



Project Type: Project A

# The Surprising Brittleness of Hidden Markov Model to Time Dependent Omitting Processes

Submitted by:

Liad Ben Shachar Itay Geva

Project Instructor: Binyamin Perets

Spring 2023

# Acknowledgments

We would like to express our sincere gratitude to **Benny Peretz**, our project instructor, for his invaluable guidance and support throughout the duration of this project. Benny provided us with the necessary resources, shared his expertise, and encouraged us to explore the underlying concepts independently. His approach of allowing us to engage with the material hands-on, rather than providing us with the answers outright, greatly enhanced our understanding and problem-solving skills.

We are also grateful to **Mr Kobi Kohai** and the **CRML Lab Stuff** for their assistance in coordinating access to laboratory facilities and equipment. Their support ensured that we had the necessary resources to conduct our experiments effectively.

Special thanks to **Mrs Orly Wigderson**, whose technical expertise and assistance with computer access significantly contributed to the success of this project. Her willingness to help and attention to detail were invaluable.

Furthermore, we would like to acknowledge that the project presented more challenges than we initially anticipated. However, these challenges provided valuable learning opportunities, and we emerged from this experience with a deeper understanding of the subject matter and improved problem-solving skills.

Lastly, we would like to acknowledge the support of our friends and family for their encouragement and understanding throughout this endeavor.

# Contents

# List of Figures

# Abstract

This study investigates the inherent limitations of Hidden Markov Models (HMMs) when subjected to time-dependent omitting processes. Despite the wide application of HMMs in analyzing sequential data, our research reveals a critical vulnerability: the model's performance significantly deteriorates in the absence of consecutive observational data. Through a series of experiments utilizing synthetic data, we examined the impact of various omission processes on the ability of standard HMM algorithms to accurately reconstruct transition matrices.

Our findings demonstrate that both Expectation Maximization and Gibbs Sampling algorithms fail to accurately infer the underlying model structure under conditions of non-consecutive data observations. The implications of these results underscore a pressing need for the development of new HMM algorithms capable of handling non-consecutive observation scenarios, promising significant advancements in fields reliant on accurate time-series analysis. Our research sets a new direction for future investigations into enhancing HMM resilience against data omission, with potential applications across a broad spectrum of scientific inquiries.

# List of Symbols

The next list describes several symbols that will be later used within the body of the document

**Probability**

| | |
|---|---|
| $\hat{X}$ | Estimator of a stochastic variable |
| $\omega$ | The chance parameter |
| $U$ | Uniform Distribution |
| $X$ | Stochastic variable |
| $x$ | Deterministic variable |
| $\mathcal{N}$ | Normal Distribution |
| $f_X(x)$ | Probability density function |

**HMM**

| | |
|---|---|
| $E$ | Emission Matrix |
| $O$ | Observation |
| $S$ | State |
| $T.I$ | Temporal Information |
| $T$ | Transition Matrix |
| $v_n$ | Probabilities of the states in discrete time $n$ |

**Other symbols**

| | |
|---|---|
| $L_1$ | Norm 1 of a matrix |

# Acronyms

**EM** Expectation Maximization.

**GT** Ground Truth.

**HMM** Hidden Markov Model.

**NN** Nearest Neighbors.

**OP** Omission Process.

**TMM** Transition Matrix Mode.

# List of Terms

**Expectation Maximization** Is an Is an iterative algorithm to reach local maxima in multivariate density function.

**Gibbs sampling** Is an iterative algorithm to reach local maxima in multivariate density function by sampling in a constrained subspace.

**Hidden Markov Model** Is an extension of The Markov Model, popular in many fields.

**Markov Model** Is a model for stochastic processes.

**Pomegranate** is a python library that includes an implementation of HMMs using EM.

# 1 Introduction

## 1.1 Time Series

Let $(\Omega, \mathcal{F}, P)$ be a probability space, and let T be an index set. A *time series* (or a *stochastic process*) is a function $Y(t, \omega)$ defined on $T \times \Omega$ such that for each fixed $t$, $Y(t, \omega)$ is a random variable on $(\Omega, \mathcal{F}, P)$. The function $Y(t, \omega)$ is often written as $Y_t(\omega)$. For a fixed $\omega$, $Y_t(\omega)$ is a deterministic function called a sample function [2].
In simpler terms, a time series is a sequence of data points $y$ each recorded at a specified time $t$. Each data point in the sequence is called an *observation* and can be constituted by any number of tracked parameters, labeled *features* [1]. In this paper the observation set $\{y\}$ will be referred to as the observation sequence, while the underlying process will be called time series. Note that these definitions are not universal.
Both the observation sequence and the time-series can be discrete or continuous. Due to the nature of digital processing, in reality, most observation sequences (which are the sample function of the stochastic process) are discrete. For convenience, we usually strive to analyze the time series devoid of the timestamps. This can be done in multiple ways and is highly dependent on the nature of the individual time series.

## 1.2 The Markov Assumption

Since a time series is a stochastic process, one can deduce all of the desired statistics of the process from the *joint distribution function*. For a finite set of random variables $\{Y_{t_1}, Y_{t_2}, \ldots, Y_{t_n}\}$ from the collection $\{Y_t : t \in T\}$, the joint distribution function is defined by

$$F_{Y_{t_1}, Y_{t_2}, \ldots, Y_{t_n}}(y_{t_1}, y_{t_2}, \ldots, y_{t_n}) = P(Y_{t_1} < y_{t_1}, Y_{t_2} < y_{t_2}, \ldots, Y_{t_n} < y_{t_n})$$

If we assume that the system has identifiable states, meaning for every $t$, $Y_t \in \{S\}$ where $\{S\}$ is a countable set, then we can instead look at the *joint probability density function* defined by

$$f_{Y_{t_1}, Y_{t_2}, \ldots, Y_{t_n}}(y_{t_1}, y_{t_2}, \ldots, y_{t_n}) = P(Y_{t_1} = y_{t_1}, Y_{t_2} = y_{t_2}, \ldots, Y_{t_n} = y_{t_n})$$

Where $y_{t_i} \in \{S\}$. Notice that we can manufacture $\{S\}$ by quantizing a continuous range. Then we can look at this joint probability as the product of $n$ independent probabilities:

$$P(Y_{t_1} = y_{t_1}, Y_{t_2} = y_{t_2}, \ldots, Y_{t_n} = y_{t_n}) = \left( \prod_{i=2}^{n} P(Y_{t_n} | Y_{t_{n-1}}, \ldots, Y_{t_1}) \right) P(Y_{t_1})$$

We dropped the equal sign notation for readability. From the above equation, we can see that the entire probability of the process is comprised by the starting probability $P(Y_{t_1})$, and the probability at time $t_n$ based on the states of the system at times $t_1, \ldots, t_{(n-1)}$.

It is useful to use the following equation with an ordered set of indices $t_1 < t_2 < \ldots < t_n$ since this implies causality in the system.
If we are to use an ordered set of indices, we get a practical way to predict the next state, where we know all $P(Y_{t_n}|Y_{t_{n-1}}, \ldots, Y_{t_1})$, since we have the trajectory of all other $Y_{t_i}$ already recorded. However, we notice that for long sequences, we get dependencies on a vastly increasing number of random variables, making the computation infeasible.
The Markov assumption is one of the most prominent methods to deal with this problem. The Markov assumption is an approximation on the dependencies on older states of the system. The $n$-the order Markov approximation is defined by:

$$P(Y_t|Y_{t-1}, \ldots, Y_1) \approx P(Y_t|Y_{t-1}, \ldots, Y_{t-n})$$

Notice that here $t \in \mathbb{N}$, since it is common practice to use regular intervals between samples, which allows us to discretize the time indices.
In essence, this approximation is an assumption on the direct dependencies of far-in-time states. A good example for why this approximation is appropriate for many time series is stock values. It is common sense that the value of a stock today will be very close to its value yesterday, thus it will be highly dependent on yesterday's value. Likewise, we can infer that it will have little to no connection to its value last year. This locality in time occurs in a lot of natural and artificial processes and allows us to use the Markov assumption.
While it is the most limited and least accurate, we usually use the first-order Markov approximation due to the simplicity of the model it gives us, as will be described in the next section. In particular, we are going to assume:

$$P(Y_t|Y_{t-1}, \ldots, Y_1) = P(Y_t|Y_{t-1})$$

We shall use the terms Markov assumption and Markov first-order approximation interchangeably [3]. Much had been discussed over the years about the asymmetry of the Markov assumption and the causality it implies [7], and its merit had sparked philosophical debate. Practically, it is a good approximation to some stochastic processes while being a bad one for others. Deciding on its use in a particular time-series often comes down to evaluating the predictive results it gives us compared to the real time-series.

## 1.3 Markov Chain

Given a discrete set of states $\{S\}_1^M$, a *Markov chain* is a discrete stochastic process $\{Y_n\}$ where:

$$P(Y_n = S_j) = \sum_{i=1}^{M} P(Y_n = S_j|Y_{n-1} = S_i)P(Y_{n-1} = S_i), \forall n > 1$$

Where $M$ is the size of $S$, and can be infinite (countable). The probability of the next state depending on the present state is the same regardless of the time index, meaning:

$$P(Y_n = S_j | Y_{n-1} = S_i) = P(Y_l = S_j | Y_{l-1} = S_i), \forall n, l \in \mathbb{N}$$

We can arrange all such probabilities in a matrix $T$, such that:

$$T_{ij} = P(Y_n = S_j | Y_{n-1} = S_i)$$

Notice that $T$ is of size $M \times M$. We also define $v_n$ as a row vector of size $M$ such that:

$$(v_n)_i = P(Y_n = S_i)$$

From the definitions of $T$, $v$, and the definition of the Markov chain, we get:

$$v_n = v_{n-1}T = (v_{n-2}T)T = v_1 T^{n-1}$$

Which means we reduced the entirety of the problem to a transition matrix $T$ and a vector of starting probabilities $v_1$ [4].
We note that the elements of $v_n$ must sum to 1, because they represent probabilities of the entire probability space ($Y_n \in \{S\}$). Similarly, each row of $T$ must sum to 1. It is also important to mention that each Markov chain has at least one invariant distribution, or steady state, which is a vector $v$ where $vT = v$, and thus if at a certain stage we get $v_n = v$, then all subsequent $v_m, m > n$ will also equal to $v$.
Markov chain analysis usually ties into time-series forecasting, in which we try to forecast the next state each time. For such use, one can see that it is enough to know $T$, since a start state can be measured rather than using the start probabilities.
One can see that we can model a time series that upholds the Markov approximation using a Markov chain. The only other assumption that needs to be made is the singularity of $T$, meaning:

$$P(Y_n = S_j | Y_{n-1} = S_i) = P(Y_l = S_j | Y_{l-1} = S_i), \forall n, l \in \mathbb{N}$$

This can be enforced by renaming any states that do not uphold this, creating new states. Since $M_{\text{new}} < N + M_{\text{old}}$, we get that $S$ is still a countable set. Though this method is viable, it is usually not required since we assume that we internalize outside effects into the inherent states, creating distinct characteristics for each state of the system, and giving them real-life meaning.

### 1.3.1 Temporal Data in a Markov Chain

Since together $T, v_1$ hold all the necessary information to analyze the time-series, and $v_1$ relates only to the first state in a sequence, it follows that $T$ holds all the temporal data of the process. We know that each Markov chain has at least one steady state, thus

for each Markov chain and for all $\epsilon > 0$ there exist $N$ such that for all $N < n \in \mathbb{N}$ the following holds:

$$\left\|T^n - T^N\right\|_1 < \epsilon$$

We consider transition matrices with higher temporal data to be cases where the minimal $N$ that satisfies the previous condition is large. This definition is not mathematical, hence we can look at a temporal index of our choosing to quantify this phenomenon. In this paper, we consider the *Temporal Index, T.I,* to be:

$$T.I = \frac{\sum_{\lambda<1} \lambda}{\sum \lambda}$$

Where $\lambda$ are the eigenvalues of the transition matrix. If we look at the decomposition of the transition matrix:

$$T = PDP^{-1}$$

where $D$ is diagonal and contains the eigenvalues, we then get that:

$$T^N = PDP^{-1} \cdot \ldots \cdot PDP^{-1} = PD^N P^{-1}$$

In order for $N$ to satisfy the previous condition we get:

$$P^{-1}D^{N+1}P = P^{-1}D^N P$$

We also know that $D_{ii}^l = \lambda_i^l$, hence we get that for all eigenvalues of T, from some power $N$ they must remain the same when multiplied by themselves. $T$ is a stochastic matrix, thus $\lambda_i \leq 1$ . For all eigenvalues that equals to one, the condition holds for all $N$. For the others $N$ must be larger than 1 in order for $\lambda_i^N = 0$ . The higher $\lambda_i$ is while not being equal to 1, the larger $N$ must be to uphold the condition. $T.I$ that we chose is one way to measure that.

## 1.4 Hidden Markov Model

The Markov chain is a very simple model, and most of the time it is too simplistic to have value. For example, take a flashlight with three intensities as the system of choice. The Markov chain of the process is shown in figure 1, where the arrows mark the probability to transfer between states.

Where the percentage marks the light intensity in each state. Now we want to measure this process and say for example we do so with a light sensor directed at the flashlight, while we darken the room as much as we can. The timer of the flashlight between state is set, and we know it, allowing us to measure in discrete time.

In an ideal world, the sensor reading would be exactly as the intensity described above, but in the real world we have inaccuracies of equipment and so we get an error between the two. Naively, from the last paragraph of the last section, we can create a state for

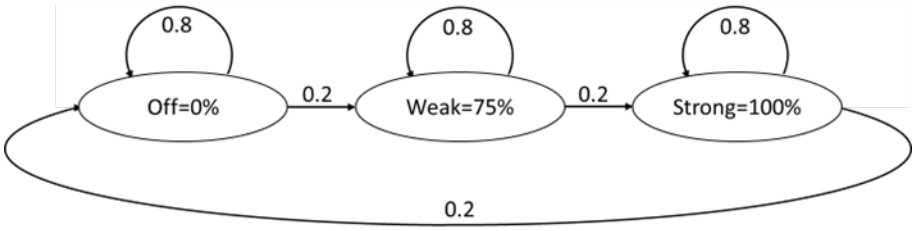



**Figure 1:** An example of Markov chain.

each reading of the sensor. This of course would defeat the purpose of understanding the underlying process, since we will lose the information of the Markov chain.
For a small enough maximum error, namely lower than 12.5%, we can use quantization in order to categorize the readings to the three states and still retain the original chain. However, for larger maximum error, we get an inherent mixing of the weak and strong states. For this case and many other, the hidden Markov Model was invented. Consider a time series $Y_t$ and consider a Markov Chain $X_t$ over the states-space $\{S\}$, a *Hidden Markov Model (HMM)* is a model in which the following relation holds [6]:

$$f_{Y_t}(O_t) = \sum_{i=1}^{M} P(O_t|X_t = S_i)P(X_t = S_i)$$

where $O_t$ is the observation at time $t$. The observation space $\{O\}$ does not need to be a countable set. Back to our previous example, we can look at the time series of the light intensity in the room as a HMM where the Markov chain in fig 1 is $X_t$, then for each $i$ there is a distribution $f_{Y_t|X_t}(o|S_i)$ that describes the distribution of observations given the system is at a known state. If $\{O\}$ is a countable set, these distribution functions (one for each $S_i$ ) can be arranged in a matrix $E$ called the *Emission Matrix*, where

$$E_{ij} = P(Y_t = O_j|X_t = S_i)$$

For convenience, the set of distributions is called the emission matrix even if $\{O\}$ is not countable. It is important to notice that like the transition matrix, the emission matrix must be Unique for all values of $t$, meaning

$$P(Y_n = O_j|X_n = S_i) = P(Y_l = O_j|X_l = S_i) \quad \forall n, l \in \mathbb{Z}$$

The distributions of the emission matrix can be of the same type (e.g Gaussian) or of different type. This lends the HMM a much better expressiveness than the Markov Chain, and thus it is applicable to more time series.
Using the HMM model, we need to find only $v_1, T$ of the Markov chain $X_t$ and the emission matrix $E$ in order to describe the time series and its underlying mechanism. As is the case with the Markov chain model, in the HMM we also are most interested in the transition matrix, since it describes the inherent states of the process.

The algorithms used to solved the HMM problem are data-based, meaning that they fit the best variables that describe a dataset of sentences that are created by the same HMM. A *sentence* in this case is a sequence of observations that are derived from a HMM model. The start and end of the sentence can be arbitrary (derived from the recording time) or a part of the model (start-of-sentence and end-of-sentence being states of the system themselves).
It is important to note that even if the emission matrix $E$ is known, the model does not collapse to a Markov chain, since mixing between states can still occur. In our example, even if we know the emission distributions of the weak and strong states, it is still not a certainty where does the observation of 87.5% comes from. Hence, the learning process of the transition matrix $T$ is the same whether or not $E$ is known. Giving the algorithms the true emission matrix is a way to cut down the convergence time.

## 1.5 Omissions

When dealing with time series we face the problem of omissions in the data. An *omission* is a missing observation in a sentence or recording. These omissions can be a result of various phenomena and can be divided to three major categories, that we shall shortly describe.
The omissions are a major hurdle to HMM algorithms, since they force us to consider a case where two successive observations $O_t$ and $O_{t+1}$, come in fact $O_s$ and $O_{s+2}$ where $O_{s+1}$ was omitted and is unknown. As mentioned before, the algorithms try to fit the variables to the observation, and this change needs to be taken into account [5].

### 1.5.1 Deterministic Omission Processes

The first type of omission is the deterministic one. these type of omissions are caused by a deterministic and usually well known reason. Let us look at an example for this type of omission. For the sake of argument, assume a worker tracks the temperature of his office each morning. On the weekend the office still has a temperature to be measured, but no worker to mark it down. In this case, the observations on the weekend are omitted from the time series. Since it is always on the weekend, it is deterministic when the omission happen.

### 1.5.2 Random Omission Processes

Continuing with our example, let us say that each day there is a chance that the worker will miss home due to important errands for the house. In such days, there will also be an omission in the data. If we assume that this probability is the same per each day, than we call this type of omission a *Random Omission*. In essence we can look at the omission process as Bernoulli Experiment to determine whether there is an observation

or omissions.

### 1.5.3 Time-Depended Omission Processes

Counter-intuitively, a *Time-Dependent Omission* is also random, but where the chance for omissions in random omissions is always the same, the chance for omission in time-dependent omission is, as the name suggests, time-dependent [9, 10].
Back to our example, let us say that the worker has two states- healthy and sick. If he is healthy is most likely to stay health, and if he is sick he is likely to stay sick for a while. This two-state systems can be described by itself as a Markov chain. On sick days, the workers takes a sick-leave and thus in our main time-series (the temperature of his office) there is an omission. This gives us an omission process that is a Markov chain which is time-dependent.
This type of omissions is hardest to counter and factor for due to the vast variety of processes that fit into this category, and the vastly different nature each of those posses. In this paper we shall look at the performances of known algorithms for the two types of omission of interest(random and time-dependent).

### 1.5.4 Factoring HMM for Omissions

Omissions in data can appear in two ways that are categorized as known and unknown omissions. These terms do not refer to the observation that was omitted from the sentence, since knowing it is not having an omission, but rather to the knowledge of an omission happening.
For this discussion we shall use the Markov chain model, since the emission matrix unnecessarily clutters the math from the underlying phenomenon. Consider a known omission, meaning that we know $X_t = S_i, X_{t-1} = ?, X_{t-2} = S_k$, then from the Markov model we get:

$$P(X_t = S_i) = P(X_t = S_i|X_{t-1} = S_j)P(X_{t-1} = S_j) = T_{ji}P(X_{t-1} = S_j)$$

Since we do not know $X_{t-1}$ we would like to look at the last state that we know:

$$P(X_t = S_i) = P(X_{t-1} = S_j|X_{t-2} = S_k)T_{ji}P(X_{t-2} = S_k) = T_{kj}T_{ji}P(X_{t-2} = S_k)$$

This expression still looks at a known $X_{t-1} = S_j$, to solve this, the naive approach is the Bayesian one:

$$P(X_t = S_i) = \sum_j T_{kj}T_{ji}P(X_{t-2} = S_k) = (T^2)_{ki}P(X_{t-2} = S_k)$$

Similarly, this can be done to any number of consecutive omissions, for the general case of $n$ omissions we get:

$$P(X_t = S_i) = (T^n)_{ji}P(X_{t-n} = S_j)$$

This method can be inserted into the algorithms in order to provide a score to a possible $T$ matrix, and fit the data with known observations.
In the unknown case this gets more complicated, because one has to first understand that there is an omission and only then factor for it. This is beyond the scope of this paper, and shall not be needed since we shall use only the known omission case, because we want to prove that under certain omission processes, the algorithms struggle even in the this simpler case.
Unknown omission is a fear in all real-life data, and thus we avoid it by using only computer generated one that fits the HMM model. This simplification shows that the phenomenon we're after lies at the HMM model itself rather than the fitness of the real data to the model (for most real data, the model is just an approximation based on Markov's assumption).

### 1.6 The Importance of Consecutive Observations

In this paper, *consecutive observations* are two successive observations that are left in the sentence, i.e neither of them are omitted. We define a *jump* of length $n$ as $n$ consecutive omissions. Given an infinite dataset and perfect reconstruction algorithms, one can independently learn $T^{\{J\}+1}$ where $\{J\}$ is the set of all jump lengths present in the data. If $0 \notin \{J\}$ it means that in the whole dataset we do not have a single case of consecutive observations. In this case, $T$ itself can't be learned, just powers of it. This present a difficult problem, since we know matrix multiplication is not *unique*.

## 2 Our Purpose

In this paper, we aim to expose a problem with all current algorithms for HMMs. We suggest that not having pairs of concurrent data points in the database breaks the existing algorithms. This predicament prevents sampling directly from $T$, and force any algorithm to infer it from its existing powers. The algorithms in use are not built to handle this specific, problem altering case. While it was proven in [reference] that HMM with random omissions is a convex problem, we suggest that the HMM-no-concurrent-observations problem is not. We shall not prove it in this paper (out of scope), but shall provide evidence in the form of non-converging algorithms that were proven to converge on the convex case. We consider the algorithm broken if it is as close to the solution (in a metric we discuss below) as a random matrix does.
We shall showcase the algorithms' brittleness by comparing their result on multiple omission processes, providing evidence that it is not a problem with the amount of data left in the database after omission. We also believe that the problem we suggest will we worse for HMMs with higher $T.I$, and shall explore it by experimenting with different HMMs that were engineered to have a specific $T.I$.

# 3 Methods

The following section details our experiment's parameters and methods. The code was written in python and is available at our git repository: `https://github.com/itaygeva/HMMOP-no-concurrent-observations`

## 3.1 Algorithms

In this project, we wanted to showcase an inherent problem to the HMM model, regardless of its solving algorithm. For that purpose, we ran our experiments on two algorithms. The first is the EM algorithm, which is the staple in the HMM field. We used the pomegranate package for that [8]. The second algorithm is a novel algorithm based on Gibbs sampling [5]. We used the package in the article for this algorithm, which is called HMMOP.

## 3.2 Omission Processes

In order to show that the algorithms break due to consecutive observations being missing, rather than the amount of missing observations, we looked at several omission processes. In this section we shall provide their mathematical notations and code implementations. For this section, we denote $Y_t$ as the HMM like before, but now we do not observe $Y_t$ directly, but rather the process after omissions, that we shall denote as $O_t$. We shall also refer to "omission process" as OP from now on.

### 3.2.1 Pass All Omission

The name of this OP is a misnomer, since no observation is being omitted. mathematically:

$$O_t = Y_t$$

This OP is used as a baseline for the algorithms, in order for us to evaluate their later performances.

### 3.2.2 Bernoulli Omissions

The simplest out of the omission processes, here we perform a Bernoulli test on whether or not a data point is being omitted. Mathematically:

$$O_t = \begin{cases} Y_t & \text{w.p } p \\ Null & \text{w.p } 1-p \end{cases}$$

Where $p$ is the probability to observing a data point. Our implementation of Bernoulli OP can be seen in listing 1.

```python
def bernoulli_experiments(p_prob_of_observation, all_full_sampled_trajs):
    """
    omits points along several trajectories using a binomial dist
    :param p_prob_of_observation: the binomial probabilty
    :param all_full_sampled_trajs: the trajectories to preform
        omissions upon
    :return: the omitted trajectories, and the omitted locations
    """
    all_relevant_observations_and_ws = []
    for vec in all_full_sampled_trajs:
        _new_vec = sample_n_points_from_traj_binom(vec,
            p_prob_of_observation)
        all_relevant_observations_and_ws.append(_new_vec)

    all_relevant_observations = [row[0] for row in
        all_relevant_observations_and_ws]
    all_ws = [row[1] for row in all_relevant_observations_and_ws]

    return all_relevant_observations, all_ws
```

**Listing 1:** Benoulli OP implementation

### 3.2.3 Markov Omission

Markov OP is an OP that is a result of a two-state Markov chain, where the states are "Observed" and "Omitted". mathematically:

$$O_t = \begin{cases} Y_t & \text{w.p } p_t \\ Null & \text{w.p } 1 - p_t \end{cases}$$

Where:

$$p_t = \begin{pmatrix} 0.5-\epsilon & 0.5+\epsilon \end{pmatrix} \begin{pmatrix} 0.5-\epsilon & 0.5+\epsilon \\ 0.5+\epsilon & 0.5-\epsilon \end{pmatrix}^{t-1}$$

We use this OP because it is very common in real life practices and thus of interest. Our implementation of this OP can be seen in Listing 2.

```python
def markov_chain_omission(epsilon, sentence):
    transmat = np.array([[0.5 - epsilon, 0.5 + epsilon], [0.5 - epsilon
        , 0.5 + epsilon]])
    state = np.random.choice([0, 1], p=[0.5 - epsilon, 0.5 + epsilon])
    w = []
    for i in range(len(sentence)):
        if state:
            w.append(i)
        state = np.random.choice([0, 1], p=transmat[state])
    omitted_sentence = sentence[w]

    if len(sentence) < 3:
        return np.array(sentence), np.arange(len(sentence))
    elif len(w) < 2:
        return markov_chain_omission(epsilon, sentence)
    else:
        if sentence.ndim == 1:
            return np.array(omitted_sentence), w
```

```
        else:
            return np.vstack(omitted_sentence), w
```

**Listing 2:** Markov OP implementation

### 3.2.4 Bernoulli of Jumps

This OP is in some regards the most important, in This, we perform a Bernoulli test for every sample that we do observe whether or not to omit the next sample. Mathematically we define a random vector $\{J.B\}$ of length the same as the sentences to undergo omission $Y_t$, where $J.B_t \sim 2 - Ber(p) \;\; \forall t > 1$ and $J.B_1 =\sim 1 - Ber(p)$. Then we define:

$$W_n = \sum_{m=1}^{n} J.B_m$$

We will forgo all $\{W_l | W_l > D\}$, where $D$ is the length of the sentence. Then:

$$O_t = \begin{cases} Y_t & t \in \{W\} \\ Null & t \notin \{W\} \end{cases}$$

One can see that for $p = 1$ we get the Pass All OP, and that for $p = 0$ we get a sentence where each second observation is omitted. Since we perform a Bernoulli test on the skip between observation, in this OP, $p$ gives us the ratio of consecutive to non consecutive observations. Our implementation of it can be seen in listing 3.

```
def consecutive_bernoulli_omission(p_prob_of_observation, sentence):
    steps = np.random.choice([1, 2], p=[p_prob_of_observation, 1 -
        p_prob_of_observation], size=len(sentence))
    steps[0] -= 1  # this is so we can start at 0
    indexes = np.cumsum(steps)  # this is to get the indexes

    w = indexes[indexes < len(sentence)]
    omitted_sentence = sentence[w]

    if len(sentence) < 3:
        return np.array(sentence), np.arange(len(sentence))
    elif len(w) < 2:
        return consecutive_bernoulli_omission(p_prob_of_observation,
   sentence)
    else:
        if sentence.ndim == 1:
            return np.array(omitted_sentence), w
        else:
            return np.vstack(omitted_sentence), w
```

**Listing 3:** Benoulli of Jumps OP implementation

### 3.2.5 Uniform Skips

In this OP we define $\{J\}_{t=1}^{D}$ where:

$$J_t = \begin{cases} 2 & \text{w.p } \frac{1}{3} \\ 3 & \text{w.p } \frac{1}{3} \\ 4 & \text{w.p } \frac{1}{3} \end{cases}$$

Similarly to the previous OP, we define:

$$W_n = \sum_{m=1}^{n} J_m$$

We will forgo all $\{W_l | W_l > D\}$, where $D$ is the length of the sentence. Then:

$$O_t = \begin{cases} Y_t & t \in \{W\} \\ Null & t \notin \{W\} \end{cases}$$

Here we get a more varied OP where non of the observations are consecutive. An implementation of this OP can be seen in listing 4.

```
def uniform_skips_omission(num_of_skips, sentence):
    skips = np.random.randint(low=1, high=num_of_skips + 1, size=
        sentence.shape)
    skips[0] -= 2  # this is so we can start at 0
    skips += np.ones_like(skips)
    indexes = np.cumsum(skips)  # this is to get the indexes

    w = indexes[indexes < len(sentence)]
    omitted_sentence = sentence[w]

    if len(sentence) < 3:
        return np.array(sentence), np.arange(len(sentence))
    elif len(w) < 2:
        return geometric_omission(num_of_skips, sentence)
    else:
        if sentence.ndim == 1:
            return np.array(omitted_sentence), w
        else:
            return np.vstack(omitted_sentence), w
```

**Listing 4:** Uniform Skips OP implementation

## 3.3 Transition Matrices

Since we are interested in the effects of the temporal data on the performance of the algorithms while the omission processes are present, we had to choose our transition matrices with care and calculate their $T.I.$ This subsection details the mathematics behind our choices and includes our implementations of each matrix. We shall hence refer to the difference in the matrices as *Trsansition Matrix Mode (TMM)*. It is important to note that we are using a - 10 states HMM throughout this paper.

### 3.3.1 Random Transition Matrix

The first transition matrix that we chose is the simplest - populating a $10 \times 10$ matrix with random samples from a uniform distribution over $[0, 1)$. mathematically:

$$\tilde{T}_{ij} \sim U([0, 1))$$

and in code:

```
self._transition_mat = np.random.rand(self._config.n_components, self._config.n_components)
```

We then normalize it to so it will be stochastic, namely:

$$T_{ij} = \frac{\tilde{T}_{ij}}{\sum_k \tilde{T}_{ik}}$$

The normalization process occurs for all transition matrices, and its implementation can be seen in listing 5.

```
def _normalize_stochastic_matrix(self, matrix):
    for row in matrix:
        row /= np.sum(row)
    return matrix
```

**Listing 5:** Stochastic transition matrix normalization implementation

A random transition matrix has a $T.I$ of about 0.5.

### 3.3.2 "Nearest Neighbors" Transition Matrix

Given the $T.I$ of a random transition matrix, it is important to find matrices with higher $T.I$, since lower values can be achieved by powering the matrices, as we shall explain later. One such matrix that achieves extremely high $T.I$ is the *nearest neighbors* matrix, in which:

$$\tilde{T}_{ij} = \exp\{|i-j|\}$$

This transition matrix gives us $T.I$ of roughly 0.8. Our code implementation of it is presented in listings 6.

```
def generate_near_biased_matrix(self):
        rows = np.arange(self._config.n_components)
        # Create a 2D array with repeated rows to represent the column indices
        columns = np.tile(rows, (self._config.n_components, 1))
        # Calculate the absolute differences element-wise
        distance_matrix = -1 * np.abs(rows - columns.T)
        stochastic_matrix = np.exp(distance_matrix)

        return self._normalize_stochastic_matrix(stochastic_matrix)
```

**Listing 6:** Nearest neighbors transition matrix implementation

### 3.3.3 "N Edges" Transition Matrix

Another process with high $T.I$ is the N-Edges process. In it, in each state the process can transition only to $N$ states, where $N < M$ which is the number of states. mathematically, we define $A_i$ for each row as set of size $N$ where each element of the set is drawn sequentially from the set $1, \ldots, M$ at random without return. Then:

$$\tilde{T}_{ij} \sim \begin{cases} U([0,1)) & j \in A_i \\ 0 & j \notin A_i \end{cases}$$

Note that $\{A_i\}_1^M$ are *i.i.d.* In this paper we used $N = 3$. Our implementation of it is shown in listing 7.

```
def generate_n_edges_matrix(self, n_edges):
      stochastic_matrix = np.zeros((self._config.n_components,
          self._config.n_components))
      for row in stochastic_matrix:
          edges_idx = np.random.choice(self._config.n_components,
              size(n_edges), replace=False)
          edges_values = np.random.random(n_edges)
          row[edges_idx] = edges_values

      return self._normalize_stochastic_matrix(stochastic_matrix)
```

**Listing 7:** N Edges transition matrix implementation

### 3.3.4 Matrix Powers

For an eigenvector $u$ with eigenvalue $\lambda$ of a matrix $T$, we get:

$$uT^n = \lambda \cdot uT^{n-1} = \ldots = \lambda^n u$$

Thus, $\lambda^n$ is an eigenvalue of $T^n$. Recall our construction of $T.I$, we can easily see that:

$$\lambda^{n \cdot N} < \lambda^N$$

It can be thus proven that this dictates that:

$$T.I\{T^n\} < T.I\{T\}$$

Therefore, a convenient way to compare between transition matrices is use a base matrix $T_1$ and its powers. Practically, in this paper we used the three base matrices described above. For random matrix we got that $T.I\{T^2\} \approx 0.1$, so we used only three powers. For the NN matrix, we chose the powers such that the $T.I$ would be at equal intervals, these powers were $[1, 3, 6, 15, 100]$. For the N-Edges case, we used [1, 2, 3, 4, 100].

## 3.4 Data Normalization

During our experiment it became apparent that we unintentionally introduced another free variable into our setup - the size of the data. initially we started with the same sentence length for each omission pipeline, as described below, and omitting observation from it. In this method, the number of observed data points is dependent on the probability in the omission process. Since the algorithms are data-based, the amount of input data they receive is critical to their performance.
After consulting with our supervisor, we decided that the best way to counter this problem is to perform data normalization. For each omission process we calculate the probability of observation prior to creating the sentences for the model, and then adjust their length

such that after omission the size of the observed data points will match the initial sentence length as described below. This equality is probabilistic.
The drawback of this method is that for lower probabilities the sentences ,including omissions, are very long, to the point that we had to forgo this process for Markov omissions and forgo the results for very low Bernoulli probabilities.

## 3.5 Wellness Metric- L1

For us to evaluate the performance of the algorithms we must have a scale to rate each run. A standard in the industry is to use some kind of norm to do so, since norms express the distance of two variables over some norm-space. In the field of HMM, the L1 metric is usually taken as the chosen metric. The mathematical expression for L1 for a matrix $X$ is:

$$\|X\|_1 = \sum_{i=1}^{M} \sum_{j=1}^{M} |X_{ij}|$$

The L1 error is calculated between the ground truth (GT), which is $T$ and the algorithm solution $\hat{T}$:

$$\left\|T - \hat{T}\right\|_1 = \sum_{i=1}^{M} \sum_{j=1}^{M} \left|T_{ij} - \hat{T}_{ij}\right|$$

There are two important normalizations that we have to make. The first is dividing by the number of states $M$. This is done in order to be able to compare results on different sizes of transition matrices. The second is dividing by the average L1 error between the GT matrix and a random matrix. We do this in order to know how the algorithms' results are compared to a random matrix.

## 3.6 Other Parameters

In order to run the tests there are many variables that need to be set. Over the course of the project we ran countless tests in order to determine a set of variables that will lead to the most accurate results, while enabling us to run a vast number of tests both in variety and for statistical significance.
Our biggest hurdle in that regard was the fact that the HMMOP algorithm is very slow and optimised for GPU. This led us to carefully choosing variables that run the fastest while still having statistical significance and convergence. Here's an itemized list of all of these parameters:

- Number of States: 10.
- State Distributions: $S_i \sim \mathcal{N}(10 \cdot i, 0.3)$
- Freeze Distributions = True (Pome).
- Number of words in a sentence: 100.
- Number of sentences: 200.

- Number of iterations: 30.
- Number of runs per configuration: 5[1].

### 3.6.1 Do the Algorithms Converge?

The algorithms we used are iterative, and take the number of iteration as a user-decided input. This means that their estimator of the transition matrix is dependent on the iterations number, k. The algorithms do converge at infinity, meaning that there exists $\hat{T}$ such that:

$$\lim_{k\to\infty} \hat{T}_k = \hat{T}$$

When we conduct our evaluation of the L1 error, we must take into consideration that we calculate it with $\hat{T}_k$ and not $\hat{T}$. We can see that:

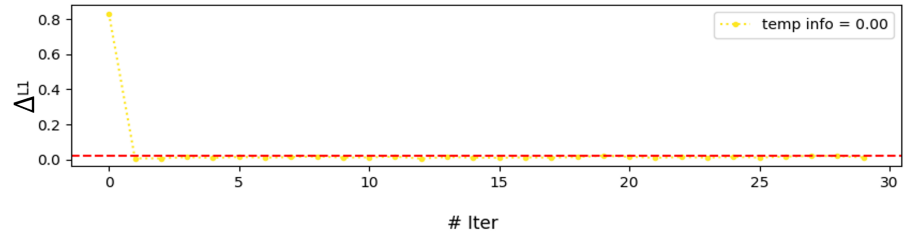
$$\left\|T - \hat{T}\right\|_1 = \left\|T - \hat{T}_k + \hat{T}_k - \hat{T}\right\|_1 \leq \left\|T - \hat{T}_k\right\|_1 + \left\|\hat{T}_k - \hat{T}\right\|_1$$

Which means we need to minimize the convergence error $\left\|\hat{T}_k - \hat{T}\right\|_1$ in order to regard the algorithm output as a solution, and comparing between different scenarios.
Algorithm convergence continues to be an open research question, and has no concrete definition. In our case, we look at the L1 convergence error, as per the equation above. When looking at the plot of the convergence error, it is hard to determine a hard line where the curve flattens, and the algorithm converges. Recall that $\hat{T}$ is not known, thus we cannot compute the error directly.
It is customary in such case to look at the derivative of the error, and determining when it is flattening using a moving average. We used this method with 5 elements in the moving average. We also decided on a threshold of 0.02 as the threshold for convergence. Mathematically:

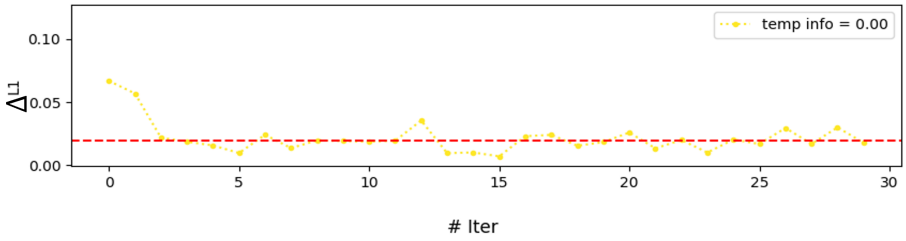
$$\text{Converged}_n = \mathbb{I}\left(\frac{1}{5}\sum_{i=n-4}^{i=n}\left|\frac{dL_1}{d\text{iteration}_i} - \frac{dL_1}{d\text{iteration}_{i-1}}\right| < 0.02\right)$$

Where $\mathbb{I}$ is the indicator function. We assume that the converged series is a step function, and define convergence at the first iteration where it is 1.
Unfortunately, we sometimes had to forgo this condition, seeing as the L1 difference vector would fluctuate ad infinitum. Usually this would happen in cases where the model had limited data. In these cases, we could usually see that there was an initial improvement in the L1 value, followed by slight fluctuations (which were larger than 0.02). We can see an example of this in figure 2b . After running a few of our configuration on 200 iterations, and seeing that the flattening that can be seen in the naked eye on the graph remains for all successive iterations, and there is no sudden improvement after a long

[1]For the Uniform Skips OP runs we used 15 runs at the same configurations, since they varied more than the other runs.




**(a)** Instance of convergence as we defined it.




**(b)** Instance of fluctuations in improvement over iterations.

**Figure 2:** Convergence of different runs as a function of iterations.

number of iterations, we decided to use 30 iterations. This is also the way we determined the parameters at the above section.

# 4 Results

## 4.1 Baseline Tests- No Omissions

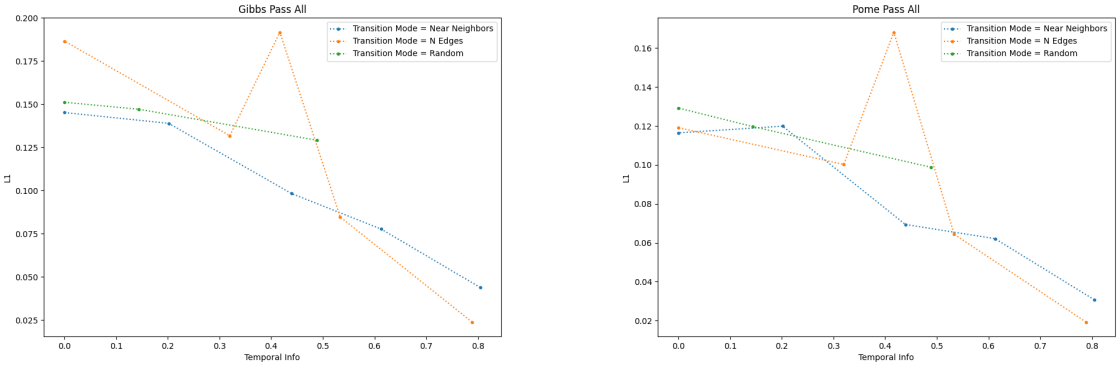
To verify that our pipeline flow works as it should, we have checked that in the case of no omissions the L1 value we obtain is sufficiently small. In Figure 3 we see that most of the tests have converged around or below 0.1, with a few above, but still below 0.2. Thus, we concluded that the pipeline works well.

## 4.2 Results per Omission Process

After running the full experiment, we analyzed the results. For each *TMM*, we looked at the L1 value, as a function of the non-omitting probability (1− omitting probability) for each omitter. Of course, we would expect a negative correlation between the L1 value and the non-omitting probability. In the following figures we can see that this assumption is substantiated. We shall analyze the results one by one.

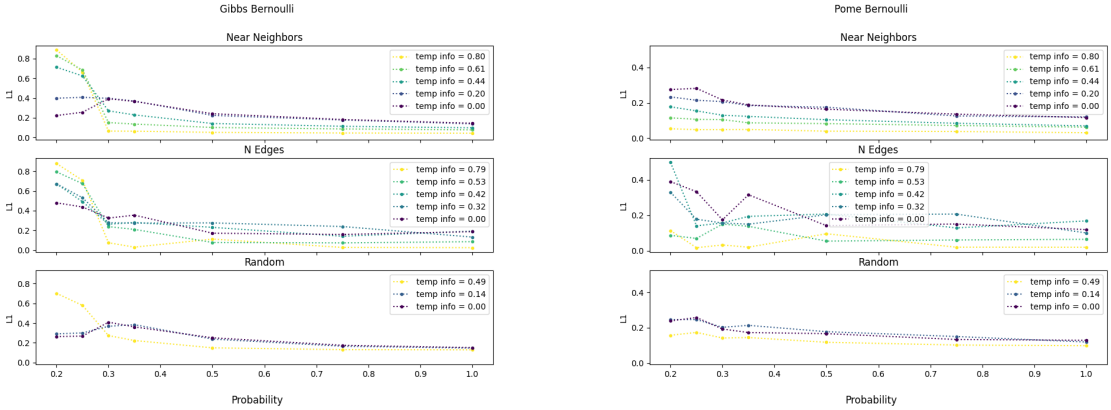
### 4.2.1 Bernoulli Omission Results

It seems that above a certain threshold of Bernoulli probabilities the algorithms manage to reconstruct the original transition matrix quite well. We can see in Figure 4 that for $p > 0.3$ both the Gibbs based algorithm and the EM based one achieved $L1 < 0.4$. In some instances, L1 is even below 0.2 (e.g. Pome Random) close to the Baseline scores.



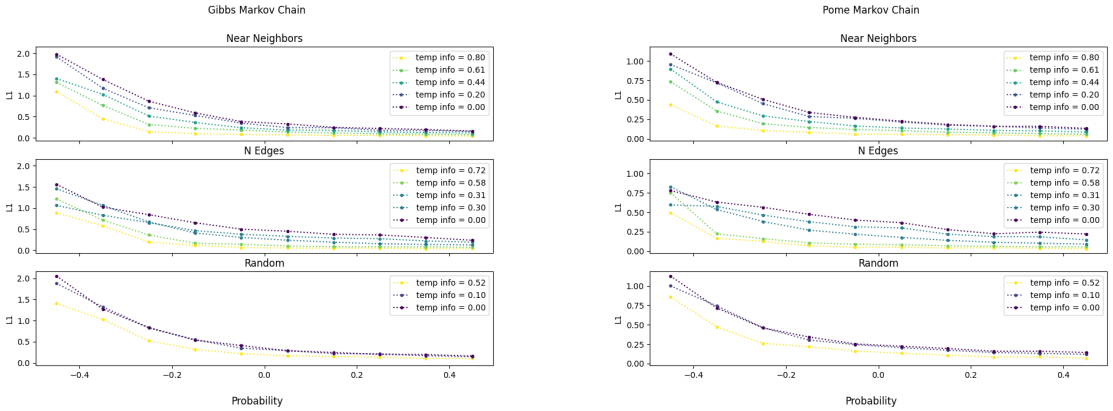

**(a)** Gibbs no omissions **(b)** EM no omissions

**Figure 3:** No omissions baseline of the algorithms. **(a)** Results of the Gibbs algorithm. **(b)** Results on the EM algorithm. Each graph contains 3 curves, one for each Transition Matrix. X axis is the $T.I$ achieved by powers of the transition matrix and the Y axis is the L1 metric we defined. The baseline is similar for each $T.I$ and each $TMM$.




**(a)** Gibbs Bernoulli omissions as a function of p. **(b)** EM Bernoulli omissions as a function of p.

**Figure 4:** Reconstruction results for Bernoulli omissions pipeline as a function of the transition matrix, the $T.I$ and the observation probability on each algorithm. **(a)** Results of the Gibbs algorithm. **(b)** Results on the EM algorithm. Each graph contains three subplots, one for each Transition Matrix. X axis is the observation probability and the Y axis is the L1 metric we defined. The different curves define different $T.I$ achieved by powers of the matrix. For $p > 0.3$ the algorithms score well, while for $p < 0.3$ the L1 is higher.




**(a)** Gibbs Markov Chain Omissions

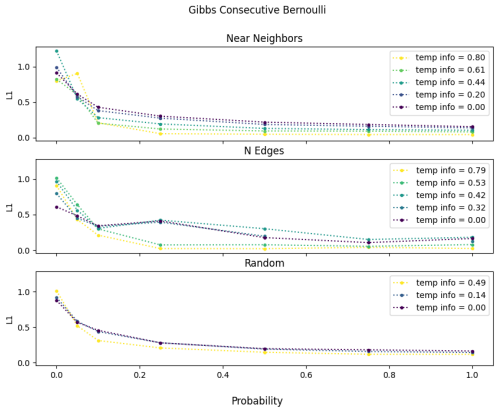
**(b)** EM Markov Chain Omissions

**Figure 5:** Reconstruction results for Markov Chain omissions pipeline as a function of the transition matrix, the $T.I$ and the observation probability on each algorithm. **(a)** Results of the Gibbs algorithm. **(b)** Results on the EM algorithm. Each graph contains three subplots, one for each Transition Matrix. X axis is the $\epsilon$ in the Markov Chain and the Y axis is the L1 metric we defined. The different curves define different $T.I$ achieved by powers of the matrix. The Markov omission have not undergone data normalization - for lower $\epsilon$ there is less data. The results look like a negative exponential.

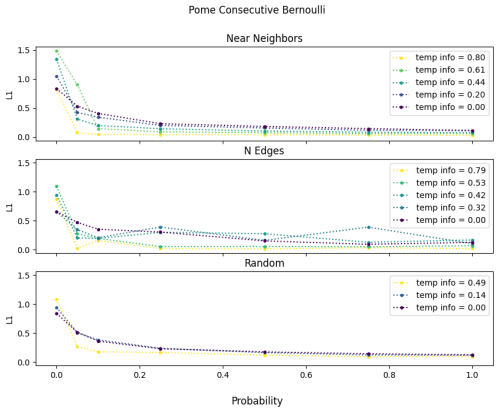
In addition, we can see that for $p > 0.3$ a clear trend occurs with regards to the $T.I$. In both algorithms, higher $T.I$ is correlated with lower L1 scores. We recall that this is in contrast to our hypothesis about the reconstruction quality with regards to the $T.I$.
However, when $p$ drops below 0.3, a change of behavior is displayed. This change is most apparent in the Gibbs based reconstruction, although it can also be seen in the EM based algorithm (N edges). Below this threshold, some of the reconstructions are markedly worse, with a sharp increase in the L1 score, up to $L1 = 0.8$. More importantly, for the Gibbs reconstruction the previously described trend flips, i.e. higher the $T.I$. correlates with high L1 score.

### 4.2.2 Markov Omission Results

In figure 5 we see the reconstruction results for the case using the Markov omission process. For lower $\epsilon$ values of the Markov omission process the reconstruction is, of course, worse. The lowest value for epsilon tested is $-0.45$, which corresponds to omission of 95% of data. This is a larger omitting percentage than any of the Bernoulli omission process. We can see that for small epsilons the L1 is up to 1 for the EM reconstruction, and up to 2 for Gibbs, as in Random or even worse than Random scores.
In addition, the counter intuitive trend that we observed in the case of the Bernoulli omission process, where high T.I. correlates to low L1 is continued here, and does not break for small probabilities.
Furthermore, it is interesting to note that as opposed to the Bernoulli omission process, the rise in L1 as the probability is lowered is quite gradual. This, and the previous




**(a)** Gibbs Bernoulli of skips




**(b)** EM Bernoulli of skips

**Figure 6:** Reconstruction results for Bernoulli of Skips ("Consecutive Bernoulli") omissions pipeline as a function of the transition matrix, the $T.I$ and the observation probability on each algorithm. **(a)** Results of the Gibbs algorithm. **(b)** Results on the EM algorithm. Each graph contains three subplots, one for each Transition Matrix. X axis is the Bernoulli probability and the Y axis is the L1 metric we defined. The different curves define different $T.I$ achieved by powers of the matrix. For $p > 0.15$ The results are comparable to baseline, but for $p = 0$ , which translates to 50% of the data, they achieve scores worse than random.

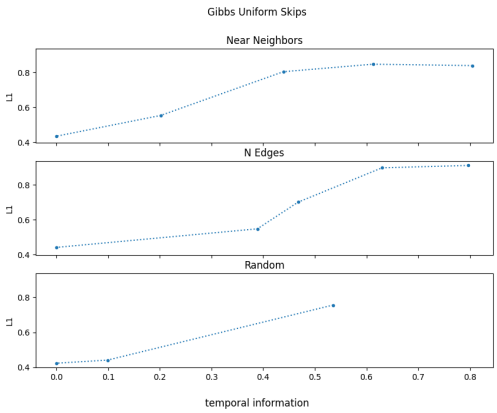
observation regarding the lack of trend inversion as seen in Bernoulli case, might be caused by lack of data normalization.

### 4.2.3 Bernoulli of Skips Omission Results

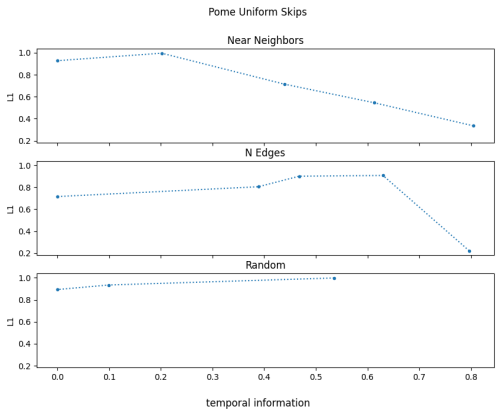
Figure 6, showing the results for the Consecutive Bernoulli omission process, appears to be quite similar to the Bernoulli omission process results. Again, we see the negative correlation between $T.I$ and L1, and again we see that over a certain probability threshold the results are quite close to the Baseline results, and below that threshold the L1 value jumps to 1.
However, in this case, that probability threshold is much lower, around 0.1. We recall that for $p = 0$ we still have 50 percent of the data, but no consecutive observations, thus it seems that only a small amount of consecutive data is needed in order to achieve results comparable to Baseline.

### 4.2.4 Uniform Skips Omission results

We should recall that the Uniform skips omission process does not receive a probability parameter. Therefore, the only discussion to be had about the reconstruction quality is with regards to the $T.I$. In graph 7 we can see opposing trends. For the Gibbs based reconstruction at figure 7a the higher the $T.I$ the higher the L1, as hypothesized. However, in figure 7b, it seems that this trend is not so conclusive for the pomegranate algorithm, and would even invert in some cases (e.g. Pome Near Neighbors).
This is a phenomenon that we have already observed in the Bernoulli omission results.




**(a)** Gibbs Uniform Skips




**(b)** EM Uniform Skips

**Figure 7:** Reconstruction results for Uniform Skips omissions pipeline as a function of the transition matrix and the $T.I$ on each algorithm. **(a)** Results of the Gibbs algorithm. **(b)** Results on the EM algorithm. Each graph contains three subplots, one for each Transition Matrix. X axis is the $T.I$ achieved by powers of the matrix and the Y axis is the L1 metric we defined. Opposite trends are observed between the algorithms. Gibbs results are better than EM's.

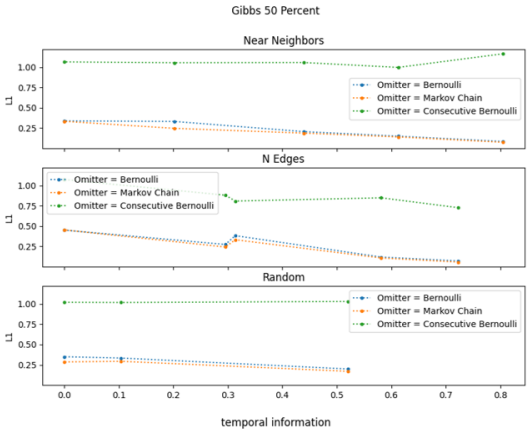
We recall that for the low observation probabilities, the Gibbs based algorithm followed our hypothesis, as in low $T.I$ leads to low L1, where as the EM based algorithm did not. It is important to note that this occurred only for low Bernoulli probabilities, where there aren't many concurrent observations. This is in agreement with the Uniform skips omission process, where there are no concurrent observations at all.
In addition, we can see that at its worse, the reconstruction is equivalent to a random matrix (i.e. $L1 = 1$) and at its best the reconstruction does not achieve Baseline quality reconstruction (the best is at around $L1 = 0.4$ excluding outliers). This is of course to be expected where we have no concurrent observations.

## 4.3 Comparison Between Different OPs - 50% of the data

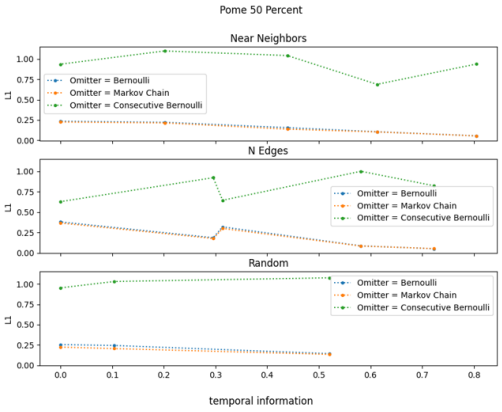
After displaying and analyzing the reconstruction results for each omitting process separately, we will now compare several different omitting processes – Bernoulli, Consecutive Bernoulli, and Markov. For each case we omitted 50% of the data, using the relevant process. It is important to note that the data was not normalized to the omission percentage, as Markov has failed to undergo data normalization.
In Figure 8 we see the comparison between the omission processes. There is a great gap between the Bernoulli and Markov processes, which achieve low L1 values

$$(< 0.5)$$

and the Consecutive Bernoulli process, which mostly fails at reconstruction. We recall that 50% omission in the case of the Consecutive Bernoulli omission process corresponds




**(a)** Gibbs 50% of the data




**(b)** EM 50% of the data

**Figure 8:** Reconstruction results for Different omission pipelines with 50% omitted data as a function of the transition matrix and the $T.I$ on each algorithm. **(a)** Results of the Gibbs algorithm. **(b)** Results on the EM algorithm. Each graph contains three subplots, one for each Transition Matrix. X axis is the $T.I$ achieved by powers of the matrix and the Y axis is the L1 metric we defined. The different curves define different omission processes. The results are prior to data normalization. Consecutive Bernoulli performs much worse, almost as random than the other two which are comparable.

to no consecutive observations. Thus, from this graph we can see the importance of consecutive observations in HMM data.

# 5 Discussion and Further Work

In this section we shall analyse the data in regards to our hypotheses, namely we shall explore the dependence of the reconstruction on concurrent observations, and on the temporal information of the process.

To that effect, we notice that the $p$ parameter in the Bernoulli of Skips OP acts as a scale between no-consecutive data ($p = 0$) and all consecutive data ($p = 1$), with a linear transition between them. Looking at the results of this graph lets us quantify the dependence of the reconstruction on the amount of consecutive observations. It can be seen that for no consecutive observation the algorithms perform worse than random, and that the decline back to baseline results is steep.

Recall that for $p = 0$ we see each second observation, meaning that we have half of the data left. This Led us to compare all the OPs with 50% of the data. this comparison has shown that the algorithms' failure is not caused by lack of data, but by lack of consecutive omissions, as both the Bernoulli and the Markov OPs achieve similar results, and the Consecutive Bernoulli results worse than random. This successfully demonstrates that the reconstruction algorithms fail in the case of non-consecutive observations.

For the Bernoulli OP, the results are only slightly worse when we omit more, until a certain threshold when especially the Gibbs algorithm gets marginally worse. It should

be noted that the normalization process happened very late in the process of the project, after deciding on 30 iteration per run and analyzing the convergence of the algorithms. We hypothesize that due to the normalization the lower probabilities, which have longer sentences overall, might not converge, and were they to converge, we would have seen almost no change in the results in small values of $p$ as this problem is convex and the algorithms are proven to converge to the solution.

It is important to not that even in the current state of ambiguous convergence, we get that very low probabilities of observation in Bernoulli omissions yield better results than Consecutive Bernoulli Omissions with $p = 0$ despite having substantially less data.

It should be noted that the algorithms perform decently when the data contains even low amounts of consecutive data. This can be seen in the sharp decline in the reconstruction quality when lowering the observation probability in the Bernoulli and Consecutive Bernoulli omission processes. We would expect this sharp decline to occur also in the Markov Chain omission process, however, the lack of data normalization could decrease the quality of reconstruction even for larger observation probability, which would conceal this sharp decline.

An interesting observation can be made when comparing no concurrent observations given by Consecutive Bernoulli Omissions with $p = 0$ and those given by the Uniform skips OP. We can see that Uniform skips perform Marginally better for all TMM and $T.I$. We can attribute this to the fact that in the case of Consecutive Bernoulli Omissions with $p = 0$ all the transitions we can infer from are even powers of the transition matrix $T$, while in the Uniform Skips case we have a product of $T^2, T^3, T^4, T^5$. We have stated before that finding $T$ from $T^2$ has no unique solution. The second problem is solvable for inversible matrices, and though the algorithms are not equipped for such task, apparently they can infer more from it naively.

In order to explore the lack of concurrent observations in Uniform Skips, we first have to detour to discuss the effect of temporal information in all our results, and in Uniform Skips in Particular. It appears that we can categorize the data into two categories in this regard; in cases where there is "enough" consecutive observations (above a certain threshold in Bernoulli[2], Consecutive Bernoulli), we get that the higher the temporal information of the ground truth transition matrix, the better the reconstruction. This is counter to our hypothesis.

We did not expect this trend to appear anywhere, and it is especially surprising to see in the case of "Uniform Skips" for EM based reconstruction, where there are no consecutive seen observations. As we have discussed, the stationary nature of low $T.I$ matrices would suggest that there would be no need for consecutive observations, as the distribution would have already reached the stationary distribution. However, these results indicate

[2]As the probability for observation lessen, so does the probability of consecutive observations which is $p^2$

otherwise. An understanding of this phenomenon requires research into the mechanisms of the algorithms themselves, and how they are affected by the temporal content of the transition matrices.
However, in the case of a lack of consecutive observations for the Gibbs based algorithm, we observed the opposite trend, which aligns with our hypothesis that for lower $T.I$ the reconstruction will improve. It is especially interesting to consider the Results of the Gibbs algorithm in the Uniform Skips OP.
Back to our discussion before, in this OP we are trying to estimate $T$ based on $T^2, T^3, T^4, T^5$. Recall from subsection 3.3.4 that the matrix powers for NN are [1,3,6,15,100], for N-Edges they are [1,2,3,4,5] and for Random they are [1,2,100]. As the $T.I$ decreases, we get that $T^2, T^3, T^4, T^5$ are more similar to $T$. when considering the powers, the $T.I$ they yield and Figure 7a we can infer that the rate of improvement as $T.I$ decreases aligns with the the similarity of $T^2, T^3, T^4, T^5$ to $T$, and thus the naive solution is also the real solution of $T$. In the case where this does not happen - in higher $T.I$, we can see that the problem of non consecutive observation still exists, and that in these cases the algorithms fail almost like random matrices.
This latest discovery leads us to believe that while we have confirmed our hypothesis in regards to non-consecutive data, further research should be made on OPs where there is no consecutive data, as we notice they do defer in "difficulty" meaning they represent inherently different mathematical problems.
In addition, one of the most important steps forward is to test this hypothesis on real world data.
Furthermore, we recommend that research be conducted into developing new algorithms which could learn the transition matrix in non-consecutive observation data. One possible path would be to learn the transition matrix powers using specific skips between observations and using those powers to reconstruct the original matrix.

# 6 Conclusions

Using different types of matrices (Near Neighbors, N Edges, Random) and their powers, we generated synthetic HMM data. This data, was either passed as is, or was partially omitted using different methods of omission. The omitted data was then passed to a model, either EM based or Gibbs sampling based, for fitting. The resulting reconstructed transition matrix was then compared to the ground truth matrix.
Running these tests with different matrices, and different omitters we have demonstrated that both the EM based model and the Gibbs sampling based model are unable to reconstruct the ground truth transition matrix, when not enough consecutive observations are seen. We have also shown that this result not the result of the amount of data used, but by the omission method. We have seen that there is inherently different behavior

between omission processes, even for ones that do not have any concurrent observations. Testing with different powers the matrices has allowed us to analyze the effect of the amount of temporal information of the matrix. We have seen that in almost all cases, the more temporal information there is, the better the reconstruction, contrary to intuition. We have found and explained that for lower amount of consecutive data in the Gibbs algorithm, the lowered the temporal information, the better the reconstruction, and explored it in the case of no concurrent observations at all.

We recommend further research, to confirm these results in further types of matrices, and especially with real-world data. Furthermore, additional research is needed to develop an algorithm that could reconstruct the transition matrix given non-consecutive observation data.

# References


[1] Peter J Brockwell and Richard A Davis. *Time series: theory and methods*. Springer science & business media, 1991.

[2] Wayne A Fuller. *Introduction to statistical time series*. John Wiley & Sons, 2009.

[3] Andrei Andreevich Markov. "The theory of algorithms". In: *Trudy Matematicheskogo Instituta Imeni VA Steklova* 42 (1954), pp. 3–375.

[4] James R Norris. *Markov chains*. 2. Cambridge university press, 1998.

[5] Binyamin Perets, Mark Kozdoba, and Shie Mannor. "Learning hidden Markov models when the locations of missing observations are unknown". In: *International Conference on Machine Learning*. PMLR. 2023, pp. 27642–27667.

[6] Lawrence Rabiner and Biinghwang Juang. "An introduction to hidden Markov models". In: *ieee assp magazine* 3.1 (1986), pp. 4–16.

[7] Richard Scheines. "An introduction to causal inference". In: (1997).

[8] Jacob Schreiber. "Pomegranate: fast and flexible probabilistic modeling in python". In: *Journal of Machine Learning Research* 18.164 (2018), pp. 1–6.

[9] Maarten Speekenbrink and Ingmar Visser. "Ignorable and non-ignorable missing data in hidden Markov models". In: *arXiv preprint arXiv:2109.02770* (2021).

[10] Hung-Wen Yeh, Wenyaw Chan, and Elaine Symanski. "Intermittent missing observations in discrete-time hidden markov models". In: *Communications in Statistics-Simulation and Computation* 41.2 (2012), pp. 167–181.