Compression and Predictive Distributions for Large Alphabet i.i.d and Markov models

Xiao Yang
Department of Statistics
Yale University
New Haven, CT, 06511
Email: xiao.yang@yale.edu

Andrew R. Barron
Department of Statistics
Yale University
New Haven, CT, 06511
Email: andrew.barron@yale.edu

***Abstract*—This paper considers coding and predicting sequences of random variables generated from a large alphabet. We start from the i.i.d model and propose a simple coding distribution formulated by a product of tilted Poisson distributions which achieves close to optimal performance. Then we extend to Markov models, and in particular, tree sources. A context tree based algorithm is designed according to the frequency of various contexts in the data. It is a greedy algorithm which seeks for the greatest savings in codelength when constructing the tree. Compression and prediction of individual counts associated with the contexts again uses a product of tilted Poisson distributions. Implementing this method on a Chinese novel, about 20.56% savings in codelength is achieved compared to the i.i.d model.**

I. Introduction

Large alphabet data compression does not have the luxury of diminishing per symbol redundancy. Luckily, symbols in a large alphabet are usually not equally probable in practice. For example, in Chinese characters, a subset of 964 characters covers 90% inputs in Chinese[1] though the vocabulary size is no smaller than 106,230 in total [2].

Coding and prediction of strings of random variables generated from an i.i.d model have been considered for the large alphabet setting with the restriction that the ordered probability or count list rapidly decreasing [3], or satisfies an envelope class property [4]. Although this i.i.d model is not the best for compression or prediction when there is dependence between successive characters, it serves as an analytical tool that more complicated models can be based on, and helps understand the behavior of coding and predictive distributions. In this paper, we propose a simple coding method particularly applicable for large alphabet compression and prediction problems, which also performs almost optimally.

Suppose a string of random variables $\underline{X} = (X_1, \ldots, X_N)$ is generated independently from a discrete alphabet $\mathcal{A}$ of size m. We allow the string length N to be variable. A special case is when N is given as a fixed number, or it can be random. In either case, $\underline{X}$ is a member of the set $\mathcal{X}^*$ of all finite length strings

$$\begin{aligned}\mathcal{X}^* &= \bigcup_{n=0}^{\infty} \mathcal{X}^n \\ &= \bigcup_{n=0}^{\infty} \{x^n = (x_1, \ldots, x_n) : x_i \in \mathcal{A}, i = 1, \ldots, n\}.\end{aligned}$$

Our goal is to code/predict the string $\underline{X}$.

Now suppose given N, each random variable X_i is generated independently according to a probability mass function in a parametric family $\mathcal{P}_\Theta = \{P_{\underline{\theta}}(x) : \underline{\theta} \in \Theta \subset R^m\}$ on $\mathcal{A}$. Thus

$$P_{\underline{\theta}}(X_1, \ldots, X_N | N = n) = \prod_{i=1}^{n} P_{\underline{\theta}}(X_i)$$

for $n = 1, 2, \ldots$ We are interested in the class of all distributions with $P_{\underline{\theta}}(j) = \theta_j$ parameterized by the simplex $\Theta = \{\underline{\theta} = (\theta_1, \ldots, \theta_m) : \theta_j \geq 0, \sum_{j=1}^m \theta_j = 1, j = 1, \ldots, m\}$.

Let $\underline{N} = (N_1, \ldots, N_m)$ denote the vector of counts for symbol $1, \ldots, m$. The observed sample size N is the sum of the counts $N = \sum_{j=1}^m N_j$. Both $P_{\underline{\theta}}(\underline{X})$ and $P_{\underline{\theta}}(\underline{X}|N = n)$ have factorizations based on the distribution of the counts

$$P_{\underline{\theta}}(\underline{X}|N = n) = P(\underline{X}|\underline{N})\, P_{\underline{\theta}}(\underline{N}|N = n),$$

and

$$P_{\underline{\theta}}(\underline{X}) \quad = \quad P(\underline{X}|\underline{N})\, P_{\underline{\theta}}(\underline{N}).$$

The first factor of the two equations is the uniform distribution on the set of strings with given counts, which does not depend on $\underline{\theta}$. The vector of counts $\underline{N}$ forms a sufficient statistic for $\underline{\theta}$. Modeling the distribution of the counts is essential for forming codes and predictions. In the particular case of all i.i.d. distributions parameterized by the simplex, the distribution $P_{\underline{\theta}}(\underline{N}|N = n)$ is the $multinomial(n, \underline{\theta})$ distribution.

In the above, there is a need for a distribution of the total count N. Of particular interest is the case that the total count is taken to be $Poisson$, because then the resulting distribution of individual counts makes them independent.

Poisson sampling is a standard technique to simplify analysis [5][6]. Here we give particular attention to the target family $\mathcal{P}_\Lambda^m = \{P_{\underline{\lambda}}(\underline{N}) : \lambda_j \geq 0,\ j = 1, \ldots, m\}$, in which $P_{\underline{\lambda}}(\underline{N})$ is the product of $Poisson(\lambda_j)$ distribution for $N_j, j = 1, \ldots, m$. It makes the total count $N \sim Poisson(\lambda_{sum})$ with $\lambda_{sum} = \sum_{j=1}^m \lambda_j$ and yields the $multinomial(n, \underline{\theta})$ distribution by conditioning on $N = n$, where $\theta_j = \lambda_j / \lambda_{sum}$. And the induced distribution on $\underline{X}$ is

$$P_{\underline{\lambda}}(\underline{X}) = P(\underline{X}|\underline{N}) P_{\underline{\lambda}}(\underline{N}).$$

978-1-4799-5186-4/14/$31.00 ©2014 IEEE

Ideally the true probability distribution $P_\lambda(\underline{X})$ could be used to code the sequence, if $\underline{\lambda}$ were known. The *regret* induced by using Q instead of $P_{\underline{\lambda}}$ is

$$R(Q, P_{\underline{\lambda}}, \underline{X}) = \log \frac{1}{Q(\underline{X})} - \log \frac{1}{P_{\underline{\lambda}}(\underline{X})},$$

where log is logarithm base 2 (We don't worry about integer constraint).

Here we can construct Q by choosing a probability distribution for the counts and then use the uniform distribution for the distribution of strings given the counts, written as P_{unif}. That is

$$Q(\underline{X}) = P_{unif}(\underline{X}|\underline{N})Q(\underline{N}). \tag{1}$$

Then the regret becomes

$$\begin{aligned} R(Q, P_{\underline{\lambda}}, \underline{X}) &= \log \frac{P_{\underline{\lambda}}(\underline{N})}{Q(\underline{N})} \\ &= R(Q, P_{\underline{\lambda}}, \underline{N}). \end{aligned}$$

However, in reality $\underline{\lambda}$ is usually unknown. Given the family $\mathcal{P}_\Lambda^m$, we could instead use the best candidate with hindsight $P_{\hat{\underline{\lambda}}}(\underline{X}) = \max_{\underline{\lambda}\in\Lambda}(P_{\underline{\lambda}}(\underline{X}))$ (corresponding to $\min_{\underline{\lambda}\in\Lambda} \log(1/P_{\underline{\lambda}}(\underline{X}))$), and compare it to $Q(\underline{X})$. The maximization is equivalent to maximizing $\underline{\lambda}$ for the count probability, as the uniform distribution dose not depend on $\underline{\lambda}$, i.e.

$$\begin{aligned} \max_{\underline{\lambda}\in\Lambda}(P_{\underline{\lambda}}(\underline{X})) &= P_{unif}(\underline{X}|\underline{N}) \max_{\underline{\lambda}\in\Lambda} P_{\underline{\lambda}}(\underline{N}) \\ &= P_{unif}(\underline{X}|\underline{N})\, P_{\hat{\underline{\lambda}}}(\underline{N}). \end{aligned}$$

Then the problem becomes: given the family $\mathcal{P}_\Lambda^m$, how to choose Q to minimize the maximized regret

$$\min_Q \max_{\underline{X}} R(Q, P_{\hat{\underline{\lambda}}}, \underline{X}) = \min_Q \max_{\underline{N}} \log \frac{P_{\hat{\underline{\lambda}}}(\underline{N})}{Q(\underline{N})}.$$

For the regret, the maximum can be restricted to a set of counts instead of the whole space. A traditional choice being $S_{m,n} = \{(N_1, \ldots, N_m) : \sum_{j=1}^m N_j = n, N_j \geq 0,\, j = 1, \ldots, m\}$ associated with a given sample size n, in which case the minimax regret is

$$\min_Q \max_{\underline{N}\in S_{m,n}} \log \frac{P_{\hat{\underline{\lambda}}}(\underline{N})}{Q(\underline{N})}.$$

As is familiar in universal coding [7][8], the normalized maximum likelihood (NML) distribution

$$Q_{nml}(\underline{N}) = \frac{P_{\hat{\underline{\lambda}}}(\underline{N})}{C(S_{m,n})}$$

is the unique pointwise minimax strategy when $C(S_{m,n}) = \sum_{\underline{N}\in S_{m,n}} P_{\hat{\underline{\lambda}}}(\underline{N})$ is finite, and $\log C(S_{m,n})$ is the minimax value. When m is large, the NML distribution can be unwieldy to compute for compression or prediction. Instead we will introduce a slightly suboptimal coding distribution that makes the counts independent and show that it is nearly optimal for every $S_{m,n'}$ with n' not too different from a target n. Indeed, we advocate that our simple coding distribution is preferable to use computationally when m is large even if the sample size n were known in advance.

To produce our desired coding distribution we make use of two basic principles. One is that the multinomial family of distributions on counts matches the conditional distribution of $N_1, \ldots, N_m$ given the sum N when unconditionally the counts are independent Poisson. Another is the information theory principle [9][10][11] that the conditional distribution given a sum (or average) of a large number of independent random variables is approximately a product of distributions, each of which is the one closest in relative entropy to the unconditional distribution subject to an expectation constraint. This minimum relative entropy distribution is an exponential tilting of the unconditional distribution.

In the Poisson family with distribution $\lambda_j^{N_j} e^{-\lambda_j}/N_j!$, exponential tilting (multiplying by the factor e^{-aN_j}) preserves the Poisson family (with the parameter scaled to $\lambda_j e^{-a}$). Those distributions continue to correspond to the multinomial distribution (with parameters $\theta_j = \lambda_j/\lambda_{sum}$) when conditioning on the sum of counts N. A particular choice of $a = \ln(\lambda_{sum}/N)$ provides the product of Poisson distributions closest to the multinomial in regret. Here for universal coding, we find the tilting of individual maximized likelihood that makes the product of such closest to the Shtarkov's NML distribution. This greatly simplifies the task of approximate optimal universal compression and the analysis of its regret.

Indeed, applying the maximum likelihood step to a Poisson count k produces a maximized likelihood value of $M(k) = k^k e^{-k}/k!$. We call this maximized likelihood the *Stirling ratio*, as it is the quantity that Stirling's approximation shows near $(2\pi k)^{-1/2}$ for k not too small. We find that this $M(k)$ plays a distinguished role in universal large alphabet compression, even for sequences with small counts k. This measure M has a product extension to counts $N_1, \ldots, N_m$,

$$M(\underline{N}) = M(N_1) \cdots M(N_m).$$

Although M has an infinite sum by itself, it is normalizable when tilted for every positive a. The *tilted Stirling ratio distribution* is

$$P_a(N_j) = \frac{N_j^{N_j} e^{-N_j}}{N_j!} \frac{e^{-aN_j}}{C_a}, \tag{2}$$

with the normalizer $C_a = \sum_{k=0}^\infty k^k e^{-(1+a)k}/k!$.

The coding distribution we propose and analyze is simply the product of those tilted one-dimensional maximized Poisson likelihood distributions for a value of a we will specify later

$$Q_a(\underline{N}) = P_a^m(\underline{N}) = P_a(N_1) \cdots P_a(N_m).$$

If it is known that the total count is n, then the regret is a simple function of n and the normalizer C_a. The choice of the tilting parameter a^* given by the moment condition $\mathbf{E}_{Q_a} \sum_{j=1}^m N_j = n$ minimizes the regret over all positive a. Moreover, value of a^* depends only on the ratio between the size of the alphabet and the total count m/n. Fig. 1 displays a^* as a function of m/n solved numerically. Given an alphabet with m symbols and a string generated from it of length n, one can look at the plot and find the a^* desired according to the m/n given, and then use the a^* to code the data.

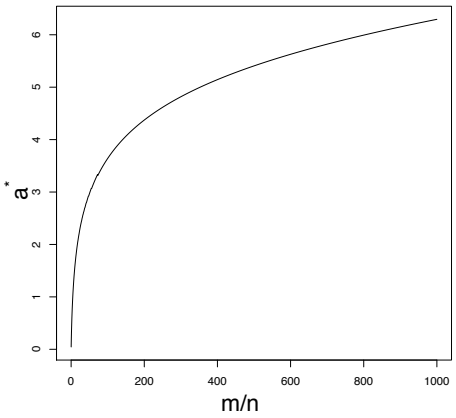

Fig. 1. Relationship between a^* and $\frac{m}{n}$.

As compared to i.i.d class, Markov sources are much richer and more realistic. Suppose given N, each random variable X_i is generated according to a probability mass function depending on its *context* (string of symbols preceding it). Following Williems' notations in [12], a tree source can be determined by a context set $\mathcal{S}$. Elements of $\mathcal{S}$ are strings of symbols from $\mathcal{A}$ or concatenation of "others" and prefixes of the contexts. "others" represents complements of the contexts in $\mathcal{S}$ with a common prefix. *Prefix* of a context is the part except the last symbol. For example, the prefix for $s = ab$ is a, and for $s = a$ equals nothing. The collection of distributions is $\mathcal{P}_{\Theta_\mathcal{S}} = \{P_{\underline{\theta}_s}(x) : \underline{\theta}_s \in \Theta_\mathcal{S} \subset R^m, s \in \mathcal{S}\}$, where $\Theta_\mathcal{S}$ is the parameter set defined later. For simplicity, we require the order of the model no larger than $T \in \{0, 1, 2, \ldots\}$, so $\mathcal{S} \in \mathcal{C}_T$, where $\mathcal{C}_T$ is the class of tree sources with order T or less.

For each context $s \in \mathcal{S}$ with a given $\mathcal{S}$, let θ_{sx} denote the probability of symbol $x \in \mathcal{A}$ showing up after s, for all $x \in \mathcal{A}$. Then $\underline{\theta}_{s\cdot} = (\theta_{s1}, \ldots, \theta_{sm})$ lies in the set

$$\Theta_\mathcal{S} = \{\underline{\theta}_{s\cdot} = (\theta_{s1}, \ldots, \theta_{sm}) : x \in \mathcal{A}, \theta_{sx} \geq 0, \sum_{x\in\mathcal{A}} \theta_{sx} = 1\}.$$

Again, we could take advantage of factorizations based on the distribution of the counts $\mathbf{N}_\mathcal{S} = (\underline{N}_{s\cdot})_{s\in\mathcal{S}}$, where $\underline{N}_{s\cdot} = (N_{s1}, \ldots, N_{sm})$ is the count for all symbols given context $s \in \mathcal{S}$

$$P_{\underline{\theta}}(\underline{X}|N=n) = P(\underline{X}|\mathbf{N}_\mathcal{S})\, P_{\underline{\theta}}(\mathbf{N}_\mathcal{S}|N=n),$$

and

$$P_{\underline{\theta}}(\underline{X}) = P(\underline{X}|\mathbf{N}_\mathcal{S})\, P_{\underline{\theta}}(\mathbf{N}_\mathcal{S}).$$

Picking the distribution for the total count to be $Poisson$ again leads to the target family $\mathcal{P}_\Lambda^{|\mathcal{S}|m} = \{P_{\underline{\lambda}}(\mathbf{N}_\mathcal{S}) : \lambda_{sj} \geq 0,\ j = 1, \ldots, m, s \in \mathcal{S}\}$, in which $P_{\underline{\lambda}}(\mathbf{N}_\mathcal{S})$ is the product of $Poisson(\lambda_{sj})$ distribution for $N_{sj}, j = 1, \ldots, m$ and $s \in \mathcal{S}$. And the induced distribution on $\underline{X}$ is

$$P_{\underline{\lambda}}(\underline{X}) = P(\underline{X}|\mathbf{N}_\mathcal{S}) P_{\underline{\lambda}}(\mathbf{N}_\mathcal{S}).$$

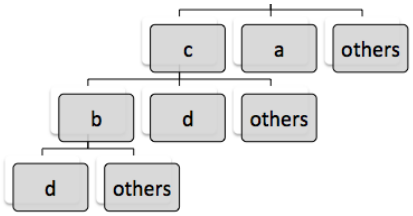

Fig. 2. An example context tree.

There are two sources of cost involved in using a tree model. One is the *coding cost* $C(\underline{X})$ for variables given the context. Another is the *description cost* $D(\mathcal{S})$ for describing the tree. Overall, we want to find Q which uses shorter codelength for sequences generated from an unknown tree source $\mathcal{S} \in \mathcal{C}_T$. That is, to minimize

$$\max_{\mathcal{S}\in\mathcal{C}_T} \max_{\underline{X}\in S} \mathcal{C}(\mathcal{S}, \underline{X}) = \max_{\mathcal{S}\in\mathcal{C}_T} \max_{\underline{X}\in S} \left(C(\underline{X}) + D(\mathcal{S})\right).$$

for a given set of sequences S. One choice of S could be $S_{m,n,\mathcal{S}} = \{\mathbf{N}_\mathcal{S} : \sum_{s\in\mathcal{S}} \sum_{j=1}^m N_{sj} = n, N_{sj} \geq 0,\ j = 1, \ldots, m, s \in \mathcal{S}\}$ associated with a given sample size n.

We use the same coding distribution as given in equation (2) for count variables conditional on each given context s. The coding distribution for the counts given s is simply the product

$$Q_{a_s}(\underline{N}_{s\cdot}) = P_{a_s}^m(\underline{N}_{s\cdot}) = P_{a_s}(N_{s1}) \cdots P_{a_s}(N_{sm}), \quad (3)$$

with a properly chosen a_s for each context $s \in \mathcal{S}$.

The tilting parameter a_s can be chosen according to the ratio of alphabet size m and the context count $N_s = \sum_{j=1}^m N_{sj}$ for all $s \in \mathcal{S}$ [3]. Using the tilted distribution $P_{\lambda_{a_s}}$ as a coding distribution, the regret is simply a sum of the individual regrets.

It is clear that when the context set $\mathcal{S}$ is provided, the coding distribution and the regret it induces is easy to obtain. Yet the questions is how to construct the context set to begin with. We adopt a method similar to Rissanen's approach in [13]. Different from the i.i.d model in which regret can be solely relied on to evaluate the performance, for Markov models different targets associated with different models need to be taken account of. Hence, we focus on the total codelength $\mathcal{C}(\mathcal{S}, \underline{X})$ and use it as a standard to evaluate the performance of different models and coding distributions. We use a greedy algorithm to build the context tree with details discussed in Section III-C. An illustrative example tree with a simple alphabet $\mathcal{A} = \{a, b, c, d\}$ is given in Fig.2.

II. I.I.D CLASS

Let S be any set of counts, then the maximized regret of using Q as a coding strategy given a class $\mathcal{P}$ of distributions when the vector of counts is restricted to S is

$$R(Q, \mathcal{P}, S) = \max_{\underline{N}\in S} \log \frac{\max_{P\in\mathcal{P}} P(\underline{N})}{Q(\underline{N})}.$$

Theorem 1: Let P_a be the distribution specified in equation (2) (Poisson maximized likelihood, tilted and normalized). The regret of using a product of tilted distributions $Q_a = \otimes_{j=1}^m P_a$ for a given vector of counts $\underline{N} = (N_1, \ldots, N_m)$ is

$$R\left(Q_a, \mathcal{P}_\Lambda^m, \underline{N}\right) = aN \log e + m \log C_a.$$

Let $S_{m,n}$ be the set of count vectors with total count n be defined as before, then

$$R\left(Q_a, \mathcal{P}_\Lambda^m, S_{m,n}\right) = an \log e + m \log C_a. \tag{4}$$

Let a^* be the choice of a satisfying the following moment condition

$$\mathbf{E}_{P_a} \sum_{j=1}^m N_j = m\, \mathbf{E}_{P_a} N_1 = n. \tag{5}$$

Then a^* is the minimizer of the regret in expression (4). Write $R_{m,n} = \min_a R(Q_a, \mathcal{P}_\Lambda^m, S_{m,n})$.

When $m = o(n)$, the $R_{m,n}$ is near $\frac{m}{2} \log \frac{ne}{m}$ in the following sense.

$$\begin{aligned} -d_1 \frac{m}{2} \log e &\leq R_{m,n} - \frac{m}{2} \log \frac{ne}{m} \\ &\leq m \log(1 + \sqrt{\frac{m}{n}}), \end{aligned} \tag{6}$$

where $d_1 = O\left((\frac{m}{n})^{1/3}\right)$.

When $n = o(m)$, the $R_{m,n}$ is near $n \log \frac{m}{ne}$ in the following sense.

$$\begin{aligned} m \log\left(1 + (1 - d_2)\frac{n}{m}\right) &\leq R_{m,n} - n \log \frac{m}{ne} \\ &\leq m \log\left(1 + \frac{n}{m} + d_3\right) \end{aligned} \tag{7}$$

where $d_2 = O(\frac{n}{m})$, and $d_3 = \frac{1}{2\sqrt{\pi}} \frac{n^2 e^2}{m(m-ne)}$.

When $n = bm$, the $R_{m,n} = cm$, where the constant $c = a^* b \log e + \log C_{a^*}$, and a^* is such that $\mathbf{E}_{P_a} N_1 = b$.

Proof: The expression of the regret is from the definition. The fact that a^* is the minimizer can be seen by taking partial derivative with respect to a of expression (4). Details of proof can be found in [3]. ∎

Remark 1: The regret depends only on the number of parameters m, the total counts n and the tilting parameter a. The optimal tilting parameter is given by a simple moment condition in equation (5).

Remark 2: The regret $R_{m,n}$ is close to the minimax level in all three cases listed in Theorem 1. The main terms in the $m = o(n)$ and $n = o(m)$ cases are the same as the minimax regret given in [14] except the multiplier for $\log(ne/m)$ here is $m/2$ instead of $(m-1)/2$ for the small m scenario. For the $n = bm$ case, the $R_{m,n}$ is close to the minimax regret in [14] numerically.

Corollary 1: Let $\mathcal{P}_\Theta^m$ be a family of multinomial distributions with total count n. Then the maximized regret $R(Q_a, \mathcal{P}_\Theta^m, S_{m,n})$ has an upper bound within $\frac{1}{2} \log 2\pi n$ above the upper bound in Theorem 1.

Proof: This can be easily seen by the following equation

$$\begin{aligned} & \log \frac{P_{\hat{\lambda}}(X_1, \ldots, X_N | N = n)}{P_{unif}(X_1, \ldots, X_n | N_1, \ldots, N_m) Q_a(N_1, \ldots, N_m)} \\ = \; & \log \frac{\prod_{j=1}^m M(N_j)}{P_{\hat{\lambda}_{sum}}(N = n) Q_a(N_1, \ldots, N_m)} \\ \simeq \; & \frac{1}{2} \log 2\pi n + \log \frac{\prod_{j=1}^m M(N_j)}{Q_a(N_1, \ldots, N_m)}. \end{aligned} \tag{8}$$

Here $\hat{\lambda}_{sum} = n$, hence the term $\frac{1}{2} \log 2\pi n$ is Stirling's approximation of $\log 1/P_{\hat{\lambda}_{sum}}(N = n)$. ∎

Remark 3: The $\frac{1}{2} \log 2\pi n$ arises because here Q includes description of the total N while the more restrictive target regards it as given.

III. Tree source

A. Coding cost

For each context $s \in \mathcal{S}$, we use a product of *tilted Stirling ratio distributions* with parameter a_s as in equation (3) to code the vector of counts $N_{s\cdot}$. The coding distribution is the product of all the $Q_{a_s}(\underline{N}_{s\cdot})$, i.e.

$$Q_a^{\mathcal{S}}(\mathbf{N}_{\mathcal{S}}) = \prod_{s \in \mathcal{S}} Q_{a_s}(N_{s\cdot}).$$

Corollary 2: Using independent tilted Stirling ratio distribution $Q_a^{\mathcal{S}}$ to code the counts in $S_{m,n,\mathcal{S}}$, the regret equals

$$R(\mathcal{P}_\Lambda^{|\mathcal{S}|m}, Q_a^{\mathcal{S}}, S_{m,n,\mathcal{S}}) = \sum_{s \in \mathcal{S}} \left(a_s N_s \log e + m \log C_{a_s}\right).$$

This can be easily seen by applying the definition.

B. Description cost

To describe a given context set $\mathcal{S}$, we use the following rule

$$D(S) = 1 + N_{branches}\left(1 + \log |\mathcal{A}|\right),$$

where $N_{branches}$ is the number of "labeled" branches in the tree. Here "labeled" means a specified symbol in the alphabet. For instance, $N_{branches} = 5$ in the example tree.

The first bit is used to describe if the model is nondegenerate (i.i.d or Markov). For each branch other than "others", we could use $\log m$ bits to convey which symbol it is, and then another 1 bit to say if it is nondegenerate. Our example tree uses $1 + 5(1 + \log 4) = 16$ bits.

C. Using codelength to construct the tree

Here we use the example in Fig.2 to describe how we construct the tree. Starting from the root which conditions on nothing (the i.i.d model), we choose the first symbol (c) that produces the most savings (if any) in codelength to condition on. Next, we consider two possible ways to expand the tree: one is another symbol (a) that makes the most savings; the other is to further condition on one more symbol (b) preceding the first found one (c). After calculating and comparing savings produced by these two candidates, we then decide which one to pick again by choosing the one with the greatest savings. Continue these procedures until conditioning provides no more

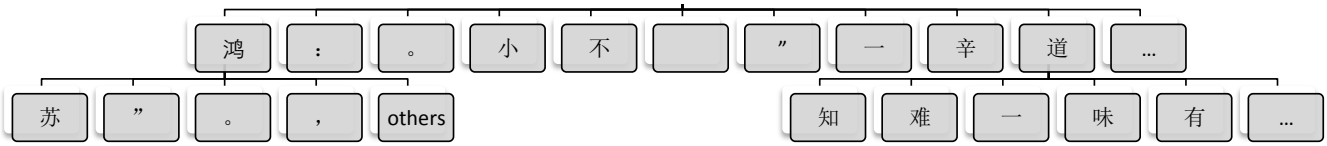

Fig. 3. Context tree for *Fortress Beseiged*.

savings or the maximum number of symbols to condition on (T) is reached, we have the context tree built. At each layer, the label "others" represents all other symbols that are not picked up in that layer, for example, "others" includes b and d in the first layer in the example.

When expanding one branch from the tree, we are conditioning on one more symbol. This leads to a different target model. We calculate the coding cost by adding up the minimized codelength for the target model and the minimax regret by using Szpankowski's approximation [14], because the minimized regret of using the tilted Stirling ratio distribution is very close to the minimax level. Then the overall codelength is used for comparison in tree construction.

IV. A REAL EXAMPLE

We apply the proposed method to a contemporary Chinese novel named 《围城》 and translated as Fortress Besieged [15]. The book contains 216,601 characters in total, and it is encoded in GB18030, the largest official Chinese character set which contains 70,244 characters [16].

We start from the i.i.d model which uses 1,954,777 bits. The first single character to condition on is 鸿, which produces 12,631 bits of savings. It is actually the first character of the protagonist's first name in the novel. It is indeed not surprising that seeing the first character highly indicates the next thing to see is the second for a two-character name.

We restrict the order of the Markov model to be no larger than 5, but in fact no context exceeding two characters is chosen. There are 342 branches in the tree, among which 95 are in the first layer, and 5 of them extends to the second layer. In fact, second layer branches are picked up only after most first layer branches have already been chosen. A small part of the tree is displayed in Fig.3.

The dots on the right stand for the rest of the model that cannot be shown. And the blank cell in the middle of the first layer is actually the space symbol. The total savings amounts to 401,922 bits (about 20.56%) as compared to the i.i.d model. Please note that existing models for tree sources are mostly designed for small alphabet compression, hence direct comparison with which would not be quite fair.

V. CONCLUSION

We consider a compression and prediction problem under both large alphabet i.i.d model and bounded tree models, and design a method to construct the context tree. Combining this method with tilted Stirling ratio distribution as coding distributions for symbols with given contexts, we have a convenient and efficient way for compression and prediction. Implementing the proposed method on a Chinese novel, considerable savings in codelength is produced compared to the i.i.d model.

ACKNOWLEDGMENT

The authors would like to thank Prof. Jun'ichi Takeuchi, Prof. Wojciech Szpankowski and Prof. Narayana Santhanam. Furthermore, we thank for the invitation to present part of the work in The Sixth Workshop on Information Theoretic Methods in Science and Engineering (Tokyo, Aug, 2013) and Center for Science of Information at Purdue University (West Lafayette, Oct, 2013). Some ideas of the subsequent work are based on feedback from these presentations.

REFERENCES

[1] I. of Applied Linguistice Ministry of Education. (2008, Nov) 2007 report on language use in china. [Online]. Available: http://www.china-language.gov.cn/14/2008_11_17/1_14_3890_0_1226884790921.html

[2] Wikipedia, "Chinese characters," December 2013. [Online]. Available: http://zh.wikipedia.org/wiki/

[3] X. Yang and A. Barron, "Large alphabet compression and predictive distributions through poissonization and tilting," *arXiv:1401.3760v1 [stat.ME]*, 2013.

[4] A. G. S. Boucheron and E. Gassiat, "Coding on countably infinite alphabets," *IEEE Transactions on Information Theory*, vol. 55, no. 1, Jan 2009.

[5] W. Feller, *An Introduction to Probability Theory and Its Applications*. Wiley, 1950, vol. 1.

[6] H. D. J. Archarya, "Tight bounds for universal compression of large alphabets," *IEEE International Symposium on Information Theory*, 2013.

[7] Y. M. Shtarkov, "Universal sequential coding of single messages," *Problems of Information Transmission*, vol. 23, no. 3, pp. 175–186, 1987.

[8] Q. Xie and A. R. Barron, "Minimax redundancy for the class of memoryless sources," *IEEE Transactions on Information Theory*, vol. 43, pp. 646–657, May 1997.

[9] I. Csiszar, "I-divergence geometry of probability distributions and minimization problems," *The Annals of Probability*, vol. 3, no. 1, pp. 146–158, Feb 1975.

[10] ——, "Sanov property, generalized I-projection and a conditional limit theorem," *The Annals of Probability*, vol. 12, no. 3, pp. 768–793, Jan 1984.

[11] J. V. Campenhout and T. Cover, "Maximum entropy and conditional probability," *IEEE Transactions on Information Theory*, vol. 27, no. 4, July 1981.

[12] F. M. J. Willems, Y. Shtarkov, and T. Tjalkens, "The context-tree weighting method: basic properties," *IEEE Transactions on Information Theory*, vol. 41, no. 3, pp. 653–664, 1995.

[13] J. Rissanen, "A Universal Data Compression Syetem," *IEEE Transactions on Information Theory*, vol. 29, no. 5, pp. 656–664, 1983.

[14] W. Szpankowski and M. J. Weinberger, "Minimax redundancy for large alphabets," *Information Theory Proceedings*, June 2010.

[15] Wikipedia, "Fortress beseiged," October 2013. [Online]. Available: http://en.wikipedia.org/wiki/Fortress_Besieged

[16] ——, "Gb18030 stardard," January 2014. [Online]. Available: http://zh.wikipedia.org/wiki/GB_18030