathcal{Z}^*| n \delta^{-1})$. Let $\mathcal{Z}_{\mathrm{AUG}} \subseteq \mathcal{X}$ of size K maximizing $\mu(\cdot)$, and let $\kappa := K(1 - \mu(\mathcal{Z}_{\mathrm{AUG}}))^{-1}$. The expected redundancy of ρ^{B} with respect to ρ^*_μ is bounded as*

$$\begin{aligned}
\mathbf{E}\left[\mathfrak{F}_n(\rho^{\mathrm{B}}, \rho^*_\mu)\right] \;\leq\; & \mu(\mathcal{Z}^*)\tfrac{|\mathcal{Z}^*|-1}{2} \log n + \\
& (1 - \mu(\mathcal{Z}^*))\kappa \log^2 n + O(\kappa \log(\kappa n) \log\log n),
\end{aligned}$$

where the expectation is with respect to both $x_{1:n} \sim \mu$ and $r_{1:n}$, the random integers chosen by Algorithm 1.

Proof (sketch). We consider a partition of the event space at each time step:

1. $x_t \in \mathcal{Z}^*$ and $G_n(\mathcal{Z}^*)$,
2. $x_t \in \mathcal{Z}^*$ and $\neg G_n(\mathcal{Z}^*)$, or
3. $x_t \notin \mathcal{Z}^*$.

In the first case, the Budget SAD correctly estimates the probability of x_t, and exhibits similar regret guarantees to the SAD. The second case contributes negligible expected redundancy, since $\Pr\{\neg G_n(\mathcal{Z}^*)\}$ is small. The final case gives rise to most of the expected redundancy, which we bound by providing bounds on z_t (which lets us estimate the probability of symbols not in the K-concise summary), which itself depends on a bound on $D_{t,K+1}$, the expected gap between the first and $K+1^{th}$ symbols in our summary. Let $r_{1:n}$ be the random draws of r within K-distinct reservoir sampling. We show that

$$\mathop{\mathbf{E}}_{x_{1:n}, r_{1:n}} [D_{t,K+1}] \leq \kappa - 1,$$

which provides the last two terms of our bound. □

Remark 1. *The expected redundancy of ρ^{B} with respect to ρ^*_μ is also bounded as*

$$\mathbf{E}\left[\mathfrak{R}_n(\rho^{\mathrm{B}}, \rho^*_\mu)\right] \leq O(n \log n),$$

and in particular the bound of Theorem 2 is meaningful whenever $\kappa \log n = o(n)$.

The three terms in Theorem 2 correspond respectively to 1) the cost of learning μ over $\mathcal{Z}^*$, 2) estimating the quantity $1 - \mu(\mathcal{Z}^*)$, and 3) the redundancy for symbols which enter, but do not remain in our reservoir. In the presence of a favourable distribution (i.e., when $(1-\mu(\mathcal{Z}^*))/(1-\mu(\mathcal{Z}_{\mathrm{AUG}}))$ is small), our algorithm is asymptotically optimal, in the sense that its per-step redundancy goes to 0 as $n \to \infty$. This is the case, for example, when the source is in the class $\mathcal{M}$ of memory-bounded models (Equation 1) and $|\mathcal{X}| \gg n$.

As a final note, Lu and Lu [2011] recently provided a $\Omega(n)$ lower bound on the regret suffered by a bounded-memory, sequential prediction algorithm in the 0–1 loss setting. We point out that our analysis does not contradict their bound since, in the worst case, the second term may grow superlinearly (see Remark 1 and Section 6 below). In this case, one may (trivially) construct a Bayesian mixture between ρ^{B} and a uniform estimator to achieve $O(n \log |\mathcal{X}|)$ regret.

5 Empirical Evaluation

The Budget SAD is especially motivated by the desire to minimize memory usage in the presence of infrequent symbols. Here we present two experiments showing the practical value of our approach.

5.1 Synthetic Image Domain

We first consider a synthetic image domain to demonstrate the effectiveness of our algorithm in the presence of symbol noise. In this domain, the source first selects one of m images uniformly at random, then corrupts this image with probability p and otherwise leaves it unchanged. Here we consider 50×60, 8-bit images and a uniform noise corruption model

Figure 2: Sample from the synthetic image source with $m = 4$ and $p = 0.9$. Symbols are corrupted using uniform pixel noise.

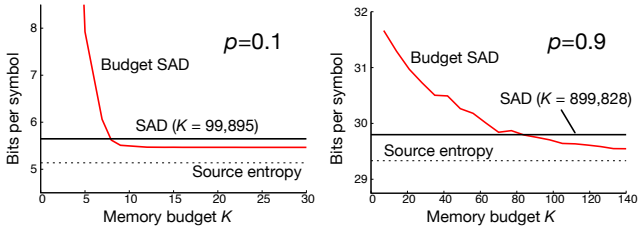

Figure 3: Redundancy for the Budget SAD as a function of maximum reservoir size. Images are encoded as 32-bit integers (i.e., $\log_2 |\mathcal{X}| = 32$).

(Figure 2; images from Wikipedia, 2015). Thus, while the source emits symbols from an alphabet $\mathcal{X}$ of size $256^{50\times60}$, only m of these symbols occur with non-negligible frequency.

We implemented the Budget SAD with a time-dependent reservoir size K to match the regret bound of Theorem 2: from a parameter $R \in \mathbb{R}^+$, we set $K_t = \lceil R\lceil \log(t+1) \rceil \rceil$. We trained both SAD and Budget SAD on one million symbols generated with $m = 4$ and $p \in \{0.1, 0.9\}$. We performed experiments with $R \in [1.0, 2.0]$ (when $p = 0.1$) and $R \in [0.5, 10.0]$ (when $p = 0.9$), with the results of each Budget SAD experiment averaged across 100 trials.

To study the behaviour of our algorithm, we map each value of R to its corresponding end-of-sequence reservoir size and report the algorithm's redundancy as a function of this value. Results for both values of p are shown in Figure 3. The Budget SAD achieves equal or lower redundancy using a small fraction of the SAD's memory budget (**0.01%** for both $p = 0.1$ and $p = 0.9$). Note that the Budget SAD even performs slightly better than the SAD in this instance, as it assigns a higher probability mass to unseen symbols. We conclude that the Budget SAD has the ability to significantly reduce memory usage.

5.2 Branch Pruning in Context Tree Weighting

Our analysis so far has focused on memoryless sources and their models. A common method of augmenting memoryless models is to embed them into a tree structure in order to guarantee low redundancy with respect to the class of k-Markov sources. One such method, Context Tree Weighting [Willems *et al.*, 1995], uses the generalized distributive law [Aji and McEliece, 2000] to efficiently perform Bayesian model aver-

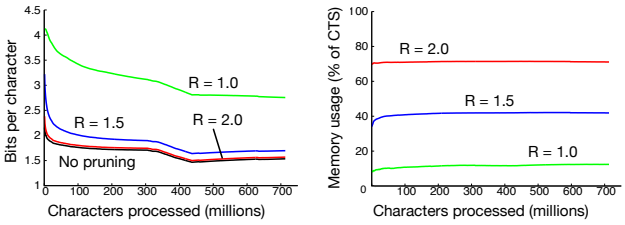

Figure 4: **Left.** Redundancy for the Budget SAD and SAD estimators. **Right.** Memory usage for the Budget SAD.

aging over tree structures.

At the core of CTW is a growing $|\mathcal{X}|$-ary context tree; each new symbol creates up to D new nodes, where D is a depth parameter. Each node corresponds to a *context* (a subsequence of $x_{1:n}$) and maintains a statistical estimator (here, the SAD). Here we consider a modified CTW in which each node uses a K-distinct reservoir to store its children: removing an item from the reservoir prunes a whole subtree.

To understand the need for branch pruning, consider the *Wikipedia Clean Text* data set [Mahoney, 2011], which we use as our sequential prediction task here. This data set is composed of 713M lowercase characters (a–z and whitespace; $|\mathcal{X}| = 27$). Within are 105M unique contexts of length at most $D = 9$. This number grows linearly over time, making a memory-bounded solution highly desirable. However, each context sees, on average, two or fewer symbols; as a result, using the Budget SAD at each node cannot significantly reduce memory usage. Branch pruning, on the other hand, is relatively effective.

For our experiment we used the more efficient CTW relative, Context Tree Switching [CTS; Veness *et al.*, 2012] and derived our parameters from its authors' open-source implementation. As with the synthetic experiment, we considered values in the range $R \in [1.0, 3.0]$. For each node u, its reservoir size was increased as $R \log(t_u + 1)$, with t_u the number of visits to this node. We set the depth parameter to $D = 9$ as larger values of D did not affect redundancy, and averaged each experiment across 10 trials.

As illustrated in Figure 4, our algorithm, through the R parameter, can smoothly trade off memory usage and redundancy (with respect to an unpruned CTS). For example, for a small redundancy ($\leq$ **5%**) we observed a **44.6%** average reduction in memory usage ($R \approx 1.76$, not illustrated). Furthermore, there was minimal variance between trials. While memory usage still remains linear in n, we believe this result is particularly significant given the Zipfian behaviour of natural language data, which in our analysis corresponds to a large κ term.

6 Discussion

In our setting, the optimal model ρ^*_μ assigns a uniform probability over the set $\mathcal{X} \setminus \mathcal{Z}^*$. Interestingly enough, the main source of redundancy in the Budget SAD is the $\kappa \log^2 n$ cost for approximating this uniform probability. In particular for some distributions (e.g. $\mu(\cdot) \propto 2^{-i}$ for $i \in \mathbb{N}$) our analysis suggests an unpleasant $O(n \log n)$ redundancy — possibly worse-than-random performance. This cost stems from our overestimation of symbols in $\mathcal{Z}^*$; it seems probable that an alternative approach, which instead overestimates $1 - \mu(\mathcal{Z}^*)$, exists.

On the other hand, with well-behaved data the Budget SAD provides significant memory savings. This is the case, for example, when the tail of μ is approximately uniform. Most count-based frequency estimators [Friedman and Singer, 1999; Teh, 2006; Veness and Hutter, 2012; Hutter, 2013] model this behaviour by setting aside a lump probability mass ε for all unobserved symbols; our z_t term serves a similar purpose.

Surprisingly, K-distinct reservoir sampling is unnecessary if the source μ is truly memoryless and stationary. In fact, in this case the first K distinct symbols in $x_{1:n}$ are just as good an approximation to $\mathcal{Z}^*$! The benefit of our approach is to (partially) inoculate the budget estimator against adversarial empirical sources; a similar role is played by the Dirichlet-multinomial prior in estimating counts.

Orlitsky *et al.* [2003] proposed two methods for achieving $o(n)$ per-symbol redundancy on *open* alphabets, where new symbols appear at a $O(n)$ rate. As an alternative to the z_t term, in this setting it may be sufficient to accurately estimate $|\mathcal{A}_t|$, the size of the alphabet at time t; work on distinct value queries suggests this is possible [Gibbons, 2001]. However, Orlitsky et al.'s method requires that we instead estimate the *prevalence*, or frequency distribution, of symbols. Whether this can be done with bounded memory is not known to us.

7 Conclusion

In this paper we presented a novel algorithm, K-distinct reservoir sampling, which incrementally maintains a sample of frequently occurring symbols within a data stream. We described how to combine K-distinct reservoir sampling with a simple count-based estimator and derived a corresponding redundancy bound. As our empirical results show, our algorithm can significantly reduce the memory requirements of both memoryless and tree-based estimators when dealing with long data sequences.

8 Acknowledgements

The author wish to thank Joel Veness for uncountably many discussions on the topic of count-based estimation, Laurent Orseau for providing thorough editorial comments, Alvin Chua and Michael Bowling for helpful discussion, and finally the anonymous reviewers for their superb feedback.

References

[Aji and McEliece, 2000] Srinivas M. Aji and Robert J. McEliece. The generalized distributive law. *IEEE Transactions on Information Theory*, 46(2):325–343, 2000.

[Bellemare *et al.*, 2013] Marc G. Bellemare, Joel Veness, and Michael Bowling. Bayesian learning of recursively factored environments. In *Proceedings of the Thirtieth International Conference on Machine Learning*, 2013.

[Cavallanti *et al.*, 2007] Giovanni Cavallanti, Nicoló Cesa-Bianchi, and Claudio Gentile. Tracking the best hyperplane with a simple budget perceptron. *Machine Learning*, 69(2-3):143–167, 2007.

[Charikar *et al.*, 2004] Moses Charikar, Kevin Chen, and Martin Farach-Colton. Finding frequent items in data streams. *Theoretical Computer Science*, 312(1):3–15, 2004.

[Cleary and Witten, 1984] John G. Cleary and Ian Witten. Data compression using adaptive coding and partial string matching. *IEEE Transactions on Communications*, 32(4):396–402, 1984.

[Cover and Thomas, 1991] Thomas M. Cover and Joy A. Thomas. *Elements of information theory*. John Wiley & Sons, 1991.

[Dekel *et al.*, 2008] Ofer Dekel, Shai Shalev-Shwartz, and Yoram Singer. The Forgetron: A kernel-based perceptron on a budget. *SIAM Journal on Computing*, 37(5):1342–1372, 2008.

[Demaine *et al.*, 2002] Erik D. Demaine, Alejandro López-Ortiz, and J. Ian Munro. Frequency estimation of internet packet streams with limited space. In *Proceedings of the Tenth Annual European Symposium on Algorithms*, pages 348–360, 2002.

[Farias *et al.*, 2010] Vivek F. Farias, Ciamac C. Moallemi, Benjamin Van Roy, and Tsachy Weissman. Universal reinforcement learning. *IEEE Transactions on Information Theory*, 56(5):2441–2454, 2010.

[Friedman and Singer, 1999] Nir Friedman and Yoram Singer. Efficient Bayesian Parameter Estimation in Large Discrete Domains. *Advances in Neural Information Processing Systems 11*, 11:417, 1999.

[Gale and Church, 1990] William Gale and Kenneth W. Church. Poor estimates of context are worse than none. In *Proceedings of the Workshop on Speech and Natural Language*, pages 283–287, 1990.

[Gibbons and Matias, 1998] Phillip B. Gibbons and Yossi Matias. New sampling-based summary statistics for improving approximate query answers. In *Proceedings of the ACM SIGMOD*, volume 27, pages 331–342, 1998.

[Gibbons, 2001] Phillip B. Gibbons. Distinct sampling for highly-accurate answers to distinct values queries and event reports. In *Proceedings of the International Conference on Very Large Databases*, 2001.

[Hutter, 2013] Marcus Hutter. Sparse adaptive dirichlet-multinomial-like processes. In *COLT*, 2013.

[Katz, 1987] Slava M. Katz. Estimation of probabilities from sparse data for the language model component of a speech recognizer. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 35(3):400–401, 1987.

[Knuth, 1981] Donald E. Knuth. *The Art of computer Programming, Volume 2: Seminumerical Algorithms, 2nd Edition*. Addison-Wesley, 1981.

[Krichevsky and Trofimov, 1981] R. Krichevsky and V. Trofimov. The performance of universal coding. *IEEE Transactions on Information Theory*, 27:199–207, 1981.

[Lu and Lu, 2011] Chi-Jen Lu and Wei-Fu Lu. Making online decisions with bounded memory. In *Algorithmic Learning Theory*, pages 249–261, 2011.

[Mahoney, 2011] Matt Mahoney. Large text compression benchmark, 2011. [Online; accessed 01-February-2015].

[Manku and Motwani, 2002] Gurmeet Singh Manku and Rajeev Motwani. Approximate frequency counts over data streams. In *Proceedings of the 28th International Conference on Very Large Databases*, pages 346–357, 2002.

[Metwally *et al.*, 2005] Ahmed Metwally, Divyakant Agrawal, and Amr El Abbadi. Efficient computation of frequent and top-k elements in data streams. In *Proceedings of the International Conference on Database Technology*, pages 398–412, 2005.

[Orabona *et al.*, 2008] Francesco Orabona, Joseph Keshet, and Barbara Caputo. The projectron: a bounded kernel-based perceptron. In *Proceedings of the 25th international conference on Machine learning*, pages 720–727. ACM, 2008.

[Orlitsky *et al.*, 2003] Alon Orlitsky, Narayana P. Santhanam, and Junan Zhang. Always good turing: Asymptotically optimal probability estimation. In *Proceedings of the 44th Anual Symposium on Foundations of Computer Science*, 2003.

[Teh, 2006] Yee Whye Teh. A hierarchical bayesian language model based on pitman-yor processes. In *Proceedings of the 21st International Conference on Computational Linguistics*, pages 985–992, 2006.

[Tjalkens *et al.*, 1993] Tj. J Tjalkens, Y.M. Shtarkov, and F.M.J. Willems. Context tree weighting: Multi-alphabet sources. In *14th Symposium on Information Theory in the Benelux*, pages 128–135, 1993.

[Tziortziotis *et al.*, 2014] Nikolaos Tziortziotis, Christos Dimitrakakis, and Konstantinos Blekas. Cover tree bayesian reinforcement learning. *The Journal of Machine Learning Research*, 15(1):2313–2335, 2014.

[Veness and Hutter, 2012] Joel Veness and Marcus Hutter. Sparse sequential Dirichlet coding. *ArXiv e-prints*, 2012.

[Veness *et al.*, 2012] Joel Veness, Kee Siong Ng, Marcus Hutter, and Michael H. Bowling. Context tree switching. In *Data Compression Conference (DCC)*, pages 327–336, 2012.

[Wikipedia, 2015] Wikipedia. Campbell's soup cans, 2015. [Online; accessed 01-February-2015].

[Willems *et al.*, 1995] Frans M. Willems, Yuri M. Shtarkov, and Tjalling J. Tjalkens. The context tree weighting method: Basic properties. *IEEE Transactions on Information Theory*, 41:653–664, 1995.