Quantifying and Estimating the Predictability Upper Bound of Univariate Numeric Time Series

Jamal Mohammed
Universiy of Zurich
Zurich, Switzerland
mjamal@ifi.uzh.ch

Michael H.Böhlen
Universiy of Zurich
Zurich, Switzerland
boehlen@ifi.uzh.ch

Sven Helmer
Universiy of Zurich
Zurich, Switzerland
helmer@ifi.uzh.ch

ABSTRACT

The intrinsic predictability of a given time series indicates how well an (ideal) algorithm could potentially predict it when trained on the time series data. Being able to compute the intrinsic predictability helps the developers of prediction algorithms immensely in deciding whether there is further optimization potential, as it tells them how close they are to what is (theoretically) achievable. We call the intrinsic predictability the predictability upper bound Π^{max} and propose a novel method for quantifying and estimating it for univariate numeric time series. So far, this has only been done for symbolic time series, even though most real-world time series are numeric by nature. We base our technique on the close relationship between entropy and predictability, utilizing the entropy rate of a time series to compute Π^{max}. Since existing entropy rate estimators, such as those based on the Lempel-Ziv compression algorithm, only work for symbolic data, we develop new estimators using tolerance thresholds for matching numeric values. We demonstrate that Π^{max} is an effective upper bound that characterizes the intrinsic predictability of a time series. We give formal proofs and we validate our arguments experimentally by comparing Π^{max} with the prediction accuracy of different state-of-the-art models on various real-world datasets from different domains.

CCS CONCEPTS

• **Mathematics of computing** → **Time series analysis**; **Information theory**.

KEYWORDS

Prediction, Information Theory, Time series

ACM Reference Format:
Jamal Mohammed, Michael H.Böhlen, and Sven Helmer. 2024. Quantifying and Estimating the Predictability Upper Bound of Univariate Numeric Time Series. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '24), August 25–29, 2024, Barcelona, Spain.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3637528.3671995

1 INTRODUCTION

Predicting (or forecasting) future events from the past plays an important role in many application fields, e.g. sales and weather forecasts, portfolio management, and prediction of (human) mobility. Our goal is to *quantify the limits of predictability* of a given time series, i.e., providing an upper bound for the ratio of potentially achievable correct predictions to all predictions for a particular dataset. When computing this upper bound, we only rely on the intrinsic properties of the data without utilizing any additional information. That is why this upper bound is also called *intrinsic predictability* [19], in contrast to *realized predictability*, which only tells us about the performance of specific prediction algorithms. Intrinsic predictability is a powerful tool when it comes to judging whether we have poor realized predictability due to a suboptimal prediction algorithm/model or because the time series is generally hard to predict [24]. For instance, a small gap between the upper bound and the currently used prediction algorithm indicates that there is little room for improvement and that it is unlikely that we can find a better algorithm. Additionally, intrinsic predictability can be utilized to benchmark prediction algorithms against an absolute measure rather than relatively against each other.

We base our technique on the entropy rate, which has been shown to be suitable for measuring how difficult it is to predict the next value in a time series [7]. The notion that entropy and predictability are related (i.e., the higher the entropy, the lower the predictability and vice versa) has been around for a long time. While many different variants of entropy have been proposed for measuring the complexity of data, e.g. approximate entropy [25], sample entropy [26], or permutation entropy [3], none of these methods have been used to quantify the predictability. Most of the studies investigating these entropies show various rates of correlation between entropy and predictability or metrics such as the mean squared error [5, 6]. To the best of our knowledge, Song et al. [31] were the first to propose a method to compute a concrete number: the *maximum predictability* $\Pi^{\max}$, which serves as an upper bound for the predictability. However, their approach only works for symbolic time series; they map location information to a fixed alphabet. Many real-world time series are numeric, though. We show that discretizing numeric values in a straightforward way (as, for instance, done by Smith et al. [30]) does not work well, up to the point of $\Pi^{\max}$ ceasing to be an upper bound for predictability.

We propose a new way to compute $\Pi^{\max}$, avoiding discretization by introducing a tolerance ϵ. Two values x and y are considered equal if their distance is within the tolerance ϵ: $|x - y| \leq \epsilon$. This preserves proximity relations between numeric values much better than the original discretization approach and leads to more accurate values for $\Pi^{\max}$. Note that a tolerance also affects the implementation of the algorithm to compute $\Pi^{\max}$. We show how to calculate $\Pi^{\max}$ using a novel numeric Lempel-Ziv algorithm. In summary, our main contributions are

KDD '24, August 25–29, 2024, Barcelona, Spain

ACM ISBN 979-8-4007-0490-1/24/08
https://doi.org/10.1145/3637528.3671995

- We propose a new upper bound for the predictability of numeric time series given its entropy rate and a tolerance ϵ, and prove the correctness of our approach.
- We implement a novel algorithm for estimating the entropy rate of a numeric time series.
- In an experimental evaluation we demonstrate the effectiveness of our technique.

2 RELATED WORK

The related work section is divided into two parts. First, we have a look at entropy-based approaches for computing or estimating the predictability of time series. In the second part, we review techniques for implementing approaches to estimate entropy rates.

Entropy is a well-known concept applied to estimating predictability [7]. Song et al. propose an approach based on the entropy rate of a (human mobility) time series to compute an upper bound of predictability [31]. Although criticized by Smith et al. [30] and Xu et al. [34] for using a coarse spatial granularity resulting in a loose upper bound, in principle the approach by Song et al. works. However, it can only be applied to symbolic time series data. This is one of the reasons why a relatively coarse granularity was used: mapping spatial areas to symbols loses the spatial relationships between the different areas. Generalizing an entropy-based approach from a discrete domain to a continuous domain is far from trivial. As we will show later, the naive discretization applied by Smith et al. causes problems in terms of the accuracy. The straightforward approach of replacing the sum in the formula for entropy with an integral (as originally suggested by Shannon [27]) leads to the so-called *differential entropy* [4, 7], which has several issues, such as easily becoming infinite for continuous probability distributions [2] and not being invariant under coordinate transformations [4].

Since the seminal work of Shannon [27, 28], many different notions of entropy, such as approximate entropy[25], fuzzy entropy [5], sample entropy[26], and permutation entropy[3], have been proposed. An advantage of these entropy types is that they can be applied to continuous domains. For instance, Chen et al. [5] have estimated the predictability of numerical time series with fuzzy entropy, approximate entropy, and sample entropy, while Garland et al [10] have done so with permutation entropy. Garland et al. demonstrate empirically that their weighted permutation entropy (WPE) is correlated to the mean absolute scaled error (MASE). The method is limited to correlation, though, as WPE and MASE are two completely different measures and there is no direct connection between the two. This lack of being directly comparable with entropy is a general issue with measures for forecast accuracy, such as MASE (even though MASE is widely considered to be the preferred measure [15]). We are not aware of any method to compute an upper bound of the best achievable forecast accuracy for a given numerical time series. Summarizing the entropy-based state-of-the-art methods, we can say that on the one hand, there are techniques that compute an upper bound for the intrinsic predictability of a time series but only work for discrete time series [31]. On the other hand, there are approaches that can be applied to numeric time series, but which only establish a correlation between forecasting accuracy and entropy and do not provide an upper bound for the intrinsic predictability. With our method, we can compute an upper bound for the intrinsic predictability for numeric time series.

We now turn to the second part of the related work section, implementing the entropy rate computation of a time series. First of all, we point out that calculating the exact value of the entropy rate is usually not possible, as the exact probability distribution of the underlying generating process is not known (we assume that we are only given the observed values). Even if we had access to the concrete probabilities, it would still be infeasible to compute the exact entropy rate, since this has high runtime costs. Consequently, the entropy rate of a time series is usually estimated rather than computed exactly. For an overview and a discussion, see [9], in which Feutrill distinguishes between parametric estimators, which assume that the data generation follows a certain model, such as a Gaussian process [7] or a Markov process [7], and non-parametric ones, which do not make this assumption. We disregard parametric estimators, as we do not make any assumptions about the underlying model. For non-parametric estimators, such as the Lempel-Ziv-based algorithm [35], the close relationship between entropy and data compression algorithms is exploited. Intuitively, the entropy rate measures how many bits of novel information are introduced by each new sample emitted from a source. When using a compression algorithm on these samples, the compression rate will be higher if less novel information is encountered and vice versa. The Lempel-Ziv (LZ) algorithm is of particular interest, as it is a *universal source code* for ergodic, stationary sources. A code for a source X (e.g. a random variable) is a mapping from the domain of X to a set of symbols. For a universal source code, the limit of the expected code length (for the number of samples going to infinity) reaches the entropy rate and the mapping is independent of the particular source [1]. The original LZ algorithm is only suitable for symbolic data, though. We implement an entropy estimator based on the Lempel-Ziv algorithm for numeric time series.

3 PRELIMINARIES

Before we begin describing our approach, we introduce some basic definitions of time series and entropy-related concepts.

Definition 1. *A **time series T** is a time-ordered sequence $T = (x_1, x_2, \ldots, x_n)$, where x_t is the value at time point t and n is the length of time series T.*

Definition 2. *A **time series data generating process** is defined via a stochastic process X, which consists of a sequence of random variables $X_1, X_2, \ldots, X_n$. The index refers to the point in time, i.e., X_t is the random variable for time point t.*

For the moment, we make the following assumption on the time series T and the stochastic process X:

- Discrete, equi-spaced intervals of time: for $t \in \{1, 2, \ldots, n\}$ the delta between any consecutive observations x_t and x_{t+1} is constant; so, X is a discrete time stochastic process.
- Boundedness of values in T: $x_t \in [min(T), max(T)]$.
- Stationarity of X: the statistical properties of T do not change over time. More formally, the probabilistic behavior of a sequence $(x_i, x_{i+1}, \ldots, x_{i+k})$ is the same as that of a shifted sequence $(x_{i+h}, x_{i+h+1}, \ldots, x_{i+h+k})$ [29].

- Ergodicity of X: the average over ensembles and time are the same (ensembles are essentially a collection of possible states). For our case, this means that the average over potential subsequences (for a time period) is the same as the average over a single very long time period [23].

As we cover numeric time series, the random variables X_t can be interpreted as continuous random variables. The definitions of entropy below assume discrete random variables, though. We resolve this seeming contradiction in Sections 5.1 and 5.2.

Definition 3. ***Entropy*** *measures the information content, quantifying the uncertainty of a random variable. For random variable Y over domain ϕ, the entropy is defined as:*

$$H(Y) = -\sum_{y\in\phi} p(y)\log_2(p(y)) \tag{1}$$

$p(y)$ is the probability that Y takes value $y \in \phi$. Entropy is usually measured in bits, thus we use the logarithm to the base 2.

Entropy is always greater or equal to 0 (assuming that $0\log_2(0) = 0$) and is maximized for a uniformly distributed random variable. Additionally, entropy always refers to a single random variable. As a time series is defined by a stochastic process made up of a sequence of random variables $X_1, X_2, \ldots, X_n$, we need to modify how to measure the uncertainty of the process. Instead of using entropy, we use the *entropy rate*, which intuitively measures the average uncertainty per symbol in the generated time series.

Definition 4. *The **entropy rate** $\mathcal{H}$ of a stochastic process X consisting of the random variables $X_1, X_2, \ldots, X_n$ is computed with the help of conditional entropy and the limit as n tends to infinity:*

$$\mathcal{H}(X) = \lim_{n\to\infty} \frac{1}{n}\sum_{t=1}^{n} H(X_t|X_{t-1}...X_1) \tag{2}$$

We should mention that the limit may not exist if the generating process X is non-stationary or non-ergodic. Moreover, for t=1 the above equation is reduced to $H(X_1)$ [7].

Definition 5. *The **conditional entropy** measures the amount of information provided by the outcome of a random variable Y given the value of another random variable Z.*

$$\begin{aligned} H(Y|Z) &= \sum_{z\in\zeta} p(z)H(Y|Z=z) \\ &= -\sum_{z\in\zeta}\sum_{y\in\phi} p(y,z)\log_2 p(y|z) \end{aligned} \tag{3}$$

In our proofs, we use the following theorems on conditional entropy. The proofs of these theorems can be found in [7], along with more details about information theory and entropy.

Theorem 1. ***Chain rule*** *for joint and conditional entropies:*

$$H(X,Y) = H(X) + H(Y|X) \tag{4}$$

Corollary 2. *A corollary of this rule is*

$$H(X,Y|Z) = H(X|Z) + H(Y|X,Z) \tag{5}$$

Theorem 3. ***Conditionality reduces entropy**:*

$$H(X|Y) \leq H(X) \tag{6}$$

4 CURRENT CHALLENGES

While Song et al. managed to connect predictability directly to entropy, they sidestepped the whole issue of symbolic versus numeric data by assuming that the data is already discretized [31]. Smith et al. continued where Song et al. had left off and explicitly discretized the geolocation data by placing a grid on top of maps [30]. The numerical value domain was essentially divided into bins and all values contained in a bin were assigned the same symbol.

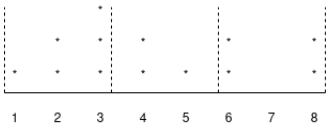


Figure 1: Binning time series values

Although binning makes it easy to determine whether a prediction is considered acceptable or not (two values sharing the same symbol match), there are still problems regarding proximity. For example, given a value of 6 and two predictions 5 and 8, intuitively 5 is a better prediction, since it is closer. However, binning rarely captures the proximity correctly. In Figure 1, we have a bin width of 2.4, but for this bin configuration, 8 would be considered a matching prediction, while 5 would not be. Assuming that two values are considered close enough when they are within distance 1.2 of each other, we have a total of 78 potentially matching pairs in Figure 1, 25 of which are within range of each other. However, using the binning depicted in the figure with dashed lines, we have 24 matching pairs, 7 of which are false positives, while missing 8 false negatives.

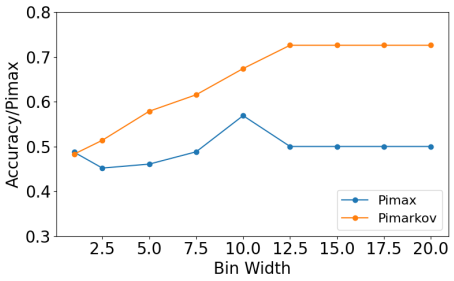


Figure 2: Proximity Issues due to binning

This is not only counterintuitive, but also leads to scenarios in which Π^{max} ceases to be an upper bound. We generated a synthetic series of integers using a first order Markov model with a total of 50 unique values and trained a model on this data. Then we computed Π^{max} for the time series, discretizing the values using different bin widths. Figure 2 shows the results. Not only does the model (labeled Pimarkov) have predictability better than the upper bound (labeled Pimax), the upper bound does not grow monotonically with the bin width (we would expect higher predictability for wider bins, due to a decrease in entropy), but it fluctuates depending on where the bounds of the bins are located. This observation highlights the failure of numeric value discretization as a method for computing the predictability upper bound of numeric sequences.

5 OUR APPROACH

Our goal is to develop a method considering the proximity of values, allowing us to compute the intrinsic predictability of a time series directly from its entropy rate. We utilize a similarity measure, i.e., we compare the time series consisting of the predicted values to the actually measured time series by determining their similarity.

5.1 Defining Predictability

First, we need to determine whether a predicted value matches the actually measured one. (We assume a forecast horizon of one, i.e., given the measured values so far, we predict the next value.) After predicting a value, we get access to the next true value, which can then be used for the following prediction. We first define predictability formally and come back to its implementation later.

Definition 6. *Given the stochastic process $X = (X_1, X_2, \ldots, X_n)$ generating the time series T, let $\hat{X} = (\hat{X}_1, \hat{X}_2, \ldots, \hat{X}_n)$ be the process generating the predictions. Given a predefined threshold ϵ, we say that there is a (prediction)* ***error****, defined by the discrete random variable E_t, at timestamp t, if*

$$E_t = \begin{cases} 1 & |x_t - \hat{x_t}| > \epsilon \\ 0 & otherwise \end{cases} \tag{7}$$

In the following, when we state that $X_t \approx_\epsilon \hat{X}_t$, we mean that $|x_t - \hat{x_t}| \leq \epsilon$ (thus, $X_t \not\approx_\epsilon \hat{X}_t$ means $|x_t - \hat{x_t}| > \epsilon$). Applying $\approx_\epsilon$ to a sequence of values means that $X_i, X_{i+1}, \ldots, X_{i+l} \approx_\epsilon \hat{X}_i, \hat{X}_{i+1}, \ldots, \hat{X}_{i+l}$ is true, iff $\bigwedge_{j=0}^{l}(X_{i+j} \approx_\epsilon \hat{X}_{i+j})$ is true.

Compared to binning, this is a subtle but crucial difference. Using a tolerance threshold for the distance between values avoids misclassifications: all pairs within distance ϵ will be matched, while those that are further apart will not. We use the definition of a prediction error to define the predictability of a given model (described by its stochastic process $\hat{X}$) for a particular time series.

Definition 7. *We distinguish different levels of* ***predictability****.*

- *The predictability $\Pi_t^{\hat{X}}$ of $\hat{X}$ for a single step t is equal to*

$$\Pi_t^{\hat{X}} = p(E_t = 0) = 1 - p(E_t = 1) \tag{8}$$

- *Calculating the average predictability for the process up to timestamp n boils down to*

$$\Pi^{\hat{X}}(n) = \frac{1}{n}\sum_{t=1}^{n}\Pi_t^{\hat{X}} \tag{9}$$

- *The overall predictability of $\hat{X}$ is the limit of Equation* (10) *with n tending to infinity:*

$$\Pi^{\hat{X}} = \lim_{n\to\infty}\frac{1}{n}\sum_{t=1}^{n}\Pi_t^{\hat{X}} \tag{10}$$

An open question remains when it comes to applying Equation (7) to determine whether two values match or not: what value should we set ϵ to? When using it in the context of a similarity measure, the time series are normalized to obtain consistent measurements [11]. Morse and Patel recommend using one quarter of the standard deviation of the normalized time series for ϵ [22]. While we could do this as well, we think a user should set the value of ϵ. This makes a lot of sense for numerical time series, as there may be requirements from the application side concerning an acceptable *margin of error* of predictions, i.e., which distance of a prediction to the actual value can still be tolerated. Essentially, our approach determines the intrinsic predictability of a time series for a given margin of error and allows a user to investigate the tradeoffs between predictability and different margins of error.

5.2 Upper Bound for Predictability

First, we look at the uncertainty introduced by predicting time series values. More precisely, given the prediction $\hat{X}_t$ for timestamp t, how much information do we need to determine the actual value X_t? In fact, we also have to figure out if $\hat{X}_t$ is a correct value or if we have made a prediction error. More formally, we have to determine the conditional entropy $H(E_t, X_t|\hat{X}_t)$. Applying the chain rule (Equation (5)), we get

$$H(E_t, X_t|\hat{X}_t) = H(E_t|\hat{X}_t) + H(X_t|\hat{X}_t, E_t) \tag{11}$$

Intuitively, $H(E_t|\hat{X}_t)$ represents the information needed to communicate that an error was made, while $H(X_t|\hat{X}_t, E_t)$ stands for the information needed to determine X_t given the prediction $\hat{X}_t$ and the knowledge that an error occurred at time t. In its current form, Equation (11) is difficult, if not impossible, to compute. We would need access to a (large) number of joint and conditional probabilities and, more importantly, the random variables X_t and $\hat{X}_t$ are, strictly speaking, continuous. Nevertheless, we can provide an upper bound, essentially treating X_t and $\hat{X}_t$ as discrete random variables covering a range of values in the continuous domain.

First, we look at $H(E_t|\hat{X}_t)$. Since conditionality reduces entropy (Equation (6)), we know that

$$\begin{aligned} H(E_t|\hat{X}_t) &\leq H(E_t) \\ &= -p(E_t = 0)\log_2 p(E_t = 0) - p(E_t = 1)\log_2 p(E_t = 1) \end{aligned} \tag{12}$$

E_t is a discrete random variable, so we are able to apply Equation (1). We distinguish two cases for the right-most part of Equation (11):

$$\begin{aligned} H(X_t|\hat{X}_t, E_t) &= p(E_t = 0)H(X_t|\hat{X}_t, E_t = 0) + \\ &\quad p(E_t = 1)H(X_t|\hat{X}_t, E_t = 1) \\ &= p(E_t = 1)H(X_t|\hat{X}_t, E_t = 1) \end{aligned} \tag{13}$$

The first part for $p(E_t = 0)$ drops out, because if we make a correct prediction, then $X_t \approx_\epsilon \hat{X}_t$ and we do not need any additional information to determine the correct value, which means $H(X_t|\hat{X}_t, E_t = 0)$ is equal to $H(X_t|X_t) = H(\hat{X}_t|\hat{X}_t)$, which is equal to zero.

That leaves us with the second part $p(E_t = 1)H(X_t|\hat{X}_t, E_t = 1)$. We know that we made an error, so $X_t \not\approx_\epsilon \hat{X}_t$, which, according to Definition 6, means that the difference between the predicted value $\hat{x_t}$ and the correct value x_t is greater than ϵ. We divide the range of all possible values into sub-intervals of width ϵ starting from $\hat{x_t}$. Figure 3 illustrates this: the correct value x_t will not be found in the gray sub-intervals (otherwise, $x_t \approx_\epsilon \hat{x_t}$ would be true), but in one of the other intervals.

We now have to determine the number of sub-intervals. Assume that we know the smallest and largest theoretically possible value of the domain of X_t, denoted by $x_{\min}$ and $x_{\max}$, respectively. If we handle predictions $\hat{X}_t$ falling outside of the range as being equal to

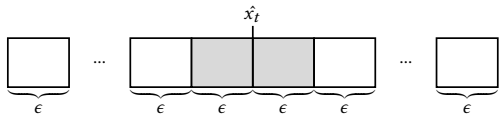


Figure 3: Making an incorrect prediction

$x_{\min}$ or $x_{\max}$, then we also guarantee that $\hat{X_t}$ has the same domain as X_t. An upper bound for the maximum number N of distinct intervals of width ϵ (in the range $[x_{\min}, x_{\max}]$) is

$$N = \frac{x_{\max} + \epsilon - (x_{\min} - \epsilon)}{\epsilon} \tag{14}$$

Consequently, knowing that we have N intervals and that the two ϵ-intervals neighboring $\hat{x_t}$ are blocked, X_t has to fall into one of the other (of at most) $N-2$ ranges. The entropy becomes maximal for a uniform distribution, so we can determine an upper bound:

$$H(X_t|\hat{X_t}, E_t = 1) \leq \sum_{i=1}^{N-2} \frac{1}{N-2} \log_2 \frac{1}{N-2} = \log_2(N-2) \tag{15}$$

Finally, we connect the entropy $H(E_t, X_t|\hat{X_t})$ to the entropy rate $\mathcal{H}(X)$ of the time series. Since the prediction model uses the past observations to make a prediction, i.e., $\hat{X_t} = f(X_1, X_2, \ldots, X_{t-1})$, $H(X_t|\hat{X_t}) = H(X_t|X_{t-1}, \ldots, X_1)$. Applying the chain rule differently to $H(E_t, X_t|\hat{X_t})$, we obtain

$$H(E_t, X_t|\hat{X_t}) = H(X_t|\hat{X_t}) + H(E_t|X_t, \hat{X_t}) \tag{16}$$

However, the entropy $H(E_t|X_t, \hat{X_t})$ is equal to zero, since knowing X_t and $\hat{X_t}$, we can directly deduce E_t without any additional information, so

$$H(E_t, X_t|\hat{X_t}) = H(X_t|\hat{X_t}) = H(X_t|X_{t-1}, \ldots, X_1) \tag{17}$$

Computing the average entropy $H(X|\hat{X})$ of predicting the whole time series then boils down to

$$H(X|\hat{X}) = \frac{1}{n} \sum_{t=1}^{n} H(X_t|X_{t-1}, \ldots, X_1) \tag{18}$$

This expression is equivalent to the one found in the entropy rate (cf. Definition 4). Combining all the previous results (Equations (11), (12), (13), (15), and (18)) yields:

$$\mathcal{H}(X) \leq -\Pi^{\hat{X}} \log_2 \Pi^{\hat{X}} - (1 - \Pi^{\hat{X}}) \left(\log_2(1 - \Pi^{\hat{X}}) - \log_2(N-2) \right) \tag{19}$$

This means, we have now connected the entropy rate of a time series directly with its predictability. The inequality in Equation (19) holds for an arbitrary prediction model $\hat{X_t}$. We are interested in an upper bound for the predictability, which means finding the largest possible value for $\hat{X_t}$ satisfying (19). We denote this upper bound by $\Pi^{\max}$ and compute its value by determining the entropy rate $\mathcal{H}(X)$ of the time series and then (numerically) solving the equation $\mathcal{H}(X) = -\Pi^{\max} \log_2 \Pi^{\max} - (1-\Pi^{\max})(\log_2(1-\Pi^{\max}) + \log_2(N-2))$ for $\Pi^{\max}$. Any other value for $\Pi^{\hat{X}}$ will result in a higher entropy and therefore a lower predictability. How to determine the entropy rate of a time series is the topic of the following section.

6 ENTROPY RATE ESTIMATION

Equation (19) makes it clear that being able to accurately estimate the entropy rate is crucial for our approach. As already indicated, there are numerous different ways to estimate the entropy rate of a time series. Many of these methods converge to the true entropy rate when the length n of the time series goes to infinity. In this case, the differences between those estimators do not play a role, as their limit is always the true entropy rate. However, when applying different entropy rate estimators to finite time series, the differences do matter. Surprisingly, work on entropy rate estimation for finite time series is few and far between. The most detailed (empirical) study we found was by Lesne et al. [17]. The authors investigate two block entropy and two Lempel-Ziv variants. However, as it is not possible to generalize block entropy estimators to numerical data, we focus on Lempel-Ziv (LZ) estimators.

Approaches based on Lempel-Ziv use the compressibility of a time series to estimate the entropy rate. This is done by parsing the time series and partitioning it into a set of distinct words. For the computation of the entropy rate, we are particularly interested in the number of words created by the parsing: the higher/lower the number of partitions, the higher/lower the entropy. Two different parsing approaches have been proposed, which, for sake of simplicity, we call LZ1 and LZ2.

6.1 Numeric Lempel-Ziv 1 (NLZ1)

LZ1 is based on the 1978 paper by Lempel and Ziv [36]. Cover and Thomas call this a tree-structured Lempel-Ziv algorithm, as it can be viewed as building a (tree-structured) dictionary [7]. Essentially, the parsing searches for the shortest prefix of the part of the time series not parsed yet that is not in the dictionary. For symbolic data, matching values during the parsing is straightforward: we can compare the values directly. For numeric data, we have to use our comparison operator $\approx_\epsilon$. Algorithm 1 shows pseudocode for the numeric LZ1 estimator (NLZ1), returning the number of partitions; $w[i, j]$ denotes the subsequence $x_i, x_{i+1}, \ldots, x_j$ of time series T (if $i > j$, then $w[i, j]$ is the empty word).

Algorithm 1: NLZ1($T = (x_1, x_2, \ldots, x_n)$)

```
1 i ← 1, D ← ∅;
2 while i ≤ n do
3     find the smallest j ≥ i such that there is a word
        w[a, b] ∈ D with w[i, j − 1] ≈_ε w[a, b] and no word
        w[a, c] ∈ D with w[i, j] ≈_ε w[a, c];
4     D ← D ∪ {w[i, j]};
5     i ← j + 1;
6 end
7 return |D|
```

We now have to convert the number of words, $c(n) = |D|$, returned by the algorithm above for time series T with length n into an entropy rate. Cover and Thomas [7] show that

$$\mathcal{H}_C(T) = \frac{c(n)(\log_2(c(n)) + 1)}{n} \to \mathcal{H}(X) \tag{20}$$

for $n \to \infty$ for a stationary ergodic sequence (their proof is based on a proof by Wyner and Ziv [33]).

When implementing NLZ1, the most expensive part is finding a matching word in the dictionary. For symbolic data, this is just a lookup in a hash table. However, due to using the more complicated comparison operator $\approx_\epsilon$, we cannot just do a hash table lookup. Clearly, we do not want to iterate through all the keys in the dictionary one by one. Consequently, we divide the dictionary into sub-dictionaries: one for each length, i.e., we have one sub-dictionary for words of length one, one for sequences of length two, and so on. In this way, we only have to look at a subset of the words stored in the dictionary. We provide details about the implementation and complexity in the appendix.

6.2 Numeric Lempel-Ziv 2 (NLZ2)

LZ2 appears in a paper by Wyner and Ziv[32] and looks for the shortest prefix of the part of the time series not parsed yet that has not been encountered anywhere in the parsed subsequence. Since we use the estimator by Kontoyiannis et al. [16], we compute the lengths of all words for every starting position within T. Again, we have to make changes to be able to deal with numerical data, utilizing the operator $\approx_\epsilon$. Algorithm 2 shows pseudocode for NLZ2.

Algorithm 2: NLZ2($T = (x_1, x_2, \ldots, x_n)$)

```
i ← 1, allocate list Λ ← (λ_1, λ_2, ..., λ_n);
while i ≤ n do
    find the smallest j ≥ i such that there is a word
      w[a, b] ∈ (x_1, x_2, ..., x_{i−1}) with w[i, j − 1] ≈_ε w[a, b]
      and no word w[a, c] ∈ (x_1, x_2, ..., x_{i−1}) with
      w[i, j] ≈_ε w[a, c];
    λ_i ← j − i + 1;
    i++;
end
return Λ
```

Again, we have to turn the returned value into an entropy rate. Kontoyiannis et al. use the following formula for doing so:

$$\mathcal{H}_K(T) = \frac{\log_2(n)}{\frac{1}{n}\sum_{i=1}^{n}\lambda_i} \tag{21}$$

They show that for $n \to \infty, \mathcal{H}_K(T)$ converges to $\mathcal{H}(X)$ for stationary ergodic processes. The motivation for computing λ_i for every position from 1 to n and averaging the values is to decrease the variance.

In NLZ2, we have to find a matching word in the part of the time series we already parsed, which means doing threshold-based comparisons again. In a first step, we build an inverted index for faster lookup of matching values. For this purpose, we traverse the time series T and for each distinct value we find, we create a list of positions where this value is found in T and store the information in a dictionary. In a final step, we create a sorted array of all the keys in the dictionary that acts as a directory for the inverted index.

After building the index, we traverse the time series value by value and look up all previously parsed positions that match the current value. For each of these positions, we find the longest matching subsequence (up to the current position). The maximum length of all these subsequences plus one (because we are looking for the shortest unseen subsequence) becomes the value of the current λ. Once we have determined the values of all λ_i in Λ, we can estimate the entropy rate by using Equation (21).

Finally, we describe how lookups in the index work. In a first step, we have to find all values in the directory of the index that match the current value in the time series. Since the entries in the directory are sorted, we use binary search to find the first matching value (and then traverse back and forth from there). For every matching value, we look up the positions in the dictionary and return those that have already been parsed. Further details are provided in the appendix.

7 EXPERIMENTAL EVALUATION

We run several experiments to evaluate our approach empirically.[1] The first experiments are conducted in an ideal setting, i.e., first, the time series satisfy all the assumptions made in Section 5 (covering the theoretical results) and, second, we can determine the exact entropy rate of the time series. We demonstrate that Π^{max} is indeed an upper bound for the predictability of numeric univariate time series and that the entropy rate can also be estimated accurately in an ideal scenario. Then, we move on to experiments using real-world data, showing that our approach is also applicable if the underlying assumptions do not hold.

7.1 Experimental setup and Datasets

We use the first 80% of each time series $(x_1, x_2, \ldots, x_r)$ to train different prediction models and to estimate Π^{max}. The remaining 20% of the time series $(x_{r+1}, x_{r+2}, \ldots, x_n)$ is used to evaluate the accuracy Π^{model} of the models. Π^{model} is computed as shown in Formula (10), i.e., the proportion of correct predictions among all predictions. We use a one-step ahead forecasting method [15], i.e., the next value is predicted based on past observations up to this point. Let x_t be the actual value and $\hat{x_t} = f(x_1, x_2, \ldots, x_{t-1})$ be the prediction of a model for time step t, respectively. Similar to Formula (7), the correctness of a prediction of a model using an error is

$$E_{model} = \begin{cases} 1 & |x_t - \hat{x_t}| > \epsilon \\ 0 & otherwise \end{cases} \tag{22}$$

Thus, the prediction accuracy of a model boils down to

$$\Pi^{model} = 1 - \frac{\sum_{t=r+1}^{n} E_{model}}{n - r} \tag{23}$$

We use the following datasets for the experiments:

- **Synthetic Datasets** are based on First Order Markov Models. In the experiments, we label the datasets using the following schema: $NxDy$, where x represents the number of states in the model, meaning the number of distinct values that are generated, and y specifies the distribution of the transition probabilities. We use the following distributions: (1) $y = u$, uniform distribution, (2) $y = s$, skewed distribution, i.e., Zipf distribution with $\alpha = 2$, and (3) $y = i$, where $i \in \{0, 1, 2, \ldots, k\}$. The parameter i determines the transition probabilities in the following way: $N - 1$ transitions, called weak transitions, have probability $p_w^i = \frac{1}{10^i N}$, and the remaining transition, called dominant transition, has the

[1] The code and data can be accessed on: https://github.com/JamalSZ/QEPUBUNTS

probability $p_d^i = 1-(N-1)p_w^i$. For $i = 0$, we have a uniform distribution; for larger i, we have more skew.

- The **S&P 500**[2] open price dataset provides a collection of daily opening prices for individual stocks. The python pandas_datareader package with parameters symbol='GSPC', start date=04-01-1990, end date= 31-10-2021 and data_source = 'yahoo' was used to load the data from Yahoo! Finance.
- The **Temperature**[3] dataset provides hourly temperature values of the city of Zurich Switzerland.
- **ETTh1** (Electricity Transformer Temperature hourly) is a dataset about the long-term deployment of electric power, consisting of two-year hourly oil temperatures of electric household power generators. [18].
- The **Precipitation**[4] dataset (CPC-CONUS) contains daily rain measurements. For the experiments, we use historical records of location (39°N, 109°W) from 1948 until 2024.

We use the real-world datasets to show that our approach also works when assumptions such as stationarity do not hold and the time series show a trend or seasonality. We ran the Augmented Dickey-Fuller[5] (ADF) stationarity test [20, 21], indicating how much each dataset deviates from stationarity (ADF critical values are: −3.431 for 1%, −2.862 for 5%, and −2.567 for 10%). Based on the p-values and the ADF statistic (see Table 1 for an overview of ADF results and trend/seasonality for all datasets), the ETTh1 dataset can be considered marginally stationary at a 1% significance level and stationary at the 5% level and shows seasonality, but no trend. The S&P 500 dataset does not meet any of the ADF criteria and is therefore clearly non-stationary, showing only a trend. For the Temperature dataset, the p-value and the ADF statistics are lower than the 1% critical value, indicating stationarity. It shows seasonality, but no trend. Finally, the Precipitation dataset has a p-value and an ADF statistic far below the critical values, strongly indicating stationarity.

	ADF Statistic	p-value	trend	seasonality
ETTh1	-3.416	0.010		✓
S&P 500	0.856	0.992	✓	
Temperature	-3.732	0.004		✓
Precipitation	-114.624	0.000	✓	✓

Table 1: Stationarity Test

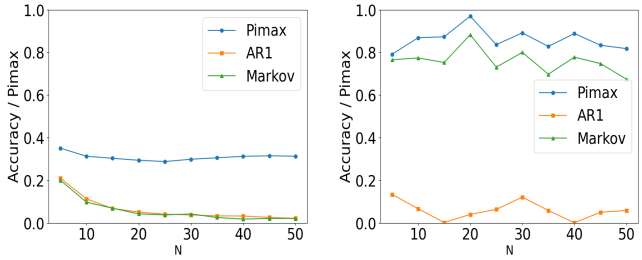
We implemented our algorithms from scratch in Python and conducted the experiments on a machine with an Intel Xeon Silver 4214 CPU with 48 cores at 1.000GHz, running Debian GNU/Linux 10 (buster) x86_64 as operating system. Solving Equation (19) for $\Pi^{\max}$ is done using the numerical solver scipy.optimize.fsolve().[6]

[2]https://finance.yahoo.com/quote/%5EGSPC/history/
[3]https://data.stadt-zuerich.ch/dataset/ugz_meteodaten_stundenmittelwerte (Accessed on 20-11-2021)
[4]Copernicus Climate Change Service, Climate Data Store, (2021): Temperature and precipitation gridded data for global and regional domains derived from in-situ and satellite observations. Copernicus Climate Change Service (C3S) Climate Data Store (CDS). DOI: 10.24381/cds.11dedf0c (Accessed on 11-06-2024)
[5]https://www.statsmodels.org/dev/generated/statsmodels.tsa.stattools.adfuller.html
[6]https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.fsolve.html

7.2 Ideal Scenario

For the ideal scenarios, we use synthetic datasets and set the tolerance level ϵ to 0 for the moment to ensure that all theoretical assumptions are met and that we can compute the exact entropy rate.

7.2.1 Π^{max} *is an Upper Bound.* We generated 20 synthetic datasets NxDy varying the number of states x and the probability distributions y for the transitions: $x \in \{5, 10, 15, \ldots, 50\}$ and $y \in \{u, s\}$. The case for $y = s$ is highly predictable, because every state has one very likely transition, while the case for $y = u$ is unpredictable, since every transition probability is the same. We trained two models on the generated time series: ARIMA of order 1 (AR(1)) and a first-order Markov model (FOMM). We cannot do better than FOMM, as it follows the underlying generating process perfectly.



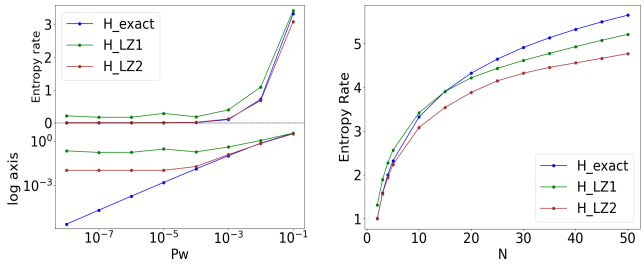
(a) Uniform $\Pi^{max,T}$ vs Π^{model} **(b) Skewed $\Pi^{max,T}$ vs Π^{model}**

Figure 4: First-Order-Markov-Model Synthetic

Figure 4 compares the predictabilities of the models and Π^{max} with each other. On the x-axis, we vary the number of different states of the Markov model and the y-axis measures the predictability (as defined in Equation 23). We make the following observations. First, Π^{max} is indeed an upper bound for these datasets, its curve is always above the ones for AR(1) and FOMM. Additionally, the general shape of the curves for FOMM and Π^{max} is very similar (in particular for the more predictable dataset with $y = s$), indicating that Π^{max} captures the potential predictability of the time series well. This is remarkable, because Π^{max} does not make any assumptions about the probability distribution underlying the generating process. Finally, as expected, the upper bound Π^{max} is also smaller for the less predictable datasets ($y = u$) and higher for the more predictable ones ($y = s$).

7.2.2 Entropy Rate Estimation. As already mentioned before, the accurate estimation of entropy rates is crucial for us. While most proposed estimators converge to the true value for time series of infinite lengths, for finite time series the situation is more complex.

For this experiment, we generated eight datasets *NxDi* with $x = 10$, $i = 0, 1, 2, \ldots, 7$ for figure 5a and 13 datasets *NxDy* with $x = 2, 3, 4, 5, 10, 15, \ldots, 50$ and $y = u$ for figure 5b. In Figure 5, we compare the quality of NLZ1 and NLZ2 with the exact entropy rate for the generated datasets. On the x-axis, we vary i, depicting the weak transition probability p_w^i (Figure 5a) and the number of states (Figure 5b). On the y-axis we show the entropy rate. We actually display the plot twice in Figure 5a: the upper half uses a linear scale, while the lower half uses a logarithmic one. Let us first look at



(a) $H_{exact}/H_{NLZ1}/H_{NLZ2}$ **(b)** $H_{exact}/H_{NLZ1}/H_{NLZ2}$

Figure 5: Entropy Estimation for different entropy levels

the NLZ1 estimator: in Figure 5a it overestimates the true entropy rate, but gets closer for large values of p_w^i (high level of entropy); in Figure 5b it starts out by overestimating the true rate for small values of x (i.e., lower level of entropy) and then gradually shifts to underestimating the rate for large values of x (i.e., higher level of entropy). This confirms empirical results by Lesne et al. in [17], who demonstrated that the accuracy of entropy rate estimators depends on the entropy level of the dataset. The classic Lempel-Ziv estimator, called LZ76 in [17], overestimated the entropy rate for low entropy levels and underestimated it for high entropy levels.

We now turn to the NLZ2 estimator. When Kontoyiannis et al. developed their estimator [16], they assumed that standard Lempel-Ziv estimators tend to overestimate the true entropy rate based on experiments conducted on a dataset consisting of four Jane Austen novels. While this assumption holds for English text, which has a relatively low entropy rate of 1.6 bits per character, Kontoyiannis et al. did not consider other datasets with higher levels of entropy, missing the fact that Lempel-Ziv estimators do not always overestimate the true entropy. Their goal was to create an estimator that, while converging to the true entropy for time series of infinite lengths, produces estimates that are lower than those of NLZ1 for finite ones. We confirm this in Figures 5a and 5b: NLZ2 generally underestimates the true entropy rate, except for datasets with very low levels of entropy (see lower left corners of figures). For our purposes, the NLZ2 estimator is usually the better estimator, as an underestimation of the true entropy rate will guarantee that Π^{max} is an upper bound. We come back to the discussion of the entropy level of datasets when investigating real-world data sets in Section 7.3. We also studied the convergence rate of the estimators: there is no substantial difference between NLZ1 and NLZ2 (the results can be found in the appendix).

7.3 Real-World Time Series

We now investigate the four real-world datasets ETTh1, S&P 500, Temperature, and Precipitation. Figures 6a, 6b, 6c, and 6d show Π^{max}, computed with NLZ1 and NLZ2, and different models trained on the data: SciNet[18], ARIMA[14], LSTM[13], and CNNLSTM[8]. On the x-axis we vary the tolerance threshold ϵ (low values of ϵ mean a high level of entropy and vice versa) and on the y-axis we display the predictability.

For the experiments, we varied the threshold level from $\epsilon^{max} = \frac{max(T)-min(T)}{2}$, where all values match each other, i.e., we have an entropy of 0, down to $\epsilon^{min} = \frac{min(x_i-x_j)}{2}, \forall x_i, x_j \in \chi$ (with χ being the set of unique values in T), where non-identical values never match, i.e., we have the highest possible entropy rate. For practical purposes, a user would have to choose a value that reflects the margin of error that is still acceptable for an application.

We observe that the models perform differently for the various datasets, but the most important question is: is Π^{max} still an upper bound for predictability for real-world datasets when many of the assumptions made for the proofs do not hold anymore (see Section 7.1). If we use the NLZ1 estimator, the answer seems to be no, as for high values of ϵ, the predictability of many models surpasses Π^{max}. It is actually more intricate, as the differences between NLZ1 and NLZ2 heavily depend on the entropy level.

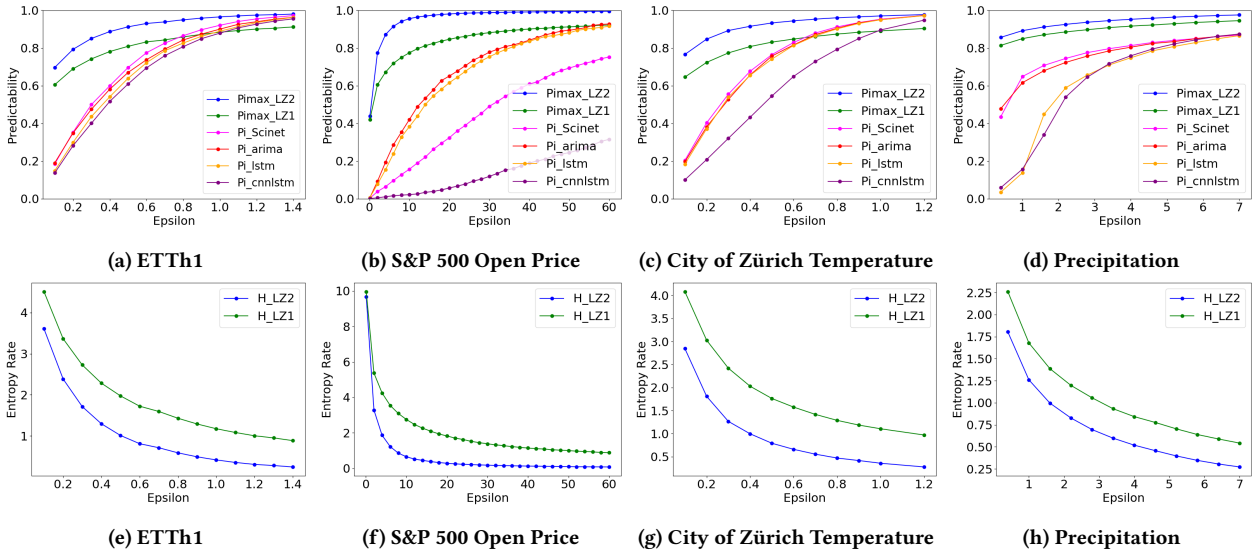
Large values of ϵ mean that we are more lenient when it comes to matching values and this has a direct impact on the entropy rate: the larger ϵ, the lower the entropy rate, which can be seen in Figures 6e, 6f,6g, and 6h for the different datasets. For low levels of entropy (i.e., large values of ϵ), as we have seen in Section 7.2.2, the NLZ1 estimator overestimates the true entropy rate and, consequently, we end up with an underestimation for Π^{max}. Significantly overestimating the true entropy rate leads to a situation in which Π^{max} ceases to be an upper bound, while underestimating the true entropy rate leads to a looser bound for Π^{max} (however, Π^{max} is still an upper bound). The NLZ2 estimator was specifically built to provide lower estimates than NLZ1, converging to the true value from below, and, therefore, is better suited for low levels of entropy. For high levels of entropy (i.e., small values of ϵ), we are faced with a different situation: at some point (e.g., in Figure 5b this was for an entropy rate between 3 and 4), NLZ1 switches from overestimating to underestimating the true entropy rate. Around this breakeven point, NLZ1 provides a very accurate estimation of the entropy rate and if we move to higher levels of entropy, NLZ1 gives us better results than NLZ2, which underestimates the true entropy rate more and more (resulting in a looser upper bound for Π^{max}). Consequently, for low levels of entropy, NLZ2 should be used, while for high levels of entropy, NLZ1 is the better estimator.

For very low levels of entropy, NLZ2 starts to overestimate the true entropy rate as well. This the case for very large values of ϵ for real-world datasets, where Π^{max} may cease to be an upper bound. Nevertheless, we argue that for practical purposes, these cases are irrelevant, since the predictability reaches values close to one, which means that we know that these scenarios are highly predictable. Additionally, very large values for the tolerance threshold ϵ are useless from the application point of view: for instance, assuming a stock price with a value of $200, making predictions within a range of +$100/-$100 of the true value is not useful.

8 CONCLUSION AND FUTURE WORK

We developed a technique for quantifying an upper bound for the predictability of univariate numeric time series and implemented an algorithm to estimate this upper bound. Unlike conventional approaches for adapting discrete algorithms to numeric data by discretizing the data, we introduce a tolerance threshold ϵ, which allows us to avoid the proximity issues caused by discretization.

In an empirical study, we validate the effectiveness of our approach and demonstrate that it can also be applied in real-world



(a) ETTh1 (b) S&P 500 Open Price (c) City of Zürich Temperature (d) Precipitation

(e) ETTh1 (f) S&P 500 Open Price (g) City of Zürich Temperature (h) Precipitation

Figure 6: (a), (b), (c), (d) Π^{max} vs Π^{model} and (e), (f), (g), (h) H_{NLZ1} vs H_{NLZ1} (real-world datasets)

settings, which do not always adhere to the conditions assumed by the theoretical framework we use. We showcase the utility of our method across a range of scenarios and datasets.

While the computation of the upper bound Π^{max} itself is robust and stable, it relies on sufficiently accurate estimations of the entropy rate of a given time series. While the theoretical properties of entropy rate estimators are largely understood, especially in scenarios modeling infinite time series, investigations into finite time series (for real-world applications) are few and far between. We plan to analyze the behavior and properties of entropy rate estimators for various finite time series with the goal of developing more robust estimators that consider the overall entropy level of a dataset, allowing us to unlock the full potential of our approach.

REFERENCES

[1] Peter Andreasen. 2001. *Universal Source Coding.* Master's thesis. University of Copenhagen, Copenhagen, Denmark.

[2] Valentina Baccetti and Matt Visser. 2012. Infinite Shannon entropy. *Journal of Statistical Mechanics: Theory and Experiment* 2013 (12 2012). https://doi.org/10.1088/1742-5468/2013/04/P04010

[3] Christoph Bandt and Bernd Pompe. 2002. Permutation Entropy: A Natural Complexity Measure for Time Series. *Phys. Rev. Lett.* 88 (Apr 2002), 174102. Issue 17. https://doi.org/10.1103/PhysRevLett.88.174102

[4] Richard E. Blahut. 2002. 25 - Information Theory and Coding. In *Reference Data for Engineers (Ninth Edition)* (ninth edition ed.), Wendy M. Middleton and Mac E. Van Valkenburg (Eds.). Newnes, Woburn, 25–1–25–31. https://doi.org/10.1016/B978-075067291-7/50027-3

[5] Weiting Chen, Jun Zhuang, Wangxin Yu, and Zhizhong Wang. 2009. Measuring complexity using FuzzyEn, ApEn, and SampEn. *Medical Engineering & Physics* 31, 1 (2009), 61–68. https://doi.org/10.1016/j.medengphy.2008.04.005

[6] Zhe Chen, Ya'an Li, Hongtao Liang, and Jing Yu. 2019. Improved Permutation Entropy for Measuring Complexity of Time Series under Noisy Condition. *Complexity* 2019 (2019), 1403829:1–1403829:12. https://doi.org/10.1155/2019/1403829

[7] Thomas M. Cover and Joy A. Thomas. 2006. *Elements of Information Theory* (2nd edition ed.). John Wiley & Sons, Inc.

[8] Jeff Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Trevor Darrell, and Kate Saenko. 2015. Long-term recurrent convolutional networks for visual recognition and description. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015.* IEEE Computer Society, 2625–2634. https://doi.org/10.1109/CVPR.2015.7298878

[9] Andrew Feutrill. 2023. *Characterisation and Estimation of Entropy Rate for Long Range Dependent Processes.* Ph. D. Dissertation. University of Adelaide, Adelaide, Australia.

[10] Joshua Garland, Ryan James, and Elizabeth Bradley. 2014. Model-free quantification of time-series predictability. *Phys. Rev. E* 90 (Nov 2014), 052910. Issue 5. https://doi.org/10.1103/PhysRevE.90.052910

[11] Dina Q. Goldin and Paris C. Kanellakis. 1995. On Similarity Queries for Time-Series Data: Constraint Specification and Implementation. In *1st Int. Conf. on Principles and Practice of Constraint Programming (CP'95).* Cassis, France, 137–153. https://doi.org/10.1007/3-540-60299-2_9

[12] Yanjun Han, Jiantao Jiao, Chuan-Zheng Lee, Tsachy Weissman, Yihong Wu, and Tiancheng Yu. 2018. Entropy Rate Estimation for Markov Chains with Large State Space. arXiv:1802.07889 [cs.LG]

[13] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation* 9, 8 (11 1997), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

[14] R.J. Hyndman and G. Athanasopoulos. 2021. *Forecasting: principles and practice* (3rd edition ed.). OTexts: Melbourne, Australia. OTexts.com/fpp3.

[15] Rob J. Hyndman and Anne B. Koehler. 2006. Another look at measures of forecast accuracy. *International Journal of Forecasting* 22, 4 (2006), 679–688. https://doi.org/10.1016/j.ijforecast.2006.03.001

[16] Kontoyiannis I., Algoet P. H., Suhov Yu. M., and Wyner A. J. 1998. Nonparametric Entropy Estimation for Stationary Processes and Random Fields, with Applications to English Text. *IEEE Transactions on Information Theory* 44 (1998), 1319–1327.

[17] Annick Lesne, Jean-Luc Blanc, and Laurent Pezard. 2009. Entropy estimation of very short symbolic sequences. *Phys. Rev. E* 79 (Apr 2009), 046208. Issue 4. https://doi.org/10.1103/PhysRevE.79.046208

[18] Minhao Liu, Ailing Zeng, Muxi Chen, Zhijian Xu, Qiuxia Lai, Lingna Ma, and Qiang Xu. 2022. SCINet: Time Series Modeling and Forecasting with Sample Convolution and Interaction. arXiv:2106.09305 [cs.LG]

[19] E.N. Lorenz. 1995. Predictability: a problem partly solved. In *Seminar on Predictability.* ECMWF, Reading, United Kingdom, 1–18.

[20] James G MacKinnon. 1994. Approximate asymptotic distribution functions for unit-root and cointegration tests. *Journal of Business & Economic Statistics* 12, 2 (1994), 167–176.

[21] James G MacKinnon. 2010. *Critical values for cointegration tests.* Technical Report. Queen's Economics Department Working Paper.

[22] Michael D. Morse and Jignesh M. Patel. 2007. An efficient and accurate method for evaluating time series similarity. In *Proc. of the ACM SIGMOD Int. Conf. on Management of Data.* ACM, Beijing, China, 569–580. https://doi.org/10.1145/1247480.1247544

[23] Athanasios Papoulis and S. Unnikrishna Pillai. 2002. *Probability, Random Variables, and Stochastic Processes* (4th ed.). McGraw Hill, Boston.

[24] Frank Pennekamp, Alison C. Iles, Joshua Garland, Georgina Brennan, Ulrich Brose, Ursula Gaedke, Ute Jacob, Pavel Kratina, Blake Matthews, Stephan Munch, Mark Novak, Gian Marco Palamara, Björn C. Rall, Benjamin Rosenbaum, Andrea Tabi,

Colette Ward, Richard Williams, Hao Ye, and Owen L. Petchey. 2019. The intrinsic predictability of ecological time series and its potential to guide forecasting. *Ecological Monographs* 89, 2 (2019), e01359. https://doi.org/10.1002/ecm.1359

[25] Steven M. Pincus. 1991. Approximate entropy as a measure of system complexity. *Proc. Natl. Acad. Sci.* 88, 6 (1991), 2297–2301. https://doi.org/10.1073/pnas.88.6.2297

[26] Joshua S. Richman and J. Randall Moorman. 2000. Physiological time-series analysis using approximate entropy and sample entropy. *American Journal of Physiology-Heart and Circulatory Physiology* 278, 6 (2000), H2039–H2049. https://doi.org/10.1152/ajpheart.2000.278.6.H2039

[27] Claude E. Shannon. 1948. A mathematical theory of communication. *Bell Syst. Tech. J.* 27, 4 (1948), 623–656. https://doi.org/10.1002/j.1538-7305.1948.tb00917.x

[28] Claude E. Shannon. 1948. A mathematical theory of communication. *Bell Syst. Tech. J.* 27, 3 (1948), 379–423. https://doi.org/10.1002/j.1538-7305.1948.tb01338.x

[29] Robert H. Shumway and David S. Stoffer. 2017. *Time Series Analysis and Its Applications* (4th edition ed.). Springer. https://doi.org/10.1007/978-3-319-52452-8

[30] Gavin Smith, Romain Wieser, James Goulding, and Duncan Barrack. 2014. A refined limit on the predictability of human mobility. *2014 IEEE International Conference on Pervasive Computing and Communications, PerCom 2014*, 88–94. https://doi.org/10.1109/PerCom.2014.6813948

[31] Chaoming Song, Zehui Qu, Nicholas Blumm, and Albert-László Barabási. 2010. Limits of Predictability in Human Mobility. *Science* 327(5968) (February 2010), 1018–1021.

[32] Aaron D. Wyner and Jacob Ziv. 1989. Some asymptotic properties of the entropy of a stationary ergodic data source with applications to data compression. *IEEE Trans. Inf. Theory* 35, 6 (1989), 1250–1258. https://doi.org/10.1109/18.45281

[33] Aaron D. Wyner and Jacob Ziv. 1991. Fixed data base version of the Lempel-Ziv data compression algorithm. *IEEE Trans. Inf. Theory* 37, 3 (1991), 878–880. https://doi.org/10.1109/18.79955

[34] Paiheng Xu, Likang Yin, ZhongTao Yue, and Tao Zhou. 2019. On predictability of time series. *Physica A-statistical Mechanics and Its Applications* 523 (2019), 345–351.

[35] Jacob Ziv and Abraham Lempel. 1977. A Universal Algorithm for Sequential Data Compression. *IEEE Transactions on Information Theory* 23 (1977), 337–343.

[36] Jacob Ziv and Abraham Lempel. 1978. Compression of individual sequences via variable-rate coding. *IEEE Trans. Inf. Theory* 24, 5 (1978), 530–536. https://doi.org/10.1109/TIT.1978.1055934

A PROPERTIES OF MARKOV CHAINS

Definition 8. *A sequence of random variables $X_1, X_2, ...X_n$ are said to form a (first order) Markov chain if the conditional distribution of any X_i depends only on X_{i-1} and is conditionally independent of $X_1..X_{i-2}$. Specifically, $X_1, X_2, ...X_n$ form a Markov chain if the joint probability mass function can be written as:*

$$p(x_1, x_2, ..x_n) = p(x_1) \cdot p(x_2|x_1) \cdot \cdots \cdot p(x_n|x_{n-1}) \tag{24}$$

Theorem 4. *Let $X = \{X_i\}$ be a stationary Markov chain with N unique states, stationary distribution μ_N and transition probability matrix P_N. Let $X_1 \sim \mu_N$. Then the exact entropy rate is:*

$$\mathcal{H}_{exact}(X) = -\sum_{ij} \mu_i \cdot P_{ij} log P_{ij}. \tag{25}$$

where the stationary distribution is the solution of the following equation

$$\mu_N = \mu_N \cdot P_N \tag{26}$$

Example 1. *Here we show how we compute the exact entropy rate of a Markov chain. Let T = {0,1,0,1,0,1,0,1} with transition probability matrix*

$$P = \begin{bmatrix} 0.5 & 0.5 \\ 0.5 & 0.5 \end{bmatrix}$$

In the general case, to compute μ_N, we first initialize μ_0 with arbitrary probabilities (summing up to one), e.g. $\mu_0 = [0.01, 0.99]$. We then iteratively compute μ_i using Equation (26) ($\mu_i = \mu_{i-1} \cdot P$) until $\mu_i \simeq \mu_{i-1}$. Since T is a uniformly distributed time series, the steady stationary distribution of T is $\mu_N = \{0.5, 0.5\}$. Finally, we calculate the entropy rate using Equation (25), obtaining
$\mathcal{H}_{exact}(T) = -\sum_{i=1}^{2}\sum_{j=1}^{2} \mu_i \cdot P_{ij} log P_{ij} = -4 * 0.5 * 0.5 \cdot log_2 0.5 = 1$

B IMPLEMENTATION AND PSEUDOCODE

Figure 7 shows the pipeline of operations needed to compute $\Pi^{\max}$. Calculating the value for N is straightforward, we determine the minimum and maximum value occurring in T and then apply Equation (14). Solving Equation (19) for $\Pi^{\max}$ is done using a numerical solver scipy.optimize.fsolve() [7] function which finds the roots of a nonlinear equation. As the most challenging part is estimating the entropy rate, we provide more details on the implementation of NLZ1 and NLZ2 here.

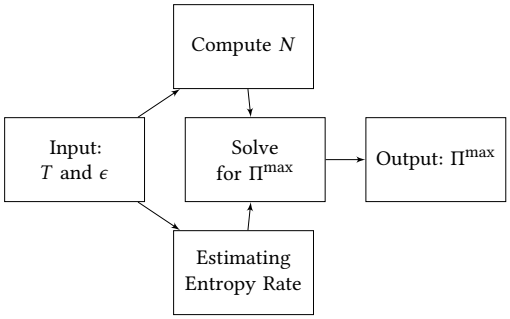


Figure 7: Framework Pipeline

B.1 Implementing NLZ1 and NLZ2

Algorithm 3 shows pseudocode for our implementation of the NLZ1 estimator and Algorithm 4 provides details about the lookup function we use.

In the NLZ2 algorithm, we first build an inverted index for faster lookup of matching values, see Algorithm 5 for pseudocode. Algorithm 6 shows pseudocode for the implementation of the actual NLZ2 estimator. Finally, we describe how lookups in the index work. We have to find all values in the directory of the index that match the current value in the time series. Since the entries in the directory are sorted, we use binary search to find the first matching value (and then traverse back and forth from there). For every matching value, we look up the positions in the dictionary and return those that have already been parsed. Algorithm 7 shows pseudocode for the lookup code.

B.2 Run Time Complexity

The run time of the algorithms presented in the previous sections depends heavily on the distribution of the values within a time series, which makes it difficult to analyze their complexity. Consequently, we look at the extreme cases, i.e., the most expensive and the least expensive one, all the other run times fall between these two extremes. We focus on the complexity of Algorithm 6, as it is more complicated than Algorithm 3. For two arbitrary values $T_i, T_j \in T$, $0 \leq i \neq j < n$, the extreme cases are:

- Case 1: $\forall i, j\ (i \neq j) : T_i \not\approx_\epsilon T_j$, i.e., T_i and T_j never overlap
- Case 2: $\forall i, j\ (i \neq j) : T_i \approx_\epsilon T_j$, i.e., T_i and T_j always overlap

[7] https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.fsolve.html

Algorithm 3: NLZ1($T = (T_1, T_2, \ldots, T_n), \epsilon$)

```
1  i ← 1, D ← ∅;
2  while i ≤ n do
3      curSeq ← T_i;
4      while LZ1lookup(curSeq, D, ϵ) do
5          i++;
6          if i ≤ n then
7              append T_i to curSeq;
8          else
9              break;
10         end
11         if not len(curSeq) in D then
12             D[len(curSeq)][curSeq] ← 1;
13             i++;
14         end
15     end
16 end
17 for i ← 1 to len(D) do
18     c_n ← c_n + len(D[i]);
19 end
20 return (c_n(log_2(c_n)+1))/n
```

Algorithm 4: NLZ1lookup(curSeq, D, ϵ)

```
1  found ← false;
2  for seq in D[len(curSeq)] do
3      i ← 1;
4      while (i ≤ len(curSeq)) ∧ (curSeq[i] ≈_ϵ seq[i]) do
5          i++;
6      end
7      if i > len(curSeq) then
8          found ← true;
9          break;
10     end
11 end
12 return found
```

Algorithm 5: NLZ2buildIdx(T)

```
1  D ← ∅;
2  for i ← 1 to len(T) do
3      D[T[i]].append(i);
4  end
5  Dir ← sorted array of keys in D;
6  return InvIdx(Dir,D)
```

Case 1. In this scenario, the loop in line 6 of Algorithm 6 will never be executed, as the set stored in the variable *matches* will always be empty. The lookup in line 5 (see also Algorithm 7), although yielding no matches (since $T_i \not\approx_\epsilon T_j$, has to search the dictionary. As the size of the dictionary is at most n and we use binary search, the run time for one search is $O(\log n)$. In Algorithm 6, the lookup

Algorithm 6: NLZ2($T = (T_1, T_2, \ldots, T_n), \epsilon$)

```
1  Λ ← {λ_1...λ_n} ;          // initialize all λ_i with 0
2  Idx ← LZ2buildIdx(T) ;
3  for cur ← 1 to n do
4      λ_max ← 0;
5      matches ← LZ2lookup(Idx, T, cur, ϵ);
6      for i in matches do
7          λ ← 0;
8          while (cur + λ < n) ∧ (i + λ < cur) do
9              if T[i + λ] ≈_ϵ T[cur + λ] then
10                 λ++;
11             end
12         end
13         if cur + λ = n then
14             return Λ
15         end
16         if λ > λ_max then
17             λ_max ← λ;
18         end
19     end
20     λ_cur ← λ_max + 1;
21 end
22 return Λ
```

is invoked n times (see line 3), thus the overall run time complexity of Case 1 is $O(n \log n)$.

Case 2. In this case, because the current value T_{cur} of the time series matches all previously seen ones $T_1, T_2, \ldots, T_{cur-1}$, the size of the set returned by the lookup (and stored in *matches*) is equal to $cur - 1$. Subsequently, the loop in line 6 of Algorithm 6 will be invoked $cur - 1$ times. The number of comparisons performed in the loop starting at line 8 is inversely proportional to the value of i: the larger i, the fewer comparisons we need to make ($cur - 1$ comparisons for $i = 1$, $cur - 2$ comparisons for $i = 2$, and so forth). Thus, in total, we have to make

$$\sum_{i=1}^{cur-1} cur - i = \sum_{i=1}^{cur-1} cur - \sum_{i=1}^{cur-1} i \tag{27}$$

$$= cur^2 - \frac{1}{2}(cur - 1)cur \tag{28}$$

$$= \frac{1}{2}cur(cur + 1) \tag{29}$$

comparisons in the inner loops (line 6 and line 8), which are executed for each iteration of the outermost loop in line 3. The outermost loop stops at cur = $\frac{n}{2}$, since we reach the end of the time series when searching for the longest match (see condition in line 8). So, the overall number of iterations is

$$\sum_{cur=1}^{\frac{n}{2}} \frac{1}{2}cur(cur + 1) \tag{30}$$

Since $O(\frac{1}{2}cur(cur+1)) = O(cur^2)$ and $O(\frac{n}{2}) = O(n)$, we obtain

$$\sum_{cur=1}^{n} cur^2 = \frac{n(n+1)(2n+1)}{6} = O(n^3) \tag{31}$$

via the square pyramidal number of n.

Therefore, the worst-case complexity of Case 2 is $O(n^3)$. Strictly speaking, there is still the cost $O(n \log n)$ of looking up the matches in line 5, but this cost is dominated by $O(n^3)$.

Algorithm 7: NLZ2lookup(Idx, T, cur, ϵ)

```
1 matches ← ∅;
2 for all i such that Idx.Dir[i] ≈_ε T[cur] do
3 |   for pos in Idx.D[Idx.Dir[i]] do
4 |   |   if pos < cur then
5 |   |   |   matches ← matches ∪ { pos };
6 |   |   end
7 |   end
8 end
9 return matches
```

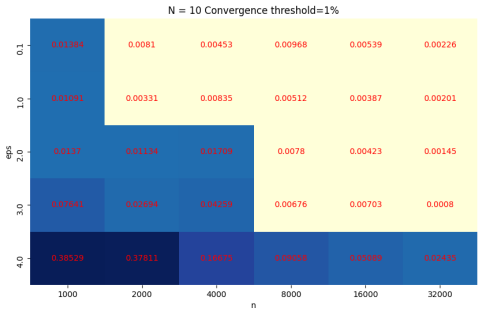
C CONVERGENCE OF THE ENTROPY RATE ESTIMATION

As we are estimating the entropy rate of finite time series, the rate of convergence of the estimation becomes a crucial aspect in determining the stability of the value Π^{max} for increasing time series lengths. It is usually difficult for real-world datasets to determine this convergence rate, as several factors influence it. Nevertheless, it would be of great help to get an indication that the length of the time series is sufficiently large to ensure a reliable value for Π^{max}. Han et al. claim that a length of $n \gg N^2$, where N is the alphabet size, is sufficient [12]. We found two issues with this approach. First of all, what is the exact meaning of "$\gg$", i.e., when is n much greater than N^2 in terms of concrete values? Second, the inequality given in [12] is too simplistic, as the convergence rate does not depend on the alphabet size alone. It is possible to have different convergence rates for the same value of N, but different levels of entropy [17].

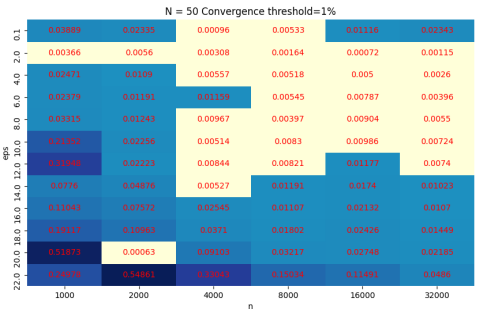
We propose an approach based on measuring the differences of the entropy rate taken from neighboring (equi-distant) points in time. More formally, if we, for instance, estimate the entropy rate for $n = 1000, 2000, 3000, \ldots$ (where n denotes the length of the prefix of a time series we use for the estimation), then $\Delta(n_1, n_2) = \mathcal{H}_{2000} - \mathcal{H}_{1000}$ measures the first difference, $\Delta(n_2, n_3) = \mathcal{H}_{3000} - \mathcal{H}_{2000}$ the second one, and so on (for computing $\mathcal{H}$ we use Algorithm 1 or 2). However, we found that using absolute values for the difference can be misleading: depending on the level of entropy, the differences vary. For high levels of entropy, the entropy rate starts out with a large value and tends to drop steeply. In contrast, for low levels of entropy, the values of the entropy rate start out smaller and decrease more slowly. Hence, we measure the relative difference:

$$\Delta_r(n_i, n_{i+1}) = \frac{\mathcal{H}_{n_{i+1}} - \mathcal{H}_{n_i}}{\max(\mathcal{H}_{n_{i+1}} - \mathcal{H}_{n_i})}. \tag{32}$$

When the relative difference drops below a certain threshold, we know that the obtained estimates are stable enough for a reliable estimation of Π^{max}.



(a) Convergence Threshold = 1% N=10



(b) Convergence Threshold = 1% N=50

Figure 8: Convergence of H with increasing n for different values of ϵ

We ran some experiments investigating the convergence by generating two datasets $NxDy$ with $x = \{10, 50\}$ and $y = u$ looking at the convergence rate of the entropy rate estimators (here we show the results for the NLZ2 estimator, the ones for NLZ1 are very similar). Like in the previous experiments, we also expected the entropy level of the data to have an effect. Figures 8a and 8b illustrate the results for $x = 10$ and $x = 50$, respectively. On the x-axis, we vary n, the length of the time series, and on the y-axis, we vary ϵ, the tolerance threshold. The numbers in the heatmaps are computed by subtracting the entropy for n from the value for the previous n according to Equation (32) (for $n = 1000$, we subtract the value from that for $n = 500$, which is not shown). Again, we observe that the level of entropy of the time series has an impact. For high levels of entropy, i.e., small values of ϵ, the estimators converge fairly quickly, while for low levels of entropy, i.e., large values of ϵ, the estimators take longer. This is an important observation, as it means that it is not sufficient to just make sure that $n \gg N^2$ as claimed by Han et al. in [12], but that we also need to consider the level of entropy of the data. For very low levels of entropy, the estimators can take quite some time to converge (bottom rows in Figures 8a and 8b). However, for practical purposes, these cases are not important; we also discuss this matter in Section 7.3.