# Final Report
## Data Compression using Probabilistic Inference

Pranav Maneriker, Dhruv Singal, Ankesh Pandey

February 20, 2017

**Abstract**

The goal of the project was to do a brief survey of the state of the art probabilistic lossless data compression algorithms and trace the history of some landmark results which led to huge improvements in these algorithms.

# 1 Introduction

Data compression is ubiquitous in modern world. Ever since Shannon laid the mathematical groundwork for information theory [12], data compression algorithms have had wide ranging implications throughout the world. Probabilistic methods allow for better data compression algorithms, achieving state of the art (in terms of efficiency) in lossless compression. Section 2 gives the mathematical framework for the data compression methods which use probabilistic inference. Section 3 surveys the classical method of Prediction by Partial Matching (or PPM) and its derivatives. These were the very first models to use Bayesian inference in the setting of lossless data compression. Section 4 deals with Sequence Memoizer, a nonparametric model using Bayesian inference to compress natural language data. Section 5 discusses PAQ, a suite of data archiving algorithms involving ensembles of PPM-like models, which are state of the art in lossless compression.

# 2 Lossless Compression

[8] introduced *arithmetic coding*, which is used by a lot of lossless compression algorithms to give near optimal results. Arithmetic coding uses a probability distribution over the symbol set to give a numeral encoding of a sequence of symbols. Essentially, lesser number of bits will be required to encode a sequence containing symbols which are predicted to occur with higher probability. The method involves successively narrowing done the encoding interval (starting from [0,1]) using the probability distribution. The decoding is just the reverse process. A variant of arithmetic coding, called *adaptive arithmetic coding*, involves updating the probability distribution in a deterministic fashion after every successive observation in the sequence.

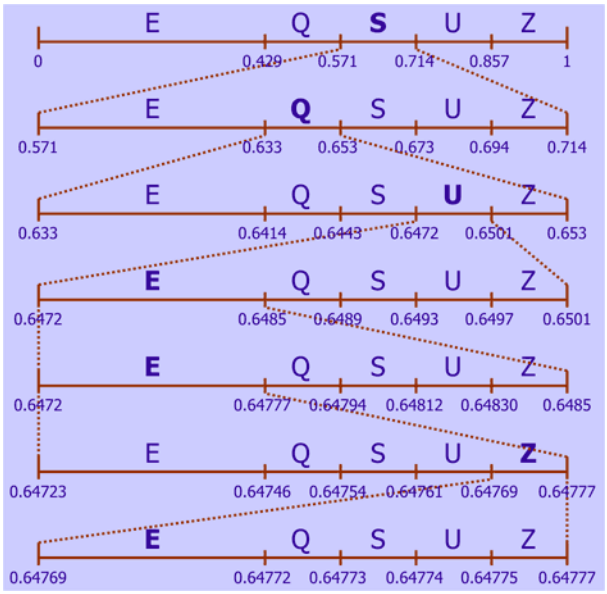


Figure 1: Arithmetic coding [2]

# 3 Prediction by Partial Matching (PPM)

This algorithm, originally proposed by Cleary and Witten[5], sequentially compresses one symbol at a time while learning context dependent probability for improving compression.

- A Markov model with context depth dependent on computational restrictions (higher order requires more storage)
- For order $o$, the current prediction is a function of the previous $o$ predictions
- The model uses the longest matching context cleverly (with a fixed set of hyper parameters) to generate probabilities

## 3.1 Advancements in PPM(PPM-DP)

Primarily, different PPM versions use modified update rules and models to define the probability of the next symbol and how the escape mechanism is handled.
Of the developments in PPM, the PPM-DP[13] algorithm is one of the most recent ones. The major ideas in this algorithm are:

### 3.1.1 Blending

Usage of a generalised blending mechanism with distinct hyper parameters based on depth and fanout to combine contexts. In particular, we use the following construction:

$$P_s(x) := \frac{\mathcal{M}_s(x) - \beta}{|\mathcal{M}_s| + \alpha}.1[x \in \mathcal{M}_s] + \frac{U_s\beta + \alpha}{|\mathcal{M}_s| + \alpha}.\begin{cases} \frac{1}{|\mathcal{X}|} & \text{if } s = \epsilon \\ P_{suf(s)}(x) & otherwise, \end{cases}$$

where $1[x \in \mathcal{M}_s]$ equals 1 if symbol x was observed at least once in context $s$ (and 0 otherwise). $\alpha$ and $\beta$ are the hyperparameters and$\mathcal{M}_s$ is the multiset of symbol counts for context $s$, and $U_s$ denotes the number of unique symbols in $\mathcal{M}_s$:

$$U_s = \sum_{x \in \mathcal{X}} 1[x \in \mathcal{M}_s]$$

### 3.1.2 Dynamic Updates

Dynamic updates to hyperparameters using gradient information:

$$\alpha_{fd} \leftarrow \alpha_{fd} + \delta.\frac{\partial logP_s}{\partial \alpha_{fd}} \qquad\qquad \beta_{fd} \leftarrow \beta_{fd} + \delta.\frac{\partial logP_s}{\partial \beta_{fd}}$$

Where $d$ is the depth and $f$ is the fanout.

# 4 Sequence Memoizer

The Sequence Memoizer by Wood et al.[15] is a deep (unbounded) smoothing Markov model. It takes the non-parametric Bayesian approach to compress sequences of data. The main features are:

- It encodes the fact that natural language sequence data exhibits power-law properties using the Pitman-Yor process as a prior
- By extending the hierarchical PYP model it includes the set of distributions in all contexts of finite length
- Due to the coagulation property of the PYP inference in full sequence model with infinite number of parameters is equivalent to inference in compact context tree with linear number of parameters

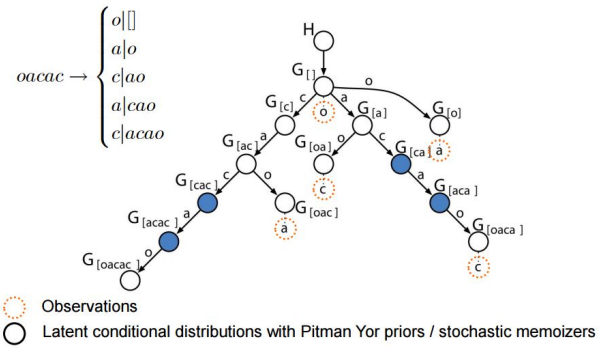


Figure 2: Graphical model Trie[14]

Due to Bayesian inference methods sequence memoizer performs consistently well irrespective of the context lengths. As a consequence of the marginalization steps due to coagulation property of hierarchical PYP, inference in the full sequence memoizer model with an infinite number of parameters is equivalent to inference in the compact context tree with a linear number of parameters.

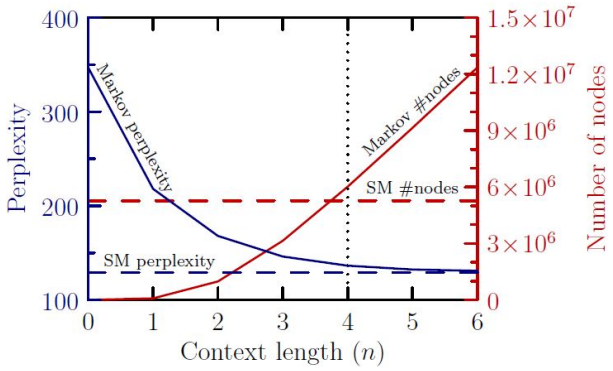


Figure 3: Performance of SM vs. $n^{th}$ order Markov models[15]

### 4.1 Improvements in Sequence Memoizer

Jan Gasthaus et al.[7] proposed improvements to the existing SM model and inference algorithm, including an enlarged range of hyperparameters, and a memory efficient representation.

#### 4.1.1 Non-Zero Concentration Parameters

We now have a more flexible setting of the hyperparameters with potentially non-zero concentration parameters(unlike SM by Wood[15]). The hyperparameter setting is: let $\alpha_\epsilon = \alpha > 0$ be free to vary at the root of the hierarchy, and set each $\alpha_u = \alpha_{\sigma(u)} d_u$ for each $u \in \Sigma^* \backslash \{\epsilon\}$. This hyperparameter settings also retain the coagulation and fragmentation properties which allow us to marginalize out many PYPs in the hierarchy for efficient inference

#### 4.1.2 Compact Representation

Memory usage is improved by using a Chinese restaurant franchise representation and storing only the minimal statistics about each restaurant required to make predictions: the number of customers and the number of tables occupied by the customers$(c_{us}, t_{us})$. Our starting point is the joint distribution over the Chinese restaurant franchise:

$$P(\{c_{us}, t_{us}, A_{us}\}, x_{1:T}) = \left( \prod_{s \in \Sigma} H(s)^{t_{\epsilon s}} \right) \prod_{u \in \Sigma^*} \left( \frac{[\alpha_u + d_u]_{d_u}^{t_u - 1}}{[\alpha_u + 1]_1^{c_u - 1}} \prod_{s \in \Sigma} \prod_{a \in A_{us}} [1 - d_u]_1^{|a|-1} \right)$$

Here, $\alpha_u$ and $d_u$ are concentration and discount parameters respectively. Integrating out the seating arrangements $A_{us}$ gives the joint distribution over $\{c_{us}, t_{us}\}$:

$$P(\{c_{us}, t_{us}\}, x_{1:T}) = \left( \prod_{s \in \Sigma} H(s)^{t_{\epsilon s}} \right) \prod_{u \in \mathcal{U}} \left( \frac{[\alpha_u + d_u]_{d_u}^{t_u - 1}}{[\alpha_u + 1]_1^{c_u - 1}} \prod_{s \in \Sigma} S_{d_u}(c_{us}, t_{us}) \right)$$

Here, $\mathcal{U}$ denote the reduced set of contexts, and redefine $\sigma(u)$ is the parent of u in $\mathcal{U}$.

#### 4.1.3 Modified Inference

Our goal is to compute the posterior of either $\{t_s, A_s\}$ or only $\{t_s\}$, for efficient operation in this new representation. Current inference algorithms for the SM and hierarchical Pitman-Yor processes operate in the Chinese restaurant franchise representation, and use either Gibbs sampling or particle filtering. Since each $c_{us}$ is determined by $c_{us}^x$ and the $t_{vs}$ at child restaurants v, it is sufficient to update each $t_{us}$, which for $t_{us}$ in the range $\{1, \ldots, c_{us}\}$ has conditional distribution

$$P(t_{us}/rest) \propto \frac{[\alpha_u + d_u]_{d_u}^{t_u - 1}}{[\alpha_{\sigma(u)} + 1]_1^{c_{\sigma(u)} - 1}} S_{d_u}(c_{us}, t_{us}) S_{d_{\sigma(u)}}(c_{\sigma(u)s}, t_{\sigma(u)s})$$

$S_d(c, t)$ typically has very high dynamic range, so care has to be taken to avoid numerical under/overflow.

## 5 PAQ

Mahoney [11] introduced the novel idea of combining predictions of various context based probabilistic models in an ensemble. The resulting data archiver called PAQ has since undergone a series of revisions. Some domain specific variants of PAQ have also been developed, going on to win the Calgary challenge [3] and the Hutter prize[1]. PAQ is widely regarded as the state of the art in lossless data compression, in terms of efficiency [4]. The current version of PAQ is called ZPAQ [10]. ZPAQ uses a *model* constituted of various *components*, each of which is an independent prediction engine which predicts the probability of the next symbol in a sequence (typically a sequence of bits). The various components can be combined in arbitrary fashion to give rise to the final architecture. A typical architecture is given here:

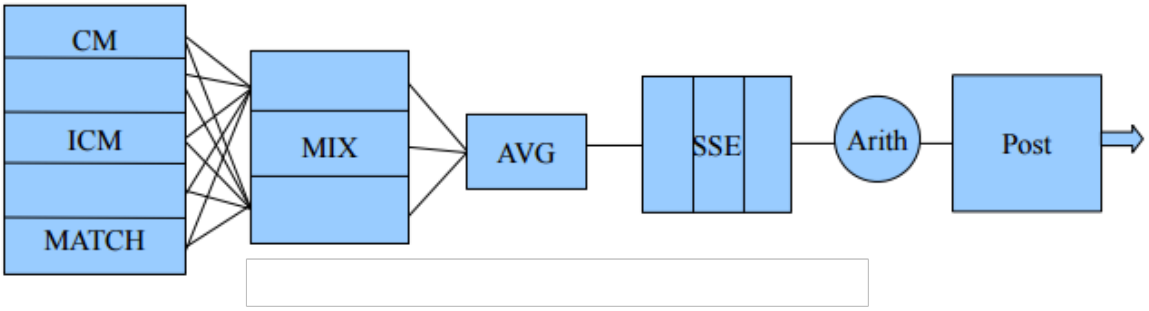


Figure 4: A possible ZPAQ architecture [10]

The first layer of the architecture are context based prediction models, which are PPM-like models predicting the probability of the next symbol. The next layers successively mix the predictions of the previous layers using the knowledge derived from the observed sequence. Some of these components are described here:

- *CM*: Context Model uses a table which maps a context (sequence of bits from the end of the current prefix) to a prediction bit. The table is adjusted according to the prediction error.
- *ICM*: Indirect Context Model uses an indirect mapping by first using a hash table to map the bit history (the history of bits observed immediately after the current context) of the contexts and the most recent bit observed after the context. It then uses a table like CM to get the prediction.
- *MATCH*: This component uses a table which matches the current context with the past occurrences of the same. The confidence of the prediction depends on the length of the match.
- *MIX2*: This component mixes the predictions of arious components using a weighted average. The weights are adjusted after every prediction according to a learning rate.
- *SSE*: Secondary Symbol Estimation uses a table (like CM) mapping the final prediction from previous layers and the current context to give the current prediction. The table is adjusted according to the prediction error.

Other variants of PAQ use different component set, mixers and architectures. For example, PAQ8 uses a neural network type mixer combining predictions from over 552 base models [9].

# 6 Experiments

We wanted to investigate the impact mixing updates on the effectiveness of compression. This reason for this is the fact that PPM-DP uses blending and Sequence Memoizer uses non-parametric models which means that they try to estimate probabilities across different context lengths. We ran ZPAQ with different architectural configurations and compared the plotted the results generated by them. Here we present the results with the Dickens corpus and Webster corpus [6].

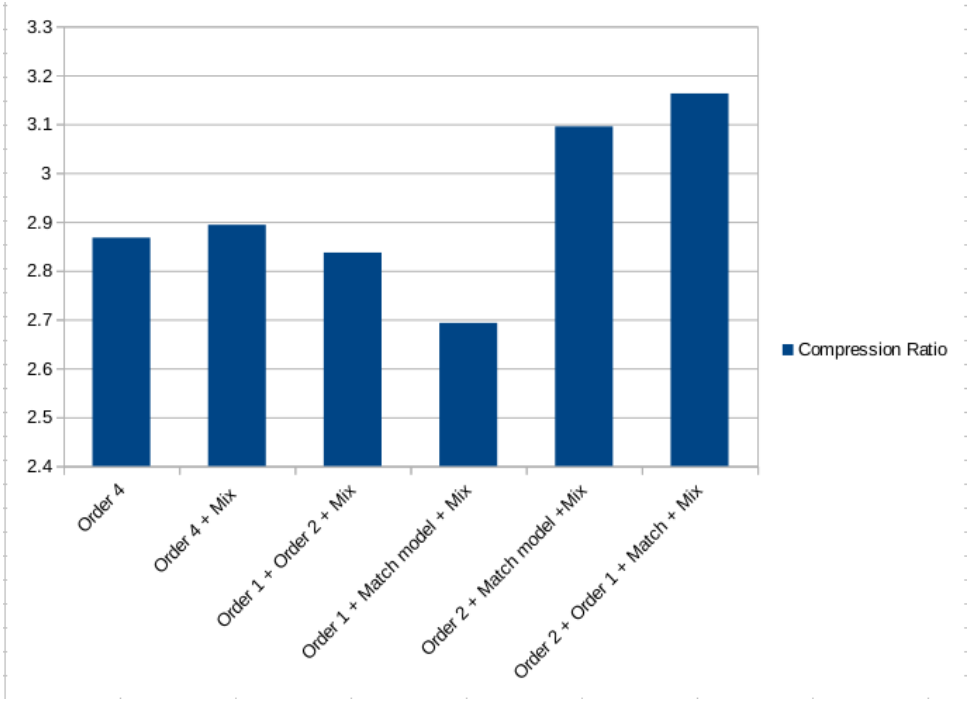


Figure 5: Compression Ratios for Dickens corpus

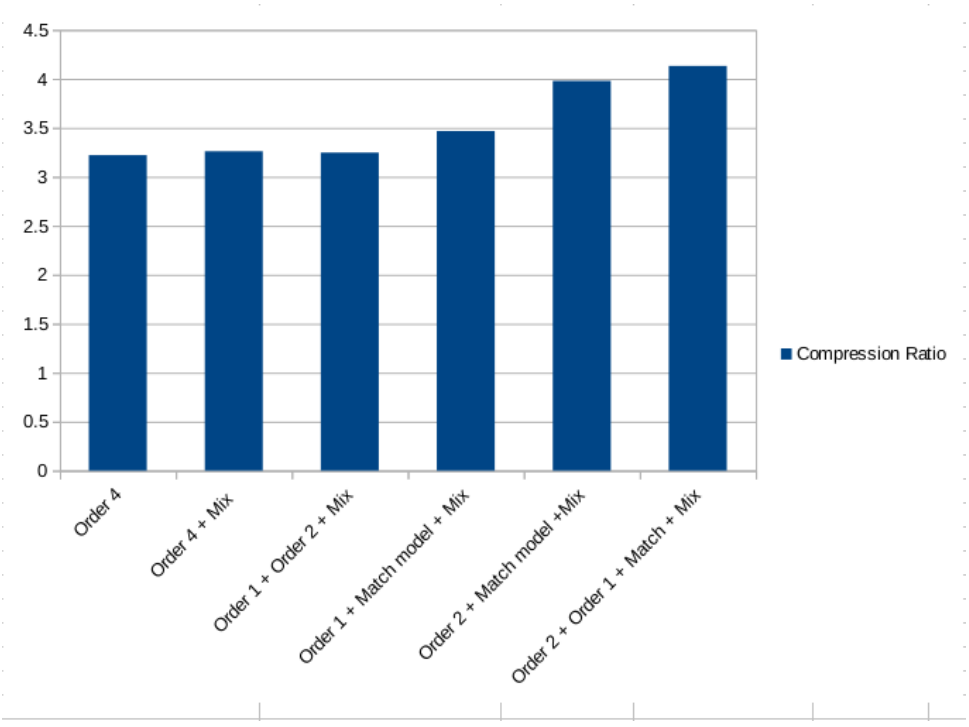


Figure 6: Compression Ratios for Webster corpus

The code and results are available here and here

# 7 Conclusion

The models of PPM-DP and SM suggest that more efficient encoding can be obtained by considering a larger number of contexts and longer contexts. However, suitable blending and mixing algorithms are quite essential in achieving good performance. PAQ already uses information from a number of context dependent models to predict the next symbol. Our experiments indicate that mixtures of lower order models are able to perform as good as higher order models which supports the hypothesis that combining models is effective. One idea here might be to massively parallelize the model predictions, since each model can run independently. Thus, in the future we can investigate distributed compression algorithms. Also, some shallow or deep learning architectures might prove to be well suited for such applications.

# References


[1] 50'000 prize for compressing human knowledge. Accessed: 2016-04-20.

[2] Context adaptive binary arithmetic coding. Accessed: 2016-04-20.

[3] Prize for compressing human knowledge. Accessed: 2016-04-20.

[4] Squash compression benchmark. Accessed: 2016-04-20.

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

[6] Sebastian Deorowicz. Silesia compression corpus, 2014.

[7] Jan Gasthaus and Yee W Teh. Improvements to the sequence memoizer. In *Advances in Neural Information Processing Systems*, pages 685–693, 2010.

[8] Paul G Howard and Jeffrey Scott Vitter. Arithmetic coding for data compression. *Proceedings of the IEEE*, 82(6):857–865, 1994.

[9] Byron Knoll. *A Machine Learning Perspective on Predictive Coding with PAQ8 and New Applications*. PhD thesis, The University of British Columbia (Vancouver, 2011.

[10] Matt Mahoney. The zpaq open standard format for highly compressed data - level 2. 2013.

[11] Matthew V Mahoney. Adaptive weighing of context models for lossless data compression. 2005.

[12] Claude Elwood Shannon. A mathematical theory of communication. *ACM SIGMOBILE Mobile Computing and Communications Review*, 5(1):3–55, 2001.

[13] Christian Steinruecken, Zoubin Ghahramani, and David MacKay. Improving ppm with dynamic parameter updates. In *Proc. Data Compression Conference*, 2015.

[14] Frank Wood, Jan Gasthaus, Cédric Archambeau, Lancelot James, and Yee Whye Teh. A stochastic memoizer for sequence data, 2009.

[15] Frank Wood, Jan Gasthaus, Cédric Archambeau, Lancelot James, and Yee Whye Teh. The sequence memoizer. *Communications of the ACM*, 54(2):91–98, 2011.