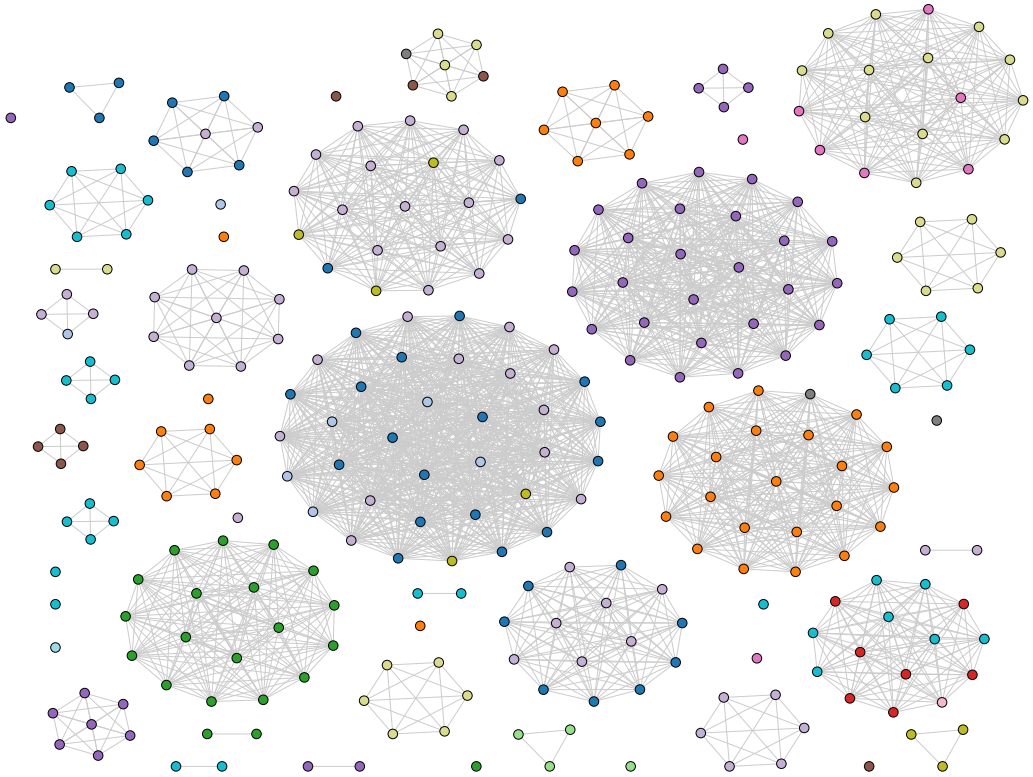


Clustering genomic signatures

A new distance measure for variable length Markov chains

Master's thesis in Computer science - algorithms, languages and logic

Joel Gustafsson
Erik Norlander

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2018

Master's thesis 2018

Clustering genomic signatures

A new distance measure for variable length Markov chains

JOEL GUSTAFSSON
ERIK NORLANDER

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2018

Clustering genomic signatures
A new distance measure for variable length Markov chains
JOEL GUSTAFSSON
ERIK NORLANDER

Supervisor:	Alexander Schliep Computer Science and Engineering
Advisors:	Peter Norberg & Martin Holmudden Institute of Biomedicine, Gothenburg University
Examiner:	Devdatt Dubhashi Computer Science and Engineering

Master's Thesis 2018
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Clustering of genomic signatures. The colours correspond to the taxonomic families of the signatures.

Gothenburg, Sweden 2018

Clustering genomic signatures
A new distance measure for variable length Markov chains
Joel Gustafsson
Erik Norlander
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

Pathogens such as bacteria and viruses are leading causes of disease worldwide, which makes it essential to identify them in DNA samples. Instead of analysing raw DNA sequences, mathematical models based on Variable Length Markov Chains (VLMCs), known as Genomic signatures, make it possible to classify DNA samples faster than with traditional alignment-based methods. To analyse a set of genomic signatures, we use clustering, which is an unsupervised machine-learning method. For the clustering of VLMCs, an accurate and fast similarity measure (distance function) is needed.

To analyse distance functions and clusters, we define metrics based primarily on the taxonomic ranks of the underlying organisms. For the distance functions, we primarily analysed whether the VLMCs within the same taxonomic rank were closest to each other. For the cluster analysis, we use the silhouette metric to determine how well separated the clusters are and define the average percentages, sensitivity, and specificity of the captured taxonomic ranks.

We present a new distance function for VLMCs, called Frobenius-intersection, which correlates accurately with the well-known Kullback-Liebler distance function, while also being several orders of magnitude faster. We use average-link clustering together with the Frobenius-intersection distance to cluster data sets of known viruses and bacteria with relatively short DNA sequences. The clusters of VLMCs correspond accurately to the Baltimore types of the viruses as well as the viruses' and bacteria's taxonomic families. However, most of the classifications of viruses are also subdivided into multiple clusters. Moreover, when combining the set of bacteria and viruses, the clusters start to mix the viruses and bacteria before finding all of the taxonomic families.

The clustering of the genomic signatures is accurate with respect to, for instance, taxonomic ordering. Therefore, it can help in identifying unclassified pathogens. Future research may reveal other causes of similarity between the genomic signatures.

Keywords: Computer science, Bioinformatics, Master's thesis, Markov chains, Variable length Markov chains, DNA clustering, Genomic signatures, Clustering, Machine learning, Unsupervised learning

Acknowledgements

We would like to express our gratitude towards our supervisor Alexander Schliep, for always pushing us in the right direction. We would also like to thank Peter Norberg and Martin Holmudden who have provided insight into the biological aspects of this work, and made their previous research and tools available to us.

Joel Gustafsson & Erik Norlander
Gothenburg, June 2018

Contents

1

Introduction

Pathogens such as fungi, bacteria, and viruses are leading causes of disease worldwide. The world health organisation published a report in 2014 [18] which shows an alarming rise in the amount of antibiotic resistant-bacteria, so-called superbugs, around the world. These superbugs make wounds, cuts, and diseases that have been treated effectively with antibiotics deadly once again. Moreover, viruses are also a considerable threat to human lives, take for instance, the recent Ebola outbreak in West Africa [28] which killed over 11 000 people. Accurate identification and classification of pathogens would make it possible to treat patients more efficiently and help make informed decisions with regards to public health. One typical approach to classification of pathogens is to analyse their genome.

The genome of an organism is the blueprint of the organism's cells. It is made up of DNA molecules which corresponds to the source code of the organism. DNA is copied to new cells and organisms, and sometimes errors referred to as mutations occur. These mutations gradually lead to evolution where many organisms share a significant portion of DNA. Organisms within the same taxonomic family (for instance, primates) share an even more significant portion of their DNA. The DNA molecules are chains of nucleotide base-pairs, usually double-stranded. Each base-pair consists of two of the four possible nucleotides: **A**denine, **C**ytosine, **G**uanine, and **T**hymine. There are four base pairs: A-T, T-A, G-C, and C-G. Because the base pairs are symmetric, one strand of the DNA molecule is always a mirror of the other, see figure 1.1. This property makes it possible to analyse DNA as a single sequence of As, Cs, Gs and Ts.

Over the last years, the methods involved in DNA sequencing (the process of determining the DNA sequence of a given organism) have evolved dramatically. The number of sequenced genomes has grown exponentially with improved experimental techniques, algorithms and hardware. These improvements have led to databases with a considerable amount of classified DNA sequences, including those of pathogens such as viruses. These databases allow researchers to have a large reference set of DNA sequences with which to compare unclassified sequences.

The typical approach to identifying unclassified DNA sequences aligns small parts of sequences (sequence reads obtained in a DNA sequencing experiment) to reference genomes. The sequence alignment is an expensive operation, the standard implemen-

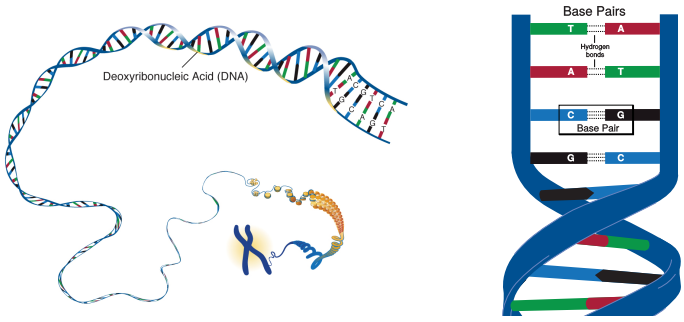


Figure 1.1: *The images illustrate the helix-shaped DNA molecule and the base pairs it is made of. Since the base pairs are symmetric, A always binds to T, C to G, and vice versa, each of the two strands is a mirror of the other. This makes it possible to think of DNA as a single long string of the four letters A, C, G, and T.*
Images courtesy of the National Human Genome Research Institute.

tation costs $O(n^2)$ [17] with n as the sequence length. In contrast, alignment-free methods rely on features of the DNA, such as GC-content (the percentage of Gs and Cs in the DNA) [16] and oligonucleotide frequencies (the frequencies of short sub-sequences of nucleotides) [2].

Genomic signatures are mathematical models that represent such features. For instance, an IID (independent and identically distributed) model captures the relative frequency of As,Cs,Gs, thus capturing the GC-content. Extending the model with one step of memory gives a first-order Markov chain which is capable of capturing more complex relationships. More advanced genomic signatures, based on Variable Length Markov Chains (VLMCs) [4, 5], have been shown to be species-specific [9] and can thus be used for analysis of DNA sequences. Furthermore, since VLMCs are compact, they allow for a faster comparison procedure than entire sequences. These models, as such, allows analysis of larger sets of data, for instance with *clustering*.

Clustering is a machine learning technique which identifies patterns in data. A clustering procedure divides data into subsets in which data points are more similar to each other than to data points in the other subsets. As an example of clustering, consider the Iris data set [7]. Iris contains the sepal length and width, as well as the petal length and width of 50 samples from three species of Iris flowers. We can apply clustering to this data set to classify each data point as belonging to one of the three species. The species are identified by which combinations of sepal/petal length and width correspond to what the clustering believes to be the three species. When presented with a flower which we do not know the species of, the resulting clusters can tell us which of the three species it most closely resembles.

1.1 Purpose

In this thesis, we present a clustering procedure for VLMCs, as well as a new distance function. The clustering may provide insight into which DNA sequences are similar, and in which respect. As such, the clustering can help identify pathogens, as well as give valuable insight into how pathogens and species evolve. However, to be able to

cluster VLMCs, we first need to define which VLMCs are similar; we need to define a distance function.

The variable and stochastic nature of the VLMCs pose a problem for a distance measure since two structurally different VLMCs can model a similar underlying sequence. However, there is a recently defined distance function for VLMCs [27]. Moreover, the literature on distances between Hidden Markov models (more complicated, but similar models), for instance, [3, 13, 30], suggests that it is possible to define several distance measures between VLMCs.

The speed of the distance calculation is also crucial. The clustering procedure may, as a worst case, require a distance calculation for every pair of VLMCs. Given an extensive data set of VLMCs, this imposes some time/complexity-constraints on the distance calculation.

1.2 Delimitations

We do not devise a novel clustering algorithm. Instead, we adapt an existing algorithm to this specific problem. We use an existing algorithm for clustering since there are existing algorithms that fulfil our needs.

We only consider genomic signatures based on VLMCs. Although this approach could be used with other underlying models, VLMCs have been shown to work well in the past, for instance, [9].

2

Theory and related research

Here, we give a formal definition of variable length Markov chains, as well as hidden Markov models, for which there are several well-known distance functions. We then present some distance functions, and give an introduction to clustering. Moreover, we present some further background on DNA, how species are classified as well as how to construct the genomic signatures (VLMCs) from DNA-sequences. Finally, we present a couple of methods which cluster DNA-sequences directly.

2.1 Markov models

Markov models are a class of stochastic models that are well suited to model a sequence of events, often represented by symbols. The simplest Markov model, the first-order Markov chain, models the next event as a function of only the current event. Higher-order and variable length Markov chains instead model a sequence of previous events which the probabilities of the next event are based upon. Finally, hidden Markov models use a more abstract relationship not directly dependent on the observed events.

2.1.1 Markov chains

A Markov chain is a discrete-state random process, specifically, a set of discrete random variables, $\{X_1, X_2, ..., X_n\}$, defined over a sample space Σ. In the context of genomics, $\Sigma = \{A, C, G, T\}$ is a common choice, corresponding to the four building blocks of DNA.

In a *first-order* Markov chain, the probability mass function (pmf) of X_n is a function of the outcome of X_{n-1}. If the outcome of X_n is $a \in \Sigma$, we define the model to be in state a. The probability mass functions of the random variables X_n can be expressed as a transition matrix, which we refer to as A. If there are $N = |\Sigma|$ symbols in the alphabet, the transition matrix for a first-order Markov chain will be of size $N \times N$. Figure 2.1 depicts a first-order Markov chain along with its parameters.

In a *higher-order* Markov chain, the next state does not depend only on the immediate previous outcome. Instead, it depends on the previous sequence of outcomes, referred to as the *context*. Formally, for a Markov chain of order k, the pmf of X_n is

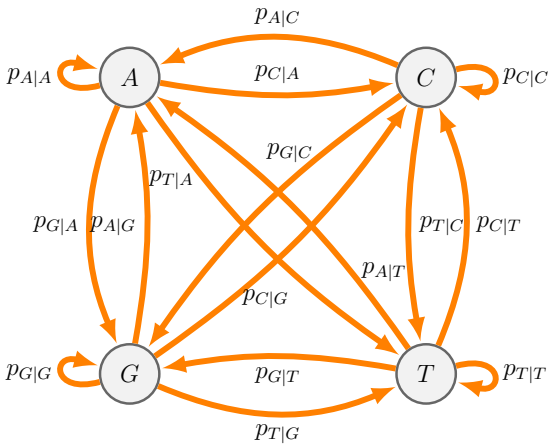


Figure 2.1: *This first-order Markov chain can be used to model a sequence of As, Cs, Gs, and Ts. The model can be used to generate a random sequence of these letters. If the model has generated a certain sequence, for instance, "ACAT", we say that the model is in state T, since it is the latest character. From this the probability that the next generated letter is A is simply* $P_{A|T}$.

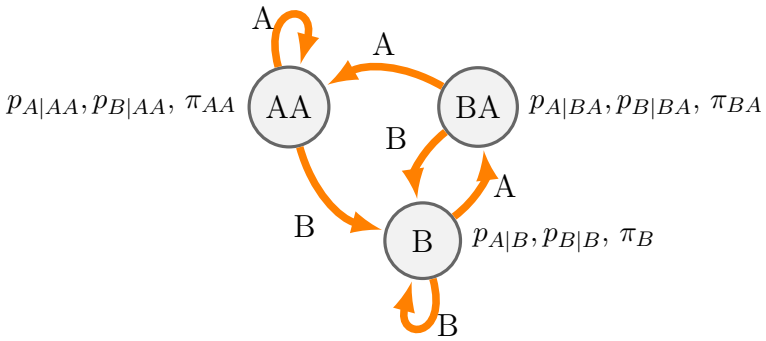
a function of the outcome of $X_{n-1}, X_{n-2}, ..., X_{n-k}$. If the outcome of X_n is $a \in \Sigma$, we define the model to be in the state corresponding to the k-suffix of the concatenation of the context of X_n and a. The transition matrix for a higher-order Markov chain will be a $S \times S$ matrix, where $S = |\Sigma|^k$ is the number of states in the model.

2.1.2 Variable length Markov chains

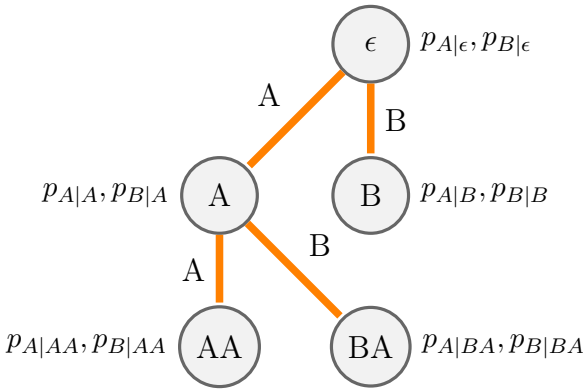
A generalisation of a higher-order Markov chain is the Variable Length Markov Chain (VLMC), for instance, [1, 22]. In these Markov chains the length of the context is allowed to vary on a state by state basis. The order of a VLMC is defined as the length of the longest context. We define the emission matrix B for a VLMC as the matrix with size $S \times N$, where S is the number of states, and N the size of the alphabet. Element b_{ij} is the probability of emitting a character j while in state i.

A VLMC can be described by a *Prediction Finite Suffix Automaton* (PFSA), see figure 2.2a for an illustration, which is a similar representation to the standard Markov chain. Each state, s, in the PFSA is labelled by a sequence, $l(s)$, of symbols from an alphabet Σ. Each transition between two states is labelled by a single symbol $a \in \Sigma$. The states are labelled as: if state s_0 has a transition to s_1 with label a, then $l(s_1)$ is a suffix of the concatenation of $l(s_0)$ and a.

Another representation of a VLMC is as a *Prediction Suffix Tree* (PST). The transition and state labelling are symmetrical to the PFSA, however, the structure is not. If state s_0 has an edge to state s_1 with label a, then $l(s_1)$ is the concatenation of a and $l(s_0)$. In a PST the current state is determined by reading the current sequence backwards from the root. For example, if given the string "BBA" and we wish to predict the next character, we read "A-B-B" from the root until we reach a terminal node. This terminal node contains the probability distribution for the next



a *A PFSA corresponding to a VLMC. π denotes the probability of starting in the corresponding state.*



b *Corresponding PST. If we want to find the probability that an A comes after the sequence BBA, we would read the sequence backwards from the root and end up in the node BA.*

Figure 2.2: *A PST and PFSA representation of a VLMC with the alphabet $\Sigma = \{A, B\}$. Note the similarities: the labelling of states and transitions, and the differences: state B does not transition to BA in the PST, and state A is needed for the tree structure.*

character. An illustration of such a PST is given in figure 2.2b. Note that, unlike the PFSA, we do not traverse the structure along the edges when continuously reading a sequence.

2.1.3 Hidden Markov models

A Hidden Markov Model (HMM), is probabilistic function of a Markov chain, and a more complex model than a VLMC. The main difference is that this model consists of two underlying random processes instead of one.

An HMM consist of S *hidden states*, whose transition probabilities can be represented as a transition matrix A of size $S \times S$. This matrix is similar to the transition matrix of a Markov chain. However, the transitions that A represent do not correspond to an output character. Instead, the HMM also has an emission matrix B. Each row in B represents the distribution of the possible output characters in a state, this makes B a matrix of size $S \times N$, where N is the size of the alphabet Σ. See figure 2.3 for an illustration, note how there are separate probabilities for emissions and transitions in the model.

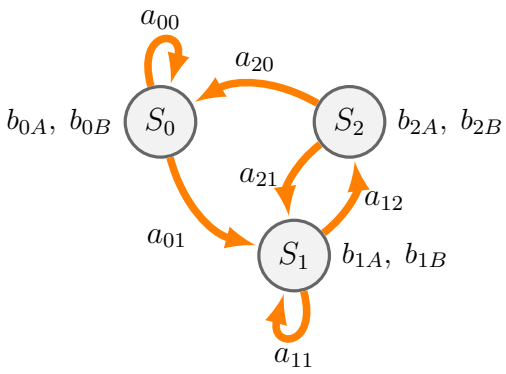


Figure 2.3: *An example of an HMM. Note how the symbols that are outputted from each state does not correspond to which state the model moves to. For example, regardless of which symbol is outputted from S_0, the model can move to either S_0 or S_1. The output symbol from S_0 is probabilistic and the probabilities are found in the emission matrix: b_{0A} is the probability that an A is generated and b_{0B} that a B is generated.*

The states of an HMM are called hidden since it is not possible to determine which state the model is in from the output of the model. This is in contrast to Markov chains, where each state depends on the observed data in a deterministic fashion.

2.2 Distance functions

A proper distance function has a number of desirable mathematical properties, which we present here. Moreover, previous research into the area of HMMs and VLMCs has revealed a number of distance functions, most which do not fully satisfy the mathematical properties.

2.2.1 Mathematical properties

Mathematically, a distance function $D(x, y)$ has to satisfy the following conditions:

Symmetry:	$D(x, y) = D(y, x)$
Non-negativity:	$D(x, y) \geq 0$
Identity:	$D(x, x) = 0$
Triangle inequality:	$D(x, z) \leq D(x, y) + D(y, z)$

Fulfilling these conditions makes the distance function a *metric*, and implicitly defines a metric space for the data points. If the data points lie in a metric space, it is easier to reason about how they relate to each other.

2.2.2 Measure of estimation error distance

Levinson et al. [12] introduce a distance function that they refer to as "measure of estimation error". It can be used as a Euclidean distance between two HMMs

(λ_1, λ_2), and measures the similarity of the emission probabilities,

$$d(\lambda_1, \lambda_2) := \sqrt{\frac{1}{MN} \sum_{j=1}^{M} \sum_{k=1}^{N} \left(b_{jk}^{(1)} - b_{p(j)k}^{(2)} \right)^2}. \tag{2.1}$$

Here, $b_{jk}^{(i)}$ is an entry in the emission matrix $B^{(i)}$, and $b_{p(j)k}^{(i)}$ corresponds to the entry in $B^{(i)}$ which minimises equation (2.1). This distance function obeys identity and non-negativity, but symmetry and triangle inequality does not hold.

2.2.3 Kullback-Leibler distance

Juang et al. [10] propose a distance measure, based on the likelihood of a generated sequence O_T, called KL-distance. The distance measure is the Kullback-Leibler number between the likelihoods, $\mu(\cdot \mid \lambda)$, of two HMMs (λ_1, λ_2),

$$d(\lambda_1, \lambda_2) := \lim_{T \to \infty} \frac{1}{T} \big(\log \mu(O_T \mid \lambda_1) - \log \mu(O_T \mid \lambda_2) \big). \tag{2.2}$$

Here, O_T is a sequence generated under λ_1, with length T. Typically, T is set to a large value, instead of using the limit $T \to \infty$. The measure obeys identity and non-negativity and can be made symmetric as

$$d_s(\lambda_1, \lambda_2) := \frac{d(\lambda_1, \lambda_2) + d(\lambda_2, \lambda_1)}{2}. \tag{2.3}$$

However, Zung et al. [30] show that the distance measure does not obey the triangle inequality.

2.2.4 Frobenius norm

Cuzzolin et al. [3] make use of the Frobenius norm as a base distance between two HMMs. They claim the Frobenius norm has both superior performance and speed compared to the KL-distance. The Frobenius norm is a standard matrix norm,

$$||M||_F := \sqrt{\sum_{i}^{m} \sum_{j}^{n} |a_{ij}|^2}. \tag{2.4}$$

The distance between λ_1, λ_2 is calculated by applying the norm to the differences between the emission matrices and transition matrices respectively, and summing the result, yielding

$$d_F(\lambda_1, \lambda_2) := ||A_1 - A_2||_F + ||B_1 - B_2||_F. \tag{2.5}$$

This distance function at least obeys identity, non-negativity, and symmetry.

2.2.5 PST-matching

Sürmeli et. al [27] propose a distance function for VLMCs, in their PST-form,

$$d(\lambda_1, \lambda_2) = \sum_{i}^{N} \sum_{j}^{M} x_{ij} w_{ij} (I \cdot \frac{\gamma_{ij}}{L_{ij}} + (1 - I) \cdot \frac{\delta_{ij} \epsilon_{ij}}{2}). \tag{2.6}$$

Since the equation (2.6) contains a sum over all pairs of states, they introduce x_{ij} as an indicator function. It has the value of 1 if i, j are the same state or the most similar state in the other model. Otherwise, x_{ij} is 0.

They call w_{ij} the match weighting component, and it is equal to the average of the occurrence probabilities of the two states i and j in their respective model. The occurrence probability is the probability of seeing that specific context in the original training sequence.

Furthermore, the distance is split into two costs: similarity-cost and dissimilarity cost. The hyper-parameter, $I \in [0, 1]$, known as the *cost type weighting component*, adjusts how much each cost should contribute to the total distance. The similarity-cost will be 0 for any two states which are not exactly the same context. Analogously, the dissimilarity cost will be 0 for any two states which are exactly the same context.

The dissimilarity cost is a measure of how different two states are,

$$I \cdot \frac{\gamma_{ij}}{L_{ij}}. \tag{2.7}$$

Here, $\gamma_{ij} \geq 0$ is the absolute difference in context length for the two states i from λ_1 and j from λ_2. The term is normalised by L_{ij} which is the larger of the two context lengths.

The probability cost is a measure of how similar the probabilities of two states are

$$(1 - I) \cdot \frac{\delta_{ij} \epsilon_{ij}}{2}. \tag{2.8}$$

Here, ϵ_{ij} is defined to be 1 only if the contexts i and j are exactly the same, it is 0 otherwise. The factor δ_{ij} is the sum of the differences of the probabilities in the two states i from λ_1 and j from λ_2

$$\delta_{ij} = \sum_{a \in \Sigma} |P_1(a \mid i) - P_2(a \mid j)|. \tag{2.9}$$

The authors do not motivate if their distance measure is a real distance metric.

2.3 Clustering

The clustering problem is well studied in the field of machine learning. To cluster a data set is the problem of finding a set of clusters C. Elements within each cluster

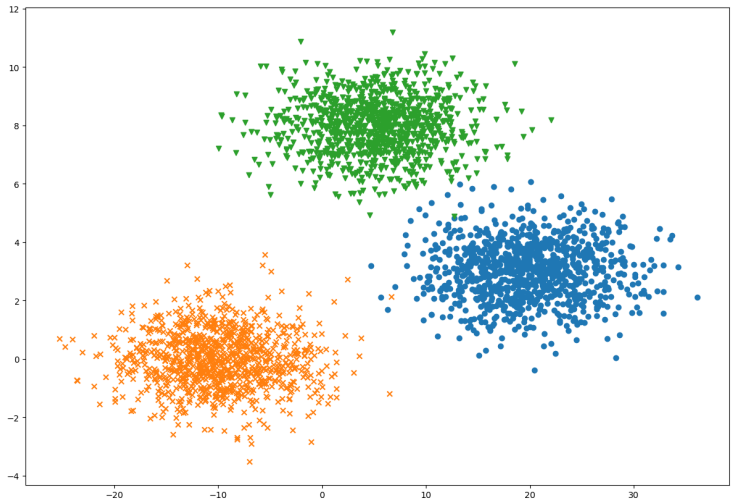


Figure 2.4: *An example of the input to a clustering procedure. The colours and markers indicate the "real" classes, the input points to a clustering procedure are generally not distinguishable from each other. The clustering task is to find the three groups of points shown in the image.*

$c \in C$ are more similar to each other than to elements of other clusters in C. We refer to elements within a cluster to be in the same *class*. Given a clustering of a data set, it is also possible to determine into which of the classes a new data point belongs.

Figure 2.4 contains an example of the input to a clustering procedure. Note that, usually, the number of clusters and which points belong to which classes are unknown parameters. Therefore, the number of clusters may need to be estimated (for instance, by running the algorithm multiple times).

2.3.1 Graph-based clustering

One type of clustering algorithm is graph-based clustering [29]. Each data point corresponds to a vertex in a graph, G, and the distance between the data points correspond to weighted edges between them. From the graph, several different schemes (for instance single-link or average-link [15]) can be applied to find a clustering. In bottom-up graph-based clustering, each vertex of the graph initially represents a cluster. Then, iteratively, the clusters are *merged* to form larger clusters.

A disadvantage to graph-based approaches is the construction of the graph, which requires distance calculations between every vertex. However, the approaches allow analysis of individual steps in the clustering, which allows for manual selection of when to stop the clustering depending on when clusters were merged.

2.4 DNA

DNA (and RNA) molecules are responsible for encoding the blueprint of their respective host. Therefore, analysis of DNA molecules serves as a good source of information about the organisms. The DNA molecules are made up of four nucleotide bases: adenine, cytosine, guanine, and thymine, shortened as A, C, G, and T, respectively. DNA molecules are typically double-stranded, each strand forming a mirror of the other, A always binds to T and C to G, and vice versa, in the other strand. As such, both strands contain the same information and it is sufficient to analyse one of them. Although uncommon, some organisms, such as certain viruses, have single-stranded DNA.

A sequence of three nucleotides (also called a triplet or codon) in the DNA sequence encodes an amino acid residue, a fundamental building block for proteins which, in turn, are crucial for the functioning of the cells. This encoding is not unique since there are more possible combinations (4^3) than there are common amino acid residues (20). A more general name for a short subsequence (not necessarily three long) of nucleotides is *oligonucleotides*.

One standard approach to analysing differences between organisms is to examine differences in their respective DNA. Since DNA sequences can be encoded as strings of characters, a common method of measuring the difference between two DNA sequences is with the Levenshtein (or edit) distance. The Levenshtein distance measures the number of operations (insertions, deletions, and substitutions), representing evolutionary mutations, that has to be applied to the two strings for them to become equal. These operations can be weighted to favour certain operations. A well known dynamic programming method for calculating the Levenshtein distance is the Needleman-Wunsch algorithm [17]. Another method that measures the difference between two sequences is the k-mer distance, the number of shared (or differing) substrings of length k.

Moreover, there are alignment-free methods which compare DNA sequences without having to calculate the edit distance. These methods usually construct a simple mathematical model of the sequence, giving a faster comparison step. Examples of alignment-free methods include measuring the difference in GC-content [16] and comparing oligonucleotide frequencies [2].

2.4.1 Organism classification

Organisms are ordered in a taxonomic hierarchy, sometimes referred to as the phylogenetic tree, corresponding to the genetic ancestry of the organisms. The ranks are domain, kingdom, phylum, class, order, family, genus and species. For instance, the family Felidae corresponds to cats, the genus Felis corresponds to small and medium-sized cats, and the species Felis silvestris corresponds to small European and African cats (including domestic cats).

The Baltimore classification is a classification system for viruses. It places viruses

into classes depending on the structure of their genome (DNA, RNA, double-stranded, etc) and method of replication.

2.5 Genomic signatures

A genomic signature refers to a model that captures the frequencies of oligonucleotides in a DNA sequence [11]. While the definition of genomic signatures does not specify what the model is, in this thesis, the underlying model is a VLMC.

2.5.1 Genomic signature generation

Dalevi et al. [4, 5] have developed a method for creating genomic signatures based on VLMCs. They start with a DNA sequence from an organism and learn a VLMC corresponding to the organism's species. They estimate the PST corresponding to a VLMC by first naively constructing a PST by adding all nodes up to a certain depth when the number of times the node shows up in the sequence is large enough. Then each node in the PST is assigned a score ($\Delta_{v,w}$), and terminal nodes with lower scores are removed iteratively. $\Delta_{v,w}$ is defined as

$$\Delta_{v,w} := \sum_{a \in \Sigma} p(a \mid v) \log \left(\frac{p(a \mid v)}{p(a \mid w)} \right) N(v). \tag{2.10}$$

Here, w is a suffix of v, with length $|v| - 1$ (w is the parent node in the PST), $N(v)$ denotes the frequency of the word v in the training sequence, and Σ is the alphabet of the model. The value of $\Delta_{v,w}$ can roughly be interpreted as: if large enough, v is sufficiently important to keep in the PST, and otherwise it either occurs too infrequently or is too similar to its parent state.

2.5.2 Species specificity

Using VLMCs as the underlying model for genomic signatures, Holmudden [9] has demonstrated that genomic signatures can be used to distinguish between species of viruses. To show this, they divided each of the DNA sequences of the viruses into two parts. For each part, they generated the corresponding genomic signature. The signature corresponding to the first part was entered into a database, and then every second signature was compared to the database. The comparison was made using KL-distance. In 25 out of 28 cases, the signatures most closely matched each other, which indicates that they accurately capture important aspects of the virus' DNA.

2.6 Clustering DNA sequences

Earlier work in the area of genome-clustering uses the DNA sequences directly as opposed to genomic signatures. We present some of these methods here, for comparison to our method.

2.6.1 Dnaclust

Ghodsi et al. propose DNACLUST [8], which uses the edit distance between two DNA sequences for the distance measure, with unit cost for insertions, deletions, and substitution. They define a similarity measure for sequences as

$$\text{similarity} = 1 - \frac{\text{edit distance}}{\text{length of shorter sequence}}. \tag{2.11}$$

Since the edit distance calculation is expensive, they also introduce an early stopping criterion. If at any point, the edit distance becomes too large (as defined by the user) the calculation is aborted, deeming the sequences as dissimilar. The prefix of the dissimilar sequence, for which the edit distance became too large, is compared with other sequences in order to see if the distance calculation can be skipped altogether.

The clustering procedure uses a user-defined cluster diameter, which corresponds to the maximum distance between any two sequences in the same cluster. The algorithm proceeds as follows: sort the sequences by their length (longest first), pick the longest unclustered sequence as a new cluster centre and place every sequence with a similarity value less than the threshold in that cluster, repeat until every sequence belongs to a cluster.

2.6.2 Uclust

Edgar et al. propose UCLUST [6], which employs a more standard approach to sequence comparisons. VSEARCH is a tool from 2016 provided by Rognes et al. [21] which implements UCLUST. The sequences are compared by the standard edit distance, however, what makes the approach fast is that the sequences are sorted by the number of short words in common. The sorting allows the comparison algorithm to utilise the fact that similar sequences tend to share short substrings. Thus allowing the comparison calculation to be skipped if two sequences do not share many short substrings.

The clustering uses the sequence comparison method to assign sequences to clusters. How this assignment is done and when new clusters are created are not discussed by the authors.

2.6.3 SlideSort

Shimizu et al. propose SLIDESORT [26], which finds pairs of similar sequences by identifying chains of shared k-mers, i.e., substrings of length k. The process starts by identifying shared k-mers in the sequences. The next step moves a window that captures the shared k-mer one step forwards, and checks again for shared k-mers, within a threshold of the edit distance of the k-mer. If there is a match, the process continues recursively. The total edit distance of the chain of shared k-mers is taken as the distance between the sequences.

The clustering procedure in SLIDESORT is a graph-based clustering algorithm with

the sequences as vertices and the distance as the weight of the edges. The minimum spanning tree of the graph is constructed, from which a clustering is created by removing the longest edges.

3

Method

To be able to cluster the VLMCs, we first define several possible distance functions. Given the distance functions, we define two clustering procedures, our data sets, and how we selected the parameters for the training procedure for the VLMCs. Finally, we present how we evaluated the distance functions and clusters.

3.1 Distances for Variable Length Markov Chains

We defined several distance functions for VLMCs, some operate on the PST structure of the VLMCs, and some are not dependent on any specific structure. We based most of the distance functions on distance functions defined for HMMs.

3.1.1 GC-distance

The GC-content of DNA sequences has been shown to be species-specific [16]. To use this property when comparing VLMCs, we defined a distance measure which compares the GC-content of two models. The root state, ϵ, of the PST contains the relative frequency of As, Cs, Gs, and Ts, in the original training sequence. This distance function is a proper metric and we defined it as

$$d_{gc}(\lambda_1, \lambda_2) := |(P_1(G \mid \epsilon) + P_1(C \mid \epsilon)) - (P_2(G \mid \epsilon) + P_2(C \mid \epsilon))|. \tag{3.1}$$

3.1.2 Measure of estimation error for VLMCs

The measure of estimation error distance function equation (2.1) can be used without modification. We defined the matrix used in the calculation to be the emission matrix of a VLMC. While this distance function is not symmetric, we defined a symmetric version by averaging the distance between λ_1 and λ_2.

3.1.3 Kullback-Leibler distance for VLMCs

The KL-distance measure can be applied to VLMCs synonymously to HMMs. However, the KL-distance requires a generated sequence s, which makes the time complexity of the calculation to be in $\mathcal{O}(|s|)$. For a long sequence, we considered this to be too slow. However, the KL-distance accurately measures if two models are

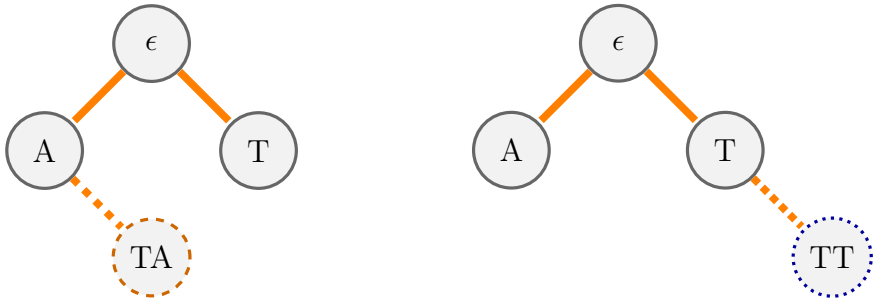


Figure 3.1: *The figure illustrates two VLMCs, λ_1, λ_2, with the alphabet $\Sigma = \{A, T\}$, represented as PSTs. Each state in the PST consists of probabilities for every letter in the alphabet, e.g. state A consists of $P(A \mid A)$ and $P(T \mid A)$.*

When using Frobenius-intersection to calculate $d_{fi}(\lambda_1, \lambda_2)$, we only consider the states that exists in both models, i.e. A, T, and ϵ.

When using Frobenius-union, we consider all states in both models. The calculation is analogous to Frobenius-intersection, except for the probabilities of model-unique states such as TA of λ_1. Since there is no corresponding state in λ_2, we choose to use the node that would be the closest ancestor to TA if it existed in λ_2, in this case A. This is an intuitive choice, since λ_2 would be in state A if TA was the latest generated sequence. The other state that is unique to a model is TT in the second model, which would be compared to state T in the first model in this example.

similar, for sufficiently large s. Therefore, we primarily used the KL-distance as ground truth for equality between models.

3.1.4 Frobenius norm distance for VLMCs

We defined the Frobenius distance only on the emission matrix of the VLMCs, since the transition and emission matrices are equivalent. Moreover, since the structure of VLMCs vary, different pairs of VLMCs have a different amount of combined states. Therefore, to make the distances comparable, we normalised the result of the calculation by the square root of the number of states. This modification makes the distance exactly the root mean squared error (we do not normalise with respect to the alphabet, since the alphabet is constant for all our genomic signatures). Moreover, the states of each VLMC represent different contexts, so the emission matrices need to be made comparable.

To compare emission matrices, we analysed the structure of the PSTs. Two PSTs always share at least the root state ϵ. Therefore, it is possible to compare the states of the PSTs directly. However, the states which the VLMCs do not share are trickier, we present two methods for dealing with them, illustrated in figure 3.1.

The intersection method, where we only compare states which both models contain. One flaw is that it does not use all the information contained in the PSTs.

Formally, let $S_i \subset \Sigma^*$ be the intersection of the states of two VLMCs λ_1, λ_2. We

compute the Frobenius-intersection distance as

$$d_{fi}(\lambda_1, \lambda_2) := \sqrt{\frac{1}{|S_i|} \sum_{s \in S_i} \sum_{a \in \Sigma} \big(P_1(a \mid s) - P_2(a \mid s)\big)^2}. \tag{3.2}$$

This distance function obeys symmetry, identity, and non-negativity. In the general case, the triangle inequality does not hold. The time complexity of the calculation is $\mathcal{O}(n|\Sigma|)$, where n is the number of states in the smaller of the two models (since the intersection of the shared states is at most all of the states in the smaller model).

The union method, which considers every state in both models. In the cases where a state exists only in one of the models, we compare the probabilities of that state to the closest ancestor of that state in the other model. Comparing a node with its closest ancestor is aligned with the pruning of states during the training of the PST, see section 2.5.1. The training procedure prune states if they either have similar probabilities to their parent state or if the state occurs infrequently. Under the assumption that the removed states have similar probabilities to their parents, comparing a missing state to its ancestor creates an accurate comparison. In the other case, where we have removed states because of a low occurrence, this method can introduce errors.

Formally, let $S_u \subset \Sigma^*$ be the union of the states of two VLMCs λ_1, λ_2. We compute the Frobenius-union distance as

$$d_{fu}(\lambda_1, \lambda_2) := \sqrt{\frac{1}{|S_u|} \sum_{s \in S_u} \sum_{a \in \Sigma} \big(\hat{P}_1(a \mid s) - \hat{P}_2(a \mid s)\big)^2}, \tag{3.3}$$

where, $\hat{P}_i(a \mid c)$ is the function which calculates the probability of character a in state c, if model i contains state c. If model i does not contain state c, it first finds the closest parent to that state in the tree.

This distance function at least obeys the identity, non-negativity, and symmetry conditions. The time complexity of the calculation is $\mathcal{O}(nd|\Sigma|)$ where n is the number of states in the union of the models, and d the max of the order of the models. The nd term is derived from the sum over the states, and having to find the closest ancestor-state, which is $\mathcal{O}(d)$.

3.1.5 PST-matching distance

PST-matching (section 2.2.5) can be used directly since it is defined for VLMCs. We set the cost type weighting component, I, to 0.5. We motivate this choice further in appendix B. The time complexity of our implementation is $\mathcal{O}((N + M) \cdot d)$, where d is the highest order of the two models, and N and M are the number of states of the two models. However, the distance function can be implemented as a parallel traversal of both PSTs in order to reach $\mathcal{O}(N + M)$.

3.2 Clustering variable length Markov chains

For the clustering of a set of n VLMCs, we implemented two graph-based clustering algorithms. Both of the clustering procedures require $\mathcal{O}(n^2)$ distance calculations.

3.2.1 Single-link clustering

Single-link clustering is defined as follows:

1. Create a vertex for every VLMC λ_i in the input, and create a cluster C_i for every vertex.
2. Calculate the distances between every pair of VLMCs, according to any distance function.
3. Merge clusters C_a, C_b with the smallest distance, according to equation (3.4).
4. Repeat 3 until a user-specified number of clusters remain.

$$D(C_a, C_b) := \min\{d(\lambda_i, \lambda_j) : \lambda_i \in C_a, \lambda_j \in C_b\} \tag{3.4}$$

Our implementation has a time complexity of $\mathcal{O}(n^2 \log n)$, where n is the number of VLMCs. We select edges by first sorting the calculated distances ($\mathcal{O}(s \log s)$, with $s = n^2$ distances), and then select the smallest edges which does not join already connected parts. This final check has a time complexity of $\mathcal{O}(n)$ per introduced edge, yielding a time complexity of $\mathcal{O}((n-k)n)$, where k is the number of clusters. Furthermore, the implementation requires $\mathcal{O}(n^2)$ memory to store the distances.

3.2.2 Average-link clustering

Average-link clustering is a similar procedure to single-link clustering. However, instead of merging clusters with the minimum single distance between them, it merges clusters with the minimum average distance. The distance between clusters C_a, C_b is defined as

$$D(C_a, C_b) := \frac{1}{|C_a||C_b|} \sum_{\lambda_i \in C_a} \sum_{\lambda_j \in C_b} d(\lambda_i, \lambda_j). \tag{3.5}$$

The average-link merge criterion is also used in the UPGMA algorithm. UPGMA produces a diagram, known as a dendrogram, which illustrates the merge steps in the clustering procedure. With genomic data, this dendrogram can represent the phylogenetic ordering of the input. Technically, UPGMA requires that the distance function is an *ultrametric*, which is a stronger metric with a different version of the triangle inequality, defined as

$$d(x, z) \leq \max\{d(x, y), d(y, z)\}. \tag{3.6}$$

Our average-link implementation has a time complexity of $\mathcal{O}(n^2 \log n)$. We stored the distances in a min-heap per cluster, which allows for $O(n)$ time for finding the

smallest average distance every iteration. Moreover, merging two clusters requires an update of all of the distances to the new, merged, cluster. Each such update was implemented in constant time, however, removing the old distances from the heaps requires $O(n)$ per heap. Therefore, instead of removing them, we keep a hash-map, from which we can remove in constant time. The hash-map is used to verify that a distance found in the heap is still relevant. The creation of the new cluster in every merge step does, however, take $O(n \log n)$ per step, which gives us a final time complexity of $O(n^2 \log n)$. Furthermore, the implementation requires $\mathcal{O}(n^2)$ memory. There are n^2 distances, and each heap and hash-map both contain an amortized maximum of n distances.

3.3 Data sets

We used two data sets of DNA from viruses, one with 33 viruses, and one with 304 viruses, which we obtained from our collaborators who are biological researchers. We also used a data set of 91 bacteria with relatively short DNA sequences (less than 5 million nucleotides long). Table 3.1 contains some additional properties of the data sets.

	# **Species**	# **Families**	# **Genera**	**Min length**	**Max**	**Average**
Small	33	5	16	30 536	295 119	127 926
Large	304	17	73	3 166	1 221 932	99 616
Bacteria	91	35	41	416 863	4 941 290	2 903 106

Table 3.1: *Properties of the different data sets used. All sequence lengths are given in the number of nucleotides. The very short family of viruses in the large data set are Retroviruses, and the long one is a Mimiviridae, which is about three times longer than the next-longest.*

From these data sets, we trained VLMCs using Dalevi et al.'s software [5]. To train VLMCs, we have to specify how many free parameters to use, which corresponds to how large the VLMCs become. For instance, a VLMC with 48 parameters correspond to a second-order Markov chain. To choose the number of parameters, we examined the accuracy of the distance functions with different numbers of parameters with respect to our metrics defined in section 3.4.

3.4 Distance function evaluation

Our criteria for determining how well the distance functions behave are based on the underlying training data. With virus DNA as the underlying data, we primarily analysed the following three properties:

- Taxonomic rank (for instance, genus and family).
- Host species.
- The Baltimore classification.

During the comparison between distance functions, we primarily examined the tax-

onomic ranks of the species corresponding to the VLMCs. This evaluation method is based on the intuition that if two species share a taxonomic rank, for instance, they are both Herpes viruses, their genomic signatures (VLMCs) should be similar. We also considered the host species of the viruses, gathered from Virus-Host DB [14], and the Baltimore classification.

Moreover, since the approach should scale to large data sets, the computational efficiency of the distance function is crucial. For this reason, we measured and put a heavy emphasis on the computation time of the distance functions.

3.4.1 Distance metrics

We defined two metrics that reflect how well the distance functions perform. They are defined for a given set of VLMCs $\Lambda = \bigcup_{i=1}^{N} \Lambda_i$ where each set Λ_i corresponds to the classes within a taxonomic rank of VLMCs, and there are a total of N classes within the rank. Both metrics assume a given VLMC, λ, in some class Λ_i. Examples of the two metrics are given in figure 3.2.

Percent of taxonomic rank is a fraction of how many of the nearest $|\Lambda_i|$ VLMCs (to λ), according to the distance function, belong to Λ_i. As measured by this metric, a perfect distance function would have a Percent rank of 1 for all VLMCs in Λ. In order to compare distance functions over the same set of VLMCs, we use the average Percent rank for all VLMCs in the set.

Distance to taxonomic rank is defined as a fraction of the average distance $\overline{d}$ from λ to VLMCs in Λ_i and the average distance $\overline{D}$ from every λ to every VLMC in Λ. An accurate distance function would have a distance rank close to zero, and a value above 1 would indicate that the distances to VLMCs within the same rank are larger than the average distance.

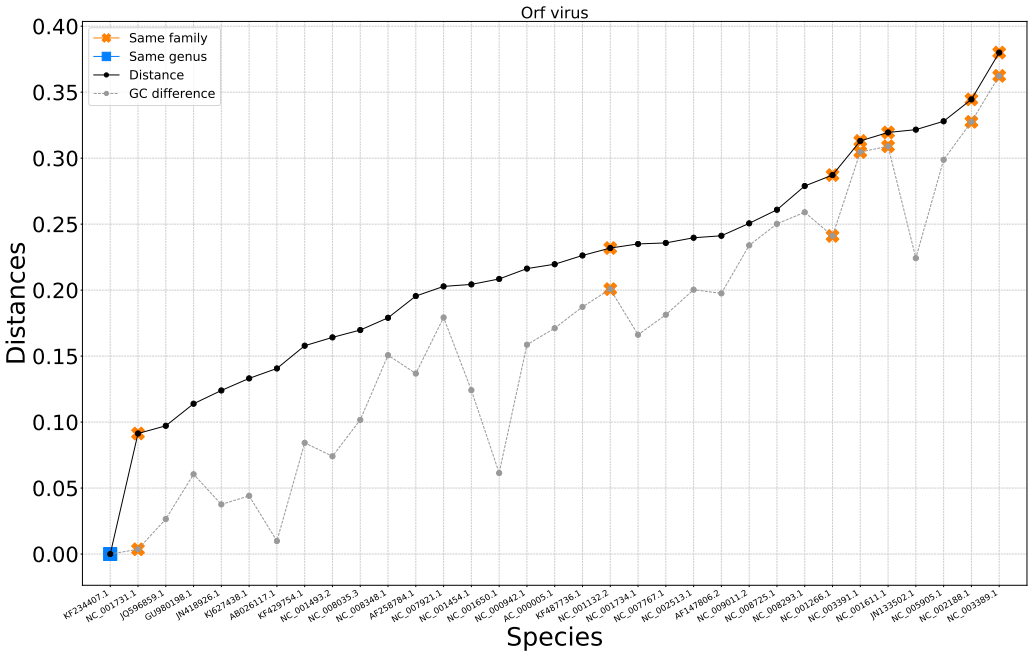
3.5 Clustering evaluation

We evaluated the output of the clustering algorithm primarily by analysing the taxonomic ranks of the species that clustered together. The optimal clustering would be one cluster per class within each rank. For the data sets with viruses, we also analysed the host species and the Baltimore classification of the clusters.

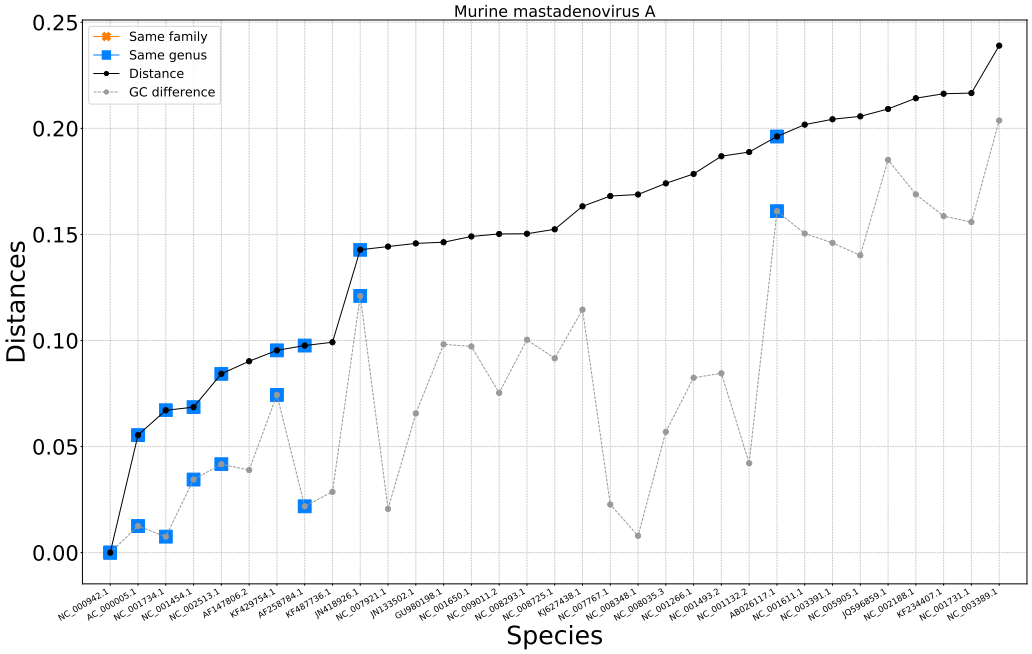
For the sake of comparison, we ran alignment-based clustering algorithms on the same data sets. Specifically, we compared to DNACLUST, SLIDESORT, and UCLUST.

3.5.1 Clustering metrics

We defined three metrics for comparison between cluster outputs. They are defined given a clustering $C = \{C_1, C_2, ..., C_k\}$ of a set of VLMCs $\Lambda = \bigcup \Lambda_i$ where each $\lambda \in \Lambda$ belongs to a class Λ_i within some taxonomic rank or Baltimore classification.



a *Plot of distance calculation for Orf virus, the values of our metrics are: a percent genus of* 1*, a percent family of* 0.25*, distance genus of* 0*, and a distance family of* 1.36*.*



b *Plot of distance calculation for Murine mastadenovirus A, the values of our metrics are: a percent genus of* 0.78*, a percent family of* 0.78*, distance genus of* 0.50*, and a distance family of* 0.50*.*

Figure 3.2: *The distance metrics in relation to distances.*

Percent of taxonomic rank is calculated by averaging how many of the genomic signatures in each cluster belongs to the same order (for instance, genus or family).

More formally, given a specific cluster C_i in the clustering, we calculate a value μ_j for each VLMC $\lambda_j \in C_i$, belonging to some class Λ_k

$$\mu_j := \frac{|C_i \cap \Lambda_k|}{|C_i|}. \tag{3.7}$$

Then we calculate our metric as

$$\text{Percent rank} := \sum_{\lambda_i \in \Lambda} \frac{\mu_i}{|\Lambda|}. \tag{3.8}$$

When clusters with genomic signatures are mostly in the same order, the metric has a value close to 1.

The silhouette is a standard clustering metric designed to measure how well a data point x lies within its own cluster. We used the average silhouette value as a metric for a given clustering. We defined the silhouette of a data point as

$$s(x) := \frac{b(x) - a(x)}{\max\{a(x), b(x)\}}. \tag{3.9}$$

Here $a(x)$ is the average distance from data point x to every other data point in the same cluster. $b(x)$ is the average distance from x to all data points in the neighbouring cluster of x, which is the cluster with lowest average distance to x that does not contain x itself.

Given a data point, the silhouette value can range from -1 to 1. The higher the value, the closer the data point lies to its own cluster compared to the neighbouring clusters, corresponding to it being well-clustered. If a silhouette value is negative, it is badly clustered since it lies closer to another cluster than to its own.

Sensitivity and Specificity, are defined by the rate of true positives vs rates of false negatives and false positives. We define them here similarly to Pipenbacher et al. [19]. A true positive is defined as a pair of VLMCs in the same cluster which also share a feature, such as taxonomic family or Baltimore class. A false positive is the opposite, a pair of VLMCs in the same cluster which do not share a feature. Finally, a false negative is a pair of VLMCs which do not belong to the same cluster, but do share a feature.

Sensitivity corresponds to the proportion of correctly classified VLMCs

$$\text{sensitivity} := \frac{\#\text{True positives}}{\#\text{True positives} + \#\text{False negatives}}. \tag{3.10}$$

Specificity corresponds to the proportion of correct classifications in the entire set of positive classifications

$$\text{specificity} := \frac{\#\text{True positives}}{\#\text{True positives} + \#\text{False positives}}. \tag{3.11}$$

A perfect procedure would have both sensitivity and specificity of 1, corresponding to every classification of VLMCs being correct. However, there is commonly a trade-off between sensitivity and specificity, where the clusters mostly contain a single feature without subdividing features into multiple clusters too much. We used these metrics primarily with respect to the taxonomic family of the genomic signatures.

4

Results

We establish the number of parameters used for the VLMCs as described in section 3.3, and then evaluate the proposed distance functions from section 3.1 using the metrics described in section 3.4. We then evaluate the clustering algorithms from section 3.2, with respect to the metrics defined in section 3.5, and illustrate the resulting clusters. Finally, we compare our results to three alignment-based clustering algorithms.

The tests are run on a computer with 8GB of RAM and an i7-6500U CPU (2.50GHz). Our source code is available at github.com/kalior/clustering-genomic-signatures.

4.1 Model generation results

When creating the VLMCs from DNA sequences, we must specify how many free parameters the models should have. We present our results as to how many parameters to select here, and then present an issue with the model training procedure.

4.1.1 Number of parameters for VLMCs

We tested varying the number of free parameters for the VLMCs, and figure 4.1 contains the results of the tests. As can be seen the metrics are quite good on models with about 10^2 parameters. Above that, however, the metrics start to decline.

Based on these results, we choose the number of free parameters of the VLMCS to 192. This number keeps the models small while still giving good results on our data set. However, there has to be a sufficient amount of data to train each of the parameters; consequently, with short sequences, a smaller amount of parameters musts be used to avoid noise. With 192 parameters, the trained VLMCs have orders as shown in table 4.1.

Creating the models from raw sequences takes a significant amount of time. The VLMC training procedure takes on average roughly 1.8s per model. The individual training time varies and is related to the sequence length of the genome.

	Min order	Average order	Max order
Small	3	3.8	7
Large	4	7.5	13
Bacteria	4	5.4	8

Table 4.1: *The order of the trained VLMCs with our three data sets. The VLMCs have been trained to have 192 parameters*

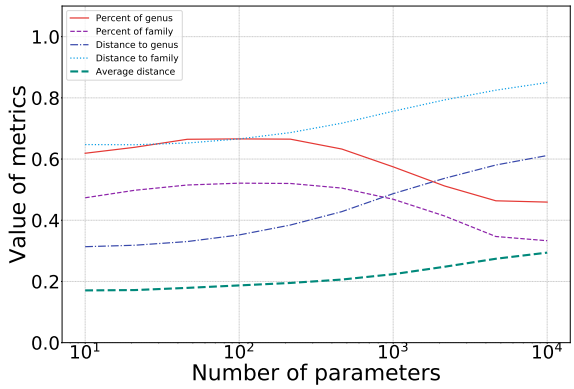


Figure 4.1: *These plots shows how our distance metrics varies with respect to our smaller data set given that we generate VLMCs of different sizes. The metrics are measured using the Frobenius intersection distance, but the results are similar for the other distance functions.*

4.1.2 Sequence length limit of the model-generation

We have been unable to train VLMCs for DNA sequences longer than 5 million nucleotides due to memory constraints. This issue has limited our data sets to viruses, which have small genomes, and those bacteria which have small genomes.

4.2 Distance function evaluation

We evaluated the distance functions for VLMCs, using VLMCs trained on the data sets and with the metrics presented in chapter 3. For the KL-distance, we selected a sequence length of 16 000, see appendix A for a motivation. We address the parameter for PST matching in appendix B.

The results for the small data set are given in table 4.2, and the larger data set in table 4.3. The simple GC-content distance captures roughly a third of the family, and is one of the worst distance functions, despite being a true distance function. The Frobenius-intersection performs about as well as the KL-distance, which is otherwise the most accurate function, while being much faster to compute. Moreover, the Frobenius-union is significantly worse than the Frobenius-intersection on the large data set, despite taking more of the information contained in the models into account. Finally, the PST matching is notably worse than even the GC-distance, which is the simplest approach.

We have also considered other distance functions for HMMs (detailed results not

shown). Lu et al.'s distance function for HMMs from [13] is too slow for our use case. Zeng et al.'s new distance function [30] is equivalent to the difference in the proportion of As, Cs, Gs, and Ts for genomic applications (comparable to GC-content). Cuzzolin et al.'s distance function from [3], requires target classes. We do not want to overfit to, for instance, the taxonomic classes, but instead, identify various causes for similarity between the genomic signatures.

Function	**% genus**	**% family**	**Dist. Genus**	**Dist. Family**	**Time**
GC-distance	0.64	0.39	0.28	0.76	0.00072s
Error estimation	0.75	0.46	0.38	0.82	5.17s
KL-distance	0.87	0.58	0.20	0.65	161.97s
Frobenius intersection	0.85	0.56	0.30	0.71	0.06s
Frobenius union	0.78	0.49	0.43	0.82	0.70s
PST matching	0.61	0.36	0.54	0.83	0.44s

Table 4.2: *Performance measure of the distance functions on the smaller data set. The sequence length for the KL calculation was* 16 000*.*

Function	**% genus**	**% family**	**Dist. Genus**	**Dist. Family**	**Time**
GC-distance	0.46	0.35	0.33	0.70	0.05s
Error estimation	0.43	0.38	0.55	0.80	245s
KL-distance	0.69	0.55	0.22	0.56	7488s
Frobenius intersection	0.67	0.52	0.38	0.69	4.2s
Frobenius union	0.50	0.43	0.59	0.83	42s
PST matching	0.39	0.29	0.64	0.86	25s

Table 4.3: *Performance measure of the distance functions on the large data set. The sequence length for the KL calculation was* 16 000*.*

In figure 4.2, figure 4.3, and figure 4.4 we plot how the Frobenius-intersection function correlates with difference in GC-content, and the cumulative captured percentages of taxonomic family and genus. The x-axis of the plots corresponds to the VLMCs as ordered by the distance function, similar to figure 3.2. There is a large variation throughout, but a large GC-difference implies a large distance (for all of the distance functions). Moreover, the distance functions quickly pick up the entire genus and family. However, the family percentages vary significantly throughout.

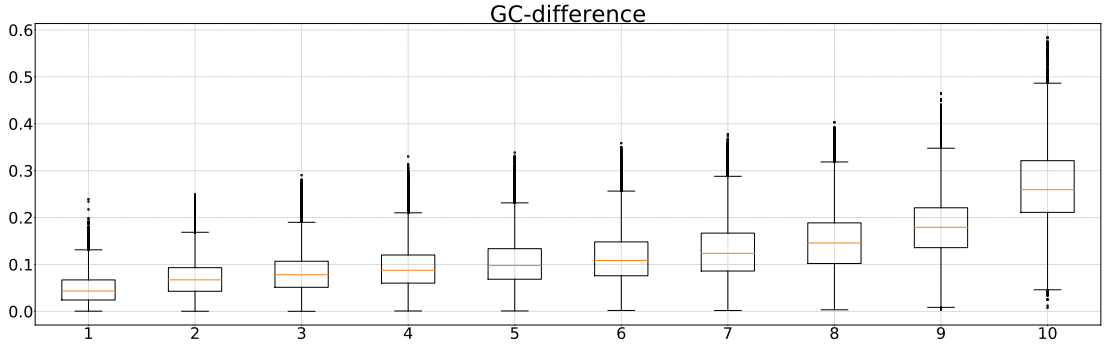


Figure 4.2: *This plot illustrates how the difference in GC-content between VLMCs correlate with our distance function.*
For each VLMC λ_i in the set, the graph shown in figure 3.2 is created. All these graphs are then combined into a single box-plot. Each box represents a certain level of proximity, e.g. the second box represents how large the GC-difference is among the models which are in the closest 20% of models but further away than the closest 10%. We can see that the difference in GC-content varies throughout, but a large GC-difference implies a large distance.

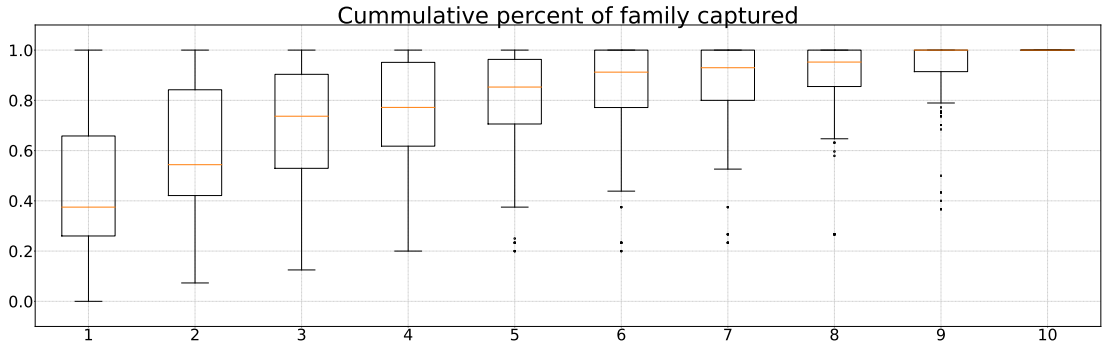


Figure 4.3: *This plot displays how many percent of the family the distance function has captured cumulatively. The last box will always be* 100%*, since the full data set has been considered at that point. For instance, the first box indicates that, on average, approximately* 50% *of the family can be found within the* 10% *closest VLMCs, but there is also a large variation.*

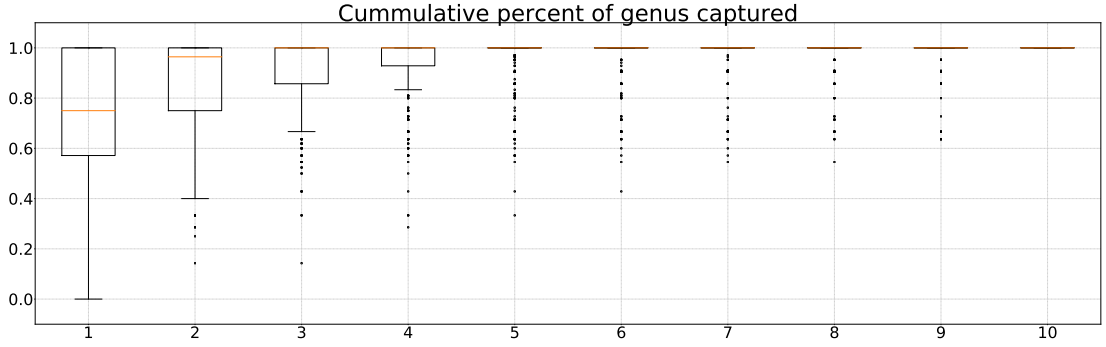


Figure 4.4: *This plot displays how many percent of the genus the distance function has captured cumulatively. The last box will always be* 100%*, since the full data set has been considered at that point. For instance, the first box indicates that, on average,* 80% *of VLMCs within the same genus are found in the* 10% *closest VLMCs. There is a large variation, but not as large as with the family.*

4.3 Clustering results

We evaluated the clustering with the distance functions evaluated in section 4.2. Here, we focus on the Frobenius-intersection and KL-distance, since they are the most promising. We present the results for the other distance functions in appendix C, but note that they perform worse than KL and Frobenius-intersection. We cluster the large data set here unless otherwise specified.

In figure 4.5 and figure 4.6 we plot how the metrics vary with respect to the amount of clusters k with single-link and average-link clustering, respectively. The single-link clustering performs worse throughout, illustrating that at least average-link clustering is needed. We, therefore, focus the remainder of our analysis on the average-link clustering. We present the values of the metrics for three selected k number of clusters from figure 4.6 in table 4.4.

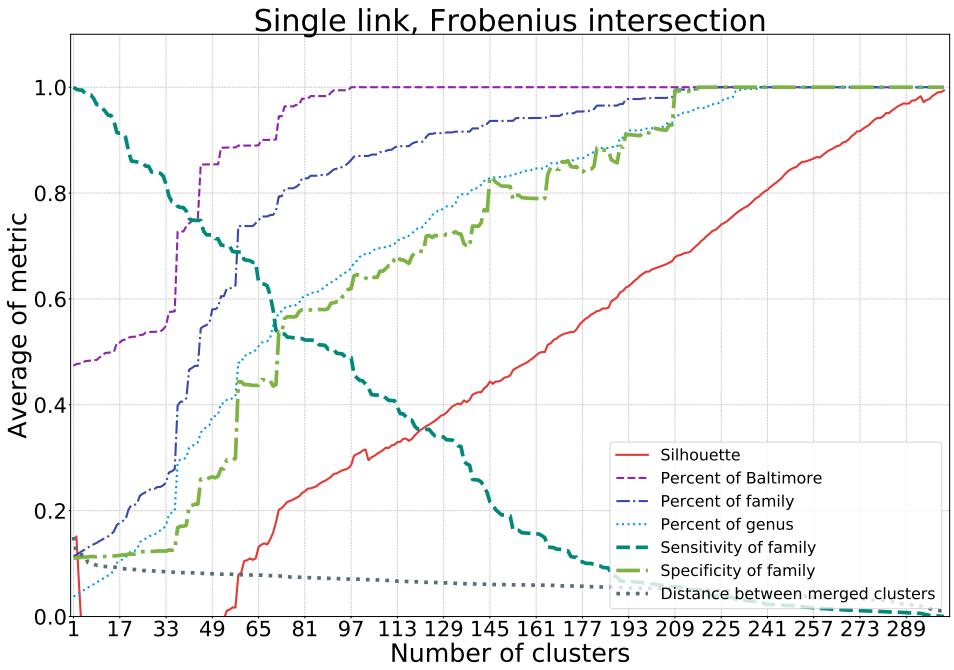
k	Sensitivity	Specificity	% family	% Baltimore	Cluster sizes			
30	0.45	0.62	0.69	0.92	10.10	4.5	1	40
50	0.29	0.66	0.78	0.97	6.06	3.0	1	36
70	0.22	0.78	0.86	0.98	4.33	2.0	1	22

Table 4.4: *The metrics for three selected numbers of clusters on the large data set and with Frobenius-intersection and average-link clustering. The cluster sizes are given as the average, median, smallest, and largest cluster size. The sensitivity and specificity are calculated with respect to the taxonomic family.*

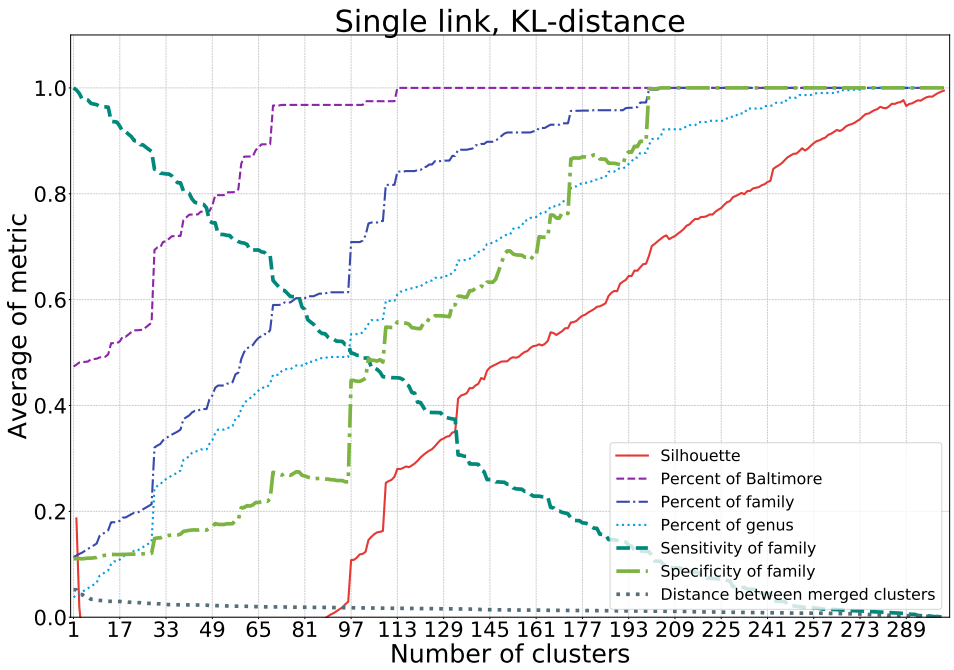
For the average-link clustering (figure 4.6) the specificity, as well as the percentages of Baltimore, family, and genus increase quickly, which represents that the clusters correspond to what the metrics are measuring. The silhouette is larger for low values of k since there are fewer clusters to compare to. The sensitivity is low throughout, indicating that there are several clusters with the same taxonomic family. The distance between the merged clusters is larger for small values of k, but decreases steadily, indicating that there are no obvious values to stop merging at. Note that the Frobenius-intersection and the KL-distance behave similarly, indicating that they identify patterns in the data equally well. Therefore, the rest of the results focus on the Frobenius-intersection distance function.

Figure 4.7 contains example output from the clustering, with the each viruses host's taxonomic class, taxonomic family, and Baltimore classification for $k = 50$ clusters. The clustering takes roughly 1s to compute. Most clusters contain a single family, but there are also unit-sized clusters and clusters with several families. In some clusters, the only common factor is the class of the host-species, note that this is an ancestor to the host-species, roughly corresponding to if the virus infects an insect, bird or vertebrate. The Baltimiore classification is very consistent throughout.

In table 4.5, we list the six largest clusters (with $k = 50$), along with which families and hosts are most prominent in those clusters. Again, most of the clusters contain one prominent family. However, the largest cluster has two roughly equally large families. Most of the clusters contain several different hosts.

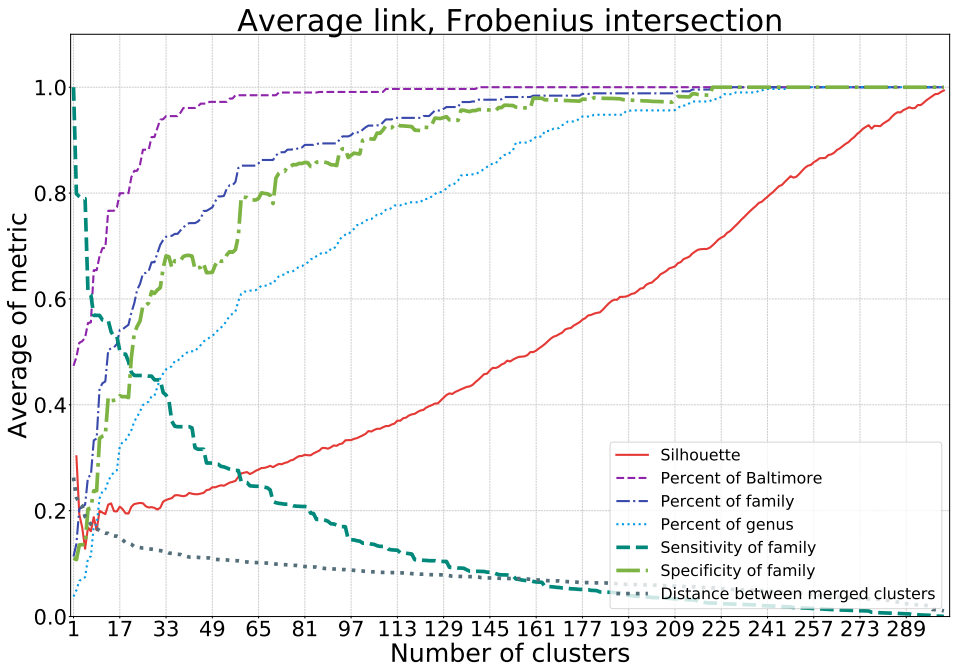


a *Frobenius intersection.*

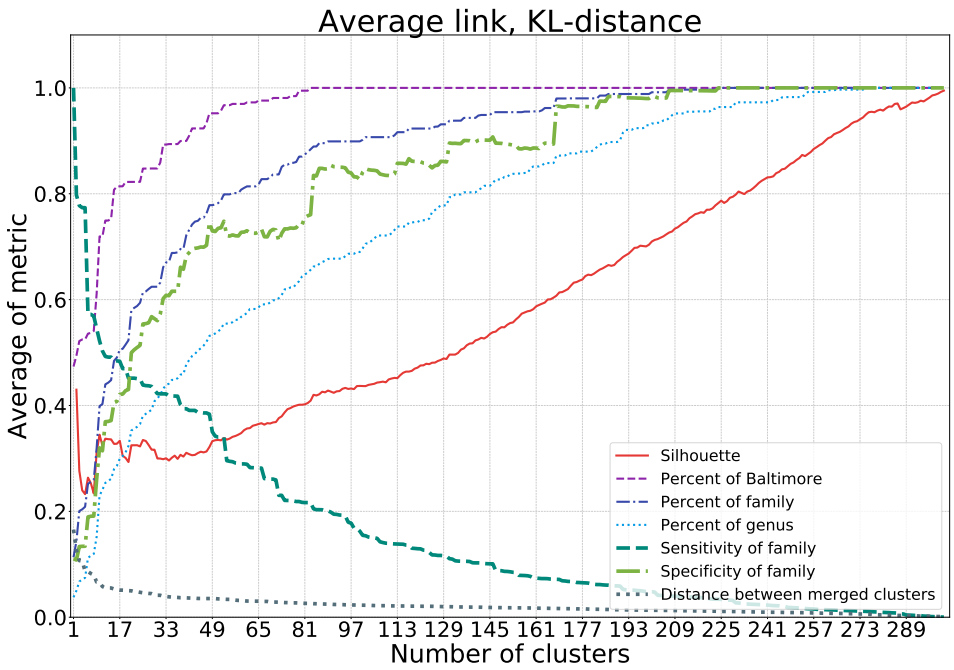


b *KL-distance.*

Figure 4.5: *Single-link clustering of the large data set with our clustering metrics with respect to different values of k number of clusters. To improve readability, the silhouette value for $k = 1$, which would always be 1, is excluded from the graphs. The merge distance is the distance between the merged clusters at that step.*

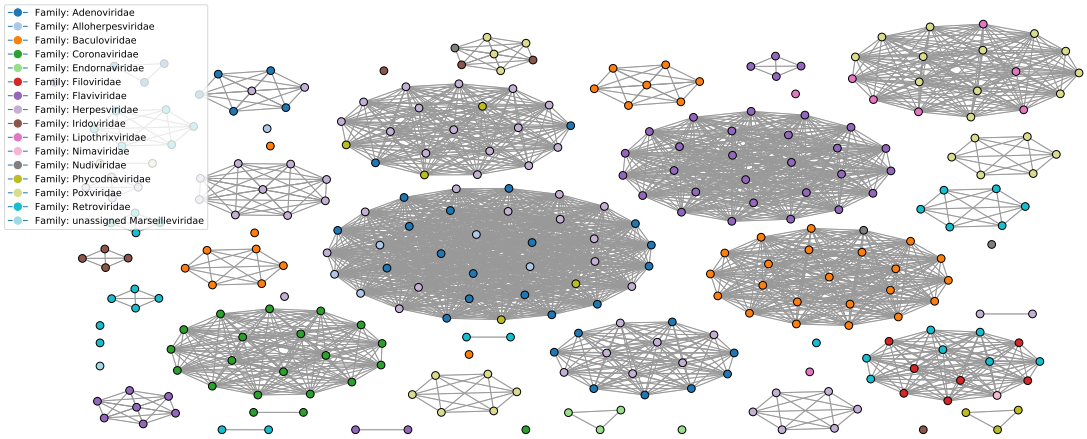


a *Frobenius intersection.*

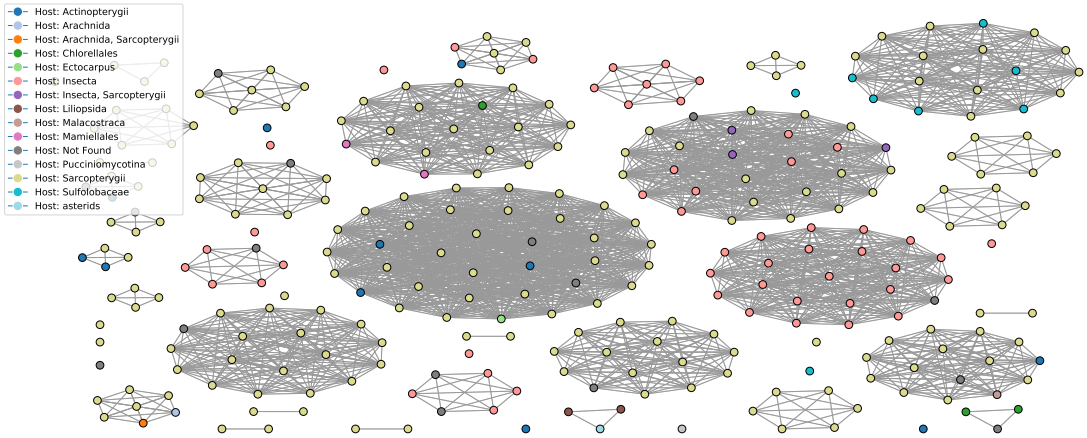


b *KL-distance.*

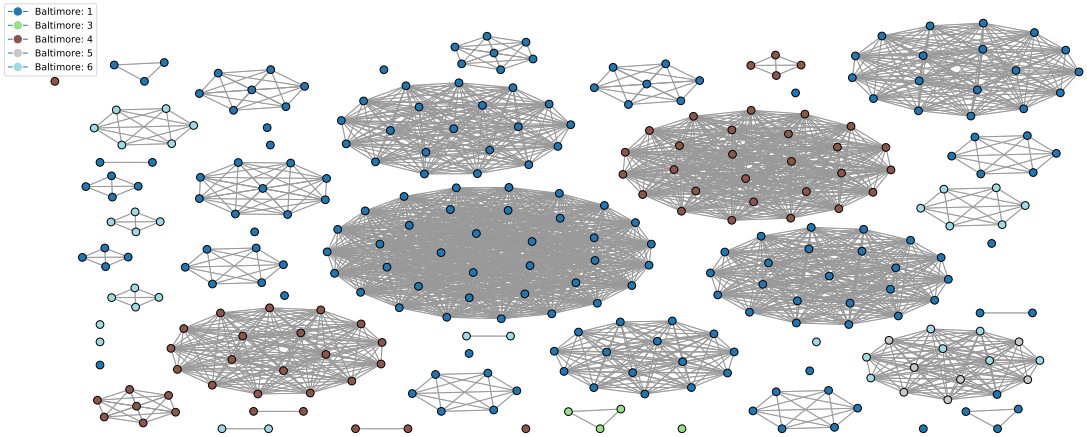
Figure 4.6: *Average-link clustering of the large data set with our clustering metrics with respect to different values of k number of clusters. To improve readability, the silhouette value for $k = 1$, which would always be 1, is excluded from the graphs. The merge distance is the distance between the merged clusters at that step.*



a *The taxonomic families of the viruses.*



b *This figure shows how the clusters of viruses correspond to their host-species' taxonomic class. For those data points for which the class "Not Found" is specified, we either did not find a host species, or we could not determine the host-species' taxonomic class. We display the taxonomic rank of class here since there are too many host species for an illustration, and also no immediately obvious correlations.*



c *The Baltimore classification of the viruses.*

Figure 4.7: *An example of which families and hosts cluster together with our method with $k = 50$, and Frobenius-intersection. The distances between clusters are not representative of how far apart the clusters are, the images only tell us which VLMCs belong to the same clusters.*

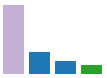
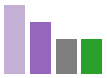
Size	Family		Host	
36	Adenoviridae: 17 Herpesviridae: 12 Alloherpesviridae: 5 Phycodnaviridae: 2		Homo sapiens: 7 Equus caballus: 6 Gallus gallus: 5 Rana pipiens: 2	
27	Flaviviridae: 27		Homo sapiens: 16 Aedes albopictus: 5 Aedes aegypti: 3 Chlorocebus aethiops: 2	
23	Baculoviridae: 22 Nudiviridae: 1		Lepidoptera: 4 Trichoplusia ni: 3 Mamestra configurata: 2 Helicoverpa armigera: 2	
20	Herpesviridae: 15 Phycodnaviridae: 3 Adenoviridae: 2		Homo sapiens: 4 Gallus gallus: 3 Ostreococcus tauri: 2 Chlorocebus aethiops: 2	
18	Poxviridae: 12 Lipothrixviridae: 6		Homo sapiens: 4 Mus musculus: 4 Acidianus convivator: 3 Acidianus: 2	
18	Coronaviridae: 18		Homo sapiens: 3 Chiroptera: 3 Sus scrofa: 2 Turdus hortulorum: 1	

Table 4.5: *The six largest clusters from figure 4.7, with the distribution of families and hosts within the clusters. Only the four largest groups within each cluster are displayed. In most clusters, there is primarily a single family, but there is no correlation with respect to the hosts. Note that the bar height is normalised per cluster and thus not comparable between clusters.*

We produce a dendrogram for the small data set, depicted in figure 4.8, illustrating the order in which clusters are merged. Most commonly, the connections are supported by the taxonomic family of the viruses. The dendrogram for the large data set is too large to display here, but the results are similar.

We cluster our set of bacteria and combine the bacteria with the large data set in figure 4.9. The silhouette values of the clustering of bacteria are larger, indicating that the created clusters are more well-separated than in the virus case. Moreover, the metrics overall seem to correspond more closely to the taxonomic families and genera (35 and 41 respectively) of the bacteria, for instance, a percent of taxonomic family of 88% at $k = 35$. With the two data sets combined, the bacteria and viruses separate reasonably well. However, with a small number of clusters, the organisms mix before merging all of the families of viruses and bacteria.

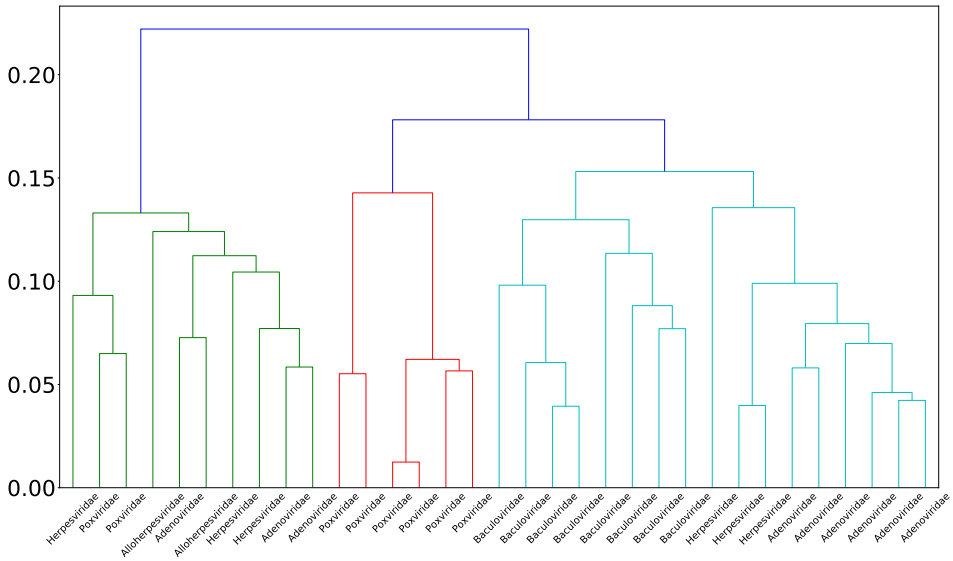
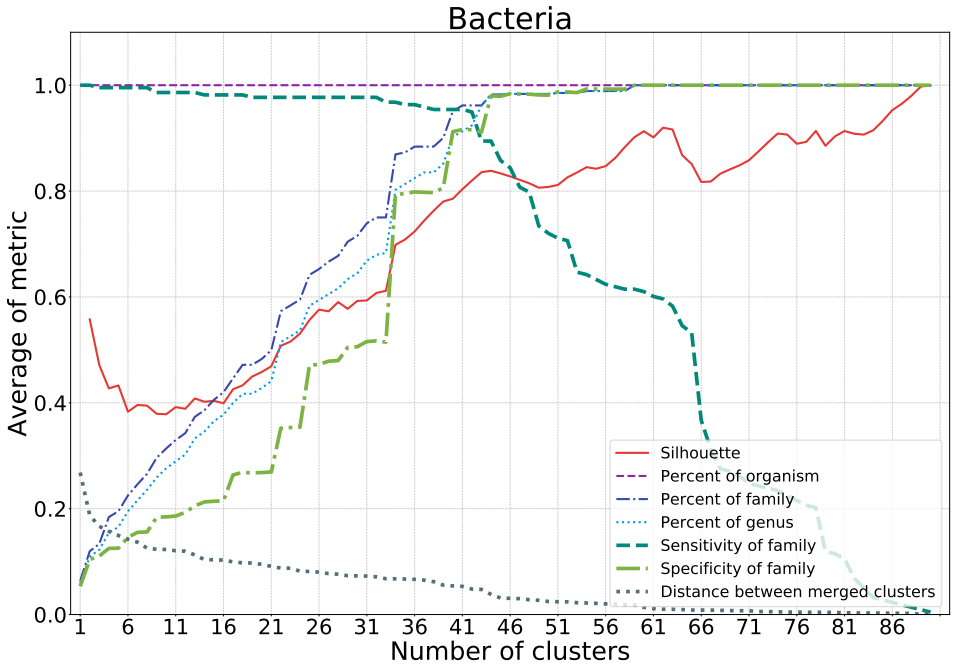
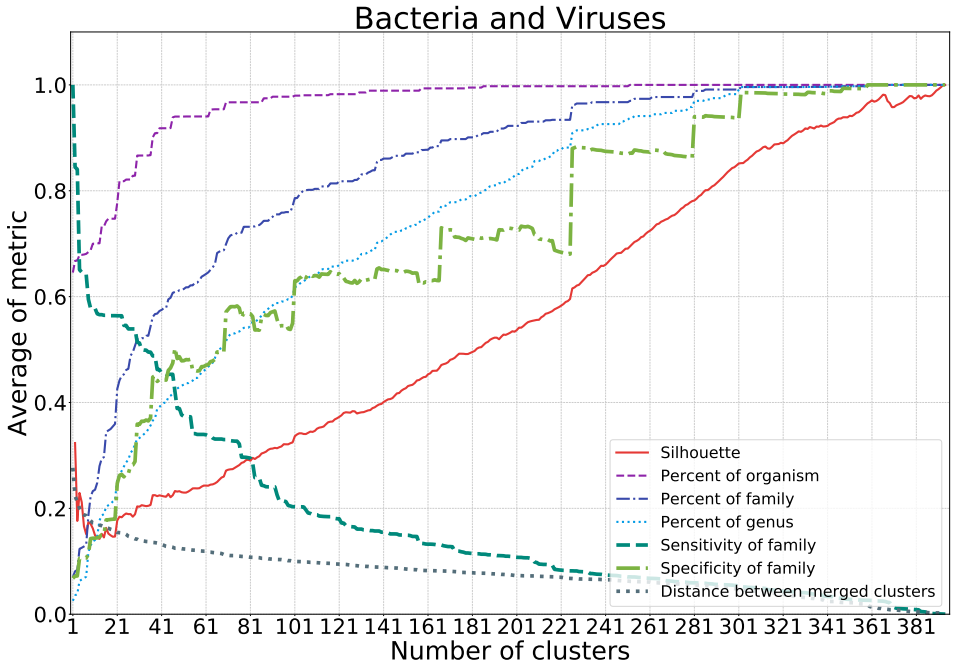


Figure 4.8: *The dendrogram for the small data set. Note how most of the early connections, and a lot of the later connections connects viruses in the same family.*



a *Bacteria data set.*



b *The virus and bacteria data set combined.*

Figure 4.9: *These images illustrate how the metrics decrease upon adding a second type of organism into the data set. The bacteria and viruses separate fairly well, but with fewer number of clusters they start to mix. Note also how much easier the bacteria set is to cluster (as indicated by the higher silhouette).*

4.4 Alignment-based clustering methods

We have attempted to run a number of alignment-based clustering methods on our small data set. In general, they suffer from issues relating to memory and execution time.

DNACLUST and SLIDESORT runs successfully on the small data set with only 160 base pairs for each DNA sequence. However, for the full length sequences, they both crash. We do not present the results here since they are not comparable with full length sequences.

We also ran VSEARCH, which contains an implementation of UCLUST, on our small data set. The clustering took 2.5 hours on our test computer and the results of our metrics are slightly worse than for our method, see table 4.6. We used an id of 0.419 for UCLUST which gave 8 clusters.

Method	**Sensitivity**	**Specificity**	**% Family**	**% Genus**
VSEARCH	0.36	0.51	0.70	0.56
Our	0.38	0.58	0.73	0.57

Table 4.6: *The results of the* VSEARCH*'s implementation of* UCLUST *versus our method.*

5

Discussion

5.1 Distance and clustering outcomes

Our method identifies the taxonomic family of viruses and bacteria with high accuracy (78% and 88% respectively), with average-link clustering and Frobenius-intersection distance. With even higher accuracy (97%), the clustering identifies the Baltimore type of the viruses in our large data set. However, the sensitivity with respect to taxonomic families is quite low which corresponds to families divided into several clusters.

Moreover, the distance function Frobenius-intersection very closely matches the results of the Kullback-Leibler distance. This correlation is clear from the metrics of the distance calculations and the clustering metrics, as well as producing similar clusters. In addition to our metrics, we could have analysed if VLMCs which are close to each other have the same order as sorted by different distance functions. However, we have primarily been interested in how well the distance functions capture, for instance, the taxonomic family, rather than requiring that they are captured in the same order.

That the Frobenius-intersection distance function works so well is somewhat surprising. There are plenty of artificial examples where the Frobenius-intersection ignores essential features in the PSTs. Take, for instance, two models containing identical probabilities for the states A, C, G, and T, but one of the models has the extra state TT, from which the probability of generating a T is 100%. These models would clearly generate very different sequences since the second model would eventually only generate Ts, but they would have a distance of 0 between them according to the Frobenius-intersection distance. However, when VLMCs are trained from sequences, the states are not independent of each other, a parent state in the PST has to contain the distribution of nucleotides for all of its children. This property allows the Frobenius-intersection to capture important aspects of the data, even when it ignores states.

5.2 Time and memory requirements

The Frobenius-intersection distance is fast. It has a time complexity of $O(m|\Sigma|)$ per distance calculation (roughly 4s on our large data set in total with n^2 distance calculations, but this could be reduced to $\binom{n}{2}$ if we take symmetry and identity into account), with m number of states in the smallest model and alphabet Σ. Note that it is not possible to create a distance function with better time complexity without excluding information. Currently, the distances are calculated sequentially in the clustering procedure. However, the task of calculating the distances is embarrassingly parallel. This property offers an additional speedup corresponding to the number of cores on the computer calculating the distances.

The average-link clustering has a time complexity of $O(n^2 \log n)$ (roughly 1s in total). For our data sets, the sequential time complexity for the distance calculations, $O(n^2 m|\Sigma|)$, dominate the total time complexity of the clustering. Moreover, the clustering requires $O(nm|\Sigma|+n^2)$ memory for the VLMCs and distances respectively.

Our method is, therefore, more efficient than alignment-based methods, which require memory and time in the order of the full DNA sequences' lengths, which is a significantly larger value than both n and m.

These time and memory complexities do not contain the generation of the VLMCs, which takes both time and memory in the order of the length of the DNA sequence. However, since this is a pre-processing step, the models only have to be generated once for every data set.

5.3 Selecting number of clusters

It is, in general, not possible to select the number of clusters to be equal to, for instance, the number of known taxonomic families in the data, and expect a good clustering. One contributing factor for this result is the fact that the average-link clustering method does not have a concept of noise. Specifically, if the data contains outliers, they can form unit-sized clusters.

For analysis of the clusters, the dendrogram, see figure 4.8, can be more helpful. The dendrogram alleviates the need to select a fixed number of clusters. Instead, it is possible to select a cut-off point on a cluster-by-cluster basis.

5.4 Analysis of unknown data

The resulting clustering of VLMCs can be used to classify new VLMCs that are generated from unclassified DNA sequences. The average distance from the new VLMCs to the clusters can be analysed to observe if any cluster is much closer than the others. The members of the nearest cluster may indicate what type of virus the new VLMCs might be.

If the input data to the clustering procedure, on the other hand, is unknown, it can be useful to identify prominent signals in the resulting clusters. For instance, signals such as early connections, large clusters, or clusters that are unaltered for extended periods. It may also be beneficial to include genetic signatures with known properties to get a better idea of what the clusters represent.

5.5 Remaining issues and future work

Our proposed method has some remaining issues which we have not addressed. Furthermore, there are some remaining questions in regards to which signals are prominent in the clusters, how much is noise, and what the origin of the signals are.

5.5.1 Genomic signature generation

We have observed that there is a wide range of parameters (between 40 and 1000, see figure 4.1) for the models for which the results of the distance functions are similar. However, we have not been able to identify whether it is possible to improve the results by selecting different amounts of parameters on a model by model basis.

Dalevi et al.'s method [5] allows a constant Δ-value cut-off for the pruning of nodes from the PST. However, this cut-off gives models with a considerable variation in size, since Δ-values grow with sequence length. Therefore, unless a cut-off is chosen per sequence, longer training sequences have enormous models, and short training sequences have small models. For this reason, we elected to use a fixed number of parameters. A more advanced heuristic for selecting the number of parameters or cut-off value may improve the accuracy of the clustering.

Furthermore, there are faster and more memory-efficient algorithms for the VLMC generation process than Dalevi et al.'s, for instance, Schulz et al.'s *Pisa* tool [25], for which, however, there is currently no implementation available. A more efficient VLMC-training implementation is needed in to analyse organisms with larger genomes, such as most bacteria, plants, and animals.

5.5.2 Signals in the data

We observe that the clusters correlate with several possible signals in the data, such as the Baltimore type of the viruses, and the taxonomic rank, while the host species is not as prevalent. However, in some cases, we have been unable to identify possible causes for clusters. Further research may reveal other biological correlations within the clusters.

5.5.3 GC-content prevalence

A large difference in GC-content implies a large distance, regardless of distance function. This correlation is present even in the cases where the VLMCs are in

the same taxonomic family or genus, see for instance figure 4.2. GC-content is a prevalent signal since each parameter in the VLMC implicitly contributes to the total GC-content. Hence, a large difference in GC-content implies differences between many parameters and therefore implies a large distance between the models. This prevalence of GC-content poses a problem for identifying taxonomic classes with high internal variability of GC-content, such as Poxviruses [23], see figure 3.2 for an example of the Poxvirus distance results.

5.5.4 Homology-bias

In the cases where the original DNA sequences are similar (homologous), we expect them to have a small distance. Two sequences that are very similar will produce similar VLMCs, in which case it is not surprising that the distance functions identifies them as similar. We are interested in identifying pairs of DNA sequences which are not homologous but still share strong signals. Holmudden [9] have shown that the genomic signatures are preserved throughout the sequence in regards to distance; however, we have not constructed a test for homology in regards to clustering.

However, we do not want to remove homology as a factor from the clustering results. If two sequences are homologous, that is a strong indication of, for instance, taxonomic ordering.

5.5.5 Learning classes

Cuzzolin et al. [3] designed a distance function for HMMs where they learn specific classes. A similar approach can be implemented for our application, by specifying classes which the distance function should learn. For instance, it may be possible to learn a distance function for taxonomic family, a distance function for host species, antibiotic resistance, etc. Each of the distance functions can then be used to cluster the data sets in order to provide a clustering based, for instance, purely on the taxonomic family.

5.5.6 Noise-resistant clustering algorithms

Since average-link clustering is sensitive to noise, it may be possible to implement a different clustering algorithm that can improve the results, two approaches follow. Mixture models [20], where the diameter of clusters can vary on a cluster by cluster basis. Neighbour-joining [24], which forms intermediate vertices which represent the clusters.

5.6 Ethics of DNA analysis algorithms

Our work can contribute to the classification of pathogens, helping with medical work in multiple areas. However, our proposed approach can also potentially be used for unethical research into humans of different social groups, or for arbitrary

classifications of species. In general, research in regards to genetics can be adopted for controversial use. Consider, for instance, the optimisation of human genes, or eugenics in general, and specifically when used for racial policies.

6

Conclusion

We define several distance functions for VLMCs which we compare to previously defined distance functions for both VLMCs and HMMs. We find that for our application of VLMCs as genomic signatures, our Frobenius-intersection distance has comparable accuracy to the KL-distance, and it is also much faster.

We cluster several sets of VLMCs using average-link clustering and find that, with high accuracy, the clusters of genomic signatures correspond to the taxonomic order of the underlying organisms. The Baltimore type of the viruses is also prominent, while the virus' host species are not equally prevalent. Further analysis may reveal other correlations within the clusters of genomic signatures.

These results are obtained through analysis of data sets consisting of viruses and bacteria with short DNA sequences. A more efficient VLMC training method is needed to train and analyse VLMCs for organisms such as animals, plants and most bacteria.

Our proposed clustering approach is also significantly faster and at least as accurate as alignment-based clustering techniques. This improvement mostly stems from having a much smaller model to compare, which still captures important parts of the DNA sequences.

With our method, it is possible to classify the viruses and bacteria of a data set accurately. Given sequenced DNA samples from patient, environmental or artificial sources, our approach offers a fast and accurate method for analysis and identification of pathogens.

Bibliography

[1] Peter Bühlmann, Abraham J Wyner, et al. "Variable length Markov chains". In: *The Annals of Statistics* 27.2 (1999), pp. 480–513.

[2] C Burge, A M Campbell, and S Karlin. "Over- and under-representation of short oligonucleotides in DNA sequences". In: *Proceedings of the National Academy of Sciences* 89.4 (1992), pp. 1358–1362. ISSN: 0027-8424.

[3] Fabio Cuzzolin and Michael Sapienza. "Learning pullback HMM distances". In: *IEEE transactions on pattern analysis and machine intelligence* 36.7 (2014), pp. 1483–1489.

[4] Daniel Dalevi, Devdatt Dubhashi, and Malte Hermansson. "A new order estimator for fixed and variable length Markov models with applications to DNA sequence similarity". In: *Statistical applications in genetics and molecular biology* 5.1 (2006).

[5] Daniel Dalevi, Devdatt Dubhashi, and Malte Hermansson. "Bayesian classifiers for detecting HGT using fixed and variable order markov models of genomic signatures". In: *Bioinformatics* 22.5 (2006), pp. 517–522.

[6] Robert C Edgar. "Search and clustering orders of magnitude faster than BLAST ". In: *Bioinformatics* 26.19 (2010), pp. 2460–2461.

[7] Ronald A Fisher. "The use of multiple measurements in taxonomic problems". In: *Annals of human genetics* 7.2 (1936), pp. 179–188.

[8] Mohammadreza Ghodsi, Bo Liu, and Mihai Pop. "DNACLUST: accurate and efficient clustering of phylogenetic marker genes". In: *BMC Bioinformatics* 12.1 (July 2011), p. 271.

[9] Martin Holmudden. "Virus Attenuation by Genome-Wide Alterations of Genomic Signatures". In: (2015).

[10] B. H. Juang and L. R. Rabiner. "A probabilistic distance measure for hidden Markov models". In: *AT T Technical Journal* 64.2 (Feb. 1985), pp. 391–408. ISSN: 8756-2324. DOI: 10.1002/j.1538-7305.1985.tb00439.x.

[11] Samuel Kariin and Chris Burge. "Dinucleotide relative abundance extremes: a genomic signature". In: *Trends in genetics* 11.7 (1995), pp. 283–290.

[12] Stephen E Levinson, Lawrence R Rabiner, and Man Mohan Sondhi. "An introduction to the application of the theory of probabilistic functions of a Markov process to automatic speech recognition". In: *The Bell System Technical Journal* 62.4 (1983), pp. 1035–1074.

[13] Chen Lu et al. "A normalized statistical metric space for hidden markov models". In: *IEEE transactions on cybernetics* 43.3 (2013), pp. 806–819.

[14] Tomoko Mihara et al. "Linking virus genomes with host taxonomy". In: *Viruses* 8.3 (2016), p. 66.

[15] Fionn Murtagh and Pedro Contreras. "Algorithms for hierarchical clustering: an overview". In: *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 2.1 (2012), pp. 86–97.

[16] A Muto and S Osawa. "The guanine and cytosine content of genomic DNA and bacterial evolution". In: *Proceedings of the National Academy of Sciences* 84.1 (1987), pp. 166–169. ISSN: 0027-8424.

[17] Saul B Needleman and Christian D Wunsch. "A general method applicable to the search for similarities in the amino acid sequence of two proteins". In: *Journal of molecular biology* 48.3 (1970), pp. 443–453.

[18] World Health Organization et al. *Antimicrobial resistance: global report on surveillance*. World Health Organization, 2014.

[19] Peter Pipenbacher et al. "ProClust: improved clustering of protein sequences with an extended graph-based approach". In: *Bioinformatics* 18.suppl_2 (2002), S182–S191.

[20] Simon Rogers and Mark Girolami. *A first course in machine learning*. CRC Press, 2016.

[21] Torbjørn Rognes et al. "VSEARCH: a versatile open source tool for metagenomics". In: *PeerJ* 4 (2016), e2584.

[22] Dana Ron, Yoram Singer, and Naftali Tishby. "Learning probabilistic automata with variable memory length". In: *Proceedings of the seventh annual conference on Computational learning theory*. ACM. 1994, pp. 35–46.

[23] Sourav RoyChoudhury, Archana Pan, and Debaprasad Mukherjee. "Genus specific evolution of codon usage and nucleotide compositional traits of poxviruses". In: *Virus genes* 42.2 (2011), pp. 189–199.

[24] Naruya Saitou and Masatoshi Nei. "The neighbor-joining method: a new method for reconstructing phylogenetic trees." In: *Molecular biology and evolution* 4.4 (1987), pp. 406–425.

[25] Marcel H Schulz et al. "Fast and adaptive variable order Markov chain construction". In: *International Workshop on Algorithms in Bioinformatics*. Springer. 2008, pp. 306–317.

[26] Kana Shimizu and Koji Tsuda. "SlideSort: all pairs similarity search for short reads". In: *Bioinformatics* 27.4 (2010), pp. 464–470.

[27] B. G. Sürmeli et al. “Unsupervised mode detection in cyber-physical systems using variable order Markov models”. In: *2017 IEEE 15th International Conference on Industrial Informatics (INDIN)*. July 2017, pp. 841–846.

[28] WHO Ebola Response Team. “After Ebola in West Africa—unpredictable risks, preventable epidemics”. In: *New England Journal of Medicine* 375.6 (2016), pp. 587–596.

[29] Agnes Vathy-Fogarassy et al. *Graph-based clustering and data visualization algorithms*. English. 2013;1; New York: Springer, 2013.

[30] Jianping Zeng, Jiangjiao Duan, and Chengrong Wu. “A new distance measure for hidden Markov models”. In: *Expert systems with applications* 37.2 (2010), pp. 1550–1555.

Appendices

A

Selecting sequence length for KL

We selected the sequence length necessary for the Kullback-Leibler distance calculation based on the following experiment. We calculated the average distance between a few selected models, and averaged over 10 re-calculations. We then increased the sequence length in log-spaced steps and selected the shortest sequence length that gave the same average distance as the longer sequences. See figure A.1 for the results of the test. We use a sequence of 16 000.

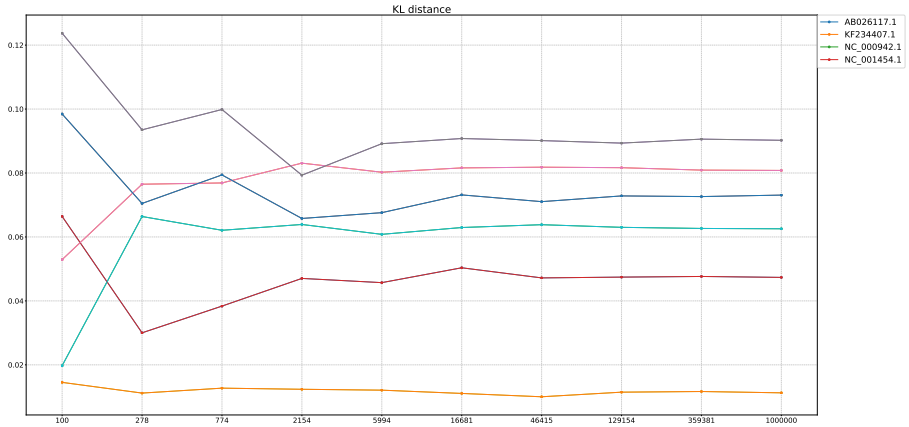


Figure A.1: *Test for how long sequence is needed for a consistent distance calculation using Kullback-Leibler distance calculation.*

B

PST-matching hyper parameter

The PST-matching performs worse than many of the other distance functions. This can be due to several reasons. The hyper-parameter I, was set to 0.5 simply because Sürmeli et al. used this value. The distance measure performed much better if we chose a value much closer to 0. However, with $I \approx 0$, the distance function essentially reduces to our Frobenius-intersection, both with slightly worse results, and a more complex model. Therefore, we have elected to set $I = 0.5$ to illustrate the full PST-matching distance function.

Sürmeli et al. also train their models in a different way than we do. This can be a factor to why PST-matching does not perform very well on our data set.

C

Clustering with other distance functions

We present the clustering metrics for some of the exclude distance functions from the results here. The single link clustering is worse throughout, and all of the distance functions are significantly worse than Frobenius-intersection and Kl-distance.

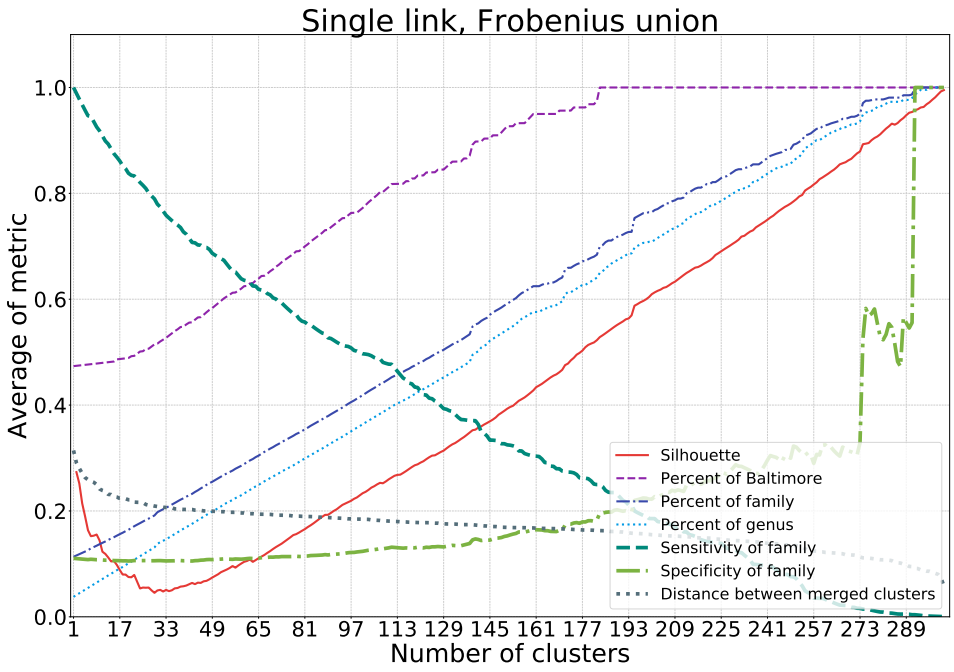


Figure C.1: *Frobenius union, single-link clustering.*

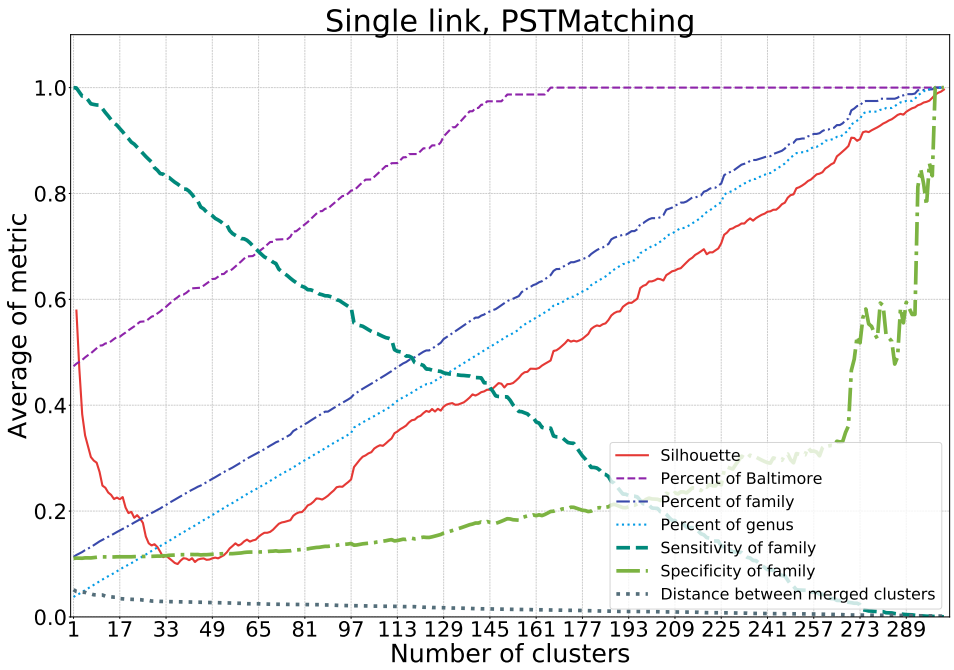


Figure C.2: *PST matching, single-link clustering.*

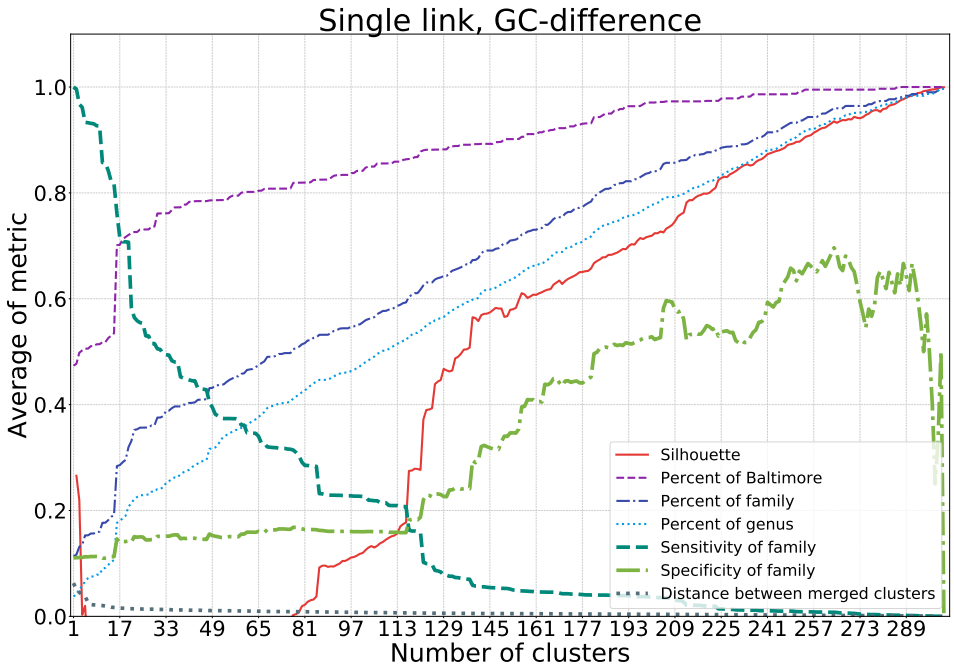


Figure C.3: *GC-content, single-link clustering.*

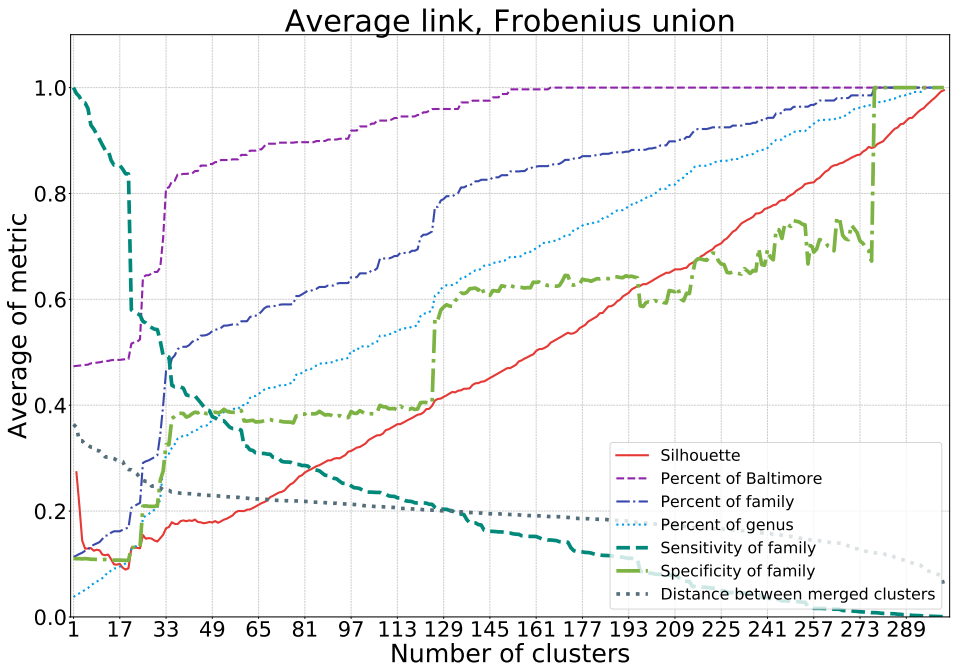


Figure C.4: *Frobenius union, average-link clustering.*

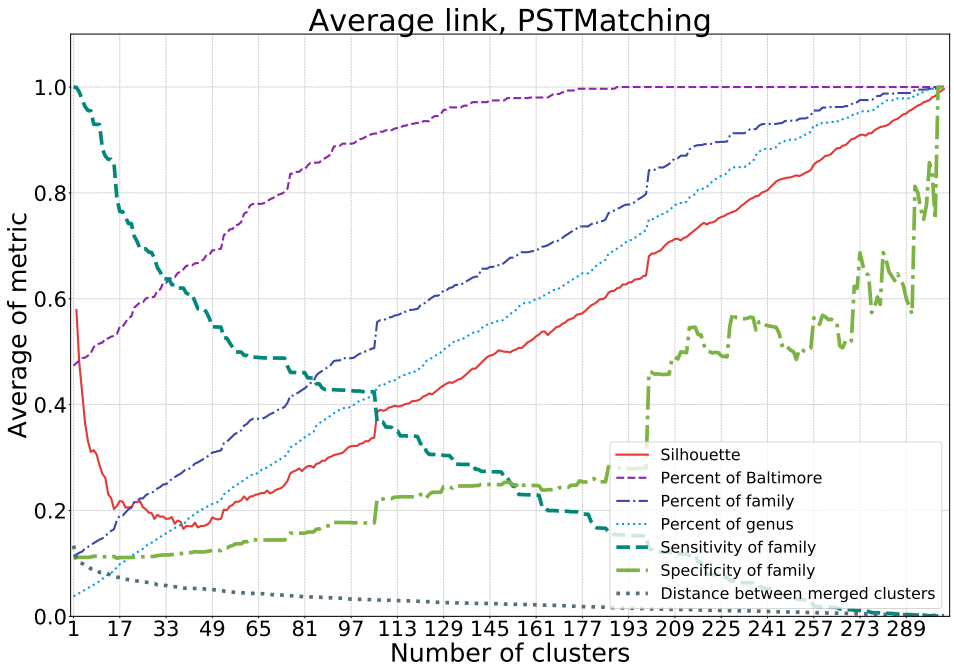


Figure C.5: *PST matching, average-link clustering.*

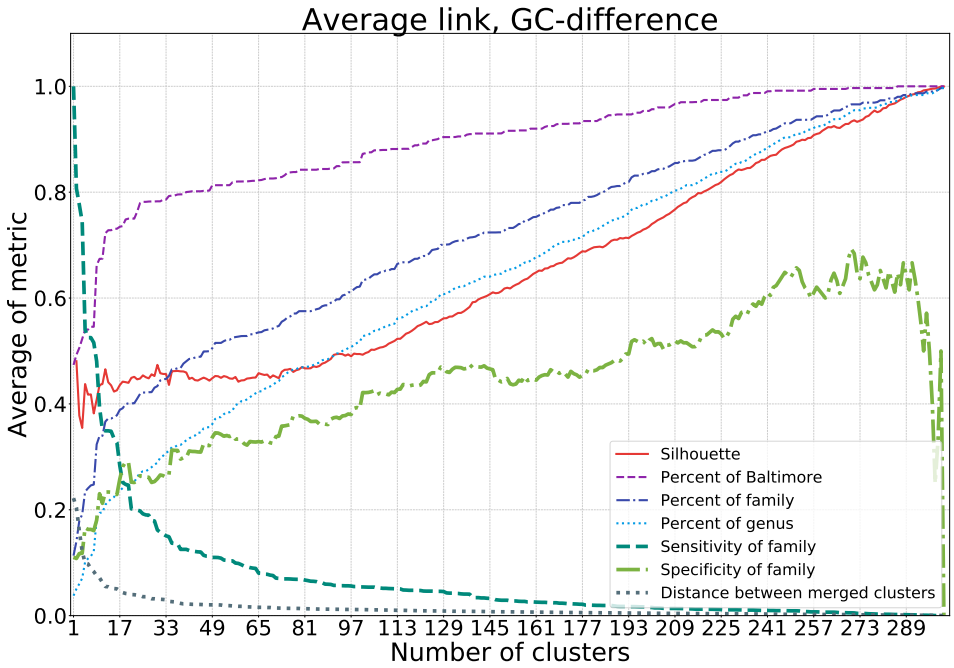


Figure C.6: *GC-content, average-link clustering.*

D

Extended data set

This data set contains multiple strains of the same species. These results indicate that the clustering easily picks up the strains of the viruses, which is not surprising since their DNA most probably is very similar.

k	Sensitivity	Specificity	% family	% Baltimore	Cluster sizes			
110	0.53	0.80	0.76	0.94	28.65	4.0	1	727
70	0.59	0.67	0.67	0.91	45.01	4.0	1	750

Table D.1: *The metrics for the extended data set.*

Size	Family	Host
727	Coronaviridae: 716 Myoviridae: 11	Homo sapiens: 280 Camelus dromedarius: 220 Sus scrofa: 180 Gallus gallus: 141
426	Adenoviridae: 244 Herpesviridae: 172 Alloherpesviridae: 7 Phycodnaviridae: 3	Homo sapiens: 246 Not Found: 82 Gallus gallus: 29 Equus caballus: 17
299	Filoviridae: 196 Siphoviridae: 48 Myoviridae: 31 Retroviridae: 8	Homo sapiens: 196 Epomops franqueti: 174 Myonycteris torquata: 174 Not Found: 63
177	Siphoviridae: 128 Myoviridae: 45 Podoviridae: 4	Mycobacterium smegmatis str. MC2 155: 113 Not Found: 48 Mycobacterium: 14 Mycobacterium smegmatis: 8
142	Myoviridae: 142	Synechococcus sp. WH 7803: 118 Synechococcus: 25
101	Poxviridae: 85 Mimiviridae: 6 Lipothrixviridae: 5 Iridoviridae: 2	Homo sapiens: 51 Mus musculus: 47 Bos taurus: 44 Acinonyx jubatus: 37
89	Herpesviridae: 69 Phycodnaviridae: 5 Adenoviridae: 3 Myoviridae: 3	Felidae: 37 Not Found: 18 Homo sapiens: 8 Chlorella variabilis: 4
81	Siphoviridae: 51 Adenoviridae: 23 Herpesviridae: 6 Filoviridae: 1	Not Found: 61 Odocoileus hemionus columbianus: 7 Pygoscelis antarcticus: 2 Mus musculus: 2
80	Herpesviridae: 80	Homo sapiens: 41 Bos taurus: 20 Macaca fascicularis: 20 Macaca leonina: 20
77	Baculoviridae: 71 Ascoviridae: 4 Polydnaviridae: 2	Not Found: 30 Lepidoptera: 9 Erinnyis ello: 6 Trichoplusia ni: 3

Table D.2: *The ten largest clusters from the extended data set, with the distribution of families and hosts within the clusters. Only the four largest groups within each cluster is displayed. In most clusters, there are primarily a single family, but the correlation with respect to host is not as clear.*