Information-Preserving Markov Aggregation

Bernhard C. Geiger*, Christoph Temmel†

*Signal Processing and Speech Communication Laboratory, Graz University of Technology, Austria
†Department of Mathematics, VU University - Faculty of Sciences, Amsterdam, The Netherlands
geiger@ieee.org, ctc@temmel.me

Abstract—We present a sufficient condition for a non-injective function of a Markov chain to be a second-order Markov chain with the same entropy rate as the original chain. This permits an information-preserving state space reduction by merging states or, equivalently, lossless compression of a Markov source on a sample-by-sample basis. The cardinality of the reduced state space is bounded from below by the node degrees of the transition graph associated with the original Markov chain.

We also present an algorithm listing all possible information-preserving state space reductions, for a given transition graph. We illustrate our results by applying the algorithm to a bi-gram letter model of an English text.

Index Terms—lossless compression, Markov chain, model order reduction, n-gram model

I. INTRODUCTION

Markov chains are ubiquitously used in many scientific fields, ranging from machine learning and systems biology over speech processing to information theory, where they act as models for sources and channels. In some of these fields, however, the state space of the Markov chain is too large to allow either proper training of the model (see n-grams in speech processing [1]) or its simulation (as in chemical reaction networks [2]). In these situations it is convenient to define a model of the process on a smaller state space, which not only allows efficient simulation (such as higher-order Markov models), but also preserves as much information of the original model as possible.

One way to reduce the cardinality of the state space of a Markov chain is to merge states, which is equivalent to feeding the process through a non-injective function. The merging usually depends on the cost function; candidate methods either rely on the Fiedler vector or other spectral criteria [3], [4], or on the Kullback-Leibler divergence rate w.r.t. some reference process [5], [6].

In addition to the model information lost by merging, the obtained process does, in general, not possess the Markov property. Modeling it as a Markov chain on the reduced state space as suggested in, e.g., [5], typically leads to an additional loss of model information. The same holds for Markov models obtained from clustered training data, as, e.g., for the n-gram class model in [7]. Consequently, there is a trade-off between cardinality of the state space, model complexity, and information loss.

Recently, we have shown the existence of sufficient conditions on a Markov chain and a non-injective function merging its states such that the obtained process is not only a kth-order Markov chain (which is desirable from a computational point-of-view), but also preserves full model information [8]. While the former property is commonly referred to as lumpability, the latter is a rather surprising one: whereas, in principle, stationary sources can be compressed efficiently by assigning codewords to blocks of samples, our result shows that in some cases lossless compression is possible on a sample-by-sample basis: Encoding is trivial, and the decoder only needs to remember the last k symbols.

Extending our previous results, we show in Section III, using spectral theory of graphs, that for an information-preserving compression, the number of input sequences merged to the same output sequence is bounded independently of the sequence length. This result allows us to estimate the minimum cardinality of the reduced state space based on the degree structure of the transition graph of the original Markov chain. Furthermore, we prove that, if a specific partition of the original state space satisfies the sufficient conditions for kth-order Markovity and information-preservation, then so does every refinement of this partition. Section IV focuses on second-order Markov chains, due to their computationally desirable properties, and presents an iterative algorithm listing all possible partitions satisfying the abovementioned sufficient conditions. We finally apply our algorithm to a bi-gram letter model in Section V.

An extended version of this paper can be found in [9].

II. PRELIMINARIES & NOTATION

Throughout this work, we deal with an irreducible, aperiodic, homogeneous Markov chain X on a finite state space ℳ and with transition matrix P. Let X_n be the nth sample of the process, and let X^2_n := {X_1, X_2, ..., X_n}. We assume that X is stationary, i.e., that the initial distribution of the chain coincides with its invariant distribution μ. Hence, for every n, the distribution P_{X_n} of X_n equals μ.

We consider a surjective lumping function g: ℳ → ℳ', with card(ℳ) := N > M := card(ℳ') ≥ 2. Abusing notation, we extend g to ℳ^n → ℳ'^n coordinate-wise and denote by g^{-1}[y] the preimage of y under g. We call the stationary stochastic process Y, defined by Y_n := g(X_n), the lumped process and the tuple (P, g) the lumping.

Since the lumping function is non-injective, a loss of information may occur, which we quantify by the conditional entropy rate

$$\overline{H}(X|Y) := \lim_{n \to \infty} \frac{1}{n} H(X^n|Y^n) = \overline{H}(X) - \overline{H}(Y)$$ (1)
where $H(\cdot)$ and $\overline{H}(\cdot)$ denote the entropy and the entropy rate (if it exists) of the argument, respectively. The lumping $(\mathbb{P}, g)$ is information-preserving iff $\overline{H}(X|Y) = 0$.

III. PREVIOUS RESULTS & EXTENSIONS

We summarize several definitions and results from [8] relevant to this work:

**Definition 1** (Preimage Count). The preimage count of length $n$ is the random variable
\[ T_n := \sum_{x \in g^{-1}[Y_n]} \Pr(X_n = x) > 0 \]  
(2)

where $[A] = 1$ if $A$ is true and zero otherwise (Iverson bracket).

In other words, the preimage count maps each sequence of length $n$ of the output process $Y$ to the cardinality of the realizable portion of its preimage.

The following characterization holds [8, Thm. 1]:
\[ \overline{H}(X|Y) = 0 \iff \exists C < \infty : \Pr(\sup_{n \to \infty} T_n \leq C) = 1 \]  
\[ \overline{H}(X|Y) > 0 \iff \exists C > 1 : \Pr(\liminf_{n \to \infty} \sqrt{T_n} \geq C) = 1 \]  
(3b)
i.e., that an almost-surely bounded preimage count (for arbitrary sequence length $n$) is equivalent to a vanishing information loss rate.

The information-preserving case (3a) can be strengthened to a deterministic version:

**Proposition 1** (Bounded Preimage Count).
\[ \overline{H}(X|Y) = 0 \iff \exists C < \infty : \sup_{n \to \infty} T_n \leq C \]  
(4)

**Proof:** See Appendix.

An interesting line for future research would be to show a deterministic analog of (3b) and its direct derivation from the Shannon-McMillan-Breiman theorem [10, Ch. 16.8].

As a corollary to Proposition 1 we get

**Corollary 1.** An information-preserving lumping $(\mathbb{P}, g)$ satisfies
\[ M \geq \min_i d_i \]  
(5)

where $d_i := \sum_{j=1}^{N} [P_{ij} > 0]$ is the out-degree of state $i$.

**Proof:** See Appendix.

A refinement does not increase the loss of information, so information-preservation is preserved under refinements. In contrast, a refinement of a lumping yielding a $k$th-order Markov process $Y$ need not possess that property; the lumping to a single state has the Markov property, while a refinement of it generally has not.

IV. AN ALGORITHM FOR SFS(2)-LUMPINGS

In this section, we present an algorithm listing all SFS(2)-lumpings, i.e., lumpings $(\mathbb{P}, g)$ yielding a second-order Markov chain and preserving full model information. There are two reasons for focusing on this particular class of lumpings: Firstly, low-order Markov models are attractive from a computational point-of-view. To obtain a first-order Markov model, the transition chain of the original chain has to satisfy overly restrictive conditions. Thus, a second-order model represents a good trade-off between computational efficiency and applicability. Secondly, compared to the general

To this end, we introduced

**Definition 2** (Single Forward Sequence [8, Def. 9]). For $k \geq 2$ a lumping $(\mathbb{P}, g)$ has the single forward $k$-sequence property (short: SFS($k$)) iff
\[ \forall y \in Y^{k-1}, g(y) \exists! x' \in y^{-1}[g(y)] : \forall x \in g^{-1}[y], x \in g^{-1}[y] \setminus \{x'\} : \Pr(X_n = x|Y^n_k = y, X_{n-k} = x') = 0 . \]  
(7)

Thus, for every realization of $Y^n_2$, the realizable preimage of $Y^n_2$ is a singleton. Therefore, SFS($k$) implies not only that $Y$ is $k$th-order Markov, but also that the lumping is information-preserving\(^1\) [8, Prop. 10]. It is a property of the combinatorial structure of the transition matrix $\mathbb{P}$, i.e., it only depends on the location of its non-zero entries, and can be checked with a complexity of $O(N^k)$ [8].

The SFS($k$)-property has practical significance: Besides preserving, if possible, the information of the original model, those lumpings which possess the Markov property of any (low) order are preferable from a computational perspective. Moreover, the corresponding conditions for the more desirable first-order Markov output, not necessarily information-preserving, are too restrictive in most scenarios (cf. [11, Sec. 6.3]).

The next result investigates a cascade of lumpings. Below, we identify a function with the partition it induces on its domain. Let $h : X \to Z$, $f : Z \to \mathcal{Y}$, and $g := h \circ f$ be $X \to \mathcal{Y}$. Clearly, (the partition induced by) $g$ is coarser than (the partition induced by) $h$ because of the intermediate application of $f$. In other words, $h$ is a refinement of $g$.

**Proposition 2** (SFS($k$) & Refinements). If a lumping $(\mathbb{P}, g)$ is SFS($k$), then so is $(\mathbb{P}, h)$, for all refinements $h$ of $g$.

**Proof:** See Appendix.

Actually, SFS($k$) implies more than $\overline{H}(X|Y) = 0$: It implies that a sequence of states of the reduced model uniquely determines the corresponding sequence of the original model, except for the first sample. Thus, the reduced model is in some sense “invertible”.

---

1Actually, SFS($k$) implies more than $\overline{H}(X|Y) = 0$: It implies that a sequence of states of the reduced model uniquely determines the corresponding sequence of the original model, except for the first sample. Thus, the reduced model is in some sense “invertible”.\[^{1}\]
case, whether a lumping \((P, g)\) satisfies the SFS\((2)\)-property can be determined by looking only at the transition matrix\(^2\) \(P\). SFS\((2)\)-lumpings have the property that, for all \(y_1, y_2 \in Y\), from within a set \(g^{-1}[y_1]\) at most one element in the set \(g^{-1}[y_2]\) is accessible:

\[
\forall y_1, y_2 \in Y; \exists! x'_2 \in g^{-1}[y_2];
\forall x_1 \in g^{-1}[y_1], x_2 \in g^{-1}[y_2] \setminus \{x'_2\}; \quad P_{x_1, x_2} = 0. \quad (8)
\]

This gives rise to

**Proposition 3.** An SFS\((2)\)-lumping satisfies

\[
M \geq \max_i d_i. \quad (9)
\]

**Proof:** We evaluate the rows of \(P\) separately. All states \(x_2\) accessible from state \(x_1\) are characterized by \(P_{x_1, x_2} > 0\). Any two states accessible from \(x_1\) cannot be merged, since this would contradict (8). Thus, all states accessible from \(x_1\) must have different images, implying \(M \geq d_{x_1}\). The result follows by considering all states \(x_1\).

In particular, Proposition 3 implies that a transition matrix with at least one positive row does not admit an SFS\((2)\)-lumping.

An algorithm listing all SFS\((2)\)-lumpings, or SFS\((2)\)-partitions, for a given transition matrix \(P\) has to check the SFS\((2)\)-property for all partitions of \(\mathcal{X}\) into at least \(\max_i d_i\) non-empty sets. The number of these partitions can be calculated from the Stirling numbers of the second kind [12, Thm. 8.2.5] and is typically too large to allow an exhaustive search. Therefore, we use Proposition 2 to reduce the search space.

Starting from the trivial partition with \(N\) blocks, we evaluate all possible merges of two states, i.e., all possible partitions with \(N - 1\) sets, of which there exist \(\frac{N(N-1)}{2}\). Out of these, we drop those from the list which do not possess the SFS\((2)\)-property. The remaining set of admissible pairs is a central element of the algorithm.

We proceed iteratively: To generate all candidate partitions with \(N - i\) sets, we perform all admissible pair-wise merges on all SFS\((2)\)-partitions with \(N - i + 1\) sets. An admissible pair-wise merge is a merge of two sets of a partition, where either set contains one element of the admissible pair. From the resulting partitions one drops those violating SFS\((2)\) before performing the next iteration. Since this algorithm generates some partitions multiple times (see the toy example in [9]), in every iteration all duplicates are removed. The algorithm is presented in Table I.

<table>
<thead>
<tr>
<th>Table I</th>
<th>Algorithm for listing all SFS((2))-lumpings</th>
</tr>
</thead>
<tbody>
<tr>
<td>1: function ListLumpings(P)</td>
<td></td>
</tr>
<tr>
<td>2: admPairs ← GetAdmissiblePairs(P)</td>
<td></td>
</tr>
<tr>
<td>3: Lumpings(1) ← merge(admPairs) (\triangleright) Convert pairs to functions</td>
<td></td>
</tr>
<tr>
<td>4: n ← 1</td>
<td></td>
</tr>
<tr>
<td>5: while notEmpty(Lumpings(n)) do</td>
<td></td>
</tr>
<tr>
<td>6: n ← n + 1</td>
<td></td>
</tr>
<tr>
<td>7: Lumpings(n) ← []</td>
<td></td>
</tr>
<tr>
<td>8: for h ∈ Lumpings(n - 1) do</td>
<td></td>
</tr>
<tr>
<td>9: for ({i_1, i_2} \in \text{admPairs}) do</td>
<td></td>
</tr>
<tr>
<td>10: if (g \leftarrow h)</td>
<td></td>
</tr>
<tr>
<td>11: (g(h^{-1}(h(i_2))) \leftrightarrow g(i_1) \triangleright i_1) and (i_2) have same image.</td>
<td></td>
</tr>
<tr>
<td>12: if (g) is SFS((2)) then</td>
<td></td>
</tr>
<tr>
<td>13: Lumpings(n) ← [Lumpings(n); g]</td>
<td></td>
</tr>
<tr>
<td>14: end if</td>
<td></td>
</tr>
<tr>
<td>15: end if</td>
<td></td>
</tr>
<tr>
<td>16: Remove duplicates from Lumpings</td>
<td></td>
</tr>
<tr>
<td>17: end while</td>
<td></td>
</tr>
<tr>
<td>18: return Lumpings</td>
<td></td>
</tr>
<tr>
<td>20: end function</td>
<td></td>
</tr>
</tbody>
</table>

2: This does not conflict with the statement, that in general the SFS\((k)\)-property depends on the combinatorial structure of \(P\). For \(k > 2\) this dependency is more complicated than for \(k = 2\).

3: Shannon used bi-grams, or digrams as he called them, as a second-order approximation of the English language [15].
sequences, so the $n$-gram model will contain a considerable amount of zero transition probabilities. Since this would lead to problems in, e.g., a speech recognition system, those entries are increased by a small constant to smooth the model, for example using Laplace’s law [1, pp. 202].

Since, by Proposition 3, an information-preserving lumping is more efficient for a sparse transition matrix, we refrain from smoothing and use the maximum likelihood estimates of the model parameters instead. We trained a bi-gram letter model of F. Scott Fitzgerald’s “The Great Gatsby”, a text containing roughly 270000 letters. To reduce the alphabet size and, thus, the run-time of the algorithm, we replaced all numbers by ‘#’ and all upper case by lower case letters. We left punctuations unchanged, yielding a total alphabet size of $N = 41$. The adjacency matrix of the bi-gram model can be seen in Fig. 1; the maximum out-degree of the Markov chain is 37.

Of the 820 possible merges only 21 are admissible. Furthermore, there are 129, 246, and 90 SFS(2)-lumpings to sets of cardinalities 39, 38, and 37, respectively. Only two triples can be merged, namely \{LB, ‘$’, ‘x’\} and \{LB, ‘,’ ‘x’\}, where LB denotes the line break. Of the more notable pair-wise merges we mention \{‘(‘,’)’\}, \{‘(‘,’z’)’\}, and the merges of ‘#’ with colon, semicolon, and exclamation mark. Especially the first is intuitive, since parentheses can be exchanged to, e.g., ‘|’ while preserving the meaning of the symbol.

Finally, we determined the lumping yielding maximum compression, i.e., the one for which $H(Y)$ is minimal. This lumping, merging \{LB, ‘$’, ‘x’\}, \{‘1’, ‘#’\}, and \{‘(‘,’)’\}, decreases the entropy from 4.3100 to 4.3044 bits. These entropies roughly correspond to the 4.03 bits derived for Shannon’s first-order model, which contains only 27 symbols [10, p. 170].

Without preprocessing, an exhaustive search is significantly more expensive: With an alphabet size of $N = 77$ and 357 admissible pairs, the first iteration of the algorithm already checks roughly 60000 distinct partitions with 75 elements, most of them satisfying the SFS(2)-condition. Proposition 3 yields $M \geq 65$.

A modified algorithm only retains the best (in terms of the entropy of the marginal distribution) partition in each iteration. This greedy heuristic achieves a compression from 4.5706 to 4.4596 bits with $M = 66$. We do not know if in this example $M = 65$ can be attained, even by an exhaustive search.

Finally, we trained a tri-gram model of the same text, without preprocessing the alphabet, and lifted the resulting second-order Markov chain to a first-order Markov chain on $\mathcal{X}^2$ (states are now letter tuples). Eliminating all non-occurring tuples reduces the alphabet size to $N = 1347$. Proposition 3 yields $M \geq 53$. There are roughly 900000 admissible pairs. A further modified algorithm, considering only 10 random admissible pairs in each iteration, achieves a compression from 8.0285 to 7.1781 bits. The algorithm terminates at $M = 579$. The question, whether a reduction to $M < 77$ is possible (thus replacing a second-order Markov model by one on a smaller state space) remains open.

VI. CONCLUSION

We presented a sufficient condition for merging states of a Markov chain such that the resulting process is second-order Markov and has full model information. We furthermore developed an iterative algorithm finding all such merges for a given transition matrix. Finally, we presented a lower bound on the cardinality of the reduced state space depending on the maximum out-degree of the associated transition graph.

The application of our algorithm to a bi-gram letter model suggests its practical relevance for model-order reduction. Future work shall investigate whether it can be successfully applied to $n$-gram models ($n > 2$) and whether it is asymptotically optimal.

ACKNOWLEDGMENTS

The authors gratefully acknowledge Franz Pernkopf, Signal Processing and Speech Communication Laboratory, Graz University of Technology, for suggesting their theoretic results.

APPENDIX

A. Proof of Proposition 1

We recall from [8] that $\overline{H}(X|Y) = 0$ implies, for all $n$,

$$
\forall \hat{x}, \hat{x} \in \mathcal{X}, \mathbf{y} \in \mathcal{Y}^{n-2}:
\Pr(X_1 = \hat{x}, Y_2^{n-1} = \mathbf{y}, X_n = \hat{x}) > 0
\Rightarrow \exists \mathbf{x} \in \mathcal{X}^n: \Pr(X^{n-1} = \mathbf{x}|X_1 = \hat{x}, Y_2^{n-1} = \mathbf{y}, X_n = \hat{x}) = 1.
$$

(10)
We thus obtain a bound on a realization $t_n$ of the preimage count (i.e., for $Y^n_i = y^n_i$)

$$t_n = \sum_{x^n_i \in \gamma^{-1}[y^n_i]} [\Pr(X^n_i = x^n_i) > 0]$$

$$= \sum_{x^n_i \in X^n} [\Pr(X^n_i = x^n_i | Y^n_i = y^n_i) > 0]$$

$$= \sum_{x^n_i \in X^n} [\Pr(X_1 = x_1, X_n = x_n | Y^n_i = y^n_i) > 0]$$

$$\times [\Pr(X^{n-1}_2 = x^{n-1}_{n-1} | Y^n_1 = y^n_1, X_1 = x_1, X_n = x_n) > 0]$$

$$\leq \sum_{x_1 \in \gamma^{-1}[n_1]} \sum_{x_n \in \gamma^{-1}[n_n]} [\Pr(X_1 = x_1, X_n = x_n | Y^n_1 = y^n_1) > 0]$$

$$\leq N^2 < \infty$$

where (a) is due to (10). Since this holds for all $n$ and all realizations, this proves

$$\Pi(X|Y) = 0 \Rightarrow \exists C < \infty: \sup_{n \to \infty} T_n \leq C$$

(11)

With (3a), the reverse implication is trivial.

### B. Proof of Corollary 1

The proof employs elementary results from graph theory:

Let $A$ denote the adjacency matrix of the Markov chain, i.e., $A_{i,j} = [P_{i,j} > 0]$. The number of closed walks of length $k$ on the graph determined by $A$ is given as [16, p. 24]

$$\sum_{j=1}^{N} \lambda_j^k$$

(12)

where $\{\lambda_j\}_{j=1}^{N}$ is the set of eigenvalues of $A$.

Let $t_X^k$ denote the number of sequences $x \in A_X^k$ of $X$ with positive probability, i.e.,

$$t_X^k = \sum_{x \in A_X^k} [\Pr(X_X^k = x) > 0]$$

(13)

Clearly, $t_X^k \geq \sum_{j=1}^{N} \lambda_j^k$. Furthermore, defining $t_Y^k$ similarly, we obtain $t_Y^k \leq M^k$. With $\lambda_{\text{max}}$ denoting the largest eigenvalue of $A$,

$$t_X^k \geq \frac{\sum_{j=1}^{N} \lambda_j^k}{M^k} \geq \left(\frac{\lambda_{\text{max}}}{M}\right)^k$$

(14)

If $\lambda_{\text{max}} > M$, then the ratio of possible length-$k$ sequences of $X$ to those of $Y$ increases exponentially. Then, the pigeonhole-principle implies that also the preimage count $T_n$ is unbounded. Thus,

$$\Pi(X|Y) = 0 \Rightarrow M \geq \lambda_{\text{max}}$$

(15)

Finally, the Perron-Frobenius theorem for non-negative matrices [17, Cor. 8.3.3] bounds the largest eigenvalue of $A$ from below by the minimum out-degree of $P$.

### C. Proof of Proposition 2

We prove the proposition by contradiction: Assume $(P, h)$ violates SFS$(k)$. Then there exists a $z \in \mathbb{Z}^{k-1}, z \in \mathbb{Z}$ such that there exist two distinct $x', x'' \in h^{-1}[z]$ and two, not necessarily distinct $x', x'' \in h^{-1}[z]$ such that

$$\Pr(X^k_2 = x' | Z^n_2 = z, X_1 = x') > 0$$

(16)

and

$$\Pr(X^k_2 = x'' | Z^n_2 = z, X_1 = x'') > 0$$

(17)

In other words, there are two different sequences $x', x''$ accessible from either the same $(x' = x')$ or from different $(x' \neq x'')$ starting states.

Now take $y = f(z)$ and $y = f(z)$. Since $h$ is a refinement of $g$, we have $h^{-1}[z] \subseteq g^{-1}[y]$ and $h^{-1}[z] \subseteq g^{-1}[y]$. As a consequence, $x', x'' \in g^{-1}[y]$ and $x', x'' \in g^{-1}[y]$, implying that $(P, g)$ violates SFS$(k)$. This proves

$(P, h)$ violates SFS$(k) \Rightarrow (P, g)$ violates SFS$(k)$.

(18)

The negation of these statements completes the proof.

### REFERENCES


